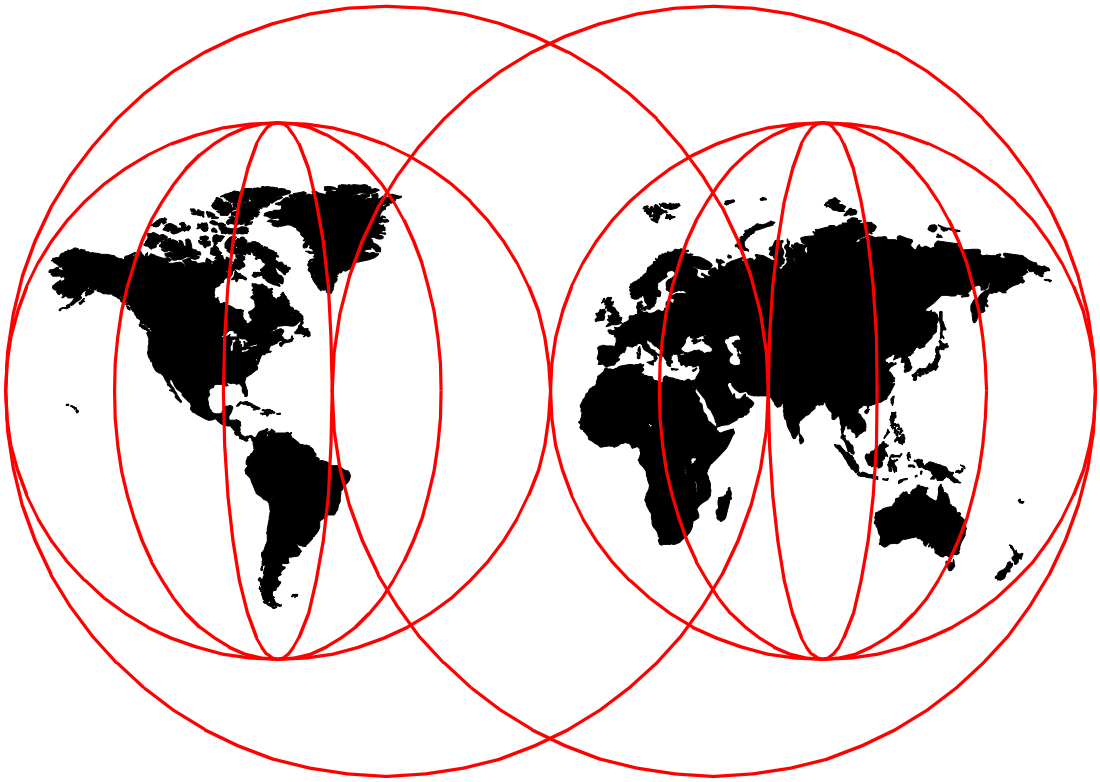
Porting UNIX Applications to OpenEdition for VM/ESA

Erich Amrehn, Neale Ferguson, Jean-Francois Jiguet



International Technical Support Organization

http://www.redbooks.ibm.com

SG24-5458-00

International Technical Support Organization

SG24-5458-00

Porting UNIX Applications to OpenEdition for VM/ESA

August 1999

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix M, “Special Notices” on page 337.

First Edition (August 1999)

This edition applies to Virtual Machine/Enterprise Systems Architecture (VM/ESA) Version 2, Release 3.0, program number 5654-030 for use with OpenEdition for VM/ESA.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HYJ Mail Station P099
522 South Road
Poughkeepsie, NY 12601-5400

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Tables

Preface

VM/ESA Version 2 and OS/390 deliver Open Systems facilities to programmers in the VM and OS/390 environments. In this redbook we discuss the reasons for porting applications. Following this we take a detailed look at a few applications we have ported and describe the facilities and techniques utilized to make them work. Special focus is placed on the business value of these applications, and how to use them in a typical VM environment. We provide significant detail on how to navigate the technical challenges and give many examples and sample code.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization Poughkeepsie Center.

Erich Amrehn is a certified Senior IT Specialist at the International Technical Support Organization, Poughkeepsie Center. Before joining the ITSO he worked as technical consultant to the IBM System/390 division for e-commerce on S/390 in Europe, the Middle East and Africa. He also has 13 years of VM experience in various technical positions in Germany and other areas in Europe and worldwide.

Neale Ferguson is Manager of Systems Services at TAB Limited in Australia. He has 18 years of experience in the VM field. He holds a bachelor's degree in computing science from the University of Technology Sydney and a masters degree in cognitive science from the University of New South Wales. His areas of expertise include VM, TCP/IP, VTAM, DB2, VSE, and P/390. He has written extensively on OpenEdition and intranets.

Jean-Francois Jiguet is an IT specialist in system management for IBM Global Services in France. He has 12 years of experience in the VM field. He has worked at IBM for 20 years. His areas of expertise include VSE, DB2, TCP/IP, VTAM, P/390 and R/390.

Thanks to the following people for their invaluable contributions to this project:

Roy Costa
International Technical Support Organization, Poughkeepsie Center

Melissa Howland
IBM Endicott

Mike Donovan
IBM Endicott

Dr. Gretchen L. Thiele
Princeton University

Comments welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in “ITSO Redbook Evaluation” on page 351 to the fax number shown on the form.
- Use the online evaluation form found at `http://www.redbooks.ibm.com/`
- Send your comments in an internet note to `redbook@us.ibm.com`

Chapter 1. Introduction to porting

Reports from Carl Greiner of the Meta Group and Barry Graham of Xephon report continuing growth in S/390. A net growth of 33% per annum is being achieved in S/390 processing power (a gross figure of 50% - the difference being due to data center consolidations). Similarly, DASD, which is estimated at 3 Gb per MIP, is also growing at a healthy rate.

At the same time, the price of this hardware is dropping rapidly. Currently, hardware accounts for around 40% of the IT budget. The Meta Group estimated that this figure will drop further as both CPU technology (via CMOS and CMOS/ECL) and DASD improve their price/performance. Some estimate a cost of $1 per megabyte for DASD in the next few years.

Once upon a time the big push to UNIX was based on the inexpensive nature of the hardware. This is very true of the desktop and departmental systems. However, to meet the needs of the *enterprise,* the UNIX high-end equipment is not “dirt-cheap”. In fact, the price of S/390 systems is dropping to such a point that the advantage once held by UNIX in this area is no longer compelling. Considering that soon hardware will no longer be the big ticket item within an enterprise, hardware cost is not going to be the deciding factor.

In terms of systems management, S/390 still offers considerable advantages over UNIX. As system management facilities are developed for the UNIX environment, the price of the software is increasing enormously. Thus, there are very sound business reasons developing for keeping your applications on the S/390 platform.

1.1 Leveraging your investment

The term *legacy* is often used to refer to applications running under S/390, and often it is used to imply that they are outmoded, inefficient, or wasteful. However, while these applications may have been around for a long time, if they are generating cash flow for your business, then they are assets to be valued and not derided. For example, in the case of the enterprise TAB Limited in Australia, such legacy systems are responsible for the turnover of in excess of $AUD3.6 billion per annum.

This heritage is not going to disappear, nor would you want it to. What you do want is for these applications to become part of the new enterprise.

For want of a better term, “Client/Server” is here to stay. While VM has been doing it since day one, in its modern incarnation “Client/Server” has passed through various stages in its development:

- The “gee-whiz” stage, where people suddenly claimed that the mainframe was dead and rushed to downsize.
- The “oh-gee-what-have-we-done” phase, where expectations were not met.
- Now it has entered a “reality” phase, where people understand where the client/server concept fits into the enterprise and that it is but a part of the overall IT facilities. People now talk of network-centric (net-centric) computing.

Therefore, the S/390 has to become an active participant in the net-centric story. We believe it should be a driving force. This is where OpenEdition is coming from.

We are taking the heritage of systems management, reliability, availability, and skills, and leveraging them by adding value in the form of open standards and interoperability.

There are a number of reasons why we believe organizations should be looking at OpenEdition as a base for UNIX-style applications:

- The arguments about reliability, availability, and serviceability are still the strongest. When equipment comes with warnings about not putting critical data on hard drives, it indicates that robustness still runs a poor second to performance.
- There is an enormous investment in S/390 that we wish to leverage.
- There is a huge suite of S/390 applications that are not going to disappear.
- There is a larger pool of applications that could be acquired “off the shelf”.
- The OS/390 UNIX Services developers cite several reasons for porting applications. Their reasons are just as valid for VM:
  - Port UNIX-type C language applications to OS/390 and possibly extend them to take advantage of unique OS/390 strengths.
  - Give workstation users access to OS/390 applications and data.
  - Use OS/390 as a large data server.
  - Use OS/390 for its access to large, high-speed tape devices for data-intensive applications.
  - Automatically back up and migrate hierarchical file systems.
  - Use OS/390 as a server for batch applications.
  - With OpenEdition DCE, access remote applications and data transparently.
  - Enterprises using OS/390 Unix Services can use any combination of these applications.

Over the last few years, there have been real pressures to look at different platforms on which to run business-critical applications. The pressures are coming from all sides, for example:

- Users want access to the latest “killer” applications.
- Computer Science graduates are being hired who are UNIX- or PC-literate only.
- There is a need to integrate the various platforms/environments within the organization.
- There is a need to reduce the cost of computing.
- There is a need to deliver solutions more rapidly.

Today VM/ESA has the tools and the environment to meet all these requirements including pipes, Web servers, RDBMS, and since VM/ESA 2.1.0, OpenEdition. The OpenEdition environment supplies the APIs and multitasking capabilities needed to develop and run many new and existing applications. The Shell and Utilities, which includes tools such as `awk`, `sed`, `make`, and `c89`, provide developers with the means to port and develop applications.

So what applications can make your VM/ESA more valuable to the organization? In the following sections we describe applications that have been ported; their

value in a business; what benefits they bring; and what lessons were learned during the porting exercise.

1.2 The bottom line

So why port applications to OpenEdition or why use applications that have already been ported? To our way of thinking it can best be summarized in the following quote:

“There is this special biologist word we use for 'stable'. It is 'dead'.” -- Jack Cohen

That is, a system needs to keep improving, to run new applications, and to leverage off what it is doing, rather than rest on its laurels. OpenEdition for VM/ESA provides an environment in which this can take place.

If you have not had to work in the UNIX world yet, you probably will soon. You can do this either on a platform that you are familiar with and one that has a heritage of applications and data that has driven your organization’s bottom line, or you can have change thrust upon you when the new UNIX server arrives in the corner of the room and you are expected to get it running and have it make money for your enterprise.

This book is an attempt to facilitate the first course of action by demonstrating what is possible under OpenEdition for VM/ESA and providing some applications and the know-how to get them running.

1.3 The resources

If you have decided to take the plunge, there are resources and software level requirements you must be aware of.

1.3.1 Software required

Appendix J, “Product service levels” on page 303 contains the SRVAPPS files of the products used during the residency. We recommend that your enterprise be at or above these levels.

The VM/ESA system we used was VM/ESA 2.3.0 at service level RSU9901. You also need a C compiler because most of the portable applications (POSIX or UNIX) are written with the C language. Since VM/ESA 2.3.0, the Language Environment for OS/390 & VM run time library is part of the VM/ESA base product.

With the base VM/ESA 2.3.0 you receive the following:

- C Runtime Support, and POSIX C Socket Library
- POSIX interfaces in CSL library
- Byte File System support
- CMS commands to work with BFS objects
- REXX and CMS Pipelines access to POSIX functions
- XEDIT support of BFS files
- CP support for POSIX
- Java JDK 1.1.4

You also need to have the following program products:

- OpenEdition Shell and Utilities (5654-030) Version 2.3.0
- C for VM/ESA compiler (5654-033) Version 3.1.0
- TCP/IP for VM FL310

Optional program products we used included:

- RSCS Networking Version 3, Release 2.0-9803
- NetView

1.3.2 Skills needed

To port applications, you need to have knowledge of the following topics:

- UNIX commands (`c89, make, grep, ls, mkdir, chmode, cat`).
- C language in order to trace, debug tools and modify C programs (this skill is not needed if you are simply going to install and customize the applications).
- UNIX korn shell scripting language in order to analyze script processes (only if you will troubleshoot or modify the packages).
- Standard VM skills to implement your work environment and to analyze errors of the application ported with TRACE and VMDUMP.

To simply install, configure and run the applications described in this redbook, you do not require very much experience with any of these skill areas.

1.3.3 Our workspace

For our porting test, we built completely new Byte File System servers. In these servers, we loaded all the packages to be ported to OpenEdition for VM/ESA.

The actual storage requirements are shown in 2.2.1, “BFSSERVE CP directory entry” on page 14.

1.3.4 The applications we ported

We have structured this book by examining the issues involved when porting an application; porting an easy application as your first step (GNU zip); a detailed look at the porting of an important infrastructure applications (GNU make); the building of an infrastructure for your VM system (for example, a system logger and a time server); the implementation of some major applications (for example Samba, LDAP, and Apache); and the addition of value-add products.

1.3.5 Other resources

There are several resources that you can access to assist in porting and using applications, all of which are to be found in Appendix N, “Related Publications” on page 339. Those that you should pay particular attention to are:

- *OpenEdition for VM/ESA Implementation and Administration Guide*, SG24-4747
- *OpenEdition for VM/ESA Sockets Reference Version 2 Release 1*, SC24-5741
- *IBM C for VM/ESA Library Reference, Volume 1 Version 3 Release 1*, SC23-3908
- *OpenEdition for VM/ESA Command Reference Version 2 Release 3.0*, SC24-5728
- “Getting Started with VM/ESA OpenEdition” presentation materials and other useful information:

http://www.ibm.com/s390/vm/openedition

- Information on packages ported to OpenEdition for VM/ESA and the packages themselves:

  http://pucc.princeton.edu/~neale/vmoe.html

- *OpenEdition for VM/ESA User's Guide Version 2 Release 1.0*, SC24-5727

1.4 Process management

In order to set up, configure, and possibly debug OpenEdition for VM/ESA application programs, you need to have a basic understanding of the process model that is used within the OpenEdition for VM/ESA environment. A complete description of the CMS multitasking process management model can be found in *CMS Application Multitasking*, SC24-5766-01. We do not repeat that discussion in this redbook.

Here, we discuss the C functions and assembler callable services (also known as syscalls). Generally, a callable service is used whenever the C Run Time Library (RTL) needs to do something which requires being in an authorized state. Some C functions are passed by the C language support to the CMS multitasking kernel via the assembler callable services. The OpenEdition for VM/ESA callable services are documented in *OpenEdition for VM/ESA Callable Services Reference Version 2 Release 3.0*, SC24-5726.

When discussing process management, we consider:

- Processes
- Threads
- Interprocess communications (IPC)
- Signals

1.4.1 Processes

Processes are created in a hierarchical structure whose depth is limited only by the virtual memory available to the virtual machine. A process may control the execution of any of its descendants by suspending or resuming it, altering its relative priority, or even terminating it. Termination of a process by default causes termination of all its descendants; termination of the root process causes termination of the session.

From a UNIX perspective, this means OpenEdition for VM/ESA assigns a *process ID* (PID) to the process. Control blocks in common storage and the CMS kernel are built to represent this piece of work. These control blocks build the infrastructure that OpenEdition for VM/ESA uses to keep track of all you do.

An OpenEdition for VM/ESA program may use a number of OpenEdition for VM/ESA services to create new processes or to enable multithreading within the process itself. There are no means to prohibit creation of new processes by an application programmer.

To control processes, the following basic services are available:

fork() function and BPX1FRK service — Create a New Process
fork() in OpenEdition for VM/ESA is not a full function fork() as defined by the IEEE 1003.1 standard. The VM/ESA implementation of fork() creates a new

process within the CMS virtual machine. Once a fork() call is issued, the next call to the CMS kernel must be some form of exec() call. In this respect, fork() is similar to spawn() on OpenEdition for VM/ESA.

spawn() function and BPX1SPN service — Spawn a Process
spawn() starts a new process, but the new child process is started with another program in the Byte File System (BFS), as indicated by the parent process on the spawn() call. After the spawn() call, the two processes continue as independent processes.

exec family of functions and BPX1EXC service — Run a Program
This does not start a new process, but replaces the program in the current process with another program as indicated on the exec call.

1.4.1.1 Spawning a new process

The spawn service is a mechanism for starting a new process. The spawned process is not a copy of the parent process. On the spawn() call, the parent process specifies the name of a BFS program to start in the child process, and the new process is started with this new program being given control at its main entry point. The child process does inherit the file descriptors from the parent process.

If your application uses pipes or shared memory and you switch to using spawn(), there are differences to be aware of.

1.4.1.2 Replacing the program in a process

If a program in a process wants to replace itself with another program, it can use the exec service. This service will preserve the current process environment, but completely replace the program that is running within that process. A successful exec call (the exec family of functions) will never return control to the calling program, but control will be passed to the main entry point of the new program that is specified on the exec call.

The exec service is typically used after a fork() by the child process to replace the parent process program with a child-specific program to perform the child-related functions of the application.

1.4.1.3 UID/GID Assignment: Process Authorization

Every virtual machine is automatically assigned a User ID (UID) and Group ID (GID), although these may be default values. Various UID/GID assignments are:

Real UID
At process creation, the real UID identifies the user who has created the process.

Effective UID
Each process also has an effective UID. The effective UID is used to determine owner access privileges of a process.

Normally this value is the same as the real UID. It is possible, however, for a program that resides in the hierarchical file system to have a special flag set that, when this program is executed, changes the effective UID of the process to the UID of the owner of the program. A program with this special flag set is said to be

a set-user-ID program. This feature provides additional permissions to users while the set-user-ID program is being executed.

Real GID
At process creation, the real GID identifies the current connect group of the user for which the process was created.

Effective GID
Each process also has an effective group. The effective GID is used to determine group access privileges of a process.

Normally this value is the same as the real GID. A program can, however, have a special flag set that, when this program is executed, changes the effective GID of the process to the GID of the owner of this program. A program with this special flag set is said to be a set-group-ID program. Like the set-user-ID feature, this provides additional permission to users while the set-group-ID program is being executed.

The real UID/GID tells us who we really are; the effective UID and GID are used for file access permission checks; the saved values of UID and GID are stored by the exec() function.

Table 1 shows the relation between the values described and how they are manipulated by various function calls.

Note: Although we only reference the setuid() function, the same applies to the GID as handled by the setgid() function.

Table 1. Summary of how UID may be changed

ID	Exec-set-uid-bit-off	Exec set-uid-bit-on	Setid() superuser
Real user ID	Unchanged	Unchanged	Set to UID
Effective user ID	Unchanged	Set from owner ID of the program file	Set to UID
Saved set-user ID	Copied from effective UID	Copied from effective UID	Set to UID

1.4.1.4 Process groups and job control
In addition to having a process ID, each process belongs to a process group. A *process group* is a collection of one or more processes. Each process group has a unique process group ID. The most important attribute of a process group is that it is possible to send a signal to every process in the group just by sending the signal to the process group leader. (When sending a kill, you must put a hyphen (-) before the PID of the process group leader in order to have the signal be propagated to the group.)

Each time the shell creates a process to run an application, the process is placed into a new process group. When the application spawns new processes, these are members of the same process group as the parent.

Some process identifiers are used for job control. The several types of process identifiers associated with a process are:

- PID: A process ID is a unique identifier assigned to a process while it runs. The PID is not returned to the system until the parent issues a wait(). Until the wait() is issued, a terminated process still has a PID and its status is halted. Each time you run a process, it has a different PID (it takes a long time for a PID to be reused by the system). You can use the PID to track the status of a process with the `ps` command or the `jobs` command, or to end a process with the `kill` command.
- PGID: Each process in a process group shares a process group ID (PGID), which is the same as the PID of the first process in the process group. This ID is used for signaling-related processes. If a command starts just one process, its PID and PGID are the same.
- PPID: A process that creates a new process is called a *parent process*; the new process is called a *child process*. The parent process (PPID) becomes associated with the new child process when it is created. The PPID is not used for job control.

1.4.2 Threads

If a program within an OpenEdition for VM/ESA process needs to work with more dispatchable units of work, but does not need the advanced functions of fork or spawn, it can use the POSIX threading services that are part of the OpenEdition for VM/ESA services.

The application program can create and manage threads via a range of special threading system service calls. To create a new thread, an application program uses the pthread_create() service. All threads on OpenEdition for VM/ESA are the equivalent of heavyweight threads on OS/390 UNIX System Services. However, POSIX threads are mapped to CMS multitasking threads, which are very lightweight by nature. The system overhead involved in creating threads on CMS is very low.

OpenEdition for VM/ESA resources, such as BFS files, sockets, pipes, and so forth, are available to all threads within a process.

1.4.2.1 Stopping Threads

An application can stop threads within their process by using pthread_kill() or pthread_cancel().

1.4.2.2 Porting applications with pthreads

See 3.1.4.17, “Porting with pthreads” on page 47.

1.4.3 Interprocess communication (IPC)

Three interesting functions in OpenEdition for VM/ESA come under the heading of interprocess communication:

- Shared memory
- Message queues
- Semaphores

These forms of interprocess communication extend the possibilities provided by the simpler forms of communication between processes: pipes, named pipes or FIFOs, signals, and sockets.

1.4.3.1 Shared memory

Shared memory is good for sharing large amounts of data. It saves you from moving the data multiple times, as is done for pipes, message queues, and sockets. Shared memory provides an efficient way for multiple processes to share data, such as control information that all processes require access to. The processes use semaphores to take turns getting access to the shared memory. For example, a server process can use a semaphore to lock a shared memory area, then update the area with new control information, use a semaphore to unlock the shared memory area, and then notify the sharing processes. Each client process sharing the information can then use a semaphore to lock the area, read it, and then unlock it again for access by other sharing processes.

Shared memory will persist even after all users detach from it, for example, if one process is attached to a shared memory segment and it terminates (either normally or abnormally) without detaching the segment. The segment does not go away.

If you want to use shared memory from C, you can use the C functions shmget(), shmat(), and shmctl(). For assembler, you can use the services:

- shmget() (BPX1MGT) — Create/Find a Shared Memory Segment
- shmat() (BPX1MAT) — Attach to a Shared Memory Segment
- shmctl() (BPX1MCT) — Perform Shared Memory Control Operations
- shmdt() (BPX1MDT) — Detach a Shared Memory Segment

The functions shmget() or BPX1MGT function allow you to define how big an area of shared memory you want. This can be from 4 K to 1 GB. When you do this function, it reserves space in a kernel data space. Shared memory is permanent, until explicitly freed or detached.

1.4.3.2 Message queues

XPG4 provides a set of C functions that allow processes to communicate through one or more message queues. A process can create, read from, or write to a message queue. Each message is identified with a “type” number, a length value, and data (if the length is greater than zero).

Message queues are good for handling small messages that are fed to a server. The intended design is that the message queue never get too deep.

A message can be read from a queue based on its type rather than on its order of arrival. Multiple processes can share the same queue. For example, a server process can handle messages from a number of client processes and associate a particular message type with a particular client process. Or the message type can be used to assign a priority in which a message should be dequeued and handled. If you build up deep queues and use multiple message types for categorizing, this can affect performance.

Message queues are persistent for the duration of the current IPL. You can write a message to a queue and another job or process can react to it right away or next week. Messages waiting in the queues are kept in kernel data spaces until they are received.

Messages can be very small (1 byte) or quite large (megabytes). For larger messages, consider using shared memory instead. One process can put

something in shared memory and a message on a queue can point to it. This avoids moving the data.

The C functions for using message queues are msgget(), msgrcv(), and msgsnd().

The callable services are:

- msgget (BPX1QGT) — Create or Find a Message Queue
- msgrcv (BPX1QRC) — Receive from a Message Queue
- msgsnd (BPX1QSN) — Send to a Message Queue

1.4.3.3 Semaphores

Semaphores, unlike message queues and pipes, are not used for exchanging data, but as a means of synchronizing operations among processes. Semaphores provide the ability to perform serialization on resources. A semaphore value is stored in the kernel and then set, read, and reset by sharing processes according to some defined scheme.

Typical uses for semaphores are serialization of shared memory, resource counting, and file locking. Frequently, semaphores are used to serialize hunks of shared memory.

The semget() function creates a semaphore or locates an existing one.

1.4.4 Signals

The basis for error handling in OpenEdition for VM/ESA C application programs is the generation, delivery, and handling of signals. Signals can be generated and delivered as a result of system events or application programming. You can code your application program to generate and send signals, and to handle and respond to signals delivered to it. These types of signal handling are supported for catching signals: ANSI C, POSIX.1, BSD, and additional functions provided by XPG4.

Examples of ways signal functions can be used are:

- **Maintenance:** Most UNIX systems send a signal to the process in the event of invalid pointers or other indications of a bug in the program. Depending on how the signal handling is set up, this can cause a core dump to be generated and used for debugging purposes by developers.
- **Communication Events:** When two programs are communicating with each other over a file descriptor (it could be a networking (IP) program, a pipe, or something else), if the recipient side of a conversation terminates (normally or abnormally), the sending party receives a SIGPIPE event.
- **Timer Functionality:** Either the alarm() or the settimer() functions cause the signal SIGALRM to be generated.
- **User/tty Interrupts:** Usually the interactive user can cause a signal to be generated by using certain key sequences. For example, Ctrl-C generates a SIGINT and Ctrl-Z causes the SIGTSTP signal to be sent to the process.
- **Interprocess Communication:** We do not recommend using signals for communication. The biggest disadvantage is that multiple signals can be lost.

  For general purpose communication, use shared memory, semaphores, messages queues, sockets, and pipes.

- **Process Tracking:** A parent is notified when a child process has terminated by waiting for SIGCHLD.

Each process has an action to be taken in response to each signal defined by the system. During the time between the generation of a signal and the delivery of a signal (when the actual action is performed), the signal is said to be pending. It is valid for the process to block it. If a signal that is blocked is generated for a process and the action for that signal is either the default action or to catch the signal, the signal remains pending for the process until the process either unblocks the signal or changes the action to ignore the signal.

A signal can be specified to be blocked in the sigprocmask() and sigsuspend() functions. Each process has a signal mask that defines the set of signals currently blocked from delivery and that is inherited by a child from its parent.

1.4.4.1 Supported Signals - POSIX(OFF)

For C for VM/ESA with POSIX(OFF), these signals are supported:

- **SIGABND:** System abend.
- **SIGABRT:** Abnormal termination (software only).
- **SIGFPE:** Erroneous arithmetic operation (hardware and software).
- **SIGILL:** Illegal or invalid instruction.
- **SIGINT:** Interactive attention interrupt by raise() (software only).
- **SIGIOERR:** Serious software error such as a system read or write. You can assign a signal handler to determine the file in which the error occurs or whether the condition is an abort or abend. This minimizes the time required to locate the source of a serious error.
- **SIGSEGV:** Invalid access to memory (hardware and software).
- **SIGTERM:** Termination request sent to program (software only).
- **SIGUSR1:** Reserved for user (software only).
- **SIGUSR2:** Reserved for user (software only).

1.4.4.2 Supported Signals - POSIX(ON)

For C for VM/ESA with POSIX(ON), these signals are supported:

- **SIGABND:** System abend.
- **SIGABRT:** Abnormal termination (software only).
- **SIGALRM:** Asynchronous time-out signal generated as a result of an alarm().
- **SIGCHLD:** Child process terminated or stopped.
- **SIGCONT:** Continue execution, if stopped.
- **SIGFPE:** Erroneous arithmetic operation (hardware and software).
- **SIGHUP:** Hangup, when a controlling terminal is suspended or the controlling process ended.
- **SIGILL:** Illegal or invalid instruction.
- **SIGINT:** Asynchronous Ctrl-C from the shell or a software generated signal.
- **SIGIO:** Completion of input or output.
- **SIGKILL:** An unconditional terminating signal.
- **SIGPIPE:** Write on a pipe with no one to read it.
- **SIGQUIT:** Terminal quit signal.
- **SIGSEGV:** Invalid access to memory (hardware and software).
- **SIGSTOP:** Stop executing.
- **SIGTERM:** Termination request sent to program (software only).
- **SIGTRAP:** Debugger event.
- **SIGTSTP:** Terminal stop signal.
- **SIGTTIN:** Background process attempting read.

- **SIGTTOU:** Background process attempting write.
- **SIGUSR1:** Reserved for user (software only).
- **SIGUSR2:** Reserved for user (software only).

Chapter 2. Setting up the VM environment

In this chapter we explain the setup of our VM environment to both port applications and to use them. We describe files we will have to configure in order to get the best out of the system.

At this point, you should have the following functions or products ready to run:

- C for VM (5654-033) Version 3.1.0
- Shell and Utilities (5654-030) Version 2.3.0
- Shared File System
- Language Environment run time library

We recommend that you install the last preventive services on VM, C for VM and LE run time library. The SRVAPPS files for all of the products used during the residency are contained within Appendix J, “Product service levels” on page 303.

2.1 Our VM system

The TOTVM1 system ran as a guest of the WTSCVMXA VM/ESA 2.3.0 system in a logical partition of a 9021-282. This LPAR had 128 MB of storage allocated to it.

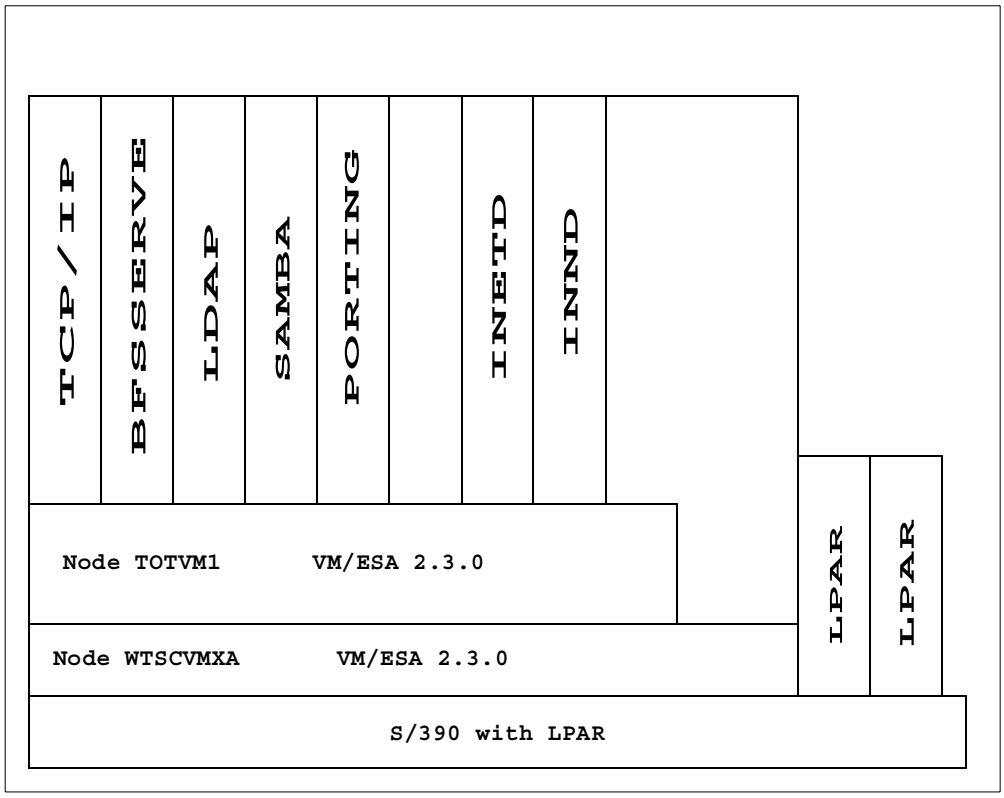


Figure 1. Our system

2.2 Adding our first users to the CP directory

To establish our environment, we made several additions to the CP directory. Initially we created two user IDs.

1. The BFSSERVE user ID is the owner of the filepool server where all objects created during the residency were stored. We used this filepool as our programmers' "sandpit"; see Figure 2.
2. The PORTING user ID is the user that we used to do our porting work. This user became our preferred user, the place where we worked during the long spring days.

2.2.1 BFSSERVE CP directory entry

This user is a Shared File System (SFS). Inside it, we installed our Byte File System (BFS). In the BFS, we created the UNIX file trees used to store our porting work. When all the ported packages were installed, we used 70000 blocks in BFS. To this, we added 70000 blocks in working space. Data disk 310 of BFSSERVE is sized at 800 cylinders.

```
USER BFSSERVE XXXXXXXX 32M 64M BG
* BFS filepool server for OpenEdition for VM/ESA porting work
IPL 190
IUCV ALLOW
IUCV *IDENT RESANY GLOBAL
MACHINE XC
OPTION MAXCONN 2000 NOMDCFS APPLMON
POSIXINFO UID 52
POSIXOPT SETIDS ALLOW
SHARE REL 1500
CONSOLE 0009 3215 T COSTA
SPOOL 000C 3505 A
SPOOL 000D 3525 A
SPOOL 000E 1403 A
LINK MAINT 0190 0190 RR
LINK MAINT 019E 019E RR
LINK MAINT 019D 019D RR
LINK MAINT 0193 0193 RR
* 191 is Run and Backup Disk
MDISK 0191 3390 485 100 VM1U1A MR
* 250 is the Control Disk
MDISK 0250 3390 585 30 VM1U1A R
MINIOPT NOMDC
* 405 is the log1 disk
MDISK 0405 3390 615 10 VM1U1A R
MINIOPT NOMDC
* 406 is the log2 disk
MDISK 0406 3390 625 10 VM1U1A R
MINIOPT NOMDC
* 260 is the catalog disk 1 MDK00001 Group=1
MDISK 0260 3390 635 50 VM1U1A R
*310 a data disk 1 MDK00002 Group=2
MDISK 0310 3390 001 800 VM1BFS R
```

Figure 2. BFS user CP directory entry - filepool for residency "sandpit"

2.2.2 PORTING CP directory entry

PORTING is the user that we used to do the package building; see Figure 3. We set the upper limit for memory to 64 MB. From time to time you need to define storage to 64 MB when there is a large compilation to perform. The PORTING user ID has no 191 minidisk; instead, it used filespace SFSTEST:PORTING as its A disk.

```
USER PORTING XXXXXXXX 32M 64M G
IPL CMS PARM AUTOCR FILEPOOL SFSTEST
IUCV ALLOW
IUCV ANY
MACHINE XC
POSIXINFO GID 0 UID 0
POSIXOPT SETIDS ALLOW QUERYDB ALLOW EXEC_SETIDS ALLOW
CONSOLE 0009 3215
SPOOL 000C 3505 A
SPOOL 000D 3525 A
SPOOL 000E 1403 A
LINK MAINT 0190 0190 RR
LINK MAINT 019E 019E RR
LINK MAINT 019D 019D RR
```

Figure 3. PORTING user CP directory entry - the programmers' workhouse

2.2.3 UID and GID explained

Users are identified by user identifications (UIDs), each of which is associated with an integer in the range of 0 to 42 949 667 295 (X'FFFFFFFF'). We will use UIDs and the integers associated with them interchangeably whenever that does not cause confusion. Users with UID=0 are given superuser privileges. A *superuser* has the special rights and privileges needed to perform administrative tasks on files, processes and users (as in CMS, when a user ID has all classes of commands). Users with UIDs less than 100 are, by convention, system users, while higher UID numbers are usually reserved for normal users.

Users are placed in groups, identified by group identifications (GIDs). Each GID is associated with an integer in the range from 0 to 42 949 667 295 (X'FFFFFFFF'). Groups with GID=0 are reserved for system groups. In most installations, other groups with GIDs under 100 are used for system functions, while groups with GIDs over 100 are used for normal groups of users.

In VM Open Edition, UID and GID are given with the CP directory statement POSIXINFO. This statement should be coded in every user entry that works with Open Edition. The POSIXINFO statement must be created *before* any device definitions, as shown:

```
POSIXINFO UID 0 GID 0
```

In order to be recognized by the system as being a given user, it is necessary to either login to the system by identifying your UID, or change from your current UID to another UID (`su` command). In order to prevent mistaken or malicious acts toward users, OpenEdition for VM/ESA provides an authentication mechanism whereby users must authenticate their identity by providing a password to the system. This is done during VM logon. Security and integrity is assumed by VM CP.

Once a user is properly identified and authenticated, CP grants the user access to all authorized information.

Each user is associated with one or more groups. When a user is identified to the system, a default GID is provided and the system grants access to all authorized information for that group. A user can change groups using the `newgrp` command. OpenEdition for VM/ESA then checks its files to determine whether or not the user is an authorized member of that group, and grants all authorized access if appropriate. Groups are described in the CP directory with POSIXGROUP statements. Ensure the entries shown in Figure 4 exist in the CP directory. They must be placed *before* any user statements and common profiles. These statements are case-sensitive.

```
    GLOBALDEFS
    POSIXGROUP  system  0
    POSIXGROUP  staff 1
    POSIXGROUP  bin 2
    POSIXGROUP  sys 3
    POSIXGROUP  adm 4
    POSIXGROUP  mail  6
    POSIXGROUP  security  7
    POSIXGROUP  nobody 4294967294
1   POSIXGROUP  news 5
2   POSIXGROUP  cron 8
3   POSIXGROUP  httpd 9
```

Figure 4. POSIXGROUP statements required in CP directory

Figure 4 notes:

1. This POSIXGROUP statement will be used for the Internet News Server (See 7.1, “INND - USENET” on page 147).
2. The cron POSIXGROUP will be used by the CROND package (See 5.5, “CROND” on page 97).
3. The httpd POSIXGROUP will be used by the Apache package (See 6.3, “Apache” on page 136).

2.3 Creating the BFS server

After putting the CP directory online, we describe in this section how to build the Shared File System and to install our Byte File System (BFSTEST) in it. The process to create a filepool is also described in *VM/ESA CMS File Pool Planning, Administration, and Operation*, SC24-5751.

2.3.1 Formatting BFSSERVE 191 minidisk

The BFSSERVE install process begins with the following steps:

1. `LOGON` to the user BFSSERVE and `FORMAT` the 191minidisk.
2. `ACCESS MAINT 193` minidisk as C.
3. `CP SET EMSG ON`

2.3.2 Creating the PROFILE EXEC

We use XEDIT to create the PROFILE EXEC for the BFSSERVE user ID; see Figure 5.

```
/* */
'ACCESS 193 B'
'SET PF10 RET'
'SET EMSG ON'
'SET AUTOREAD OFF'
'SET RUN ON'
'EXEC FILESERV START'
```

Figure 5. PROFILE EXEC for BFSSERVE

2.3.3 Defining BFSTEST DMSPARMS

For the BFSSERVE user ID, we used XEDIT to create BFSTEST DMSPARMS file; see Figure 6.

```
ADMIN PORTING EWEB001 2VMVMZ30
ADMIN EWEB002 EWEB003 EWEBLOG EWEBADM RMAINT
NOAUDIT
BACKUP
SAVESEGID CMSFILES
FILEPOOLID BFSTEST
NOCRR
NOLUNAME
USERS 50
REMOTE
```

Figure 6. BFSTEST DMSPARMS file used to define the BFS filepool.

2.3.4 Defining BFS POOLDEF

In this file, we defined minidisks used by the filepool server:

```
MAXUSERS=1000
MAXDISKS=500
DDNAME=CONTROL          VDEV=250
DDNAME=LOG1             VDEV=405
DDNAME=LOG2             VDEV=406
DDNAME=BACKUP   DISK    FN=CONTROL     FT=BACKUP    FM=A
DDNAME=MDK00001         VDEV=260       GROUP=1  BLOCKS=8983
DDNAME=MDK00002         VDEV=310       GROUP=2  BLOCKS=300098
```

Figure 7. POOLDEF statements for BFS filepool

The DDNAME=BACKUP statement is used to specify where the backup file will be. In our case, it is on the 191 minidisk. This file will be used to back up the control data of the BFS filepool server.

Note: This backup does not save your data loaded in BFS.

To back up your data, you have to use FILEPOOL BACKUP or FILEPOOL UNLOAD commands.

2.3.5 Building the BFSTEST server

During this step, minidisks are formatted and the Shared File System structure is built. Execute the following commands:

1. `FILESERV GENERATE`
2. `FILESERV BACKUP`
3. `FILESERV START`

FILESERV GENERATE formats the minidisk used by SFS and creates the directory, the control data, and the log spaces.

FILESERV BACKUP backs up the control data and clears the filepool log of the Shared File System server.

FILESERV START starts the server in multiple user mode.

If you get the message `DMS5BB3045I Ready for operator communications`, it means the BFS filepool is ready to be used. Enter `#CP DISC` to close the console session on your terminal.

2.3.6 Enrolling users to the BFSTEST filepool

Next, we logged on to the PORTING user ID and enrolled all users that needed access to the filepool BFSTEST. The PORTING user ID is an SFS (BFS) administrator ID (refer to 2.3.3, “Defining BFSTEST DMSPARMS” on page 17).

```
ENROLL USER PORTING BFSTEST: (BFS
```

2.3.7 Updating execs to load the objects

Stay in the PORTING user ID and do the following:

1. Access `2VMVMZ30 49E` minidisk (or, if Shell and Utility are installed in SFS, then access `VMSYS:2VMVMZ30.SHELL.BUILDTST`).
2. Copy the `SHELL LOADBFS` to your A disk.
3. Access the `MAINT 193` minidisk.
4. Copy `BFS LOADBFS` to your A disk.

   Files with a filetype of `LOADBFS` are used by the `LOADBFS EXEC` to build the directory tree and to load files from the CMS environment to the Byte File System.

   The BFS LOADBFS file is used to create the CMS Byte File System tree and load CMS base objects.

   The SHELL LOADBFS file is used to create the Shell and Utilities tree and load its objects.
5. Run this command: `XEDIT BFS LOADBFS`
6. Change all occurrence of filepool `VMSYS:` to `BFSTEST:`
7. Run this command: `XEDIT SHELL LOADBFS`
8. Change all occurrences of filepool `VMSYS:` to `BFSTEST:`

2.3.8 Loading the BFS

Start the filepool load with all the objects defined in `BFS LOADBFS` and `SHELL LOADBFS`.

1. Enter the following:

```
LOADBFS BFS
LOADBFS SHELL
```

2. Check that you have no errors in `BFS LBFSLOG` and `SHELL LBFSLOG`.

In case of error during LOADBFS. skip to 2.4.1, “Create PROFILE EXEC for the PORTING user ID” on page 19 and 2.4.3, “Starting the SHELL” on page 21 for more information, then return here to do the following:

1. Enter `cd /` (to switch you to the root of the BFS directory).
2. Enter `rm -fR *` (to remove all the files and subdirectories created during the LOADBFS command).
3. Increase the size of your filepool or correct the errors.
4. Restart loading of the filepool with `LOADBFS BFS` and `LOADBFS SHELL`.

2.3.9 Creating a second playground

In our testing, we created a second “sandpit” so that multiple testers could execute the porting work without interfering with each other. We called the second sandpit BFS2. We followed the same procedure described in 2.3, “Creating the BFS server” on page 16.

2.4 Establishing the OpenEdition for VM/ESA environment

Once the BFSTEST filepool has been prepared, you need to customize some OpenEdition for VM/ESA files.

2.4.1 Create PROFILE EXEC for the PORTING user ID

PROFILE EXEC will be used to set all the required environment variables.

```
   /* Profile EXEC */
1  'SEGMENT LOAD CEEEV003 (SYSTEM'
2  'SEGMENT LOAD VMLIB'
   'RTNLOAD * (FROM VMLIB SYSTEM GROUP VMLIB)'
   'SET RDYMSG SMSG'
   'SYNONYM' USERID()
   'SET INSTSEG ON E'
   'CP TERM BREAKIN GUESTCTL'
   'CP SET MSG ON'
   'CP SET EMSG TEXT'
   'CP SET IMSG ON'
3  'OPENVM MOUNT /../VMBFS:BFSTEST:ROOT/ /'
   'GLOBAL MACLIB DMSGPI HCPGPI MVSXA'
4  'GLOBAL TXTLIB SCEELKED SCEEOBJ SQLLIB CMSLIB'
5  'GLOBAL LOADLIB SCEERUN'
   exit Rc
```

Figure 8. PROFILE EXEC for user PORTING

Figure 8 notes:

1. This logical segment is loaded to ensure that at IPL time the storage is available for LE to use (if something were to use this storage before LE/370 could load it, there would be severe performance penalties).
2. VMLIB contains many highly used routines. We load these routines using the RTNLOAD command.
3. The root directory of the working filepool is mounted. This statement gives a default access to our filepool for CMS session.
4. SCEELKED and SCEEOBJ are automatically used by the c89 EXEC, but if you were to invoke the prelinker and loader directly, you would need these.
5. SCEERUN contains the non-shared version of all the LE/370 routines.

2.4.2 3270 considerations

UNIX evolved as a character-oriented operating system, while S/390 operating systems have their roots in the record-oriented world. It is amusing to read a UNIX bigot stating that S/390 languages were record-oriented because of their punch card legacy while they have the far superior byte mechanism.

In a large enterprise, it would be suicide for a system to handle interrupts generated by every key stroke; that is why 3270s have enter keys. This requires some adjustments to the way a UNIX application may work under OpenEdition. Similarly, the character set used by a UNIX programmer is slightly different than that used on S/390 (and we do not mean just ASCII/EBCDIC).

2.4.2.1 Escape Sequences

On a UNIX workstation the Ctrl key is used extensively to send control sequences to the system. This will not work with a 3270 terminal, which has no control key nor any way of generating the sequence a control key produces. To support applications which rely on such sequences, special 3270 character sequences are used.

By default, this appears as: <esc-char><signal><enter>

where:

```
<esc-char> = ¢
<signal>   = character such as 'C', 'V', 'D', or 'Z'
<enter>    = enter key
```

Thus, to signal Ctrl-C, you would enter: `¢C<enter>`

The meanings of some of the escape sequences are:

- kill process: `<CTRL-C> = ¢C`
- end of input: `<CTRL-D> = ¢D`
- terminate foreground job: `<CTRL-V> = ¢V`
- suspend foreground job: `<CTRL-Z> = ¢Z`

2.4.2.2 VM Terminal Characters

Some characters, with either special meaning to UNIX applications or favored by them, tend to clash with the S/390 system. For example, under VM/ESA, the default line delete character is the cent (¢) sign. Similarly, the open quotes (“) sign and the at (@) sign both have special meaning. To avoid this clash, either change

to always use different characters, or change to different characters during the course of your OpenEdition for VM/ESA work.

In our case, we adopted the second course and used a SHELL EXEC (see Figure 9 on page 22) to override the VM defaults each time anyone entered the shell.

2.4.2.3 Code pages

Typically, those who use the US code page with a 3270 keyboard do not have the open bracket ([) character or the close bracket (]) character available. The PC keyboard utility, 3174 utility, or 3472 setup will allow you to make these characters available by remapping the keyboard. Depending on the code page, however, they may be X'BA' and X'BB' instead of X'AD' and X'BD'. Under CMS, this is easy to fix with SET OUTPUT and SET INPUT commands.

Today many countries have different code page and keyboard arrangements and to work with VM Open Edition, you may have to do some checking and modification.

The OpenEdition for VM/ESA environment uses the EBCDIC code page (IBM-1047). This code page is described in *OpenEdition for VM/ESA User's Guide Version 2 Release 1.0*, SC24-5727.

The first point to determine is are you able to type and display all the characters you need? If some of your keys do not display the desired characters, depending on your 3270 emulation program, you may be able to change the key assignment. Appendix D, "Keyboard setting for Enetwork Personal Communication" on page 213 shows an example of how to perform this key assignment.

The next point to determine is does your XEDIT session interpret the correct key and display correctly all the EBCDIC codes?

In our case, the answer to this second question was no, so we had to correct the shortcomings detected. To do so we used two commands: `SET OUTPUT` and `SET INPUT`. These commands cause CMS to interpret an EBCDIC value received from the keyboard or to be sent to the display as another EBCDIC value.

To verify your XEDIT environment, follow Appendix D.2, "SCREEN EXEC" on page 215.

Note: The `SET OUTPUT` or `SET INPUT` command stays active until you next IPL CMS or issue these commands without operands.

2.4.3 Starting the SHELL

The OpenEdition for VM/ESA environment is started with the command `OPENVM SHELL`. SHELL is the UNIX command environment and gives access to the commands defined by the Shell and Utilities. All the commands available are described in *OpenEdition for VM/ESA Command Reference Version 2 Release 3.0*, SC24-5728.

To simplify the startup of OpenEdition for VM/ESA, we created a `SHELL EXEC` with all the required commands in it as shown in Figure 9 on page 22.

```
   /* Shell EXEC */
1  'SET OUTPUT AD' 'BA'X
   'SET OUTPUT BD' 'BB'X
2  'SET INPUT BA AD'
   'SET INPUT BB BD'
3  'CP TERM ESCAPE OFF'
   'CP TERM LINEND OFF'
   'CP TERM LINEDEL OFF'
4  'PIPE CMS OPENVM UNMOUNT /'
5  'VMLINK CVM'
6  'OPENVM MOUNT /../VMBFS:BFSTEST:ROOT/ /'
7  'OPENVM SHELL'
   /* reset to default value */
8  'SET INPUT BA BA'
   'SET INPUT BB BB'
   'SET OUTPUT AD' 'AD'X
   'SET OUTPUT BD' 'BD'X
   'PIPE CMS OPENVM UNMOUNT /'
   exit
```

Figure 9. The SHELL EXEC used to invoke the OpenEdition for VM/ESA Shell

Figure 9 notes:

1. The hex character X'AD' is displayed as X'BA' ('[') and for the next line, the hex character X'BD' is displayed as X'BB' (']').
2. The '[' is interpreted as X'AD' and for the next line, the ']' character is interpreted as X'BD'.
3. TERMINAL LINEND, ESCAPE, LINEDEL are set OFF, as the default characters have special meaning to the shell.
4. Just in case we had inadvertently mounted another file system, we issue an unmount command. To suppress the error message that is received when nothing is mounted, we included `OPENVM UNMOUNT` in a `PIPE` command.
5. Access the C compiler filespace with the VMLINK facility:

   The USERPROD NAMES file was created and placed on the Y disk containing the following nicknames:

```
   :nick.TCPIP     :title.TCP/IP for VM
1                  :product.TCPMAINT 592 RR <+120 Z>

   :nick.CVM       :title.C/VM 3.1
                   :product..DIR SFSTEST:CVM.
```

   Figure 10. USERPROD NAMES file used during testing

   Figure 10 note:

   1. The TCPIP product disk must be accessed *after* the disks containing the LE/370 C header files (for example, STDIO H), as there are header files on this disk that share the namespace of the LE/370 files and will lead to compilation errors if searched first (for example, SOCKET H).
6. Mount the ROOT filespace from the BFSTEST filepool as the root directory.
7. Enter the shell.

8. When we leave the shell, the next statements reset the mappings defined in steps 2 to 7.

Now by entering `SHELL`, the OpenEdition for VM/ESA environment is invoked with our customized changes.

2.4.3.1 Dealing with the EMSG Setting

The default action of the C runtime library routines is to turn off the EMSG setting. This means that any messages with this attribute will not be displayed on the console. If you need to reIPL CMS in the middle of, for example, a compilation or linkedit, the EMSG setting will not be reset.

This behavior may be modified by setting a global variable:

```
GLOBALV SELECT CENV SETL _KEEP_EMSG=Y
```

2.4.4 Creating our first tool

In VM Open Edition, there is only one editor called `sed`. This editor is unusable for normal editing functions. With CMS, we are used to a great editor called XEDIT. XEDIT is able to access and update files residing in the Byte File System.

To use XEDIT while in the shell environment may be a little cumbersome:

```
cms "xedit 'some/file.ext' (name bfs"
```

This can be rather inconvenient given the number of times you will want to edit files. To call XEDIT, we wrote our first shell script program. The shell script is an interpretive language used to write UNIX command procedures (it is similar to REXX for CMS and to `.bat` files in Windows).

The purpose of our script program is to call XEDIT from our OpenEdition for VM/ESA session.

1. On the command line type: `cms 'x /bin/x (name bfs'`

   Remember in UNIX or OpenEdition for VM/ESA environment, commands are case-sensitive. There is no automatic translation to uppercase as in CMS.

2. In the file just opened, insert a line:

   ```
   cms "X '$1' (name bfs"
   ```

3. Save this file.

4. To be able to use the new `x` command, we gave execute rights on the x object to every user ID authorized to access our BFS directory using the `chmod` command.

   ```
   chmod 755 /bin/x
   ```

To test your first program, type `x a.file` and you should have an XEDIT session for a file called a.file.

2.4.5 Customizing /etc/profile

The UNIX world uses different profile levels to initialize the user environment during logon of a new user.

In OpenEdition for VM/ESA, the profiles are read in the following order:

1. /etc/profile

This is the main profile used by all users connected to this Byte File System (BFS). It sort of corresponds to the SYSPROF EXEC used at CMS IPL time.

2. /u/<userid>/.profile in your home directory (where <userid> is equivalent to your CMS userid)

   .profile is your private profile. You can customize it to your needs just like PROFILE EXEC.

The Shell and Utilities delivers a sample profile in /etc/profile.sample which you can customize; see Figure 11.

1. Copy profile.sample:

   ```
   cp /etc/profile.sample /etc/profile
   ```

2. Use XEDIT to modify it:

   ```
   x /etc/profile
   ```

```
   # sample /etc/profile
   # export TZ=EST5EDT
   # export LC_ALL='En_US'
   cms 'VMFCLEAR'
   date '+%A %B %Od %r %Z %Y'
   # The MAILMSG will be printed by the shell every MAILCHECK seconds
   # (default 600) if there is mail in the MAIL system mailbox.
   export MAIL=/usr/mail/$LOGNAME
   export MAILMSG="You have new mail."
1  if [ $HOME = "/" ]; then
      export HOME=/u/$LOGNAME
   fi
2  PATH=$HOME:$HOME/bin:$HOME/usr/local/lib:/usr/local/etc
   :./:/usr/local/bin:/usr/bin:/etc:/bin
3  cd $HOME
4  PS1='[$LOGNAME - $PWD]>'
   PS2="> "
5  alias help="cms help oshell"
   alias man="cms help oshell"
   alias dir="ls -laA"
   alias mkdirs="mkdir -p"
   if [ -s "$MAIL" ]           # This is at Shell startup.  In normal
   then echo "$MAILMSG"        # operation, the Shell checks periodically
   fi
6  CLASSPATH=/usr/mqm:/usr/mqm/com/ibm/mq:/usr/mqm/com/ibm/mqservices:/usr
   /java/classes:/usr/java/lib/classes.zip:/usr/NetRexx/lib
   /NetRexxC.zip:/usr/NetRexx/lib/NetRexxR.zip:./:
   /usr/java/classes:/usr/java/lib/classes.zip
   export CLASSPATH
7  export PATH=$PATH:/usr/NetRexx/bin:/usr/java/bin
```

Figure 11. /etc/profile used during testing

Figure 11 notes:

1. HOME is a UNIX variable that specifies the default position in the directory root. If the user does not have an initial working directory specified in their CP directory entry, we will set it to /u/$LOGNAME, where $LOGNAME is the VM user ID in lowercase.
2. PATH defines where the shell will search for an executable object. The search order is from left to right. We have put our working directory first so

that the shell will search it first. Using our working directory as the top of the search order is acceptable in a test environment, but not for a production system.

3. With this command we go to our home directory: `cd $HOME`.
4. PS1 identifies the shell prompt. Our setting is "[userid - working directory]>".
5. Use to set an alias of a command `alias`.
   - `help` results in HELP OSHELL <operands>.
   - `man` (the UNIX "man" page processor) is aliased to cms HELP.
   - `dir` is the third alias set and it uses the `ls` command to generate a DOS-like presentation of files in directories.
   - `mkdirs` is an alias needed to install the DCRON package correctly.
6. `CLASSPATH` is used by Java and NetRexx to define the search order for their objects.
7. `export` marks each variable name so that the current shell makes it automatically available to the environment of all commands run from that shell.

2.4.6 Some useful Shell commands and utilities

In order to become productive within the Shell, it is necessary to become familiar with several of the commands that you will require. This section is not exhaustive as it is simply an introduction to a few of the most commonly used commands. Further information may be obtained from *OpenEdition for VM/ESA Command Reference Version 2 Release 3.0*, SC24-5728.

All the commands used within the Shell are case sensitive; this is a major difference for the VM user.

alias

Use `alias` to tailor commands: for example, if you do not like using the command `rm` , you can use `alias` to allow you to refer to it as erase:

```
alias erase=rm
```

Similarly, if you find it useful to invoke `grep` with the `-i` option, then you can use an `alias` to invoke `grep` with the `-i` option automatically included:

```
alias grep="grep -i"
```

ar

This command is for maintaining archive libraries. The archive library is a collection of files, usually object files. Using `ar`, you can perform various operations on archive libraries, such as creating a new library, adding members to an existing library, deleting members from a library, extracting members from a library, printing a table of contents for a library, and so on.

This command is useful for collecting common object files for subsequent use by other applications. Archive libraries are used extensively by all our packages.

Example:

```
ar -t libgdbm.a
__.SYMDEF
dbmopen.o
dbmdelete.o
dbmfetch.o
dbmstore.o
dbmseq.o
dbmclose.o
dbmdirfno.o
dbmpagfno.o
dbmrdonly.o
gdbmopen.o
gdbmdelete.o
gdbmfetch.o
gdbmstore.o
```

Figure 12. Using the ar command to display the contents of library

awk

You may encounter this command during your porting endeavors. `awk` is a file processing language that is well suited to data manipulation and retrieval of information from text files.

An awk program consists of any number of user-defined functions and rules of the form:

```
pattern {action}
```

chmod

Use this command to change access permission to a file or directory. To use this command you must be a super user, the owner of the file or directory, or have appropriate privileges.

This command is useful after porting an application and assigning authorities when releasing to the enterprise.

In this example we grant execute privileges on a file:

```
chmod +x /home/apache/testcmd.sh
```

chown

This command sets the user ID (UID) to owner for the files and directories named by pathname arguments. The owner can be a user name from the user profile, or it can be a numeric user ID.

If you include a group name, that is, if you specify owner followed immediately by a colon (:) and then group with no intervening spaces, such as owner:group, `chown` also sets the group ID to group for the files and directories named.

In this example we change the ownership of the directory and all files within to the user apache in the group httpd:

```
chown -R apache:httpd /usr/local/apache
```

cms
Pass the command to CMS for processing:

```
cms q accessed a
Mode  Stat     Files  Vdev  Label/Directory
A      R/W       183  DIR   SFSTEST:NEALE.
```

cp
Copy a file.

c89
Compiles, prelinks, and builds IBM C for VM/ESA programs. First, `c89` performs the compilation phase (including preprocessing) by compiling all operands of the file.c form. The result of each compile step is a file.o file. If all compilations are successful, or if only file.o and no file.c files are specified, `c89` proceeds to the module build phase, which consists of a prelink step and CMS module generation step.

In the module build phase, `c89` combines all file.o files from the compilation phase along with any file.o operands that were specified. Any file.a and -l libname operands that were specified are also used.

diff
This command attempts to determine the minimal set of changes needed to convert a file whose name is specified by the first argument into the file specified by the second argument.

By default (“normal diff”), output consists of descriptions of the changes in a style like that of the ed text editor. A line indicating the type of change is given. The three types are a (append), d (delete), and c (change).

The context output (“context diff”) format starts with a two-line header, which looks like this:

```
*** from-file from-file-modification-time
--- to-file to-file-modification time
```

Next come one or more segments of differences; each segment shows one area where the files differ. Context format segments look like this:

```
***************
*** from-file-line-range ****
  from-file-line
  from-file-line...
--- to-file-line-range ----
  to-file-line
  to-file-line...
```

The lines of context around the lines that differ start with two space characters. The lines that differ between the two files start with one of the following indicator characters, followed by a space character:

- '!' A line that is part of a group of one or more lines that changed between the two files. There is a corresponding group of lines marked with '!' in the part of this segment for the other file.

- '+' An “inserted” line in the second file that corresponds to nothing in the first file.
- '-' A “deleted” line in the first file that corresponds to nothing in the second file.

If all of the changes in a segment are insertions, the lines of from-file are omitted. If all of the changes are deletions, the lines of to-file are omitted. See A.2.6, “Update builtin.c” on page 185 for an example of a diff file.

find
Searches a given file hierarchy specified by path, finding files that match the criteria given by expression. Each directory, file, or special file encountered in the hierarchy is “passed through” the command arguments, and if a match is found, then an action defined by the command arguments occurs.

grep
Searches files for one or more pattern arguments. It does plain string, basic regular expression, and extended regular expression searching.

`grep` is very useful when undertaking porting as it allows you to search for things like entry points in source files you may be having problems with. When teamed with the piping facilities of OpenEdition for VM/ESA, you can get `grep` to perform some large and complex searches for you and have action taken on the files found. Combined with `find`, it is a powerful tool for searching through entire directory structures, as in the following example, which scans all C source files in the current directory and all of its subdirectories for occurrences of the string ‘fork’:

```
find ./ -name "*.c" | xargs grep -i "fork"
```

ls
Lists files and directories. If the path name is a file, `ls` displays information on the file according to the requested options. If it is a directory, `ls` displays information on the files and subdirectories therein. `ls` is analogous to the CMS listfile command.

make
`make` is a very useful command but we will be replacing it with gnu `make`. `make` helps you manage projects containing a set of interdependent files, such as a program with many source and object files, or a document built from source files, macro files, and so on. `make` keeps all such files up to date with one another. If one file changes, `make` updates all the other files that depend on the changed file.

mv
Move a file. This effectively renames a file.

pax
Of all the commands you will need to deal with, `pax`, `make`, and `c89` are the ones you will probably deal with more than others. `pax` reads and writes archive files. An archive file concatenates the contents of files and directories, and can also record such information as file modification dates, owner names, and so on. You can therefore use a single archive file to transfer a directory structure from one machine to another, or to back up or restore groups of files and directories.

`pax` understands cpio, tar, and ustar archive formats (compressed and uncompressed). `pax` will also perform character set conversion; for example, from

ASCII to EBCDIC (usually ISO8859 to IBM-1047). This is a handy utility. Be careful, though, if your archive contains a mixture of ASCII and binary data.

rm
Remove a file or directory structure.

sed
The `sed` command applies a set of editing subcommands contained in a script to each argument input file. If you did not specify any input file, `sed` reads the standard input.
`sed` is very useful in pipelines and is used extensively in most configuration scripts accompanying packages.

Here is an example of what `sed` is capable of:

```
find ./ -name "*.c,v" | sed 's/,v//g' | xargs grep "PATH"
```

This will look for all files in the current and subsequent directories with an extension of c,v. `sed` then strips the ,v of the results of the `find` command. `xargs` then uses the results of `sed` and builds a `grep` command which searches for occurrences of the word PATH in the C source files.

&
Use `&` at the end of a command string to cause the command to run in a background process.

```
 find ./ -name "*.c" -type f &
[1]     1609
[neale - /home/neale/XSB/emu]>
./auxlry.c
./util.c
./xsb_error.c
```

Chapter 3. Porting application guide

In this section we examine some generic issues relating to the selection and use of ported applications.

For those already familiar with the UNIX world, the GNU initiative will not be new to you. For those who are not, we provide an explanation of what GNU is and what it means to you.

The most common porting issues are described in detail to help you understand what needs to be done if you decide to port an application from another UNIX platform.

OpenEdition for VM/ESA is not a full XPG4 or UNIX branded system. For most applications this is not a problem as not many systems are (although OS/390 is). This has implications to the porter, and so is explained in this chapter.

3.1 S/390 and VM/ESA porting issues

Anyone who has ever ported an application from one flavor of UNIX to another would be aware that not all UNIX systems were created equal. Sometimes the differences are small (for example going from version to version) or quite large (for example moving from SystemV to QNX). Usually, the issues involve the lack of APIs, the use of proprietary APIs, or different file system conventions.

The same is true for porting applications from another flavor of UNIX to VM/ESA. However, there are two other issues that differentiate ports to this platform from anywhere else.

3.1.1 ASCII versus EBCDIC

This is an issue that has to be addressed by anyone porting applications to S/390, so a lot of the traps have been exposed and the legwork has been done for us. The difficulty with character set issues is that the potential problem areas can be difficult to find.

Some sage words of advice come from the *OS/390 UNIX System Services Porting Guide* http://www.s390.ibm.com/oe/bpxa1por.html on

`http://www.s390.ibm.com/oe/bpxa1por.html`

“Programmers who are porting need to keep a clear head as to where they are and who is responsible for translation. When debugging, it is always worth asking yourself if the problem could be an ASCII/EBCDIC issue first.”

3.1.1.1 Contiguity and collating sequence

The most common source of problems is the assumption about the way characters are encoded: that is, in ASCII the characters “a” through “z” are sequential and “A” through “Z” come *before* “a” through “z”. Of course, this is not true in EBCDIC.

Thus, the following code fragment in applications requires attention:

```
if ((c >= 'A') & ((c <= 'Z')) { .... /* Is c an alphabetic character? */
   c = 'a' + (c - 'A');     /* Convert c to lowercase */
```

Figure 13. ASCII character set issues

In this example the cure is quite straightforward:

```
1  #if !define(__OPEN_VM)
   if ((c >= 'A') && ((c <= 'Z'))      /* Is c an uppercase character? */
      c = 'a' + (c - 'A');     /* Convert c to lowercase */
   #else
2  if (isupper(c))
3     c = tolower(c);
   #endif
```

Figure 14. Equivalent EBCDIC code

Figure 14 notes:

1. We used conditional compilation in this case. The #if statement translates to: “if the symbol __OPEN_VM has not been defined (that is, if we are not on a VM platform)”.

   Why not replace the lines altogether? The answer is that some UNIX platforms may not have the islower() API, and if you want your changes to be incorporated in the application by the package owner, you only add your statements and do not remove the work of others. There are circumstances where such removal is warranted, such as when you have the permission of the owner or there is a fundamental logic flaw that needs fixing.
2. If the character is uppercase.
3. Convert it to lower case.

3.1.1.2 Hard-coded ASCII in C code and shell scripts

The preceding change was quite simple. However, more complex problems can be introduced by programmers writing code in peculiar ways. For example, the use of hex or octal values instead of characters is not common but it does happen. The following example shows how problems can occur:

```
switch(c)
{
   case 0x30 :
      i *= 10;
      break;
   case 0x41 :
      i *= 100;
      break;
   case 0x61 :
      i *= 1000;
      break;
   case 0x0a :
      printf("%d\n",i);
   default :
      break;
}
```

Figure 15. ASCII considerations: The use of numeric values

If you were porting this fragment of code you would have to make a decision: Is the program really expecting to see "X'30'" or the character 0? If the answer is the former then the code can remain as is. However, if the latter is true you would end up with something like the following:

```
#ifndef __OPEN_VM
# define CR 0x0a
#else
# define CR 0x15
#endif
switch(c)
{
   case '0' :
      i *= 10;
      break;
   case 'A' :
      i *= 100;
      break;
   case 'a' :
      i *= 1000;
      break;
   case CR :
      printf("%d\n",i);
   default :
      break;
}
```

Figure 16. EBCDIC equivalent using character constants

Note that the lines have been changed, not just added to. In this instance you are justified in doing so because you are clarifying the intent of the code and not introducing any potentially unsupported APIs.

One of the most common ASCII dependencies we have come across in shell scripts is the use of the octal constant '\012'. This can easily be converted to '\n' or a C preprocessor variable used. Similarly, the use of '\015' is quite prevalent; but, fortunately, the EBCDIC equivalent is also '\015' or '\r'.

3.1.1.3 Using code generated by lexx or yacc

Often, packages contain C code generated by the `lexx` or `yacc` utilities. This code will probably contain ASCII dependencies and will not work. The code needs to be generated on VM/ESA by rerunning the utilities. Note that this may introduce EBCDIC dependencies, making the code less portable to other systems, but at least it will work on S/390.

For example, y.tab.c is typically generated by `yacc` and there should be commands in the package's makefile instructing make how to invoke `yacc` to rebuild y.tab.c. There should also be a comment in y.tab.c that specifies the source file that `yacc` processed to generate y.tab.c.

3.1.1.4 Applications that talk to arbitrary remote systems via sockets

These applications typically have to assume all text they receive is ASCII and they send out all text as ASCII. They have to convert the data locally as they go along.

Consider a file transfer program (ftp) client: a user will type a command such as:

```
dir foobar
```

The ftp server does not want to see these characters in EBCDIC, so the client must convert the data to ASCII before they are written to the socket. Likewise, if you simply write the data coming from the server to the user's screen, it will be meaningless because it will be in ASCII. The client must first convert the data to EBCDIC. This is true even if the server is running on an EBCDIC system, such as OS/390 or VM/ESA.

However, you must be careful to convert only text data. Some applications may mix binary and text in a data stream. For instance, the server might send 2 bytes of binary data preceding a block of text to represent the number of bytes in the block, or a checksum value.

3.1.1.5 Code that relies on byte order of data may not be portable

PC systems are *little endian*, that is, the byte in a numeric field with the lowest address is the least significant. However, S/390 and most UNIX systems are *big endian*. This typically affects integer and floating point data. If an application is responsible for transferring such data between platforms, you need to do one of the following:

- Write data exchange logic.
- Translate data to text, transfer as text, and then recreate as binary.

3.1.1.6 Identifying potential code changes

Now that you know what to do when you encounter an ASCII dependency, how can you identify such a dependency? The following commands will help you do so:

```
find ./ -name "*.[ch]" | xargs grep -i "'a'" >ascii.mods
         1                2                3
```

Figure 17. Identifying ASCII dependencies: Example 1

Figure 17 notes:

1. Find all the files in this and subsequent directories that have an extension of 'h' or 'c' (header and source files).
2. Examine these files for occurrences of 'a' or 'A'. This will identify statements as in Figure 13 on page 32.
3. Pipe the results to file ascii.mods.

```
find ./ -name "*.sh" | xargs grep "012'"
         1               2
```

Figure 18. Identifying ASCII dependencies: Example 2

Figure 18 notes:

1. Find all the files in this and subsequent directories that have an extension of sh (shell script files).

2. Examine these files for occurrences of ”012’”. This will help locate instances of ‘\012’.

```
find ./ -name "*.c" | xargs grep -i "0xdf"
        1                  2
```

Figure 19. Identifying ASCII dependencies: Example 3

Figure 19 notes:

1. Find all the files in this and subsequent directories that have an extension of c (c source files).
2. Examine these files for occurrences of “0xdf’”. This value is used in an ASCII-specific technique of converting characters to uppercase.

3.1.1.7 Commands and functions that handle conversion

There are shell commands, CMS commands, and C functions that handle ASCII to EBCDIC conversion.

Here are two shell commands that are useful:

- `iconv`

  For example, the command:

  ```
  iconv -f IBM-1047 -t IS08859-1 words.txt >converted.txt
  ```

  converts the file words.txt from the IBM-1047 standard code set to the ISO 8859-1 standard code set and stores it in the file named converted.txt.

- `pax`

  For example, the command:

  ```
  pax -wf testpgm.pax -o from=IBM-1047,to=ISO8859-1 /tmp/posix/testpgm
  ```

  backs up the /tmp/posix/testpgm directory, which is in the character set IBM-1047, into an archive file that is targeted to an ASCII character set.

  The CMS PIPE command stage XLATE lets you convert files between multiple codepages.

The C functions __atoe(), __atoe_l(), __etoa(), and __etoa_l() also perform ASCII-EBCDIC conversion.

3.1.2 Forking and spawning

For VM/ESA users, the major consideration in conversion efforts is that VM does not support `fork`(). There are compelling reasons why the developers did not implement it (at least in the first incarnation of OpenEdition for VM). Consider what `fork`() actually does and how applications use it (see Section 1.4.1, “Processes” on page 5). `fork`() clones another process which has the same programs, data, and files as the original. This means an entirely new address space has to be created. VM developers observed that most applications which used `fork`() did so in order to execute another program. That is, the programmers were prepared to suffer the overhead of cloning a process just so they could then clear the space and run another program. This is overkill. Therefore, the VM developers implemented `spawn`() which combines the `fork`() and `exec`() paradigm in a single call: use the spawn service to create a child process to run the specified

executable file. This service combines the semantics of the fork and exec services.

The other use of fork() is for reasons of multitasking. Often it would be more efficient if the application used the POSIX threads support instead. In instances where a fork() cannot be avoided there are techniques involving spawn() you can use to simulate it.

The spawn() function creates a new process, loads a specified program into the new process, and begins execution. This creates a process that is nearly an exact duplicate of the process created when fork() and exec() are used together.

In the fork() plus exec() scenario, fork() creates a new child process that is nearly an exact duplicate of the original parent process. Both the parent and the child process continue execution at the return from fork(). The child process calls exec() to replace the program inherited from the parent and begins execution of the new process image file (program).

A point that must be understood in this scenario is that there may be additional processing performed by the child process before the exec() is done, or there may be no exec() at all. After the fork(), an exact copy of the parent process exists in the child process. All memory maps, variables, pointers, and so on, exist in the child just as they did in the parent. If you examine the buffers in the child, you can see exactly what was in the parent before the fork(). This allows the child to refer to these as if nothing has happened. The child can do one of the following:

- Continue processing the program inherited from the parent
- Do some additional processing, possibly in preparation for the exec()
- Immediately do an exec()

In contrast, when spawn() is executed there is no intermediate stage in the process. The program started by the exec() initiated by the child in the scenario described above is immediately available; however, all memory maps, variables, and so on, are lost to the child process. If any information in the parent process is required by the child, that information must be provided by an alternative method.

To locate instances of fork() or vfork() use a command such as:

```
find ./ -name "*.[ch]" | xargs grep -i "fork" >fork.mods
         1                      2              3
```

Figure 20. Identifying instances of fork()

Figure 20 notes:

1. Locate all source files.
2. Look for occurrences of the string fork, FORK, vfork, or VFORK (vfork() is a variant of fork()).
3. Pipe the results into a file fork.mods so you can examine the results closely.

3.1.3 Sample conversions

The following two examples have been extracted from *OpenEdition for VM/ESA Advanced Application Programming Tools Version 2 Release 1.0*, SC24-5729.

3.1.3.1 Example 1

In this code fragment, an application uses the `fork`() function to create a child process. The child then becomes the process group leader of a new process group that runs in the foreground and invokes the `exec`() function to run another application.

```
   /*************************************************************/
   /* Issue fork to create a child that gets into its own process*/
   /* group, puts itself in the foreground, then issues exec()   */
   /*************************************************************/
   if ((pid = fork()) == -1)
       return(-1);
   else {
       if (pid == 0) {                                /* child */
           if (setpgid((pid_t) 0, (pid_t) 0) == -1)
               exit(-1);
           else {
               if (tcsetpgrp(cterm_fd, getpgrp()) == -1)
                   exit(-1);
               else {
                   execve("/prog", prog_args, my_env);
                   exit(-1);
               }
           }
       }
       else                                           /* parent */
           retpid = waitpid(pid, &status, 0);
   }
```

Figure 21. Fork to spawn conversion example 1: Fork version

```
/*************************************************************/
/*                                                           */
/*       Issue spawn to create a child process that is       */
/*       in its own process group and is in the foreground.  */
/*                                                           */
/*************************************************************/

inherit.flags = SPAWN_SETGROUP | SPAWN_SETTCPGRP;
inherit.pgroup = SPAWN_NEWPGROUP;
inherit.ctlttyfd = cterm_fd;

if ((pid = spawn("/prog", 0, NULL, &inherit, prog_args, my_env)) == -1)
    return(-1);
else
retpid = waitpid(pid, &status, 0);
```

Figure 22. Fork to spawn conversion example 1: Spawn version

3.1.3.2 Example 2

This example demonstrates how to convert an application from using `fork`() to using `spawn`() when the parent's signal environment differs from the child's. In this example, the parent must ignore specific signals. The child should ignore these

signals only if they were being ignored at the time this code fragment was invoked, otherwise they should be set to the default action. In addition, the parent must block the SIGCHLD signal, while the child must run with the signal mask that was in place at the time this code received control.

```
/*********************************************************************/
/*      Execute the command specified by the string pointed to       */
/*      by 'cmd'.  The environment of the executed command shall     */
/*      be as if a child process were created using the fork()       */
/*      function, and the child process invoked the 'sh'             */
/*      utility using the execl() function as follows:               */
/*          execl(<shell path>, "sh", "-c", cmd, (char *)0);         */
/*                                                                   */
/*      This function shall ignore the SIGINT and                    */
/*      SIGQUIT signals, and block the SIGCHLD signal, while         */
/*      waiting for the command to terminate.                        */
/*      This function shall not return until the child               */
/*      process has terminated.                                      */
/*********************************************************************/
int              stat;
pid_t            pid;
sigset_t         saveblock;
struct sigaction sa;
struct sigaction savintr;
struct sigaction savequit;

sa.sa_handler = SIG_IGN;
sigemptyset(&sa.sa_mask);
sa.sa_flags = 0;
sigemptyset(&savintr.sa_mask);
sigemptyset(&savequit.sa_mask);
sigaction(SIGINT, &sa, &savintr);
sigaction(SIGQUIT, &sa, &savequit);
sigaddset(&sa.sa_mask, SIGCHLD);
sigprocmask(SIG_BLOCK, &sa.sa_mask, &saveblock);

if ((pid = fork()) == 0) {               /* child */
    sigaction(SIGINT, &savintr, 0);
    sigaction(SIGQUIT, &savequit, 0);
    sigprocmask(SIG_SETMASK,&saveblock, 0);
    execl("/bin/sh", "sh", "-c", cmd, (char *)0);
    _exit(127);
}
if (pid == -1)
    stat = -1;
else {
    while (waitpid(pid, &stat, 0) == -1) {
        if (errno != EINTR) {
            stat = -1;
            break;
        }
    }
}
sigaction(SIGINT, &savintr, 0);
sigaction(SIGQUIT, &savequit, 0);
sigprocmask(SIG_SETMASK, &saveblock, 0);
return(stat);
```

Figure 23. Fork to spawn conversion example 2: Fork version

```
/*********************************************************************/
/*      Execute the command specified by the string pointed to       */
/*      by 'cmd'.  The environment of the executed command shall     */
/*      be as if a child process were created using the fork()       */
/*      function, and the child process invoked the 'sh'             */
/*      utility using the execl() function as follows:               */
/*          execl(<shell path>, "sh", "-c", cmd, (char *)0);         */
/*                                                                   */
/*      This function shall ignore the SIGINT and SIGQUIT signals,   */
/*      and block the SIGCHLD signal, while waiting for the command  */
/*      to terminate. This function shall not return until the child */
/*      process has terminated.                                      */
/*                                                                   */
/*      This version uses spawn() instead of fork(), exec().         */
/*                                                                   */
/*      Set up the spawn() inheritance structure so that:            */
/*      1) SIGQUIT and SIGINT will be set to their default actions   */
/*         in the child, if the process had not set them to SIG_IGN  */
/*         prior to calling this function. If either had been set to */
/*         SIG_IGN prior to the call, do nothing because signals set */
/*         be ignored in the parent will be ignored in the child.    */
/*      2) The child will inherit the signal mask of the parent,     */
/*         before the parent invoked this function.                  */
/*                                                                   */
/*      Get the pointer to environ and pass it to spawn(), so that   */
/*      the child inherits the parent's environment variables.       */
/*                                                                   */
/*********************************************************************/
int               stat;
pid_t             pid;
struct sigaction  sa;
struct sigaction  savintr;
struct sigaction  savequit;

struct inheritance inherit;
char *args[4] = { "sh", "-c", NULL, NULL };
extern char **environ;

args[2] = (char *)cmd;
inherit.flags = 0;
sigemptyset(&inherit.sigdefault);

sa.sa_handler = SIG_IGN;
sigemptyset(&sa.sa_mask);
sa.sa_flags = 0;
sigemptyset(&savintr.sa_mask);
sigemptyset(&savequit.sa_mask);
sigaction(SIGINT, &sa, &savintr);
sigaction(SIGQUIT, &sa, &savequit);
```

Figure 24. Fork to spawn conversion example 2: Spawn version (part 1 of 2)

```
/*********************************************************************/
/*|  If SIGINT and SIGQUIT were not being ignored at the time we     */
/*|  were invoked, we need to pass them in the sigdefault field      */
/*|  of the inheritance structure so that they will be set to the    */
/*|  default action in the child, and not be ignored, like in        */
/*|  the parent.                                                     */
/*********************************************************************/
if (savintr.sa_handler != SIG_IGN)  {
    sigaddset(&inherit.sigdefault, SIGINT);
    inherit.flags |= SPAWN_SETSIGDEF;
}
if (savequit.sa_handler != SIG_IGN)  {
    sigaddset(&inherit.sigdefault, SIGQUIT);
    inherit.flags |= SPAWN_SETSIGDEF;
}

if ((pid = spawn("/bin/sh", 0, NULL, &inherit,
          (const char **)args,
          (const char **)environ )) == -1) {
    switch (errno) {
          case ENOENT : stat = (127 << 8);
          break;
          default :     stat = -1;
          break;
    }
}
else {
    while (waitpid(pid, &stat, 0) == -1) {
          if (errno != EINTR) {
                stat = -1;
                break;
          }
    }
}
sigaction(SIGINT, &savintr, 0);
sigaction(SIGQUIT, &savequit, 0);
sigprocmask(SIG_SETMASK, &inherit.sigmask, 0);
return(stat);
```

Figure 25. Fork to spawn conversion example 2: Spawn version (part 2 of 2)

3.1.4 Other gotchas and hints

The following sections are hints gathered from both personal experience and as described in *OS/390 UNIX System Services Porting Guide* http://www.s390.ibm.com/oe/bpxa1por.html at:

`http://www.s390.ibm.com/oe/bpxa1por.html`

3.1.4.1 The magic value

Many UNIX systems support an executable text file that contains a *magic number* (also known as *pound bang, shbang* or *hashbang*). This is a text file beginning with #!pathname, for example:

```
#!/usr/bin/perl
```

The VM/ESA shell does not support this.

For systems that do not support it, the book *Programming PERL 5*, ISBN 1-5659-2149-6 recommends replacing

```
#!/usr/bin/perl
```

with this:

```
eval 'exec /usr/bin/perl -S $0 ${1+"$@"}'
if 0;
```

3.1.4.2 Set

The set command in OpenEdition for VM/ESA returns all shell variable values enclosed in double quotes. This behavior, documented in *OpenEdition for VM/ESA Command Reference Version 2 Release 3.0*, SC24-5728, may require modification of shell scripts that rely on either no quotes or single quotes in the returned data.

Both set and export display even simple variable values enclosed in double quotes:

```
ac_cv_newval="abc"
ac_cv_otherval="zyx"
```

Scripts that save a configuration value with the use of `set` or `env` might at first save values in this way:

```
ac_cv_newval=${ac_cv_newval='"abc"'}
ac_cv_otherval=${ac_cv_otherval='"zyx"'}
```

Executing the script a second time might result in:

```
ac_cv_newval=${ac_cv_newval='"\"abc\""'}
ac_cv_otherval=${ac_cv_otherval='"\"zyx\""'}
```

And a third time:

```
ac_cv_newval=${ac_cv_newval='"\"\\\"abc\\\"\""'}
ac_cv_otherval=${ac_cv_otherval='"\"\\\"zyx\\\"\""'}
```

If a configuration file is generated by a script, you might avoid this sort of problem by careful use of an `sed` filter to remove the unnecessary double quotes:

```
... | sed "s/=\'"'"\(.*\)"'"'/='\1'/" | ...
```

3.1.4.3 Testing for character strings

Different shells provide different means to test for character strings. The VM/ESA shell does not support *double brackets*, but you can do things like:

```
if [ "$resp" = yes ]
then
.
.
.
fi
```

Alternatively, you could use either of the following conditions to accomplish the same thing:

```
if [ X$resp = Xyes ]
if [ "X$resp" = Xyes ]
```

Also, test cannot do lexical greater-than or less-than comparisons, as in this example:

```
a="abc"
b="def"
if [[ $a > $b ]]
then
print "a bigger than b"
else
print "a smaller than b"
fi
```

This sort of script would have to be rewritten as:

```
a="abc"
b="def"
if expr $a \> $b
then
print "a bigger than b"
else
print "a smaller than b"
fi
```

3.1.4.4 Arithmetic expressions inside parentheses

By default, the VM/ESA shell does not accept statements like:

```
((x=y+z))
```

You must either:

- Use `let` (the command for which ((...)) is often synonymous):
  `let x=y+z`

or

- Turn on the *korn* shell option:
  `set -o korn (or set -K)`
  `((x=y+z))`

3.1.4.5 C language differences

There are some C language differences that you might encounter when using C for VM/ESA 3.1, as follows:

- The *const* qualifier may be regarded more strictly when considering if two function types are compatible.
- char versus unsigned char or signed char are not considered equivalent. That is, IBM's default is that char is an unsigned char, yet char and unsigned char are not always compatible. You can control this with the pre-processor directive:

  `#pragma chars (signed)`

  or

  `#pragma chars (unsigned)`
- You cannot initialize static or extern variables, except with constant expressions, which can also include references to the address of previously defined static or external variables.

3.1.4.6 The man pages

VM/ESA does not have an `nroff` formatter. If your program has `man` pages, you will need to take the `nroff` source to another system (for example, AIX) and format it there.

3.1.4.7 Users and passwords

UNIX uses the /etc/passwd file to keep track of every user on the system. This file contains the username, real name, identification information, and basic account information for each user.

With OpenEdition for VM/ESA, the CP Directory is used to keep track of every user on the system. There is no /etc/passwd file.

Users and their UIDs and passwords, and groups and their GIDs, are defined to the CP Directory. Each OpenEdition user has statements which allow the user to invoke an initial command and an initial working directory. Figure 26 illustrates two of these statements.

```
USER SAMBA DVM 32M 64M DG 75
 POSIXINFO UID 0 GNAME system IWDIR /home/samba
 POSIXOPT SETIDS ALLOW QUERYDB ALLOW EXEC_SETIDS ALLOW
 SCR INA PIN STA YEL CPO TUR INR RED
 OPTION QUICKDSP
 MACH XC
 SHARE RELATIVE 200 200
 ACCOUNT DAEMON SAMBA
 IPL CMS PARM AUTOCR FILEPOOL SFSTEST
 IUCV ALLOW
 IUCV ANY
 CONSOLE 0009 3215
 SPOOL   000C 2540 READER A
 SPOOL   000D 2540 PUNCH A
 SPOOL   000E 1403
```

Figure 26. Sample CP Directory entry showing POSIX related statements

Figure 26 notes:

1. The POSIXINFO statement defines the UID, GID (or group name), and the initial working directory.
2. The POSIXOPT statement defines extra privileges the user has.

3.1.4.8 Daemons

You may be porting a program that uses daemons. On VM/ESA, daemons run authorized (they have superuser authority) and can issue authorized functions such as the following to change the identity of a user's process:

- `setuid()`
- `seteuid()`

3.1.4.9 Ordering options and operands

In accordance with the POSIX.2 and XPG4 standards, options and operands of utilities cannot be mixed: all options must appear before operands. You cannot put the -o option at the end of the command. The X/Open compliance suites specifically test to ensure that `c89` does not allow the mixing of operands and options.

3.1.4.10 Exporting functions and variables

To avoid exposing unnecessary external variables and functions, selectively export external variables and functions by using #pragma export for C instead of the EXPORTALL compiler option.

If you export variables or functions unnecessarily, it has the following effects:

- It increases the size of your DLL, thus increasing the load and initialization costs when the DLL is first referenced by another program.
- It severely limits storage optimization (global variable coalescing and pruning of unreachable or 100% inlined code do not occur).

3.1.4.11 Compiler options

LANGLVL(EXTENDED): The C library contains several functions that are extensions to the SAA CPI Level 2 definition. These library functions are available only if the LANGLVL(EXTENDED) compile-time option is in effect.

LANGLVL(EXTENDED) can also make the compiler more “forgiving”, and allow you to compile code that it ordinarily would complain about (type mismatches, and the like).

To specify this option to `c89`, use:

```
-Wc,langlvl(extended)
```

but take care to protect the parentheses from shell interpretation.

LANGLVL(ANSI): This makes the compiler behave in a way consistent with the ANSI standard. This option causes the symbol _STDC_ to be defined. This is an important preprocessor variable, tested for extensively. Our preferred language level setting was:

```
-Wc,langlvl(ansi)
```

HWOPTS(STRING): If your processor has the string assist operations available (most modern CPUs do), make sure you specify this option as it will buy you big performance gains when any of the string functions are used in a C program (for example: `strcpy(), strcat(), strchr()`.

To specify:

```
-Wc,hwopts(string)
```

For enhanced diagnostic messages: use the -g option for compile/linkedit.

For automatic inlining: use the -O option for compile/linkedit. This is recommended for any performance-sensitive final code.

When using DLLs: we recommend that you use the compile option -Wc,dll.

For listings: use the -V option for compile/linkedit.

To indicate that symbols required by POSIX.1, POSIX.1a, and POSIX.2 are made visible: use -D_OPEN_SYS. Any symbols defined by the _OPEN_THREADS macro are also made visible. Additional symbols can be made visible if any of these standards explicitly allows the symbol to appear in the header in question or if the symbol is defined to be a VM/ESA UNIX extension.

3.1.4.12 Linking DLLs

Use the -Wb,p,dll option.

3.1.4.13 Limitations of `c89`

There are a number of small problems with the `c89` command:

- The default options for the preprocessor (`c89 -E`) include the SOURCE directive, which results in a warning stating the PPONLY and SOURCE options are incompatible.
- The CMS load sets a return code of 4 for unresolved entry points, as it does for invalid card format messages. The `c89` command assumes a return code of 4 is a warning and will reflect a 0 status back to the shell. The `make` command will interpret this as everything is okay and continue processing. However, an unresolved entry point is serious and will mean the module may not run successfully. You must inspect all logs when building packages to see if such events have occurred.
- Statements of the form:

  ```
  c89 -c test.c -DPARM="-Iincparm -Lvermin"
  ```

  will fail as the operands enclosed in quotes (“) will be interpreted as flags for the `c89` command. This is a shortcoming with the parsing logic of the command.

3.1.4.14 Conditional compilation

One of the problems programmers have is writing code that can work on many different machines. In theory, C code is portable; in reality, many machines have little differences that must be accounted for. The compiler allows the programmer great flexibility in changing the way code is generated through the use of conditional compilation. If you find that a function works differently or a header file declaration is different under VM/ESA, you can insert #ifdef and #endif statements in the code. For example:

```
#ifdef __OPEN_VM
# include <stdlib.h>
#else
# include <malloc.h>
#endif
```

3.1.4.15 Portable header files

In many porting exercises there will be times when header files expected by the package are not available. In such cases, a number of actions can be taken. They are summarized in the following table.

Table 2. Header file portability issues

Header File	Suggested Action
<access.h>	VM/ESA's equivalent interfaces are in <unistd.h> as per POSIX and XPG4.
<ar.h>	No official equivalent. The XPG4 package does install a copy (see Section 3.2, "XPG4 supplementary APIs" on page 48).
<arpa/ftp.h>	No equivalent. You can *borrow* a file from a UNIX system and use it.
<cur01.h>	No equivalent. Curses not supported under VM/ESA
<dir.h>	VM/ESA supports the <dirent.h> as per POSIX and XPG4.
<macros.h>	No equivalent at this time.
<select.h>	Use <sys/time.h> as per XPG4. This header contains the prototype for select() and macros like FD_SET.
<sys/ldr.h>	No equivalent at this time.
<sys/mnctl.h>	No equivalent at this time.
<sys/mode.h>	This header is not portable. An include for <fctnl.h> is more portable.
<sys/param.h>	This header is often unnecessary. Try removing the includes for it and see what falls out.
<sys/ptrace.h>	No equivalent at this time.
<sys/reg.h>	No equivalent at this time.
<sys/vmount.h>	No equivalent at this time.
<sys/vnode.h>	No equivalent at this time.
<termio.h>	VM/ESA supports <termios.h> as per POSIX and XPG4.
<usersec.h>	No equivalent at this time.
<userpw.h>	No equivalent at this time.

3.1.4.16 Access to socket header files using `c89`

To get access to the UNIX socket header files when compiling, consider using the following options when you invoke the c89 utility:

- -D_OPENVM

  This enables logic in include files that is unique to OpenEdition for VM/ESA (such as referencing variables defined in the DLL).

- -D_OE_SOCKETS

  Defines a BSD-like socket interface for the function prototypes and structures involved. This option is useful if you are porting a BSD4.3-conforming application to VM/ESA. Because XPG 4.2 sockets, _XOPEN_SOCKETS, contain some prototypes that require *const* on some input parameters, using this option instead would save you from editing the ported source to change some of the socket or IP address resolution calls. _OE_SOCKETS can be used with _XOPEN_SOURCE_EXTENDED 1, and the XPG4.2 socket interfaces will be replaced with the BSD-like interfaces.

- -D_ALL_SOURCE

Defines all of the functionality that is currently available with VM/ESA OpenEdition. In addition, defining _ALL_SOURCE makes a number of symbols visible that are not permitted under ANSI, POSIX or XPG4, but which are provided as an aid to porting C-language applications to VM/ESA.

Note: If a source program can be ported to VM/ESA just by defining _ALL_SOURCE, then it is possible to set this option on the command line invocation of the compiler:

c89 -D _ALL_SOURCE ....

If you do not specify any option, you will be using Universal UNIX (UU) sockets.

For detailed information about the effect of these and other compile-time flags see *IBM C for VM/ESA Library Reference, Volume 1 Version 3 Release 1*, SC23-3908 .

3.1.4.17 Porting with pthreads

This list of differences encountered with pthreads and mutexes on VM/ESA and OS/390 UNIX System Services was originally created by customers who subscribe to the mvs-oe mailing list.

Most of these differences exist because VM/ESA OpenEdition implemented the POSIX.4a draft 6 standard rather than the final version, POSIX.1c draft 10. The book *Pthreads Programming* by Nichols, Buttlar, and Farrell (ISBN 1-56592-115-1) has a chapter on these differences.

- **alarm()**

  The function `alarm()` will send the SIGALRM signal only to the thread that called `alarm()`. On some other platforms, SIGALRM can be sent to any thread in the process.

- **Thread-safe variants of POSIX routines**

  The pthreads standard defines thread-safe variants of existing POSIX functions (for example, `strtok_r` instead of `strtok`); however, these are not available under OpenEdition for VM/ESA. IBM's response is that under VM/ESA the normal versions are thread-safe, so you can use them directly. Because the two variants have differing prototypes, this represents a problem if you are porting code which contains the thread-safe variants. You have to either change the code or provide your own versions of the `_r` routines which map onto the "normal" ones. Here is an example of how to use a macro to create your own version:

  ```
  #define strtok_r(s,sep,lasts) strtok(s,sep)
  ```

  This will cause all occurrences of `strtok_r()` in the source file to be replaced by `strtok()` during the preprocessor stage of the compilation.

- **Static initialization of mutexes**

  VM/ESA does not allow you to statically initialize mutexes with `pthread_mutex_initialiser()`. To initialize mutexes, use `pthread_attr_init()` and `pthread_mutexattr_init()`. In addition to those two, you may want to use `pthread_mutexattr_setkind_np()`.

- `pthread_delete_key()`

  VM/ESA does not provide the `pthread_delete_key()` function.

- `pthread_attr_setdetachstate()`

VM/ESA does not define `pthread_create_detached()`, and the call to `pthread_attr_setdetachstate()` is slightly different, so look at the interface. You can set a variable to __DETATCHED and use this on the call instead.

- **Defaults for DETACHSTATE**

  For `pthread_attr_setdetachstate()`, VM/ESA and other platforms vary in their defaults for DETACHSTATE.

- **Process-shared attribute for mutexes**

  VM/ESA does not support mutexes shared across processes. You can use semaphores instead.

- `pthread_getspecific()`

  `pthread_getspecific()` has a slightly different prototype under VM/ESA than that specified in the standard.

- **Value returned on error**

  When VM/ESA pthread functions encounter an error, they return -1 and set errno. Other platforms return the error number as the function value. For example, `pthread_mutex_trylock()` returns -1 and errno contains EBUSY when the lock is occupied, instead of the POSIX-specified behavior to return a value of EBUSY.

- `sigwait()`

  VM/ESA implements sigwait as:

  ```
  int sigwait(sigset_t *set);
  ```

  In the standard it is:

  ```
  int sigwait(sigset_t *set, int *sig);
  ```

  VM/ESA returns the signal that interrupts the sigwait function as a return-value; the standard has it being returned in *sig.

  Therefore, sigwait has a different prototype under VM/ESA and OS/390 than in the standard. The confusion over which prototype to use extends to other platforms.

3.2 XPG4 supplementary APIs

When VM/ESA 2.3.0 was released, the OpenEdition portion of the product was enhanced with around 165 new APIs. In this one release many of the most commonly used APIs were now available to the programmer to use. However, there are a few key APIs missing that we found were needed when porting some of the applications described in this book.

Code has been written to provide some of these APIs, as well as others that do not fall under the banner of XPG4 but are commonly used. The following list describes the APIs and their function. In this list “nop” means the function was implemented as a no-operation (return after doing nothing):

- **truncate** - Truncate a file given a file descriptor (implemented using the `ftruncate()` API)
- **__passwd** - Validate or change a password.
- **daemon** - Become a system daemon (nop)
- **crypt** - Encrypt a piece of text

- **strcmpi** - Compare strings ignoring case (implemented using `strcasecmp()`)
- **syslog** - Write to the system log
- **getwd** - Get working directory (implemented using `getcwd()`)
- **writev** - Write a vector of buffers to a file (for non-socket files)
- **setrlimit** - Set resource limit (nop)
- **getrlimit** - Get resource limit (nop)
- **openlog** - Open connection to the system log
- **closelog** - Close connection to the system log
- **setlogmask** - Set logging characteristics of current connection to system log
- **getrusage** - Get resource usage
- **nice** - Change a process' priority (nop)

These functions have been packaged as a library known as libxpg4.a. This file is included in most link commands of the packages we ported. In addition, a couple of these functions are in a CMS TXTLIB named XPG4LIB.

In addition to the APIs described, there are a number of useful utility commands, header files, and shell scripts that we found were needed in doing our work. These were also packaged into our XPG4 extension suite.

3.2.1 XPG4 installation

This package can be downloaded from the accompanying CD-ROM or from the OpenEdition web sites listed in Section N.4, "Resources on the Internet" on page 340. Directions for downloading are in Section K.4, "The CD-ROM download methods" on page 316. Installation is as simple as entering the `make install` command:

```
make install
/bin/sh MakeTxtlib
File XPG4LIB TXTLIB not found
cp libxpg4.a /usr/lib/libxpg4.a
cp hostname /usr/local/bin
cp whoami /usr/local/bin
cp syslog.h /usr/include
cp sysexits.h /usr/include
cp syslog.3 /usr/local/man/man3
cp ar.h /usr/include
cp install-sh /usr/local/bin
cp mkinstalldirs /usr/local/bin
chmod 0755 /usr/local/bin/hostname
chmod 0755 /usr/local/bin/domainname
chmod 0755 /usr/local/bin/getps
chmod 0755 /usr/local/bin/whoami
chmod 0755 /usr/local/bin/install-sh
chmod 0755 /usr/local/bin/mkinstalldirs
```

3.2.2 Using XPG4 utility commands

The utility commands produce the following:

```
hostname
TOTVM1

domainname
ITSO.IBM.COM

getps
pid    user     state    command
------ -------- -------- -------
  7706    neale  Running
  5662    neale  Running sh -L
  4967    neale  Running getps

whoami
neale
```

Figure 27. XPG4 utility commands

3.2.2.1 Socket file limitations

Normally, when a process is spawned, file descriptors from the parents may be inherited by the child process. However, this is not true if those file descriptors are for sockets. This problem has been recognized by IBM and an APAR accepted and closed FIN (Fixed If Next release). Until this PTF becomes available, there is a need to pass socket file descriptors between different processes.

Why? Typically, a TCP/IP-based application will wait until work is requested of it by listening to a specific port (for example, a web server may listen on port 80). When a connection is accepted, the application will spawn a child process that does the real work and then goes back to listening for more work. To accomplish this, the child process inherits from its parent the socket on which the connection was accepted. The child then reads or writes to this socket to communicate with its partner (for example, a Web browser).

Fortunately, there is a workaround to this problem, which is used extensively by the packages described in this book. This workaround can be found in Appendix E, “Circumventing the Socket File Inheritance Limitation” on page 217.

Chapter 4. Porting the essential tools (GNUgzip, GNUmake)

The binary files of these tools are available on the CD-ROM associated with this book or:

`http://pucc.princeton.edu/~neale/vmoe.html`

We have reproduced the entire porting procedure in order to demonstrate what is involved in undertaking the porting tasks.

4.1 What is GNU

The GNU project was conceived in 1983 and has developed a complete free software system (recursively) named GNU (GNU is Not UNIX) that is upwardly compatible with UNIX. (Richard Stallman's initial document on the GNU project is called the *GNU Manifesto*; it has been translated into several other languages.)

The word *free* above pertains to intellectual freedom, not price. You may or may not pay a price to get GNU software. Either way, once you have the software you have three specific freedoms in using it:

1. The freedom to copy the program and give it away to your friends and co-workers.
2. The freedom to change the program as you wish, since you have full access to source code.
3. The freedom to distribute an improved version and thus help build the community. (If you redistribute GNU software, you may charge a fee for the physical act of transferring a copy, or you may give away copies.)

In 1971, when Richard Stallman started his career at MIT, he worked in a group which used free software exclusively. Even computer companies often distributed free software. Programmers were free to cooperate with each other, and often did.

The GNU members decided to make the operating system compatible with UNIX because the overall design was already proven and portable, and because compatibility makes it easy for UNIX users to switch from UNIX to GNU.

The initial goal of a free, UNIX-like operating system has been achieved. By the 1990s, GNU members had either found or written all the major components except one: the kernel. Then Linux, a free kernel, was developed by Linus Torvalds. Combining Linux with the almost-complete GNU system resulted in a complete operating system: a Linux-based GNU system. Estimates are that hundreds of thousands of people now use Linux-based GNU systems, including Slackware, Debian, Red Hat, and others.

4.2 Porting GNU gzip

With our VM environment established we were ready for our first porting task. We chose to port `gzip` first because it is simple and it is used by nearly all other packages described in this book. The build of `gzip` will be performed using the `make` command as supplied from Shell and Utilities for VM.

4.2.1 What is gzip

The compression utility gzip is designed to be a replacement for compress. Its main advantages over compress are much better compression and freedom from patented algorithms. The GNU Project uses gzip as the standard compression program for its system. By default, gzip currently uses the LZ77 algorithm used in zip 1.9 (the portable pkzip-compatible archiver). The gzip format was, however, designed to accommodate several compression algorithms.

The gunzip utility included in the gzip package can currently decompress files created by gzip, compress or pack. The detection of the input format is automatic. For the gzip format, gunzip checks a 32 bit Cyclic Redundancy Check (CRC). For pack, gunzip checks the uncompressed length. The compress format was not designed to allow consistency checks.

The gzip utility produces files with a .gz extension. gunzip is able to decompress .z files (packed or gziped).

gzip is free software. Under the terms of the GNU General Public License, a copy of which is provided under the name COPYING (included in the gzip package), you can redistribute it and/or modify it as you wish.

4.2.1.1 PKWare zip versus GNU gzip

The name "gzip" was a very unfortunate choice because zip and gzip are two really different programs, although the actual compression and decompression sources were written by the same persons. A different name should have been used for gzip, but it is too late to change now.

zip is an archiver: it compresses several files into a single archive file. gzip is a simple compressor: each file is compressed separately. Both share the same compression and decompression code for the "deflate" method. PKWare unzip can also decompress old zip archives (implode, shrink and reduce methods). GNU gunzip can also decompress files created by compress and pack. zip 1.9 and gzip do not support compression methods other than deflation (zip 1.0 supports shrink and implode). Better compression methods may be added in future versions of gzip. Since zip will always stick to absolute compatibility with pkzip, it is constrained by PKWare, which is a commercial company. The gzip header format is deliberately different from that of pkzip to avoid such a constraint.

On UNIX, gzip is mostly useful in combination with tar. GNU tar 1.11.2 has a -z option to invoke gzip automatically. tar -z compresses better than zip, since gzip can then take advantage of redundancy between distinct files. The drawback is that you must scan the whole tar.gz file in order to extract a single file near the end; unzip can directly seek to the end of the zip file. There is no overhead when you extract the whole archive anyway. If a member of a .zip archive is damaged, other files can still be recovered. If a .tar.gz file is damaged, files beyond the failure point can not be recovered.

4.2.2 Loading gzip sources

There are 3 ways to obtain gzip:

1. Use the CD-ROM included in this book.
2. Use Internet site: http://www.gzip.org/index.html

3. FTP files from `prep.ai.mit.edu:/pub/gnu` (sources in gzip-*.tar or .shar or .tar.gz). Make sure that you use BINARY for data transfer.

The method to upload the binaries from CD-ROM or from the network to your VM system is described in Appendix K, “General instructions for downloading packages” on page 311.

At this point, `gzip.tar.Z` is available in the directory /home/porting/ of our VM Byte File System (BFS). We are in the PORTING userid and we have run the SHELL EXEC to enter an OpenEdition for VM/ESA session.

1. To verify file availability, enter: `ls -l /home/porting/gzip.tar.Z`
2. Uncompress the file by typing: `uncompress gzip.tar.Z`

```
[porting - /home/porting]>
ls -l
total 648
-rw-rw-rw-   1 porting     system    327865 Apr 12 09:55 gzip.tar.Z
[porting - /home/porting]>
uncompress gzip.tar.Z
[porting - /home/porting]>
```

Figure 28. Uncompressing gzip

3. Unarchive the tar file with ASCII-to-EBCDIC translation with the command:

   ```
   pax -o from=ISO8859-1,to=IBM-1047 -rf gzip.tar
   ```

```
[porting - /home/porting]>
pax -o from=ISO8859-1,to=IBM-1047 -rf gzip.tar
[porting - /home/porting]>
ls -l
total 1544
drwxrwxrwx   1 porting     system         0 Apr 12 10:08 gzip-1.2.4
-rw-rw-rw-   1 porting     system    798720 Apr 12 09:55 gzip.tar
```

Figure 29. Loading gzip directories from TAR file

Note: steps 2 and 3 could be done in one step using:

```
pax -o from=ISO8859-1,to=IBM-1047 -rzf gzip.tar.Z
```

Verify that everything is OK by moving, with the `cd` command, to the directories created by `pax` and displaying the objects with the `ls` command.

```
[porting - /home/porting]>
cd gzip-1.2.4
[porting - /home/porting/gzip-1.2.4]>
ls
COPYING         configure       gzip.doc        revision.h      zdiff.in
ChangeLog       configure.in    gzip.h          sample          zforce.1
INSTALL         crypt.c         gzip.info       tailor.h        zforce.in
Makefile.in     crypt.h         gzip.texi       texinfo.tex     zgrep.1
NEWS            deflate.c       inflate.c       trees.c         zgrep.in
README          getopt.c        lzw.c           unlzh.c         zip.c
THANKS          getopt.h        lzw.h           unlzw.c         zmore.1
TODO            gpl.texinfo     match.S         unpack.c        zmore.in
algorithm.doc   gzexe.1         msdos           unzip.c         znew.1
amiga           gzexe.in        nt              util.c          znew.in
atari           gzip.1          os2             vms
bits.c          gzip.c          primos          zdiff.1
```

Figure 30. Verifying directories

The gzip files are loaded, as you can see in Figure 30, and we are now ready to build `gzip` from the source files.

`gzip` is built using the following commands:

1. `configure`

   `configure` is a shell script (Figure 31 on page 55 shows a console protocol of `configure`). This shell script checks the features and the compiler options available. `configure` will create a file named Makefile. This file will be used by the `make` command.

```
configure
checking how to run the C preprocessor
checking for underline in external names
1 nm: not found
checking for assembler
checking for install
checking for AIX
checking for minix/config.h
checking for POSIXized ISC
checking for DYNIX/ptx libseq
checking for ANSI C header files
checking for string.h
checking for stdlib.h
checking for memory.h
checking for fcntl.h
checking for time.h
checking for unistd.h
checking for utime.h
checking for sys/utime.h
checking for directory library header
checking for dirent.h
checking for closedir return value
checking for Xenix
checking for return type of signal handlers
checking for size_t in sys/types.h
checking if `#!' works in shell scripts
checking for gzip to derive installation directory prefix
        chose installation directory prefix
creating config.status
creating Makefile
```

Figure 31. Running configure

Figure 31 note:

1. `nm` is a command to examine libraries. It is not required, so it can be ignored.

2. `make clean`

   With this command, all the unnecessary objects in the gzip directory are removed.

```
make clean
rm -f *.o gzip gunzip ungzip zcat add sub a.out core
rm -f zcmp zdiff zgrep zmore znew zforce gzexe _gztest*
rm -f *.aux *.cp *.cps *.dvi *.fn *.fns *.ky *.kys *.log
rm -f *.pg *.pgs *.toc *.tp *.tps *.vr *.vrs
[porting - /home/porting/gzip-1.2.4]>
```

Figure 32. Cleaning gzip directories

3. `make`

   The `make` tool manages object dependencies between sources, macros, object files and so forth. `make` will rebuild only the objects affected by a change. To do its job, `make` uses the Makefile that was built by `configure`. Figure 33 on page 56 shows a console protocol of `make`.

```
[porting - /home/porting/gzip-1.2.4]>
make
c89 -c -DHAVE_UNISTD_H=1 -DDIRENT=1 -DVOID_CLOSEDIR=1  gzip.c
c89 -c -DHAVE_UNISTD_H=1 -DDIRENT=1 -DVOID_CLOSEDIR=1  zip.c
c89 -c -DHAVE_UNISTD_H=1 -DDIRENT=1 -DVOID_CLOSEDIR=1  deflate.c
c89 -c -DHAVE_UNISTD_H=1 -DDIRENT=1 -DVOID_CLOSEDIR=1  trees.c
c89 -c -DHAVE_UNISTD_H=1 -DDIRENT=1 -DVOID_CLOSEDIR=1  bits.c
c89 -c -DHAVE_UNISTD_H=1 -DDIRENT=1 -DVOID_CLOSEDIR=1  unzip.c
c89 -c -DHAVE_UNISTD_H=1 -DDIRENT=1 -DVOID_CLOSEDIR=1  inflate.c
c89 -c -DHAVE_UNISTD_H=1 -DDIRENT=1 -DVOID_CLOSEDIR=1  util.c
c89 -c -DHAVE_UNISTD_H=1 -DDIRENT=1 -DVOID_CLOSEDIR=1  crypt.c
WARNING CBC3356 ./crypt.c:6     Compilation unit is empty.
c89 -c -DHAVE_UNISTD_H=1 -DDIRENT=1 -DVOID_CLOSEDIR=1  lzw.c
c89 -c -DHAVE_UNISTD_H=1 -DDIRENT=1 -DVOID_CLOSEDIR=1  unlzw.c
c89 -c -DHAVE_UNISTD_H=1 -DDIRENT=1 -DVOID_CLOSEDIR=1  unpack.c
c89 -c -DHAVE_UNISTD_H=1 -DDIRENT=1 -DVOID_CLOSEDIR=1  unlzh.c
c89 -c -DHAVE_UNISTD_H=1 -DDIRENT=1 -DVOID_CLOSEDIR=1  getopt.c
WARNING CBC3180 STDIO H G2:140   Redeclaration of built-in function _gtca
                      ignored.
c89  -o gzip gzip.o zip.o deflate.o trees.o bits.o unzip.o inflate.o util.o
   crypt.o lzw.o unlzw.o unpack.o unlzh.o getopt.o
rm -f gunzip zcat
ln gzip gunzip
ln gzip zcat
```

Figure 33. Building gzip tool

4. `make install`

   This creates the directories (as shown in Figure 34 on page 57) /usr/local/man, /usr/local/bin, /usr/local/lib, usr/local/info and /usr/local/man/man1.

   The install process puts all `gzip` objects into production.

```
make install
if test ! -d /usr/local/man; then  mkdir /usr/local/man; fi
for dir in /usr/local/bin /usr/local/bin /usr/local/lib /usr/local/lib
/usr/local/info /usr/local/man/man1 ; do  if test ! -d ${dir};
then  mkdir ${dir}; fi;  done
sed -e "" -e "s|BINDIR|/usr/local/bin|" ./zdiff.in > zdiff
chmod 755 zdiff
sed -e "" -e "s|BINDIR|/usr/local/bin|" ./zgrep.in > zgrep
chmod 755 zgrep
sed -e "" -e "s|BINDIR|/usr/local/bin|" ./zmore.in > zmore
chmod 755 zmore
sed -e "" -e "s|BINDIR|/usr/local/bin|" ./znew.in > znew
chmod 755 znew
sed -e "" -e "s|BINDIR|/usr/local/bin|" ./zforce.in > zforce
chmod 755 zforce
sed -e "" -e "s|BINDIR|/usr/local/bin|" ./gzexe.in > gzexe
chmod 755 gzexe
cp gzip /usr/local/bin/gzip
for f in zdiff zgrep zmore znew zforce gzexe; do cp ${f} /usr/local/bin/${f}; done
rm -f /usr/local/bin/zcmp; ln /usr/local/bin/zdiff /usr/local/bin/zcmp
for f in gunzip ungzip zcat ; do  rm -f /usr/local/bin/${f}; done
for f in gzip gunzip zcat zdiff zgrep zmore znew zforce gzexe zcmp;
do  rm -f /usr/local/man/man1/${f}.1; done
cd .; for f in gzip gzexe; do  cp ${f}.1 /usr/local/man/man1/${f}.1; done
cd .; for f in zdiff zgrep zmore znew zforce; do  cp ${f}.1
/usr/local/man/man1/${f}.1; done
cd /usr/local/man/man1; if test ln = so; then  echo .so man1/gzip.1 > zcat.
1; echo .so man1/zdiff.1 > zcmp.1; echo .so man1/gzip.1 > gunzip.1; chmod 6
44 zcat.1 zcmp.1 gunzip.1; else  ln gzip.1  zcat.1; ln zdiff.1 zcmp.1; ln gzip.
gunzip.1; fi
cd .; for f in gzip.info* ; do cp ${f}  /usr/local/info/${f}; done
```

Figure 34. Putting gzip in production

`gzip` is now ready to be used.

4.3 Porting GNU make

We now have a working `gzip` and are ready to port a more complicated application, “GNU `make`”. We will show step by step how to port this UNIX application to OpenEdition for VM/ESA.

4.3.1 About GNU make

`make` is a tool which controls the generation of executables and other non-source files of a program from the program's source files.

GNU `make` was written by Richard Stallman and Roland McGrath.

`make` gets its knowledge of how to build your program from a file called the makefile, which lists each of the non-source files and how to compute it from other files. When you write a program, you should write a makefile for it, so that it is possible to use `make` to build and install the program.

4.3.2 Capabilities of make

GNU `make` includes the following functionality:

- It enables the end user to build and install your package without knowing the details of how that is done -- because these details are recorded in the makefile that you supply.
- It figures out automatically which files it needs to update, based on which source files have changed. It also automatically determines the proper order for updating files, in case one non-source file depends on another non-source file.
- As a result, if you change a few source files and then run `make`, it does not need to recompile all of your program. It updates only those non-source files that depend directly or indirectly on the source files that you changed.
- It is not limited to any particular language. For each non-source file in the program, the makefile specifies the shell commands to compute it. These shell commands can run a compiler to produce an object file, the linker to produce an executable, or `ar` to update a library.
- It is not limited to building a package. You can also use `make` to control installing or deinstalling a package, generate tag tables for it, or anything else you want to do often enough to make it worthwhile writing down how to do it.

4.3.3 make Rules and Targets

A rule in the makefile tells `make` how to execute a series of commands in order to build a target file from source files. It also specifies a list of dependencies of the target file. This list should include all files (whether source files or other targets) which are used as inputs to the commands in the rule of a makefile.

Here is what a rule looks like:

```
target:   dependencies ...
          commands
          ...
```

When you run `make`, you can specify particular targets to update; otherwise, `make` updates the first target listed in the makefile. Of course, any other target files needed as input for generating these targets must be updated first.

`make` uses the makefile to figure out which target files ought to be brought up to date, and then determines which of them actually need to be updated. If a target file is newer than all of its dependencies, then it is already up to date, and it does not need to be regenerated. The other target files do need to be updated, but in the right order: each target file must be regenerated before it is used in regenerating other targets.

4.3.4 Advantages of GNU make

- GNU `make` has many powerful macro features that can be used in makefiles, beyond what other `make` versions have. It can also regenerate, use, and then delete intermediate files which need not be saved.
- GNU `make` also has a few simple features that are very convenient. For example, the `-o` file option which says “pretend that source file has not changed, even though it has changed.” This is extremely useful when you add a new macro to a header file. Most versions of `make` will assume they must

therefore recompile all the source files that use the header file; but GNU `make` gives you a way to avoid the recompilation, in the case where you know your change to the header file does not require it.

4.3.5 Getting GNU make

You can get GNU `make` by:

- Copying it from the CD-ROM included with this book.
- Downloading it from:

  `http://www.ibm.com/s390/vm/openedition`
- Downloading it via ftp from the following FTP site or one of its mirrors.

  `ftp://ftp.gnu.org or one of its mirrors.`

`make` is available in a gzip format file named `make-3.77.tar.gz` The filename extensions of make-3.77 have the following meanings:

- `.tar` (the files included in this package have been stacked with TAR utility)
- `.gz` (then the tar file has been compressed with gzip)

4.3.6 Loading GNU make in BFS

How to upload the binaries from CD-ROM or from the network to your VM system is described in Appendix K, “General instructions for downloading packages” on page 311.

We assume that the file `make-3.77.tar.gz` is available in the directory /home/porting.

To install all objects included in the `make-3.77` package, do the following steps:

1. Uncompress the package using:

   `gzip /home/porting/make-3.77.tar.gz`
2. Unarchive the files included in make-3.77.tar using:

   `pax -o from=ISO8859-1,to=IBM-1047 -rf make-3.77.tar`

   This command creates the directory /home/porting/make-3.77 and loads objects in the subdirectories of `/make-3.77`
3. Move to the directory `/make-3.77` with `cd /home/porting/make-3.77`
4. We recommend reading the following files in the directory /home/porting/make-3.77
   - INSTALL file describes the installation process
   - README file gives the latest information
5. We also recommend reading information about the `ar.h` given in 3.1.4.15, “Portable header files” on page 45

To be able to build GNU `make`, we need the `ar.h` file. To install `ar.h` follow the install steps described in 3.2.1, “XPG4 installation” on page 49, or create an `ar.h` file with the sample given in Appendix C., “ar.h” on page 211.

4.3.7 Running configure

`configure` is a shell script that checks our work environment and sets many variables used during compile and linkedit phases. It is run using the following steps:

1. Start the configuration process with: `configure`

   The different messages received during the `configure` process are explained in Appendix F.1, “GNU make installation log” on page 221.

2. When the `configure` process is completed, we have to declare in `make.h` that VM is POSIX-capable.

   xedit the file `make.h` and add the bold statements shown in Figure 35 on lines 24-26 (#if defined ... , #define ... , #endif).

```
 make.h  V 134  Trunc=134 Size=477 Line=17 Col=1 Alt=0

00016 along with GNU Make; see the file COPYING.  If not, write to
00017 the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.  */
00018
00019 /* AIX requires this to be the first thing in the file.  */
00020 #if defined (_AIX) && !defined (__GNUC__)
00021  #pragma alloca
00022 #endif
00023
00024 #if defined (_POSIX_SOURCE) && !defined (POSIX) && defined (__OPEN_VM)
00025  #define POSIX
00026 #endif
00027
```

Figure 35. Modifying make.h

4.3.8 Compiling GNU make

To build the GNU `make` objects, we use the command `sh ./build.sh` as shown in Figure 36.

```
[porting - /home/porting/make-3.77]>
sh ./build.sh
compiling main.c...
WARNING CBC3296 ./make.h:259   #include file <alloca.h> not found.
compiling commands.c...
WARNING CBC3296 ./make.h:259   #include file <alloca.h> not found.
compiling job.c...
WARNING CBC3296 ./make.h:259   #include file <alloca.h> not found.
compiling dir.c...
WARNING CBC3296 ./make.h:259   #include file <alloca.h> not found.
compiling file.c...
WARNING CBC3296 ./make.h:259   #include file <alloca.h> not found.
compiling misc.c...
¢C
```

Figure 36. First error messages received with sh ./build.sh

We used `¢C` to stop the compile process, as it was evident that we had too many errors.

The console message in Figure 36 showed us that every error happened on line 259 of the `make.h` file.

To find out what the problem was, we xedited the file: `x make.h`

```
 make.h  V 134  Trunc=134 Size=477 Line=254 Col=1 Alt=0

00253
00254 #ifdef;__GNUC__
00255 #undef;alloca
00256 #define;alloca(n);__builtin_alloca (n)
00257 #else;/* Not GCC.  */
00258 #ifdef;HAVE_ALLOCA_H
00259 #include <alloca.h>
00260 #else;/* Not HAVE_ALLOCA_H.  */
00261 #ifndef;_AIX
00262 extern char *alloca ();
00263 #endif;/* Not AIX.  */
00264 #endif;/* HAVE_ALLOCA_H.  */
00265 #endif;/* GCC.  */
00266
```

Figure 37. Editing make.h

On line 259 in Figure 37, the include of alloca.h is dependent on the variable HAVE_ALLOCA_H, which is set in config.h.

Our solution was to xedit config.h and modify the bold lines as shown in Figure 38:

```
 config.h  V 80  Trunc=80 Size=298 Line=30 Col=1 Alt=0

00029
00030 /* Define if you have alloca, as a function or macro.  */
00031 #undef HAVE_ALLOCA
00032
00033 /* Define if you have <alloca.h> and it should be used (not on Ultrix).  *
/
00034 #undef HAVE_ALLOCA_H
00035
00036 /* Define if you don't have vprintf but do have _doprnt.  */
00037 /* #undef HAVE_DOPRNT */
00038
```

Figure 38. Undefine HAVE_ALLOCA

Next we restarted the compile of make: `sh ./build.sh` and checked the console messages in Figure 39 on page 62.

```
 sh ./build.sh
compiling main.c...
compiling commands.c...
compiling job.c...
compiling dir.c...
compiling file.c...
compiling misc.c...
compiling read.c...
compiling remake.c...
compiling rule.c...
compiling implicit.c...
compiling default.c...
compiling variable.c...
compiling expand.c...
compiling function.c...
compiling vpath.c...
compiling version.c...
compiling ar.c...
compiling arscan.c...
compiling signame.c...
compiling getopt.c...
WARNING CBC3180 STDIO H Y2:140   Redeclaration of built-in function _gtca ignore
d.
compiling getopt1.c...
WARNING CBC3180 STDIO H Y2:140   Redeclaration of built-in function _gtca ignore
d.
compiling glob/fnmatch.c...
compiling glob/glob.c...
WARNING CBC3213 ./glob/glob.c:233   Macro name __stat cannot be redefined.
ERROR CBC3023 ./glob/glob.c:740   Expecting function or pointer to function.
ERROR CBC3023 ./glob/glob.c:885   Expecting function or pointer to function.
ERROR CBC3023 ./glob/glob.c:960   Expecting function or pointer to function.
ERROR CBC3023 ./glob/glob.c:1186  Expecting function or pointer to function.
compiling remote-stub.c...
linking make...
DMSOVF2113E Object does not exist: './glob.o'
done
mv: make.new: EDC5129I No such file or directory.
[porting - /home/porting/make-3.77]>
```

Figure 39. Console log

As you see in Figure 39, there was a new error: a WARNING message generated during the compile of getopt.c and getopt1.c. The error occurred in STDIO H Y2 at line 140.

To find out what the source of the warning message was, we looked in STDIO H Y2 by typing: `cms x stdio h y2`

Note: This command is prefixed with `cms` to bypass our shell alias x and instead invoke the Xedit command directly since the file we want to edit is an ordinary CMS minidisk file, not a BFS file.

```
 STDIO    H        Y2  V 80  Trunc=80 Size=794 Line=133 Col=1 Alt=0

00132
00133   #ifndef __gtca
00134     #define __gtca() _gtca()
00135     #ifdef __cplusplus
00136       extern "builtin"
00137     #else
00138       #pragma linkage(_gtca,builtin)
00139     #endif
00140     const void *_gtca(void);
00141   #endif
00142
00143   #ifndef __temp
```

Figure 40. XEDITing STDIO H

The reason for the warning message was that in `stdio h y2`, `_gtca` was redefined because the `__gtca` variable was not set.

Our solution was to set `__gtca` correctly by adding a `#define __gtca` in config.h as shown on lines 11-13 in Figure 41.

```
 config.h  V 80  Trunc=80 Size=302 Line=0 Col=1 Alt=3


00000 * * * Top of File * * *
00001 /* config.h.  Generated automatically by configure.  */
00002 /* config.h.in.  Generated automatically from configure.in by autoheader.
 */
00003
00004 /* Define if on AIX 3.
00005    System headers sometimes define this.
00006    We just want to avoid a redefinition error message.  */
00007 #ifndef _ALL_SOURCE
00008 /* #undef _ALL_SOURCE */
00009 #endif
00010
00011 #ifdef __OPEN_VM
00012   #define  __gtca
00013 #endif
00014
00015 /* Define if using alloca.c.  */
00016 /* #undef C_ALLOCA */
00017
```

Figure 41. Modification in config.h

Note: The variable `__OPEN_VM` is set when we work under OpenEdition for VM/ESA. We use this variable each time we set VM-specific options.

Before we restarted the compilation, we also changed some c89 compiler options. These options are set in the `build.sh` file.

To obtain explanations about c89 options you can either:

- Access `man c89` in your OpenEdition for VM/ESA session (`man` is an alias for help)

- Read the appropriate section in the *OpenEdition for VM/ESA Command Reference Version 2 Release 3.0*, SC24-5728.

c89 uses a `-W` flag to set compiler and linkedit options. It has the following format:

```
-W phase, option(,option)...
```

The choices for phase are:

0 or c specifies the compile phase

b specifies the module build phase

This phase includes prelinker processing, the loading of the resulting CMS TEXT file via the CMS LOAD command, and the creating of the module file by the CMS GENMOD command.

The choices for option are:

l to pass options to the LOAD command

g for the GENMOD command

p for the prelinker

For example, to write the prelink map to stdout, specify:

```
c89 -W b,p,map file.c
```

The current setting for CFLAGS in build.sh is -W c,nosource. With this setting, we have no listing during the compile phase (c89 compiler generate files with extension `.lst`). For ease of debugging, we allowed the .lst file to be generated by overriding this option. We could set `-W c,source` for a source listing or `-W c,list` to have combined source and assembly listing.

LDFLAGS defines options for linkedit. We changed the setting to '`-W b,l,map`' to obtain a loadmap (`.loadmap`) to be created during CMS LOAD processing.

In build.sh, as you can see in Figure 42, the oldest options are commented out by using the # character. We inserted the bold lines (27, 29) and we also moved option `-o make.new` at line 68 before the option `$objs`.

```
 build.sh  V 252  Trunc=252 Size=70 Line=21 Col=1 Alt=0

00020 # the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
00021
00022 # See Makefile.in for comments describing these variables.
00023
00024 srcdir='.'
00025 CC='c89'
00026 # CFLAGS='-g -W c,nosource'
00027 CFLAGS='-g -W c,list'
00028 CPPFLAGS=''
00029 LDFLAGS='-W b,l,map'
00030 defines='-DHAVE_CONFIG_H -DLIBDIR="${libdir}" -DINCLUDEDIR="${includedir}"'
*****
*****
00067 echo linking make...
00068 $CC $LDFLAGS -o make.new $objs $LOADLIBES
00069 echo done
00070 mv -f make.new make
```

Figure 42. Modification in build.sh

We then restarted the compile with the command: `sh ./build.sh`

The next errors to be analyzed were during glob.c compile at line 233, as you can see in Figure 43.

```
 sh ./build.sh
compiling main.c...
compiling commands.c...
compiling job.c...
compiling dir.c...
compiling file.c...
compiling misc.c...
compiling read.c...
compiling remake.c...
compiling rule.c...
compiling implicit.c...
compiling default.c...
compiling variable.c...
compiling expand.c...
compiling function.c...
compiling vpath.c...
compiling version.c...
compiling ar.c...
compiling arscan.c...
compiling signame.c...
compiling getopt.c...
compiling getopt1.c...
compiling glob/fnmatch.c...
compiling glob/glob.c...
WARNING CBC3213 ./glob/glob.c:233   Macro name __stat cannot be redefined.
ERROR CBC3023 ./glob/glob.c:740   Expecting function or pointer to function.
ERROR CBC3023 ./glob/glob.c:885   Expecting function or pointer to function.
ERROR CBC3023 ./glob/glob.c:960   Expecting function or pointer to function.
ERROR CBC3023 ./glob/glob.c:1186  Expecting function or pointer to function.
compiling remote-stub.c...
linking make...
DMSOVF2113E Object does not exist: './glob.o'
done
mv: make.new: EDC5129I No such file or directory.
[porting - /home/porting/make-3.77]>
```

Figure 43. Console log

To fix the warning from compile edit `glob.c`, be aware that the file `glob.c` is in directory `./make-3.77/glob`

```
 ./glob/glob.c  V 80  Trunc=80 Size=1346 Line=233 Col=1 Alt=0

00232 #ifndef __GNU_LIBRARY__
00233 # define __stat stat
00234 # ifdef STAT_MACROS_BROKEN
00235 #  undef S_ISDIR
00236 # endif
00237 # ifndef S_ISDIR
00238 #  define S_ISDIR(mode) (((mode) & S_IFMT) == S_IFDIR)
00239 # endif
00240 #endif
00241
00242 #ifdef _LIBC
```

Figure 44. Display glob.c

As you see in Figure 44 on page 65, the program `glob.c` tries to define a `__stat` variable, but this variable has already been defined during include of <sys/stat.h> at line 34 of `glob.c`.

We resolved this by adding the bold statements from Figure 45 (`#undef __stat`) before line 233 in the original `glob.c` file shown in Figure 44 on page 65.

```
 glob/glob.c  V 80  Trunc=80 Size=1349 Line=231 Col=1 Alt=10

00230 #endif
00231
00232 #ifndef __GNU_LIBRARY__
00233   #ifdef __OPEN_VM
00234     #undef __stat
00235   #endif
00236 # define __stat stat
00237 # ifdef STAT_MACROS_BROKEN
```

Figure 45. Modify glob.c

We restarted the compile with the command: `sh ./build.sh`

```
[porting - /home/porting/make-3.77]>
sh ./build.sh
compiling main.c...
compiling commands.c...
compiling job.c...
compiling dir.c...
compiling file.c...
compiling misc.c...
compiling read.c...
compiling remake.c...
compiling rule.c...
compiling implicit.c...
compiling default.c...
compiling variable.c...
compiling expand.c...
compiling function.c...
compiling vpath.c...
compiling version.c...
compiling ar.c...
compiling arscan.c...
compiling signame.c...
compiling getopt.c...
compiling getopt1.c...
compiling glob/fnmatch.c...
compiling glob/glob.c...
compiling remote-stub.c...
linking make...
DMSLIO201W The following names are undefined:
 ALLOCA   SETLINEB STRSIGNA GETLOADA
done
```

Figure 46. Console log

As this figures shows, we made some progress, but there were still some problems to fix to complete the porting.

The errors we received in Figure 46 on page 66 were created during linkedit of GNU `make`; the error log can be found in the file `make.map`. The message DMSLIO201W was generated because no text file with that name could be located. The question was did we need ALLOCA, SETLINEB, STRSIGNA and GETLOADA? Were we missing any function by not including them ?

We started solving the first missing function by verifying if ALLOCA existed on our disks. It did not.

Our solution took advantage of the fact that, for VM, there is an equivalent function to `alloca`(), called `malloc`(). To declare that we wanted to use `malloc`() instead of `alloca`(), we edited build.sh and added at line 30 `-Dalloca=malloc` (see Figure 47). This, in effect, changed all occurrences of `alloca`() to `malloc`().

```
 build.sh  V 252  Trunc=252 Size=70 Line=28 Col=1 Alt=0

00027 CFLAGS='-g -W b,l,map'
00028 CPPFLAGS=''
00029 LDFLAGS=''
00030 defines='-DHAVE_CONFIG_H -DLIBDIR="${libdir}" -DINCLUDEDIR="${includedir}"
 -Dalloca=malloc '
00031 ALLOCA=''
00032 LOADLIBES=''
00033 extras=''
00034 REMOTE='stub'
00035
00036 # Common prefix for machine-independent installed files.
```

Figure 47. Modifying build.sh

We again restarted the compile with: `sh ./build.sh`

```
[porting - /home/porting/make-3.77]>
sh build.sh
compiling main.c...
compiling commands.c...
compiling job.c...
compiling dir.c...
compiling file.c...
compiling misc.c...
compiling read.c...
compiling remake.c...
compiling rule.c...
compiling implicit.c...
compiling default.c...
compiling variable.c...
compiling expand.c...
compiling function.c...
compiling vpath.c...
compiling version.c...
compiling ar.c...
compiling arscan.c...
compiling signame.c...
compiling getopt.c...
compiling getopt1.c...
compiling glob/fnmatch.c...
compiling glob/glob.c...
compiling remote-stub.c...
linking make...
DMSLIO201W The following names are undefined:
 SETLINEB STRSIGNA GETLOADA
done
```

Figure 48. Console log

According to the new console log in Figure 48, the ALLOCA error had been solved, but we still had some work to do with the functions SETLINEB, STRSIGNA and GETLOADA undefined.

Our next task was to fix the next unresolved reference in `config.h` and locate `setline`. We found that HAVE_SETLINEBUF variable is defined. If you look in CMS, you see we don't have SETLINEB * * on our disks.

Our solution was to undefine HAVE_SETLINEBUF as shown in Figure 49.

```
 config.h  V 80  Trunc=80 Size=302 Line=226 Col=1 Alt=0

00225
00226 /* Define if you have the setlinebuf function.  */
00227 #undef HAVE_SETLINEBUF
00228
00229 /* Define if you have the setregid function.  */
00230 #define HAVE_SETREGID 1
```

Figure 49. Modifying config.h

We restarted the compile by the command: `sh ./build.sh`

```
[porting - /home/porting/make-3.77]>
sh build.sh
compiling main.c...
compiling commands.c...
compiling job.c...
compiling dir.c...
compiling file.c...
compiling misc.c...
compiling read.c...
compiling remake.c...
compiling rule.c...
compiling implicit.c...
compiling default.c...
compiling variable.c...
compiling expand.c...
compiling function.c...
compiling vpath.c...
compiling version.c...
compiling ar.c...
compiling arscan.c...
compiling signame.c...
compiling getopt.c...
compiling getopt1.c...
compiling glob/fnmatch.c...
compiling glob/glob.c...
compiling remote-stub.c...
linking make...
DMSLIO201W The following names are undefined:
 STRSIGNA GETLOADA
done
```

Figure 50. Console log

There were still two undefined functions to solve: STRSIGNA and GETLOADA.

We resolved this by undefining STRSIGNAL and GETLOADAVG in the file config.h, as shown in Figure 51 on page 70. Be aware that `configure` during its process created two entries for the variable HAVE_GETLOADAVG in config.h, so we undefined both entries.

```
 config.h  V 80  Trunc=80 Size=302 Line=46 Col=1 Alt=2

00045
00046 /* Define if your system has its own `getloadavg' function.  */
00047 #undef HAVE_GETLOADAVG
00048
*****
*****
00204
00205 /* Define if you have the getloadavg function.  */
00206 #undef HAVE_GETLOADAVG
00207
*****
*****
00246
00247 /* Define if you have the strsignal function.  */
00248 #undef HAVE_STRSIGNAL
00249
00250 /* Define if you have the wait3 function.  */
```

Figure 51. Modifying config.h

We restarted the compile again with the command: `sh ./build.sh`

```
sh build.sh
compiling main.c...
compiling commands.c...
compiling job.c...
compiling dir.c...
compiling file.c...
compiling misc.c...
compiling read.c...
compiling remake.c...
compiling rule.c...
compiling implicit.c...
compiling default.c...
compiling variable.c...
compiling expand.c...
compiling function.c...
compiling vpath.c...
compiling version.c...
compiling ar.c...
compiling arscan.c...
compiling signame.c...
ERROR CBC3045 ./signame.c:307   Undeclared identifier sys_siglist.
compiling getopt.c...
compiling getopt1.c...
compiling glob/fnmatch.c...
compiling glob/glob.c...
compiling remote-stub.c...
linking make...
DMSLIO201W The following names are undefined:
 STRSIGNA GETLOADA
done
```

Figure 52. Console log

We believed we had solved all the problems, but the console log in Figure 52 tells us we had a new problem.

To solve this new error in signame.c, we examined signame.lst and found that the error happened because the variable `sys_siglist` is not defined. We checked in config.h and saw that `sys_siglist` is a variable dependent upon the preprocessor variable SYS_SIGLIST_DECLARED. We looked in SIGNAL H and saw that `sys_siglist` is not defined in SIGNAL H.

The solution for this was to undefine SYS_SIGLIST_DECLARED at line 154 and undefine HAVE_SYS_SIGLIST at line 182 of config.h, as shown in Figure 53.

```
 config.h  V 80  Trunc=80 Size=302 Line=153 Col=1 Alt=0

00001 --------------------  152  line(s) not displayed --------------------
00153 /* Define if `sys_siglist' is declared by <signal.h>.  */
00154 #undef SYS_SIGLIST_DECLARED
00155 --------------------  26  line(s) not displayed --------------------
00181 /* Define this if the C library defines the variable `sys_siglist'.  */
00182 #undef HAVE_SYS_SIGLIST
00183 --------------------  1  line(s) not displayed --------------------
00184 /* Define this if the C library defines the variable `_sys_siglist'.  */
00185 /* #undef HAVE__SYS_SIGLIST */
00186 --------------------  117  line(s) not displayed --------------------
```

Figure 53. Modifying config.h

We again restarted the compile with the command: `sh ./build.sh`

```
sh ./build.sh
compiling main.c...
compiling commands.c...
compiling job.c...
compiling dir.c...
compiling file.c...
compiling misc.c...
compiling read.c...
compiling remake.c...
compiling rule.c...
compiling implicit.c...
compiling default.c...
compiling variable.c...
compiling expand.c...
compiling function.c...
compiling vpath.c...
compiling version.c...
compiling ar.c...
compiling arscan.c...
compiling signame.c...
compiling getopt.c...
compiling getopt1.c...
compiling glob/fnmatch.c...
compiling glob/glob.c...
compiling remote-stub.c...
linking make...
WARNING EDC4011: Unresolved writable static references are detected.
DMSLIO201W The following names are undefined:
 GETLOADA
done
```

Figure 54. Console log

Now we were down to only one unresolved entry point.

To resolve the GETLOADA undefined problem, we:

1. Searched under CMS to see if a GETLOADA H file existed or if getloadavg existed in any header file. The answer was No.
2. Searched in directory /make-3.77 using command `ls -l getloada*` to see if we had an object by that name. The search command and the result is shown in Figure 55. As you see there, the `getloadavg.c` does exist.

```
[porting - /home/porting/make-3.77]>
ls -l getloada*
-rw-r--r--   2 porting     system     26579 Jul 23  1998 getloadavg.c
```

Figure 55. Searching getloadavg

To solve this problem we had to compile `getloadavg.c` so it could be used during linkedit. We did this by putting it in the list (`build.sh`) of files to be compiled. Then in addition we opened `build.sh` and changed line 33 to `extras='getloadavg.c'`.

```
 build.sh  V 252  Trunc=252 Size=70 Line=28 Col=1 Alt=0

00027 CFLAGS='-g -W b,l,map'
00028 CPPFLAGS=''
00029 LDFLAGS=''
00030 defines='-DHAVE_CONFIG_H -DLIBDIR="${libdir}" -DINCLUDEDIR="${includedir}"
 -Dalloca=malloc '
00031 ALLOCA=''
00032 LOADLIBES=''
00033 extras='getloadavg.c'
00034 REMOTE='stub'
00035
00036 # Common prefix for machine-independent installed files.
```

Figure 56. Modifying build.sh

We restarted the compile with the command: `sh ./build.sh`

```
[porting - /home/porting/make-3.77]>
sh ./build.sh
compiling main.c...
compiling commands.c...
compiling job.c...
compiling dir.c...
compiling file.c...
compiling misc.c...
compiling read.c...
compiling remake.c...
compiling rule.c...
compiling implicit.c...
compiling default.c...
compiling variable.c...
compiling expand.c...
compiling function.c...
compiling vpath.c...
compiling version.c...
compiling ar.c...
compiling arscan.c...
compiling signame.c...
compiling getopt.c...
compiling getopt1.c...
compiling glob/fnmatch.c...
compiling glob/glob.c...
compiling remote-stub.c...
compiling getloadavg.c...
linking make...
done
```

Figure 57. Console log

According to the console log in Figure 57 there were no more errors. The next step was to check if fork() was used.

4.3.9 Checking usage of fork()

To check if the `fork`() function was used in our source files, we ran the following command:

```
find ./ -name "*.[c h]" | xargs grep -i "fork"
```

```
find ./ -name "*.[c h]" | xargs grep -i "fork"
./commands.c:   fork off a child process to run the first command line in the se
quence.  */
./config.h:/* Define if you have <vfork.h>.  */
./config.h:/* #undef HAVE_VFORK_H */
./config.h:/* Define vfork as fork if vfork does not work.  */
./config.h:#define vfork fork
./function.c:     /* MSDOS can't fork, but it has `popen'.
./function.c:   pid = vfork ();
./function.c:     perror_with_name (error_prefix, "fork");
./function.c:      /* Amiga can't fork nor spawn, but I can start a program with
./job.c:     and forking a useless shell all the time leads to inefficiency. */
./job.c:      /* Fork the child process.  */
./job.c:        /* Fork failed!  */
./job.c:        perror_with_name ("vfork", "");
./job.c:      child->pid = vfork ();
./job.c:          /* Fork failed!  */
./job.c:          perror_with_name ("vfork", "");
./make.h:#ifdef HAVE_VFORK_H
./make.h:#include <vfork.h>
./misc.c:     run in a child fork whose stdout is piped.  */
./remote-cstms.c:  pid = vfork ();
./remote-cstms.c:      /* The fork failed!  */
./remote-cstms.c:      perror_with_name ("vfork", "");
./w32/subproc/proc.h:#define E_FORK            104

./w32/subproc/sub_proc.c:                     pproc->lerrno = E_FORK;

[porting - /home/porting/make-3.77]>
```

Figure 58. Locate fork function

The console log in Figure 58 of the `find` command showed that the function `vfork`() was used in:

- config.h
- function.c
- job.c
- make.h
- misc.c
- remote-cstms.c
- proc.c
- sub_proc.c

The files proc.h and sub_proc.c are for Windows and therefore we could ignore them.

In config.h, the reference redefines only vfork to fork, thus there was no change for us to make here.

In misc.c, this was only a comment, and there were no changes to make.

In make.h, <vfork.h> included depends on HAVE_VFORK_H variable, and in our case this variable was not set. So we did not change anything here.

4.3.9.1 Modifying job.c

To fix job.c the following steps were taken:

1. Edit job.c and locate vfork()
2. To be able to use spawn() instead of vfork(), we had to add the following bold statements to job.c (see Figure 59 on page 76 and Figure 60 on page 77). The changes are explained in 3.1.2, “Forking and spawning” on page 35.

```
00963 #ifdef __OPEN_VM
00964      child->remote = 0;
00965      inherit.flags = 0;
00966      child->pid    = spawnp((const char *) argv[0], 0, NULL, &inherit,
00967                             (const char **) argv,
00968                             (const char **) child->environment);
00969      if (child->pid < 0)
00970      {
00971         spawncmd = malloc(strlen(argv[0]) + 8);
00972         strcpy(spawncmd,"/bin/");
00973         strcat(spawncmd,argv[0]);
00974         child->pid = spawn((const char *) spawncmd, 0, NULL, &inherit,
00975                            (const char **) argv,
00976                            (const char **) child->environment);
00977         if (child->pid < 0)
00978         {
00979            strcpy(spawncmd,"/bin/sh");
00980            for (n_arg = 0; argv[n_arg] != 0; n_arg++);
00981                n_arg       = (n_arg + 2) * sizeof(argv[0]);
00982                childarg    = (char **) malloc(n_arg);
00983                childarg[0] = spawncmd;
00984            for (n_arg = 0; argv[n_arg] != 0; n_arg++)
00985               childarg[n_arg+1] = argv[n_arg];
00986            childarg[n_arg+1] = NULL;
00987            child->pid = spawn((const char *) spawncmd, 0,
00988                                             NULL, &inherit,
00989                                  (const char **) childarg,
00990                        (const char **) child->environment);
00991            if (child->pid < 0)
00992            {
00993               /* Spawn failed!  */
00994               free(*childarg);
00995               free(spawncmd);
00996               unblock_sigs ();
00997               perror_with_name ("spawn failed for ", argv[0]);
00998               goto error;
00999            }
01000            free(*childarg);
01001         }
01002         free(spawncmd);
01003       }
01004       if (debug_flag)
01005       {
01006        int i_arg;
01007
01008          printf ("Spawning child x%08lx %s ",
01009                  argv[0], child);
01010          for (i_arg = 0; argv[i_arg] != NULL; i_arg++)
01011             printf ("%s ",argv[i_arg]);
01012          printf ("PID %d\n",child->pid);
01013       }
01014       unblock_sigs ();
01015 # endif  /* _OPENVM  */
01016     }
*****
*****
01644 #if !defined (_AMIGA) && !defined (__MSDOS__) && !defined (__OPEN_VM)
01645 /* UNIX:
*****
```

Figure 59. Modifying job.c (part 1 of 2)

```
*****
01666 }
01667 #endif /* !AMIGA && !__MSDOS__ && !VM */
```

Figure 60. Modifying job.c (part 2 of 2)

4.3.9.2 Modifying function.c

The modifications needed in function.c are shown in Figure 61, Figure 62 on page 78, and Figure 63 on page 79.

```
 function.c  V 80  Trunc=80 Size=1631 Line=25 Col=1 Alt=4

00024 #include "commands.h"
00025
00026 #ifdef __OPEN_VM
00027 #include <spawn.h>
00028 #endif
00029 #ifdef _AMIGA
00030 #include "amiga.h"
*****
*****
00037
00038 #ifdef __OPEN_VM
00039 #define PIPBUFLEN 512
00040 #endif
00041
00042 static char *string_glob PARAMS ((char *line));
*****
*****
00376 #endif
00377
00378 #ifdef __OPEN_VM
00379    struct inheritance inherit;
00380    int fdmap[3];
00381    char *spawncmd;
00382    char **childarg;
00383    int  n_arg;
00384 #endif
00385
00386 ;/* Expand the command line.  */
*****
*****
00531 #ifndef __OPEN_VM
00532 ;pid = vfork ();
00533 ;if (pid < 0)
00534 ;  perror_with_name (error_prefix, "fork");
00535 ;else if (pid == 0)
00536 ;  child_execute_job (0, pipedes[1], argv, envp);
00537 ;else
00538 #endif /* !VM */
00539 #  endif /* ! __MSDOS__ */
```

Figure 61. Modifying function.c (part 1 of 3)

```
00540 # endif /* WINDOWS32 */
00541 #ifdef __OPEN_VM
00542   inherit.flags            = 0;
00543   fdmap[STDIN_FILENO]      = STDIN_FILENO;
00544   fdmap[STDOUT_FILENO]     = pipedes[1];
00545   fdmap[STDERR_FILENO]     = STDERR_FILENO;
00546   shell_function_completed = 0;
00547   shell_function_pid       = spawnp((const char *) argv[0], 3, fdmap,
00548                                     &inherit,
00549                                     (const char **) argv,
00550                                     (const char **) envp);
00551   if (shell_function_pid < 0)
00552   {
00553      spawncmd = malloc(strlen(argv[0]) + 8);
00554      strcpy(spawncmd, "/bin/");
00555      strcat(spawncmd, argv[0]);
00556      shell_function_pid = spawnp((const char *) spawncmd, 3,
00557                                  fdmap, &inherit,
00558                                  (const char **) argv,
00559                                  (const char **) envp);
00560      if (shell_function_pid < 0)
00561      {
00562          strcpy(spawncmd,"/bin/sh");
00563          for (n_arg = 0; argv[n_arg] != 0; n_arg++);
00564          n_arg       = (n_arg + 2) * sizeof(argv[0]);
00565          childarg    = (char **) malloc(n_arg);
00566          childarg[0] = spawncmd;
00567          for (n_arg = 0; argv[n_arg] != 0; n_arg++)
00568             childarg[n_arg+1] = argv[n_arg];
00569          childarg[n_arg+1] = NULL;
00570          shell_function_pid = spawn((const char *) spawncmd, 3,
00571                                     fdmap, &inherit,
00572                                     (const char **) childarg,
00573                                     (const char **) envp);
00574          if (shell_function_pid < 0)
00575          {
00576             free(spawncmd);
00577             perror_with_name (error_prefix, "spawn");
00578          }
00579          free(*childarg);
00580       }
00581       free(spawncmd);
00582    }
00583    if (debug_flag)
00584    {
00585     int i_arg;
00586
00587       printf ("Spawning %s ", argv[0]);
00588       for (i_arg = 0; argv[i_arg] != NULL; i_arg++)
00589          printf ("%s ",argv[i_arg]);
00590       printf ("PID %d\n",shell_function_pid);
00591    }
00592 #endif /* VM */
00593 ;  {
```

Figure 62. Modifying function.c (part 2 of 3)

```
00605 #ifndef  __OPEN_VM
00606 ;    /* Record the PID for reap_children.  */
00607 ;    shell_function_pid = pid;
00608 #endif /* !VM */
00609 #ifndef  __MSDOS__
00610 #ifndef  __OPEN_VM
00611 ;    shell_function_completed = 0;
00612
00613 ;    /* Free the storage only the child needed.  */
00614 ;    free (argv[0]);
00615 ;    free ((char *) argv);
00616
00617 ;    /* Close the write side of the pipe.  */
00618 ;    (void) close (pipedes[1]);
00619 #endif /* VM */
00620 #endif
*****
*****
00642 #ifndef __OPEN_VM
00643 #ifdef EINTR
00644 ;    while (cc > 0 || errno == EINTR);
00645 #else
00646 ;    while (cc > 0);
00647 #endif
00648 #else /* VM */
00649 #ifdef EINTR
00650      while (cc == PIPBUFLEN || errno == EINTR);
00651 #else
00652      while (cc == PIPBUFLEN);
00653 #endif
00654 #endif /* !VM */
*****
*****
00661 #ifndef __OPEN_VM
00662 ;    (void) close (pipedes[0]);
00663 #endif /* !VM */
00664 #endif
```

Figure 63. Modifying function.c (part 3 of 3)

4.3.9.3 remote-cstms.c

The last instance where `fork`() was used was not modified because it was used only with remote procedure call, a feature we do not use.

4.3.10 Rebuilding GNU make

After all these changes were complete, we had to rebuild the GNU `make` package using the command `sh ./build.sh`. As you can see in Figure 64 on page 80, there were no more errors during compile and linkedit.

```
[porting - /home/porting/make-3.77]>
sh ./build.sh
compiling main.c...
compiling commands.c...
compiling job.c...
compiling dir.c...
compiling file.c...
compiling misc.c...
compiling read.c...
compiling remake.c...
compiling rule.c...
compiling implicit.c...
compiling default.c...
compiling variable.c...
compiling expand.c...
compiling function.c...
compiling vpath.c...
compiling version.c...
compiling ar.c...
compiling arscan.c...
compiling signame.c...
compiling getopt.c...
compiling getopt1.c...
compiling glob/fnmatch.c...
compiling glob/glob.c...
compiling remote-stub.c...
compiling getloadavg.c...
linking make...
done
```

Figure 64. Console log

4.3.11 Using make

We were now ready to start our new `make`.

As a simple test, we issued: ./`make --version`

Using `./` before `make` forces the shell to use the `make` available in the current directory.

```
[porting - /home/porting/make-3.77]>
./make --version
DMSABE141T Operation exception occurred at 80000002 in routine make
Enter DEBUG, VMDUMP, or BEGIN
DEBUG
DMSDBG989I The state of the virtual machine at time of ABEND follows:
Mode of virtual machine = XC PSW = 03080000 80000002
GPR  0 = 00000698 000000C5 00DAB294 00000000
GPR  4 = 00EC4A04 00000000 009E7394 009E737C
GPR  8 = 00000000 01F540F0 00EC4B18 80DAADE0
GPR 12 = 00E9D910 0160CC58 80DAAF56 0160C5C0
FPR  0 =  0000000000000000      00000000000000000 E 00
FPR  2 =  0000000000000000      00000000000000000 E 00
FPR  4 =  0000000000000000      00000000000000000 E 00
FPR  6 =  0000000000000000      00000000000000000 E 00
External old PSW       = 030E0000 8115B1B6
SVC old PSW            = 03081000 80DAB17C
Program old PSW        = 03080000 80000002
Machine check old PSW  = 00000000 00000000
Input/Output old PSW   = 030C0000 8115B1B6
AR   0 = 00000000 00000000 00000000 00000000
AR   4 = 00000000 00000000 00000000 00000000
AR   8 = 00000000 00000000 00000000 00000000
AR  12 = 00000000 00000000 00000000 00000000
Enter DEBUG, VMDUMP, or BEGIN
```

Figure 65. The first abend

Instead of make giving us back a version number we got an abend, as illustrated in Figure 65.

Do not try to get a VMDUMP. It is not necessary and helpful at this time.

4.3.12 Debugging make

To begin the debug process, we first had to restart our CMS environment with the following steps:

1. Get in CP mode
2. IPL CMS
3. Restart the shell environment

To debug make, we used the CP trace facility. To start the trace type in the shell environment, we issued: `cms trace prog` and reentered the command `./make --version`.

```
1   cms trace prog
    [porting - /home/porting/make-3.77]>
    ./make --version
     -> 00A4AF24  CLI   957E7000    4B619481    CC 0
    *** 00A4AF24    PROG    0005 -> 00F3E028        ADDRESSING
2   D G
    GPR  0 =  00000100  80A77BFA  80D754EA  80A4A8DE
    GPR  4 =  80E664B4  009EA040  009EE050  4B619481
    GPR  8 =  009EA028  0160CEC1  80D7542A  80E663D0
    GPR 12 =  00A2F910  00A001E0  80A4AEC6  009D4EE0
3   D TA4AF10.20
    R00A4AF10  70004400 C1ACD703 D140D140 4400C1AC E6 *....A.P.J J ..A.*
    R00A4AF20  5870D144 957E7000 47803676 47000000    *..J.n=.........*
4   D TA48000.100
    R00A48000  47000000 47000002 90ECD00C 053047F0 E6 *..............0*
    R00A48010  30180014 CE030103 00A4802C C3C5C5E2    *........u..CEES*
    R00A48020  E3C1D9E3 000058F0 306A050F 00A57220    *TART...0.....v..*
    R00A48030  00000000 00000000 00000000 00000000    *................*
    R00A48040  FFFF004C 00000000 00000000 00000000    *...<............*
    R00A48050  00000000 00000000 00000000 00000000    *................*
    R00A48060  00A48012 00000000 00000000 00000000    *.u.............*
    R00A48070  00000000 00ADE520 00ADE548 00000000    *......V...V.....*
    R00A48080  47F0F026 01C3C5C5 00000098 00002640    *.00..CEE...q... *
    R00A48090  47F0F001 183F58F0 C31C184E 05EF0000    *.00....0C..+....*
    R00A480A0  000047F0 303A90E8 D00C58E0 D04C4100    *...0...Y.....<..*
```

Figure 66. Using trace prog

Figure 66 notes:

1. With the command `CMS TRACE PROG`, CP stops whenever it detects a program exception and displays the instruction responsible for the exception. In this case we got an addressing exception at address A4AF24.
2. The command `D G` displays the general purpose registers.
3. The command `D TA4AF10.20` displays the contents of the storage around the abend address, which we noted was executable code.
4. We did a second display before A4AF24 and tried to find CEESTART as an eye catcher. CEESTART indicates the beginning of the C program. We found it at address A48000. The executable code starts at +80 (address A48080).

Using the displayed information to solve the problem, we could say:

The error was at address: A4AF24

We subtracted the origin address: A48000

The result was: 002F24 (actual offset in the `make` program)

We looked in the make.map shown in Figure 67 on page 83 to find what module was the culprit. The loadmap make.map was generated by the CMS GENMOD command.

Note: LDFLAGS in build.sh was set to '`-W b,l,map`' so that a load map of make was produced.

```
 make.map  V 100  Trunc=100 Size=292 Line=0 Col=1 Alt=0


00000 * * * Top of File * * *
00001  CEESTART SD 00010000          RMODE ANY AMODE ANY
00002  .000025  SD 00010080          RMODE ANY AMODE ANY
00003  MAIN        00012890
00004  LOG@WORK    0001D4E8
00005  DIE         0001D098
00006  CEEMAIN  SD 0001F220          RMODE ANY AMODE ANY
00007  .000026  SD 0001F230          RMODE ANY AMODE ANY
00008  FATAL@ER    000231D8
00009  CHOP@COM    00022200
00010  DELETE@C    00023EB8
00011  PRINT@CO    00024168
```

Figure 67. Reading make.map

According to Figure 67, CEESTART begins at 10000. This means the error address was: 10000+2F24= 12F24, which is in the MAIN routine at offset 2EA4 (12F24-10080=2EA4).

Next, we had a look in `main.lst` (see Figure 68) at address 2EA4 to find the problem.

```
 main.lst  V 133  Trunc=133 Size=14356 Line=4578 Col=1 Alt=0


04578                                        * int do_not_define;
04579                              00806 |   * register char *ep = envp[i]
i];
04580  002E7C  5880  D080         00806 |             L     r8,128(,r13)
04581  002E80  5870  D090         00806 |             L     r7,144(,r13)
04582  002E84  8970  0002         00806 |             SLL   r7,2
04583  002E88  5A70  8008         00806 |             A     r7,8(,r8)
04584  002E8C  D203  D144  7000   00806 |             MVC   324(4,r13),0(r7)
04585                                        *
04586                                        *       /* by default, everything
 gets defined and exported */
04587                              00809 |   *       do_not_define = 0;
04588  002E92  4400  C1AC         00809 |             EX    r0,HOOK..STMT
04589  002E96  D703  D140  D140   00809 |             XC    320(4,r13),320(r1
3)
04590                                        *
04591                              00811 |   *       while (*ep != '=')
04592  002E9C  4400  C1AC         00811 |             EX    r0,HOOK..STMT
04593  002EA0  5870  D144         00811 |             L     r7,324(,r13)
04594  002EA4  957E  7000         00811 |             CLI   0(r7),126
04595  002EA8  4780  ****         00811 |             BZ    @7L24
04596  002EAC  4700  0000         00811 |             NOP
04597  002EB0                     00811 |      @7L23  DS    0D
04598                              00812 |   *         ++ep;
04599  002EB0  4400  C1AC         00812 |             EX    r0,HOOK..STMT
```

Figure 68. Analyzing main.lst

The error was on line 4594. Apparently, this failed because `*ep` was not correctly set. `*ep` should get its value at line 4579 from `envp[i]`.

The reason *ep was not set is that main() in OpenEdition for VM/ESA does not support a third parameter. To circumvent this, we used an external environ array.

So, if we have code like the following:

```
main(int argc, char **argv, char **envp)
```

it can be changed to:

```
extern char **environ;
#define envp environ
main(int argc, char **argv)
```

4.3.12.1 Modifying main.c

To correct the error and make the code work, we modified main.c as shown in Figure 69.

```
 main.c  V 89  Trunc=89 Size=2453 Line=638 Col=1 Alt=4

00639 #if !defined(_AMIGA) && !defined(__OPEN_VM)
00640 int
00641 main (argc, argv, envp)
00642 int argc;
00643 char **argv;
00644 char **envp;
00645 #else
00646 int main (int argc, char ** argv)
00647 #endif /* AMIGA & VM */
*****
*****
00653   struct dep *read_makefiles;
00654   PATH_VAR (current_directory);
00655 #ifdef __OPEN_VM
00656    extern char **environ;
00657    #define envp environ
00658 #endif
00659 #ifdef WINDOWS32
```

Figure 69. Modifying main.c

We rebuilt the package (using command sh ./build.sh) and issued the command ./make --version again.

The result is shown in Figure 70.

```
[porting - /home/porting/make-3.77]>
./make --version
GNU Make version , by Richard Stallman and Roland McGrath.
Copyright (C) 1988, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98
        Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.

Report bugs to <bug-make@gnu.org>.

[porting - /home/porting/make-3.77]>
```

Figure 70. Output of make --version

4.3.13 Testing GNU make

`make` had passed its first test, but we had to do more extensive testing with this new `make` using the command:

```
./make check
```

```
./make check
/home/porting/make-3.77/./make check-recursive check-local
make[1]: Entering directory `/home/porting/make-3.77'
Making check in glob
make[2]: Entering directory `/home/porting/make-3.77/glob'
/home/porting/make-3.77/./make
make[3]: Entering directory `/home/porting/make-3.77/glob'
make[3]: Nothing to be done for `all'.
make[3]: Leaving directory `/home/porting/make-3.77/glob'
make[2]: Leaving directory `/home/porting/make-3.77/glob'
The system uptime program believes the load average to be:
uptime
1 uptime: GSU6039 uptime: not found
make[1]: [check-loadavg] Error 127 (ignored)
The GNU load average checking code believes:
./loadavg
Error getting load average: EDC5000I No error occurred.
make[1]: [check-loadavg] Error 1 (ignored)
/home/porting/make-3.77/./make all-recursive
make[2]: Entering directory `/home/porting/make-3.77'
Making all in glob
make[3]: Entering directory `/home/porting/make-3.77/glob'
make[3]: Nothing to be done for `all'.
make[3]: Leaving directory `/home/porting/make-3.77/glob'
make[2]: Leaving directory `/home/porting/make-3.77'
2 Couldn't find make-test-* regression test suite.
make[1]: Leaving directory `/home/porting/make-3.77'
[porting - /home/porting/make-3.77]>
```

Figure 71. Doing make check

We got two errors during `./make check`

Figure 71 notes:

1. uptime: we do not have this command with Open Edition for VM, so it could be ignored.
2. make-test-*: this file does not exist, so it could be ignored.

At this point, we could say that `GNU make` was ready to run.

4.4 Installing make

Feeling confident, we put the new `make` in production with the command:

```
./make install
```

```
[porting - /home/porting/make-3.77]>
./make install
Making install in glob
make[1]: Entering directory `/home/porting/make-3.77/glob'
make[1]: Nothing to be done for `install'.
make[1]: Leaving directory `/home/porting/make-3.77/glob'
/bin/sh ./mkinstalldirs /usr/local/bin
  ./install-sh -c make /usr/local/bin/make
/bin/sh ./mkinstalldirs /usr/local/info
 ./install-sh -c -m 644 ./make.info /usr/local/info/make.info
 ./install-sh -c -m 644 ./make.info-1 /usr/local/info/make.info-1
 ./install-sh -c -m 644 ./make.info-2 /usr/local/info/make.info-2
 ./install-sh -c -m 644 ./make.info-3 /usr/local/info/make.info-3
 ./install-sh -c -m 644 ./make.info-4 /usr/local/info/make.info-4
 ./install-sh -c -m 644 ./make.info-5 /usr/local/info/make.info-5
 ./install-sh -c -m 644 ./make.info-6 /usr/local/info/make.info-6
 ./install-sh -c -m 644 ./make.info-7 /usr/local/info/make.info-7
 ./install-sh -c -m 644 ./make.info-8 /usr/local/info/make.info-8
 ./install-sh -c -m 644 ./make.info-9 /usr/local/info/make.info-9
 ./install-sh -c -m 644 ./make.info-10 /usr/local/info/make.info-10
/home/porting/make-3.77/./make install-man1
make[1]: Entering directory `/home/porting/make-3.77'
/bin/sh ./mkinstalldirs /usr/local/man/man1
 ./install-sh -c -m 644 ./make.1 /usr/local/man/man1/make.1
make[1]: Leaving directory `/home/porting/make-3.77'
[porting - /home/porting/make-3.77]>
```

Figure 72. Doing make install

`make` is now ready to be used. The `path` statement in `/etc/profile` (see Figure 11 on page 24) sets the search order so that GNU `make` will be used whenever `make` is entered.

Chapter 5. Infrastructure packages

This section describes the packages that provide the infrastructure necessary for successfully building and running business applications. The packages provide support functions such as:

- Environmental support (for example, logging or automatic daemon activation)
- Program development tools
- Prerequisites for other applications

5.1 Standard PROFILE EXEC

We enrolled each of the daemons described in this book as a user in the SFSTEST filepool:

```
ENROLL USER <daemon> SFSTEST (Blocks 500
```

This filespace served as their A-disk area. Each of the daemons adopted the following standard PROFILE EXEC.

```
   /* Standard PROFILE EXEC */
1  'VMLINK TCPIP'
2  'CP TERM LINEDEL OFF ESCAPE OFF TABCHAR OFF CHARDEL OFF'
   'SET INPUT'
3  'SEGMENT LOAD CEEEV003 (SYSTEM'
4  'SEGMENT LOAD VMLIB (SYSTEM'
   'SET OUTPUT AD' 'BA'X
   'SET OUTPUT BD' 'BB'X
   'SET INPUT BA AD'
   'SET INPUT BB BD'
   'OPENVM MOUNT /../VMBFS:BFSTEST:ROOT/ /'
   'OPENVM SHELL'
   exit
```

Figure 73. Standard PROFILE EXEC.

Figure 73 notes:

1. To use TCP/IP services, we link to the production client disk (TCPMAINT 592).
2. Prevents CP terminal handling from interfering with our commands (for example @, ¢, ").
3. Load the LE/370 segment (CEEEV003).
4. Load the VMLIB segment (contains the non-multitasking routines).

5.2 SYSLOGD

Many, if not most, serious UNIX applications use a facility known as `syslog` to write and distribute messages. The actual logging of these messages can occur on the local system or a remote system; the logs can be sent to specific files, to the system operator, or to other users. In most of the applications described in this book you will see excerpts from the system log created by these applications.

`syslog` allows you to consolidate your consoles, just as PROP does under VM. With `syslog`, important messages from all your UNIX platforms can be consolidated on a VM/ESA system.

The ability of `syslog` to route messages of various types then allows you to distribute specific message types to people responsible for a given application/function. The message traffic can also be directed to the VM system operator for integration with PROP. Although we have not implemented it, it would be possible for an action routine to be invoked by PROP, on the receipt of specific messages, that would initiate a process to control the remote system.

It is also possible to provide the reverse of this process. That is, VM messages could be sent to a remote UNIX system for consolidation and action. This would involve providing PROP (or the NetView *NCCF service) with routines which used the `syslog` service.

5.2.1 SYSLOGD CP directory entry

```
   USER SYSLOGD DVM 32M 128M BDG
1   POSIXINFO UID 0 GNAME system
    SCR INA PIN STA YEL CPO TUR INR RED
2   ACCOUNT DAEMON SYSLOGD
3   OPTION QUICKDSP
    MACH XC
    SHARE RELATIVE 200 200
    IPL CMS PARM AUTOCR FILEPOOL SFSTEST
    CONSOLE 009 3215
    SPOOL   00C 2540 READER A
    SPOOL   00D 2540 PUNCH A
    SPOOL   00E 1403
    LINK MAINT    190 190 RR
    LINK MAINT    19D 19D RR
    LINK MAINT    19E 19E RR
```

Figure 74. SYSLOGD directory entry

Figure 74 notes:

1. SYSLOGD requires root authority.
2. For accounting records, we group our daemons under a single account.
3. For good performance we used the option quick dispatch.

5.2.2 SYSLOGD BFS requirements

We adopted a convention that the user is given its own filespace, which is mounted as its home directory. This is done with the following steps:

1. Enroll the user in the filepool:

   ```
   ENROLL USER SYSLOGD BFSTEST (BFS BLOCKS 500
   ```

2. Create a mount point by creating a mount external link:

   ```
   OPENVM CREATE EXTLINK /home/syslogd MOUNT /../VMBFS:BFSTEST:SYSLOGD/
   ```

5.2.3 SYSLOGD installation and configuration

Install and configure SYSLOGD using the following steps:

1. Follow the FTP instructions in Appendix K, “General instructions for downloading packages” on page 311.
2. Enter the shell and utilities environment.
3. Enter the SYSLOGD subdirectory:

   ```
   cd SYSLOGD
   ```

4. Install the package:

   ```
   make install
   ```

5. Create an /etc/syslog.conf file:

```
# Critical messages go to VM System Operator
*.crit                *OPERATOR
# Dump debug messages to console
*.debug               CON
# News notices go to VM Users MAINT and NEALE
news.notice           MAINT,NEALE
# Syslog debug messages can be logged to file
syslog.debug          syslog.log
# Non-syslog debug messages can be logged to file
*.debug               debug.log
# News and Mail debug messages can be logged to own file
news.debug,mail.debug newsmail.log
```

Figure 75. /etc/syslog.conf file as used during residency

6. Create an /etc/syslog.conf file for each UNIX box you want to consolidate.

   On the AIX platform we used in the lab, we inserted the following lines into the /etc/syslog.conf file it used:

```
*.debug         @totvm1.itso.ibm.com
*.crit          @totvm1.itso.ibm.com
*.debug         @totvm1.itso.ibm.com
```

Figure 76. /etc/syslog.conf file used on AIX platform

7. Logon to the SYSLOGD user.
8. Create a PROFILE EXEC as shown in 5.1, “Standard PROFILE EXEC” on page 87.
9. Enter the shell by executing the PROFILE EXEC.
10. Create .profile:

```
# Start SYSLOG daemon in background
syslogd &
```

Figure 77. .profile for SYSLOG daemon

11.Re-IPL CMS to start daemon:

```
IPL
CMS Level 14, RSU 9901  02/22/99
TCP/IP for VM linked RR as 120 file mode Z

IBM
Licensed Material - Property of IBM
5654-030 (C) Copyright IBM Corp. 1995
(C) Copyright Mortice Kern Systems, Inc., 1985, 1993.
(C) Copyright Software Development Group, University of Waterloo, 1989.

All Rights Reserved.

U.S. Government users - RESTRICTED RIGHTS - Use, Duplication, or
Disclosure restricted by GSA-ADP schedule contract with IBM Corp.

IBM is a registered trademark of the IBM Corp.

Monday April 12 01:45:50 PM EDT 1999

1 [1]     3528
[syslogd - /home/syslogd]>
SYSLOG Daemon starting for TOTVM1
SYSLOG server listening on port 514
SYSLOG server monitoring /dev/log
```

Figure 78. Starting the SYSLOG daemon

Figure 78 note:

1. [1] 3528 are the job and process identifiers respectively.

12.Check if the daemon is running using the `ps` (process status) command:

```
ps -ef
    UID       PID       PPID  STIME    TTY       TIME COMMAND
syslogd       4196          1 12:44:39 tty       0:00
syslogd       2997       4196 12:44:39 tty       0:00 sh -L
syslogd       3528       2997 12:44:39 tty       0:00 syslogd
syslogd       4950       2997 12:44:43 tty       0:00 ps -ef
```

13.On another user that has access to the TCP client disk (that is, has issued VMLINK TCPIP) check if the daemon is listening as it should:

```
netstat conn
VM TCP/IP Netstat Level 310

Active Transmission Blocks
User Id  Conn  Local Socket           Foreign Socket           State
-------- ----  ------------           --------------           -----
FTPSERVE 1026  *..FTP-C               *..*                     Listen
SMTP     1024  *..SMTP                *..*                     Listen
SMTP     UDP   *..1068                *..*                      UDP
EWEB003  1001  *..81                  *..*                     Listen
EWEB002  1002  *..81                  *..*                     Listen
EWEB001  1003  *..81                  *..*                     Listen
PORTMAP  UDP   *..PMAP                *..*                      UDP
PORTMAP  1006  *..PMAP                *..*                     Listen
VMNFS    UDP   *..2049                *..*                      UDP
SYSLOGD  UDP   *..514                 *..*                      UDP
```

Figure 79. Checking status of SYSLOG daemon

14.Within the shell on any user, test SYSLOG using the `syslogger` utility that was built with the SYSLOG daemon:

```
syslogger -is -p crit -t TEST Here is a test of a critical message
<10>Apr 15 15:33:05 TEST[5075]:Here is a test of a critical message
 02 Apr 15 15:33:05   NEALE   TEST[5075]:Here is a test of a critical message
```

Figure 80. Using syslogger to test SYSLOGD

15.From a platform that is forwarding its syslog traffic to your VM system:

```
[risc71:/etc]# logger -i -t RISC -p crit Test from RISC71

02 Apr 15 15:42:23   risc71.itso.ibm.com   Message forwarded from risc71
: RISC[4204]: Test from RISC71
```

Figure 81. Using logger on other platform to test SYSLOGD

16.Using XEDIT, check the /var/adm/debug.log file for messages:

```
Apr 15 15:24:14 NEWS nnrpd[6839]:donovan.endicott.ibm.com timeout
Apr 15 15:24:14 NEWS nnrpd[6839]:donovan.endicott.ibm.com exit articles 7
groups 3
Apr 15 15:24:14 NEWS nnrpd[6839]:donovan.endicott.ibm.com times user 0.433
system 0.033 elapsed 973.000
Apr 15 15:33:05 NEALE TEST[5075]:Here is a test of a critical message
Apr 15 15:31:00 risc71.itso.ibm.com Message forwarded from risc71: secd[7240]
: AS_REQ: ISSUE: authtime 924204660, host c2f8b0a6-c0fd-11d2-951a-02608c2d20
f5@ncacn_ip_tcp:9.12.0.50[42481], hosts/merlin.itso.ibm.com/self@itso.ibm.com
for krbtgt/itso.ibm.com@itso.ibm.com
Apr 15 15:38:05   risc71.itso.ibm.com   Message forwarded from risc71:
reboot: Rebooted by root.
Apr 15 15:38:05   risc71.itso.ibm.com   Message forwarded from risc71:
reboot: Rebooted by root.
```

Figure 82. Sample SYSLOGD output in /var/adm/debug.log

5.3 INETD - the Internet super daemon

This program invokes all Internet services as needed. Connection-oriented services are invoked each time a connection is made by creating a process (in this or another virtual machine). This process is passed the connection as file descriptor 0, and is expected to do a getpeername() to find out the source host and port.

Datagram-oriented services are invoked when a datagram arrives; a process is created and passed a pending message on file descriptor 0. Datagram servers can either connect to their peer, freeing up the original socket for `inetd` to receive further messages on, or take over the socket, processing all arriving datagrams and, eventually, timing out. The first type of server is said to be multi-threaded; the second type of server is single-threaded.

INETD uses a configuration file that is read at startup and, possibly, at some later time in response to a hangup signal. The configuration file is free format, with fields given in the order shown below. Continuation lines for an entry must begin with a space or tab. All fields must be present in each entry.

```
service name                 must be in ETC SERVICES
socket type                  stream/dgram
protocol                     must be in ETC PROTOCOL
wait/nowait                  single-/multi-threaded
user                         user to run daemon as
server program               full path name
server program arguments     maximum of MAXARGS (20)
```

Figure 83. Description of fields required in INETD configuration file

Comment lines are indicated by a '#' in column 1. A sample configuration file used during the residency is as follows:

```
chargen     stream tcp nowait root   internal
chargen     dgram  udp nowait root   internal
discard     stream tcp nowait root   internal
discard     dgram  udp nowait root   internal
echo        stream tcp nowait root   internal
echo        dgram  udp nowait root   internal
sift-uft    stream tcp nowait root   /home/neale/posixuft/uftd
daytime     stream tcp nowait root   daytimed -d -T %s
time        stream tcp nowait root   daytimed -t -T %s
imap2       stream tcp nowait root   /home/GCC/imap/imapd/imapd %s %s
```

Figure 84. INETD configuration file

The entries marked "internal" are functions provided by the INET daemon and are useful for testing IP connectivity:

```
1  pipe tcpclient totvm1 19 linger 5 | cons
    .<(+|&!$*);¬-/,%_>?`:#@'=" .<(+|&!$*);¬-/,%_>?`:#@'="

   .<(+|&!$*);¬-/,%_>?`:#@'=" .<(+|&!$*);¬-/,%_>?`:#@'="

2  pipe literal fsjgkj hdkjgh | tcpclient totvm1 9 linger 5 | cons
   Ready;

3  pipe literal Hello | tcpclient totvm1 7 linger 5| cons
   Hello
```

Figure 85. Invoking the internal clients of INETD

Figure 85 notes:

1. Chargen (port 19) generates lines of characters.
2. Discard (port 9) is analogous to the CMS Pipelines "hole" stage.
3. Echo (port 7) just does that: it echoes back what you sent.

5.3.1 INETD CP directory entry

To set up the INETD server we used the following CP directory entry:

```
   USER INETD DVM 32M 64M BDG 75
1   POSIXINFO UID 0 GNAME system
    SCR INA PIN STA YEL CPO TUR INR RED
2   OPTION QUICKDSP
3   ACCOUNT DAEMON INETD
    MACH XC
    SHARE RELATIVE 200 200
    IPL CMS PARM AUTOCR FILEPOOL SFSTEST
    IUCV ALLOW
    IUCV ANY
    CONSOLE 0009 3215
    SPOOL   000C 2540 READER A
    SPOOL   000D 2540 PUNCH A
    SPOOL   000E 1403
    LINK MAINT 0190 0190 RR
    LINK MAINT 019D 019D RR
    LINK MAINT 019E 019E RR
```

Figure 86. INETD directory entry

Figure 86 notes:

1. INETD requires root authority.
2. For good daemon performance we set quick dispatch on.
3. For accounting records we group our daemons under a single account.

5.3.2 INETD BFS requirements

Consistent with the convention we decided upon, the user is given its own filespace, which is mounted as its home directory. This is done with the following steps:

1. Enroll the user in the filepool:

   ```
   ENROLL USER INETD BFSTEST (BFS BLOCKS 500
   ```

2. Create a mount point by creating a mount external link:

   ```
   OPENVM CREATE EXTLINK /home/inetd MOUNT /../VMBFS:BFSTEST:INETD/
   ```

5.3.3 INETD installation and configuration

Install and configure INETD using the following steps:

1. Follow the FTP instructions in Appendix K, “General instructions for downloading packages” on page 311.
2. Enter the shell and utilities environment.
3. Enter the INETD subdirectory:

   ```
   cd INETD
   ```

4. Install the package:

   ```
   make install
   ```

5. Copy the inetd.conf template to the /etc directory:

   cp inetd.conf /etc

6. Edit the /etc/inetd.conf file:

   x /etc/inetd.conf

   Tailor so that the resulting file looks like:

```
1    chargen     stream tcp nowait root   internal
2    chargen     dgram  udp nowait root   internal
3    discard     stream tcp nowait root   internal
4    discard     dgram  udp nowait root   internal
5    echo        stream tcp nowait root   internal
6    echo        dgram  udp nowait root   internal
7    daytime     stream tcp nowait root   daytimed -d -T %s
8    time        stream tcp nowait root   daytimed -t -T %s
```

Figure 87. INET daemon configuration file

Note: Items 1-8 correspond with items in Figure 88 on page 95.

7. Logon to the INETD user and create a PROFILE EXEC as shown in 5.1, “Standard PROFILE EXEC” on page 87.

8. Use XEDIT to create a /home/inetd/.profile:

```
# Start the INET daemon
inetd &
```

9. Re-IPL CMS to start daemon.

10. Check if daemon is running:

```
ps -ef
     UID        PID       PPID    STIME TTY       TIME COMMAND
   inetd       3696          1 13:55:40 tty       0:00
   inetd       3199       3696 13:55:41 tty       0:00 sh -L
   inetd       5943       3199 13:57:17 tty       0:00 ps -ef
   inetd       6285          1 13:57:13 tty       0:00 inetd
```

11.On another user check if INETD is listening as it should:

```
netstat conn
VM TCP/IP Netstat Level 310

Active Transmission Blocks
User Id  Conn  Local Socket           Foreign Socket         State
-------  ----  ------------           --------------         -----
FTPSERVE 1026  *..FTP-C               *..*                   Listen
SMTP     1024  *..SMTP                *..*                   Listen
SMTP     UDP   *..1068                *..*                    UDP
EWEB003  1001  *..81                  *..*                   Listen
EWEB002  1002  *..81                  *..*                   Listen
EWEB001  1003  *..81                  *..*                   Listen
PORTMAP  UDP   *..PMAP                *..*                    UDP
PORTMAP  1006  *..PMAP                *..*                   Listen
VMNFS    UDP   *..2049                *..*                    UDP
SYSLOGD  UDP   *..514                 *..*                    UDP
1 INETD  1020  *..19                  *..*                   Listen
2 INETD  UDP   *..19                  *..*                    UDP
3 INETD  1039  *..9                   *..*                   Listen
4 INETD  UDP   *..9                   *..*                    UDP
5 INETD  1040  *..7                   *..*                   Listen
6 INETD  UDP   *..7                   *..*                    UDP
7 INETD  1041  *..13                  *..*                   Listen
8 INETD  1027  *..37                  *..*                   Listen
```

Figure 88. INET daemon status

Note: Items 1-8 match correspond with items in Figure 87 on page 94.

5.4 DAYTIMED - daytime and time daemon

You will note in Figure 87 on page 94, which defines the INET daemon services, that entries 7 and 8 define services known as “daytime” and “time”. These are services defined in Request for Comments RFC-867 and RFC-868 which provide date and time services.

There are a number of utilities available for PCs that will allow them to co-ordinate their clock with the time service.

5.4.1 Daytime daemon

The daytime daemon returns the date and time to the client as an ASCII character string.

5.4.1.1 TCP-Based

One daytime service is defined as a connection-based application on TCP. A server listens for TCP connections on TCP port 13. Once a connection is established, the current date and time is sent out as an ASCII character string (and any data received is thrown away). The service closes the connection after sending the quote.

5.4.1.2 UDP-Based

Another daytime service is defined as a datagram-based application on UDP. A server listens for UDP datagrams on UDP port 13. When a datagram is received,

an answering datagram is sent containing the current date and time as an ASCII character string (the data in the received datagram is ignored).

5.4.2 Time daemon

This daemon provides a site-independent, machine-readable date and time. The time service sends back to the originating source the time in seconds since midnight on 1st January, 1900.

One motivation for using this service arises from the fact that not all systems have a date/time clock, and all are subject to occasional human or machine error. The use of time-servers makes it possible to quickly confirm or correct a system's idea of the time, by making a brief poll of several independent sites on the network.

5.4.2.1 TCP-Based

When used via TCP the time service works as follows (`S` = Server; `U` = User):

```
S: Listen on port 37 (45 octal).
U: Connect to port 37.
S: Send the time as a 32 bit binary number.
U: Receive the time.
U: Close the connection.
S: Close the connection.
```

The server listens for a connection on port 37. When the connection is established, the server returns a 32-bit time value and closes the connection. If the server is unable to determine the time at its site, it should either refuse the connection or close it without sending anything.

5.4.2.2 UDP-Based

When used via UDP the time service works as follows:

```
S: Listen on port 37 (45 octal).
U: Send an empty datagram to port 37.
S: Receive the empty datagram.
S: Send a datagram containing the time as a 32 bit binary number.
U: Receive the time datagram.
```

The server listens for a datagram on port 37. When a datagram arrives, the server returns a datagram containing the 32-bit time value. If the server is unable to determine the time at its site, it should discard the arriving datagram and make no reply.

5.4.3 DAYTIMED installation

DAYTIME will be activated by the INETD server and thus does not require a separate virtual machine of its own (although there is no reason not to assign one). It will automatically be started when required.

1. Follow the FTP instructions in Appendix K, “General instructions for downloading packages” on page 311.
2. Change to the DAYTIME directory:

   ```
   cd DAYTIME
   ```

3. Install the binaries (see F.7, “DAYTIMED installation log” on page 258 for the detailed installation log):

   ```
   make install
   ```

4. Verify that the service is working:

   The first pipe will interrogate the data service of the Daytime daemon and return the time/date oin printable form. The second pipe interrogates the time services of the daemon and returns the printable hex version of the binary time that it receives in seconds.

```
pipe tcpclient totvm1 13 linger 30 | xlate from 437 to 1047 | console
Fri Apr 30 11:12:37 1999

pipe tcpclient totvm1 37 linger 30 | spec 1-* c2x 1 | console
BAD44713
```

Figure 89. Verifying the operation of the DAYTIME services

5.5 CROND

`crond` is a background daemon that parses individual crontab files and executes commands on behalf of the users in question.

`crond` is responsible for scanning the crontab files and running their commands at the appropriate time. The `crontab` utility manages a user's access to `cron` by copying, creating, listing, and removing crontab files. If invoked without options, `crontab` copies the specified file, or the standard input if no file is specified, into a directory that holds all users' crontabs.

A crontab file consists of lines of six fields each. The fields are separated by spaces or tabs. The first five are integer patterns that specify the following:

```
minute (0-59),
hour (0-23),
day of the month (1-31),
month of the year (1-12),
day of the week (0-6 with 0=Sunday).
```

Each of these patterns may be either an asterisk (meaning all legal values) or a list of elements separated by commas.

An element is either a number or two numbers separated by a minus sign (meaning an inclusive range). Note that the specification of days may be made by two fields (day of the month and day of the week). Both are adhered to if specified as a list of elements. The sixth field of a line in a crontab file is a string that is executed by the shell at the specified times.

Some examples include:

1. Clean up dump files every weekday morning at 3:15 am:

   ```
   15 3 * * 1-5 find $HOME -name CEEDUMP* 2>/dev/null |  xargs rm -f
   ```

2. Send a birthday greeting:

   ```
   0 12 14 2 * cms msg JOHN Happy Birthday! Time for lunch.
   ```

3. As an example of specifying the two types of days:

   ```
   0 0 1,15 * 1
   ```

would run a command on the first and fifteenth of each month, as well as on every Monday. To specify days by only one field, the other field should be set to *, for example:

```
0 0 * * 1
```

would run a command only on Mondays.

The `crontab` program communicates with `crond` through the cron.update file, which resides in the crontabs directory, usually /var/spool/cron/crontabs.

This is accomplished by appending the filename of the modified or deleted crontab file to cron.update which `crond` then picks up to resynchronize or remove its internal representation of the file.

`crond` has a number of built-in limitations to reduce the chance of it being used incorrectly. Potentially infinite loops during parsing are dealt with via a fail-safe counter, and user crontabs are generally limited to 256 crontab entries. Crontab lines may not be longer than 1024 characters, including the newline.

Whenever `crond` must run a job, it first creates a daemon-owned temporary file that is opened in update mode for its exclusive use (O_EXCL and O_APPENDattributes) to store any output, then changes its user and group permissions to match that of the user the job is being run for, then spawns `bin/sh -c` to run the job. The temporary file remains under the ownership of the daemon to prevent the user from tampering with it. Upon job completion, `crond` verifies the secureness of this temporary file and, if it has been appended to, mails the file to the user.

Unlike `crontab`, the `crond` program does not leave an open descriptor to the file for the duration of the job's execution, as this might cause `crond` to run out of descriptors. When the `crontab` program allows a user to edit his crontab, it copies the crontab to a user-owned file before running the user's preferred editor. The suid `crontab` program keeps an open descriptor to the file, which it later uses to copy the file back, thereby ensuring the user has not tampered with the file type.

`crond` always synchronizes to the top of the minute, checking the current time against the list of possible jobs. The list is stored such that the scan goes very quickly, and `crond` can deal with several thousand entries without taking any noticeable amount of CPU time.

5.5.1 CROND CP directory entry

The following CROND CP directory entry was used during our porting project:

```
   USER CROND TOTVM1PW 32M 64M G
1    ACCOUNT DAEMON CROND
     IPL CMS PARM AUTOCR FILEPOOL SFSTEST
     MACHINE XC
2    POSIXINFO UID 59 GNAME cron
     CONSOLE 0009 3215
     SPOOL 000C 3505 A
     SPOOL 000D 3525 A
     SPOOL 000E 1403 A
     LINK MAINT 0190 0190 RR
     LINK MAINT 019E 019E RR
     LINK MAINT 019F 019F RR
     LINK MAINT 019D 019D RR
```

Figure 90. CROND CP directory entry

Figure 90 notes:

1. This server is in the same accounting group as the others.
2. CROND does not require root privileges and has its own group.

5.5.2 CROND BFS requirements

Consistent with the convention we decided upon, the user is given its own filespace, which is mounted as its home directory:

1. Enroll the user in the filepool:

   ```
   ENROLL USER CROND BFSTEST (BFS BLOCKS 500
   ```

2. Create a mount point by creating a mount external link:

   ```
   OPENVM CREATE EXTLINK /home/crond MOUNT /../VMBFS:BFSTEST:CROND/
   ```

5.5.3 CROND installation and configuration

Install and configure CROND using the following steps:

1. Follow the FTP instructions in Appendix K, “General instructions for downloading packages” on page 311.
2. Enter the shell and utilities environment.
3. Enter the CROND subdirectory:

   ```
   cd crond
   ```

4. Install the package:

   ```
   make install
   ```

5. Give ownership of the home directory to crond:

   ```
   chown -R crond:cron /home/crond
   chown -R crond:cron /../VMBFS:BFSTEST:CROND/
   ```

6. Make a directory in which to place the crontab files and give ownership to crond:

   ```
   mkdir -p /var/spool/cron/crontabs
   chown -R crond:cron /var/spool/cron
   ```

7. Logon to the user CROND and create a PROFILE EXEC as shown in 5.1, “Standard PROFILE EXEC” on page 87.
8. Re-IPL CMS to enter the shell.
9. Create a /home/crond/.profile:

```
# Start the CRON daemon
crond -l8 >>/var/spool/cron/cron.log 2>&1 &
```

Figure 91. .profile for CROND

10. Re-IPL CMS to invoke the `crond` daemon.
11. On this user, use the `crontab` program to create a personal crontab with the following two lines:

```
crontab -
* * * * *  date >>/tmp/test
* * * * *  date
¢D
```

12. Check the log output of `crond` to ensure the cron entries are being run once a minute; check /tmp/test to ensure the date is being appended to it once a minute.

```
cat /var/spool/cron/cron.log
30-Apr-99 12:13  crond V2.4 dillon, started
30-Apr-99 12:16  USER crond pid 12423 cmd date
30-Apr-99 12:16  USER crond pid 12072 cmd date >>/tmp/test
30-Apr-99 12:17  USER crond pid 12073 cmd date
30-Apr-99 12:17  USER crond pid 12424 cmd date >>/tmp/test
30-Apr-99 12:18  USER crond pid 12425 cmd date
30-Apr-99 12:18  USER crond pid 12074 cmd date >>/tmp/test

cat /tmp/test
Fri Apr 30 12:16:01 EDT 1999
Fri Apr 30 12:17:02 EDT 1999
Fri Apr 30 12:18:02 EDT 1999
```

After you are through testing cron, delete the entries with `crontab -e` or `crontab -d` . (Note `crontab -e` will bring up an XEDIT session for the file /var/spool/cron/crontabs/<userid>.)

5.6 REGINA (REXX for UNIX)

For anyone brought up on a diet of REXX, shell scripts are almost indigestible. We could live with Perl but we would always choose REXX if it were available. CMS REXX is usable, to a degree, under OpenEdition (that is, it can be invoked and you can use the OPENVM calls to do a lot of the work). However, nothing quite beats a native implementation.

5.6.1 Regina prerequisite tools

Regina builds C source code from syntactic and lexical definition files. These definition files require special tools for processing into C. Download and install the following tools according to Appendix K, “General instructions for downloading packages” on page 311:

5.6.1.1 BYACC

This is a yacc replacement that provides better code than the yacc supplied with the Shell and Utilities. For more information about the supplied yacc see *OpenEdition for VM/ESA Advanced Application Programming Tools Version 2 Release 1.0*, SC24-5729. Berkeley Yacc is an LALR(1) parser generator. Berkeley Yacc has been made as compatible as possible with AT&T Yacc. Berkeley Yacc can accept any input specification that conforms to the AT&T Yacc documentation. Specifications that take advantage of undocumented features of AT&T Yacc will probably be rejected.

To install:

1. Follow the FTP instructions in Appendix K, “General instructions for downloading packages” on page 311.
2. Enter the Shell environment.
3. Enter the following command before make install:

   ```
   cp /make-3.74/install-sh/urs/local/bin/install-sh
   ```

4. Change to the byacc directory:

   ```
   cd byacc
   ```

5. Install the package:

   ```
   make install
   Installing yacc in /usr/local/bin
   ```

5.6.1.2 FLEX

Flex is a superior tool that replaces the lex command which comes with the Shell and Utilities. Flex is a tool for generating programs that recognizes lexical patterns in text. It reads the given input files, or its standard input if no file names are given, for a description of a scanner to generate. The description is in the form of pairs of regular expressions and C code, and generates as output a C source file, which defines a routine. This file is compiled and linked with the library to produce an executable. When the executable is run, it analyzes its input for occurrences of the regular expressions. Whenever it finds one, it executes the corresponding C code.

To install:

1. Follow the FTP instructions in Appendix K, “General instructions for downloading packages” on page 311.
2. Change to the flex directory:

   ```
   cd flex-2.5.4
   ```

3. Install the package (see F.9, “FLEX installation log” on page 258 for a detailed log of the installation):

   ```
   make install
   ```

5.6.2 Regina installation

See Appendix A, “Step-by-Step guide to porting Regina” on page 181 for a complete description of how Regina was ported to VM/ESA.

To install the binaries:

1. Follow the FTP instructions in Section K, “General instructions for downloading packages” on page 311.
2. Enter the Regina subdirectory:

   ```
   cd Regina-0.08c
   ```

3. Install the package:

   ```
   make install
   ```

5.6.3 Using Regina

Using the REXXCPS EXEC to benchmark, we have obtained the following results (your results may vary):

1. Regina - in debug mode and unoptimized:

```
----- REXXCPS 2.1 -- Measuring REXX clauses/second -----
 REXX version is: REXX-Regina_0_08c 4.50 23 Jul 1997
       System is: UNIX
       Averaging: 5 measures of 3 iterations

     Performance: 33300 REXX clauses per second
```

Figure 92. Regina performance: Debug mode and unoptimized

2. Regina - not in debug mode and unoptimized:

```
----- REXXCPS 2.1 -- Measuring REXX clauses/second -----
 REXX version is: REXX-Regina_0_08c 4.50 23 Jul 1997
       System is: UNIX
       Averaging: 5 measures of 3 iterations

     Performance: 73097 REXX clauses per second
```

Figure 93. Regina performance: No debug mode and unoptimized.

3. After turning on all the turbo options and turning off debug:

```
----- REXXCPS 2.1 -- Measuring REXX clauses/second -----
 REXX version is: REXX-Regina_0_08c 4.50 23 Jul 1997
       System is: UNIX
       Averaging: 5 measures of 3 iterations

     Performance: 91070 REXX clauses per second
```

Figure 94. Regina performance: Debug mode off and optimized

4. Native CMS

```
----- REXXCPS 2.1 -- Measuring REXX clauses/second -----
 REXX version is: REXX370 4.01 20 Dec 1996
       System is: CMS
       Averaging: 5 measures of 3 iterations

     Performance: 93981 REXX clauses per second
```

Figure 95. REXX performance: Native CMS

5. Compiled REXX:

```
----- REXXCPS 2.1 -- Measuring REXX clauses/second -----
 REXX version is: REXXC370 4.00 27 Oct 1994
       System is: CMS
       Averaging: 5 measures of 3 iterations

     Performance: 442739 REXX clauses per second
```

Figure 96. REXX performance: Compiled REXX

Having REXX available under OpenEdition has paid immediate dividends. The SYSLOG daemon produces copious amounts of data over the course of several days. Rather than running out of BFS space or having to continually add more, or manually do things to the log files, we have automated the procedure using REXX. In the .profile of the SYSLOG user are the following:

```
# start the INET daemon
rexx archive.rexx
syslogd &
```

Figure 97. New .profile for INET invoking Regina archiver

This invokes the following procedure, which archives any large log files and clears them out. The alternative was to use `cron` (see 5.5, “CROND” on page 97) to invoke this procedure at defined intervals.

```
/* Archive large log files */
Log.1 = 'debug'; Log.2 = 'newsmail'
Log.3 = 'syslog'
Log.0 = 3
do I_Log = 1 to Log.0
   call ARCHIVE Log.I_Log
end
exit

ARCHIVE:
   parse arg LogName
   say 'Checking' LogName 'log file size'
   X = 'ls'('-l /var/adm/'LogName'.log')
   parse var X . . . . Size .
   Size = Size + 0
   if Size > 1048576 then
   do
      say 'mv /var/adm/'LogName'.log /var/adm/'LogName'.archive'
      'cp /dev/null /var/adm/'LogName'.log'
   end
return
```

Figure 98. archive.rexx will archive SYSLOG files

5.7 RCS

The Revision Control System (RCS) manages multiple revisions of files. RCS automates the storing, retrieval, logging, identification, and merging of revisions.

RCS is useful for text that is revised frequently, for example programs, documentation, graphics, papers, and form letters.

When the authors or contributors make updates to packages, they are usually distributed in the form of diff files. Facilities such as RCS or SCCS can take these diff files and apply them against the source code to produce an updated version, much in the same way that CMS uses update files.

The basic user interface is extremely simple. The novice only needs to learn two commands: `ci` and `co`. The command `ci`, short for “check in”, deposits the contents of a file into an archival file called an RCS file. An RCS file contains all revisions of a particular file. `co`, short for “check out”, retrieves revisions from an RCS file.

5.7.1 Functions of RSCS

The functions of RCS are as follows:

- Store and retrieve multiple revisions of text: RCS saves all old revisions in a space-efficient way. Changes no longer destroy the original, because the previous revisions remain accessible. Revisions can be retrieved according to ranges of revision numbers, symbolic names, dates, authors, and states.
- Maintain a complete history of changes: RCS logs all changes automatically. Besides the text of each revision, RCS stores the author, the date and time of check-in, and a log message summarizing the change. The logging makes it easy to find out what happened to a module, without having to compare source listings or track down colleagues.
- Resolve access conflicts: When two or more programmers wish to modify the same revision, RCS alerts the programmers and prevents one modification from corrupting the other.
- Maintain a tree of revisions: RCS can maintain separate lines of development for each module. It stores a tree structure that represents the ancestral relationships among revisions.
- Merge revisions and resolve conflicts: Two separate lines of development of a module can be coalesced by merging. If the revisions to be merged affect the same sections of code, RCS alerts the user about the overlapping changes.
- Control releases and configurations: Revisions can be assigned symbolic names and marked as configurations of modules that can be described simply and directly.
- Automatically identify each revision with name, revision number, creation time, author, and so forth. The identification is like a stamp that can be embedded at an appropriate place in the text of a revision. The identification makes it simple to determine which revisions of which modules make up a given configuration.
- Minimize secondary storage: RCS needs little extra space for the revisions (only the differences). If intermediate revisions are deleted, the corresponding deltas are compressed accordingly.

5.7.2 RCS installation

Install RCS using the following steps:

1. Download, unzip and unarchive according to the procedure described in Appendix K, “General instructions for downloading packages” on page 311.

2. Change to the RCS directory:

```
cd RCS/src
```

3. Install using the make command:

```
make install
./../mkinstalldirs /usr/bin
for p in ci co ident merge rcs rcsclean rcsdiff rcsmerge rlog; do \
  .././install-sh -c $p /usr/bin/$p; \
done
```

4. Check that things work:

```
ci -V
RCS version 5.7
```

5.7.3 GNU Diff installation

GNU diff is used by RCS to create “diff” files from the RCS repository. To install:

1. Download, unzip and unarchive according to the procedure described in Appendix K, “General instructions for downloading packages” on page 311.
2. Install using the make command:

```
make install
/bin/sh ./mkinstalldirs /usr/local/bin /usr/local/info
for p in cmp diff diff3 sdiff; do \
  ./install-sh -c $p /usr/local/bin/`echo $p | sed 's,x,x,'`; \
done
{ test -f diff.info || cd .; } && \
for f in diff.info*; do \
  ./install-sh -c -m 644 $f /usr/local/info/$f; \
done
```

3. Check that it works:

```
rcsdiff -V
RCS version 5.7
```

5.7.4 Getting started with RCS

Suppose you have a file f.c that you wish to put under control of RCS. If you have not already done so, make an RCS directory with the command:

```
mkdir  RCS
```

Then invoke the check-in command:

```
ci  f.c
```

This command creates an RCS file in the RCS directory, stores f.c into it as revision 1.1, and deletes f.c. It also asks you for a description. The description should be a synopsis of the contents of the file. All later check-in commands will ask you for a log entry, which should summarize the changes that you made.

Files in the RCS directory are called RCS files; the others are called working files. To get back the working file f.c in the previous example, use the checkout command:

```
co  f.c
```

This command extracts the latest revision from the RCS file and writes it into f.c. If you want to edit f.c, you must lock it as you check it out with the command:

```
co  -l  f.c
```

You can now edit f.c.

Suppose, after some editing, you want to know what changes you have made. The command:

```
rcsdiff  f.c
```

tells you the difference between the most recently checked-in version and the working file. You can check the file back in by invoking:

```
ci  f.c
```

This increments the revision number properly.

If `ci` complains with the message:

```
ci error: no lock set by your name
```

then you have tried to check in a file even though you did not lock it when you checked it out. Of course, it is too late now to do the checkout with locking, because another checkout would overwrite your modifications. Instead, invoke

```
rcs  -l  f.c
```

This command will lock the latest revision for you, unless somebody else got ahead of you already. In this case, you will have to negotiate with that person.

Locking assures that you, and only you, can check in the next update, and avoids nasty problems if several people work on the same file. Even if a revision is locked, it can still be checked out for reading, compiling, and the like. All that locking prevents is a check-in by anybody but the locker.

If your RCS file is private, that is, if you are the only person who is going to deposit revisions into it, strict locking is not needed and you can turn it off. If strict locking is turned off, the owner of the RCS file need not have a lock for check-in; all others still do. Turning strict locking off and on is done with the commands:

```
rcs  -U  f.c     and     rcs  -L  f.c
```

If you do not want to clutter your working directory with RCS files, create a subdirectory called RCS in your working directory, and move all your RCS files there. RCS commands will look first into that directory to find needed files. All the commands discussed above will still work, without any modification. (Actually, pairs of RCS and working files can be specified in three ways: (a) both are given, (b) only the working file is given, (c) only the RCS file is given. Both RCS and working files may have arbitrary path prefixes; RCS commands pair them up intelligently.)

To avoid the deletion of the working file during check-in (in case you want to continue editing or compiling), invoke

```
ci  -l  f.c     or     ci  -u  f.c
```

These commands check in f.c as usual, but perform an implicit checkout. The first form also locks the checked in revision, the second one does not. Thus, these

options save you one checkout operation. The first form is useful if you want to continue editing, the second one if you just want to read the file. Both update the identification markers in your working file (see 5.7.5, “Automatic identification” on page 107).

You can give `ci` the number you want assigned to a checked in revision. Assume all your revisions were numbered 1.1, 1.2, 1.3, etc., and you would like to start release 2. The command:

```
ci  -r2  f.c     or     ci  -r2.1  f.c
```

assigns the number 2.1 to the new revision. From then on, `ci` will number the subsequent revisions with 2.2, 2.3, etc. The corresponding `co` commands:

```
co  -r2  f.c     and     co  -r2.1  f.c
```

retrieve the latest revision numbered 2.x and the revision 2.1, respectively. A `co` command without a revision number selects the latest revision on the trunk, that is, the highest revision with a number consisting of two fields. Numbers with more than two fields are needed for branches. For example, to start a branch at revision 1.3, invoke

```
ci  -r1.3.1  f.c
```

This command starts a branch numbered 1 at revision 1.3, and assigns the number 1.3.1.1 to the new revision.

5.7.5 Automatic identification

RCS can put special strings for identification into your source and object code. To obtain such identification, place the marker:

```
$Id$
```

into your text, for instance inside a comment. RCS will replace this marker with a string of the form:

```
$Id: filename revision date time author state$
```

With such a marker on the first page of each module, you can always see with which revision you are working. RCS keeps the markers up to date automatically. To propagate the markers into your object code, simply put them into literal character strings. In C, this is done as follows:

```
static char rcsid[] = "$Id$";
```

The command ident extracts such markers from any file, even object code and dumps. Thus, ident lets you find out which revisions of which modules were used in a given program.

You may also find it useful to put the marker Log into your text, inside a comment. This marker accumulates the log messages that are requested during check-in. Thus, you can maintain the complete history of your file directly inside it. There are several additional identification markers.

5.8 Patch

`patch` will take a patch file containing any of the three forms of difference listing produced by the `diff` program and apply those differences to an original file, producing a patched version.

`patch` is useful for updating code with changes that persons external to your organization have made to source files.

By default, the patched version is put in place of the original, with the original file backed up to the same name with the extension ".orig" or "~" , or as specified by the -b switch.

`patch` will try to skip any leading garbage, apply the diff, and then skip any trailing garbage. Thus you could feed an article or message containing a diff listing to patch, and it should work.

If the entire diff is indented by a consistent amount, this will be taken into account.

With context diffs (see “diff” on page 27), and to a lesser extent with normal diffs, `patch` can detect when the line numbers mentioned in the patch are incorrect, and will attempt to find the correct place to apply each hunk of the patch.

5.8.1 Patch installation

Install `patch` using the following steps:

1. Download, unzip and unarchive according to the procedure described in Appendix K, “General instructions for downloading packages” on page 311.
2. Change to the patch directory:

   ```
   cd patch-2.0
   ```

3. Edit makefile and change line 25 to:

   ```
   mansrc=/usr/local/man
   ```

4. Install using the make command:

   ```
   make install
   ```

5. Verify the operation of patch:

   ```
   patch -v
   $Header: /cm/src_uci/usr/public/patch/RCS/patch.c,v 1.1 88/06/23 20:20:20
   sources Exp $
   Patch level: 12
   ```

5.8.2 Using patch

In the following example we will create a source file, place it under the control of RCS, check out a revision of the source file, make changes, create a "diff" file using the `rcsdiff` command, then send those diffs to someone who has a copy of the original file and who will use patch to update their copy.

1. Create the source file:

   ```
   x source.file

   patch will take a patch file containing any of the three forms
   of difference listing produced by the DUFF program and apply
   ```

```
those quick brown foxes to an original file.
```

2. Check in the file to RCS:

```
ci source.file
RCS/source.file,v  <--  source.file
enter description, terminated with single '.' or end of file:
NOTE: This is NOT the log message!
>>
Original
>>
.
initial revision: 1.1
done
```

3. Check out a revision of the file:

```
co source.file
RCS/source.file,v  -->  source.file
revision 1.1
done
```

4. Update the file:

```
x source.file
patch will take a patch file containing any of the three forms
of difference listing produced by the diff program and apply
those differences to an original file.

End of patch paragraph and example.
```

5. Use `rcsdiff` to find the differences between the original and updated version:

```
rcsdiff -wcb source.file >source.diff
===================================================================
RCS file: RCS/source.file,v
retrieving revision 1.1
diff -wcb -r1.1 source.file
*** /tmp/00005366.PO041499.SI095301.X0000205.Y0000000 Fri May 14 09:53:02
1999
--- source.file Fri May 14 09:52:43 1999
***************
*** 1,3 ****
  patch will take a patch file containing any of the three forms
! of difference listing produced by the DUFF program and apply
! those quick brown foxes to an original file.
--- 1,5 ----
  patch will take a patch file containing any of the three forms
! of difference listing produced by the diff program and apply
! those differences to an original file.
!
! End of patch paragraph and example.
```

6. Send the diffs to someone with the original file and use `patch` to update it:

```
patch -p source.file <source.diff
Hmm...  Looks like a new-style context diff to me...
The text leading up to this was:
--------------------------
|*** /tmp/00005366.PO041499.SI095301.X0000205.Y0000000 Fri May 14 09:53:02
1999
|--- source.file Fri May 14 09:52:43 1999
--------------------------
```

```
Patching file source.file using Plan A...
Hunk #1 succeeded at 1.
done
```

7. Look at the updated file:

```
patch will take a patch file containing any of the three forms
of difference listing produced by the diff program and apply
those differences to an original file.

End of patch paragraph and example.
```

Chapter 6. Configuring and running business applications

In this chapter we look at the applications that turn OpenEdition from something you can appreciate at a technical level into a workhorse for an organization. Several applications are examined in detail to see what they do and how they are installed, customized, and exploited.

6.1 Samba

The Samba software suite is a collection of programs that implements the Server Message Block (SMB) protocol for UNIX systems. This protocol is sometimes also referred to as the Common Internet File System (CIFS), LanManager or NetBIOS protocol.

6.1.1 What is SMB

This is a big question.

The very short answer is that it is the protocol by which a lot of PC-related machines share files and printers and other information such as lists of available files and printers. Operating systems that support this natively include Windows NT, OS/2, and Linux, and add on packages that achieve the same thing are available for DOS, Windows, VMS, UNIX of all kinds, MVS, and more. Apple Macs and some Web browsers can speak this protocol as well. Alternatives to SMB include Netware, NFS, Appletalk, Banyan Vines, and Decnet; many of these have advantages but none are both public specifications and widely implemented in desktop machines by default.

The Common Internet Filesystem (CIFS) is what the new SMB initiative is called. For details watch:

`http://samba.org/cifs`.

6.1.2 Why use SMB

There are several reasons why one might use SMB, including:

- Many people want to integrate their Microsoft- or IBM-style desktop machines with their UNIX or VMS servers.
- Others want to integrate their Microsoft servers with UNIX or VMS servers. This is a different problem than integrating desktop clients.
- Others want to replace protocols like NFS, DecNet and Novell NCP, especially when used with PCs.

6.1.3 What Samba can do

Here is a very short list of what Samba includes, and what it does. For many networks this can be simply summarized by “Samba provides a complete replacement for Windows NT, Warp, NFS or Netware servers.”

- A SMB server, to provide Windows NT and LAN Manager-style file and print services to SMB clients such as Windows 95, Warp Server, smbfs and others.
- A NetBIOS (rfc1001/1002) name server, which among other things gives browsing support. Samba can be the master browser on your LAN if you wish.

- An ftp-like SMB client so you can access PC resources (disks and printers) from UNIX, Netware and other operating systems.
- A tar extension to the client for backing up PCs.
- A limited command-line tool that supports some of the NT administrative functionality, which can be used on Samba, NT workstation and NT server.

For a much better overview refer to the user survey at the following web site:

`http://samba.org/samba`

6.1.4 Samba components

The Samba suite is made up of several components. Each component is described in a separate manual page. We strongly recommend that you read the documentation that comes with Samba and the manual pages of those components that you use.

smbd
The `smbd` daemon provides the file and print services to SMB clients, such as Windows 95/98, Windows NT, Windows for Workgroups or LanManager. The configuration file for this daemon is described in smb.conf.

nmbd
The `nmbd` daemon provides NetBIOS name serving and browsing support. The configuration file for this daemon is described in smb.conf.

smbclient
The `smbclient` program implements a simple ftp-like client. This is useful for accessing SMB shares on other compatible servers (such as Windows NT), and can also be used to allow a UNIX box to print to a printer attached to any SMB server (such as a PC running Windows NT).

testparm
The `testparm` utility allows you to test your smb.conf configuration file.

testprns
The `testprns` utility allows you to test the printers defined in your printcap file.

smbstatus
The `smbstatus` utility allows you to list current connections to the smbd server.

nmblookup
The `nmblookup` utility allows NetBIOS name queries to be made from the UNIX machine.

make_smbcodepage
The `make_smbcodepage` utility allows you to create SMB code page definition files for your smbd server.

smbpasswd
The `smbpasswd` utility allows you to change SMB-encrypted passwords on Samba and Windows NT servers.

6.1.5 Samba CP directory entry

```
1  USER SAMBA DVM 32M 64M BDG 75
2  POSIXINFO UID 0 GNAME system
3  POSIXOPT SETIDS ALLOW QUERYDB ALLOW EXEC_SETIDS ALLOW
   SCR INA PIN STA YEL CPO TUR INR RED
4  OPTION QUICKDSP
   MACH XC
   SHARE RELATIVE 200 200
5  ACCOUNT DAEMON SAMBA
   IPL CMS PARM AUTOCR FILEPOOL SFSTEST
   IUCV ALLOW
   IUCV ANY
   CONSOLE 0009 3215
   SPOOL   000C 2540 READER A
   SPOOL   000D 2540 PUNCH A
   SPOOL   000E 1403
   LINK MAINT    0190 0190 RR
   LINK MAINT    019D 019D RR
   LINK MAINT    019E 019E RR
```

Figure 99. Samba CP Directory Entry

Figure 99 notes:

1. Give Samba as much storage as you can afford. Samba uses the shared memory APIs (shmem) for better performance.
2. Samba requires root authority.
3. Samba needs to access the user/group database.
4. As a server, Samba gets preferential treatment.
5. We keep our daemons in a single accounting group.

6.1.6 Samba BFS requirements

Consistent with the convention we decided upon, the user is given its own filespace which is mounted as its home directory:

1. Enroll the user in the filepool:

   ```
   ENROLL USER SAMBA BFSTEST (BFS BLOCKS 500
   ```

2. Create a mount point by creating a mount external link:

   ```
   OPENVM CREATE EXTLINK /home/samba MOUNT /../VMBFS:BFSTEST:SAMBA/
   ```

6.1.7 Samba installation and configuration

Install and configure Samba using the following steps:

1. Follow the FTP instructions in Appendix K, “General instructions for downloading packages” on page 311.
2. Enter the shell environment.
3. Create a symbolic link to the Samba directory to make it simpler:

   ```
   ln -s samba-1.9.18p10 samba
   ```

4. Enter the Samba source directory subdirectory:

   ```
   cd samba/source
   ```

5. Install the package:

   ```
   make install
   ```

6. Logon to the SAMBA user.
7. Create a PROFILE EXEC a shown in Section 5.1, “Standard PROFILE EXEC” on page 87.
8. Re-IPL CMS to enter the shell.
9. Change to the main Samba directory:

   ```
   cd /usr/local/samba
   ```

10. As this is an EBCDIC platform, make the ASCII codepages file inaccessible by renaming it:

    ```
    mv lib/codepages lib/codepages.ASCII
    ```

11. Copy the sample Samba configuration file:

    ```
    cp /home/samba/examples/smb.conf.default lib/smb.conf
    ```

12. Edit the Samba configuration file as show in Section F.6.2, “Samba configuration file” on page 253.
13. Create an /etc/printcap file:

```
poke3130
pokd3130
```

Figure 100. /etc/printcap file used during residency

14. Create an INETD configuration file inetd.samba specifically for Samba:

```
netbios-ssn stream tcp nowait root   /usr/local/samba/bin/smbd %s
```

Figure 101. inetd.samba file used during residency

15. Create a /home/samba/.profile:

```
      # Start up Samba specific INETD and the NMB daemon
      PATH=/usr/local/samba/bin:$PATH
1     inetd inetd.samba &
2     nmbd -D &
```

Figure 102. .profile or INETD

Figure 102 notes:

1. `inetd` is told to read the file inetd.samba and run in the background. `inetd` will wait for connections for Samba and then spawn the `smbd` command.
2. The Samba name server (`nmbd`) is started as a daemon in the background.

6.1.8 Using Samba

Once Samba had been installed and customized, we could then attempt to access the resources. The LAN configuration we used is shown in the following illustration:

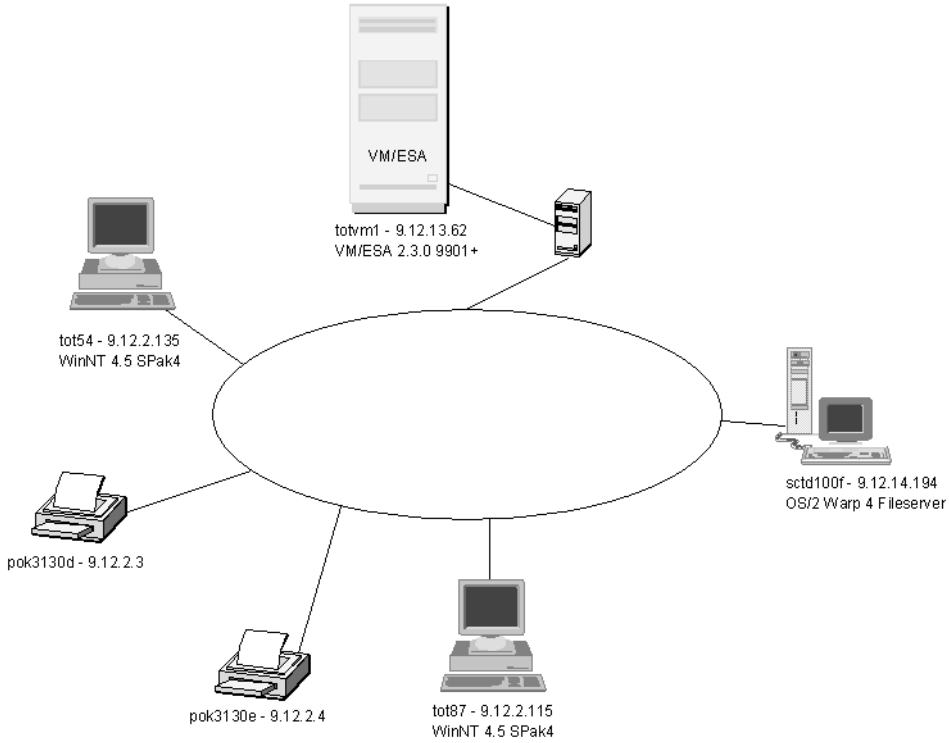


Figure 103. LAN configuration for VM/ESA Samba server

The Samba server was not in the same TCP/IP sub-net as the PC clients or the printers. This meant that the server was unable to broadcast its presence to the potential clients. However, we were able to tell Samba to announce itself to the Warp 4 File server using the “remote announce” keyword in the smb.conf file (see F.6.2 “Samba configuration file” on page 253).

To see if a PC client was able to locate our Samba server we used the “Find” application in WinNT to attempt to locate it.

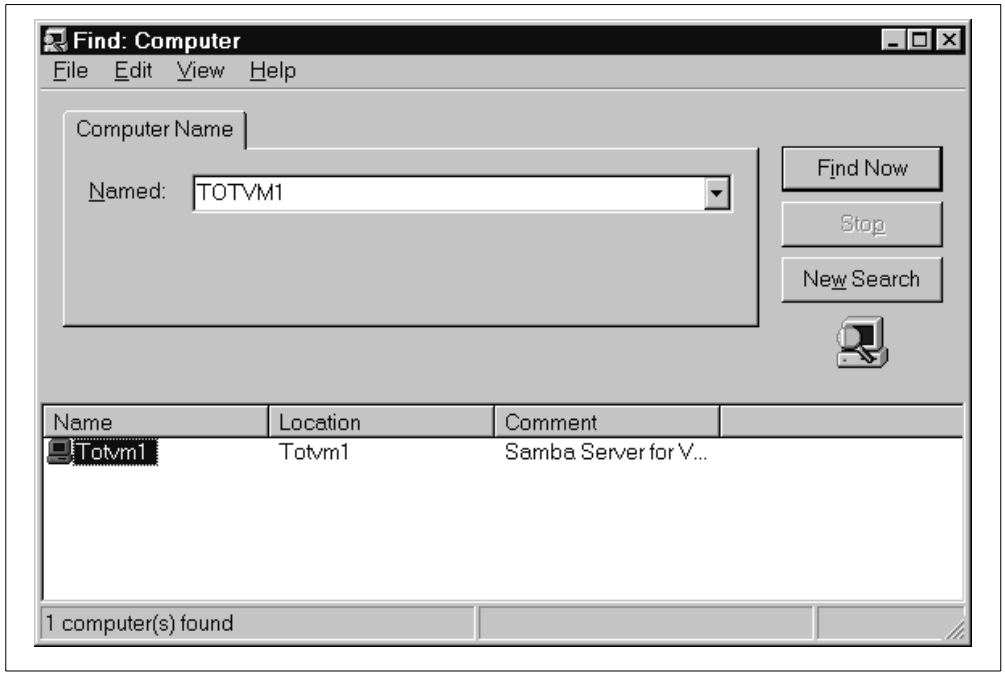


Figure 104. Using the FIND application to locate Samba

In fact, the PC client was able to find the server and you will note the “Comment” field of the preceding illustration shows the name we specified for the server in the smb.conf file.

By double-clicking on the icon in the “Find” window, the resources available for use on the Samba server are displayed.

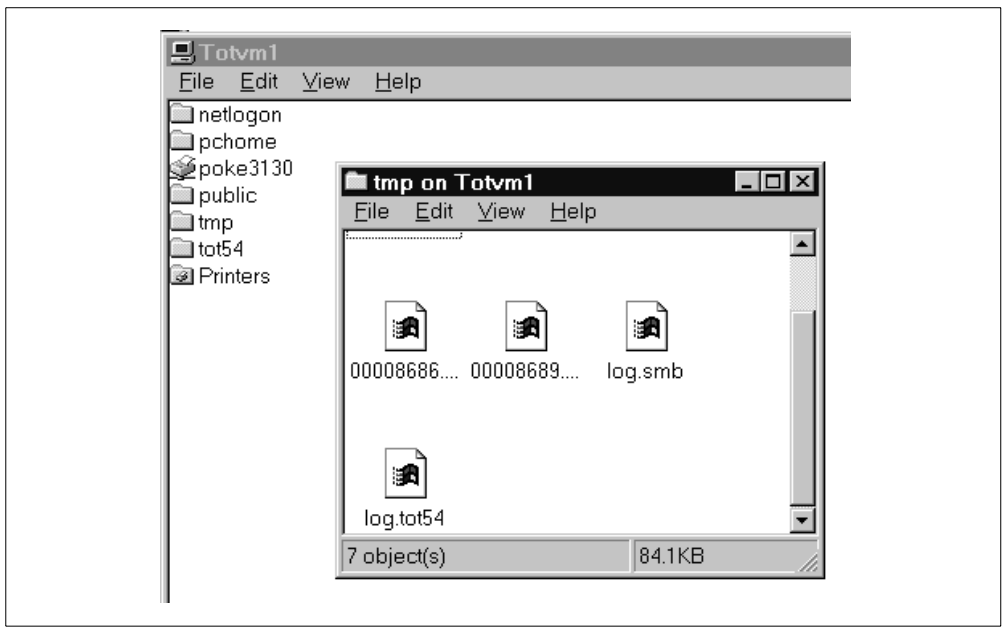


Figure 105. Examining resources on Samba

Note how each of the icons in the window in the background has a corresponding entry in the smb.conf file. In this instance we used the “tmp” folder:

```
[tmp]
   comment = Temporary file space
   path = /tmp
   read only = no
   public = yes
```

Next we examined the contents of one of the folders by double-clicking on the “tmp” folder. From this window we were able to manipulate the objects within the folder. In the following example we deleted a file.

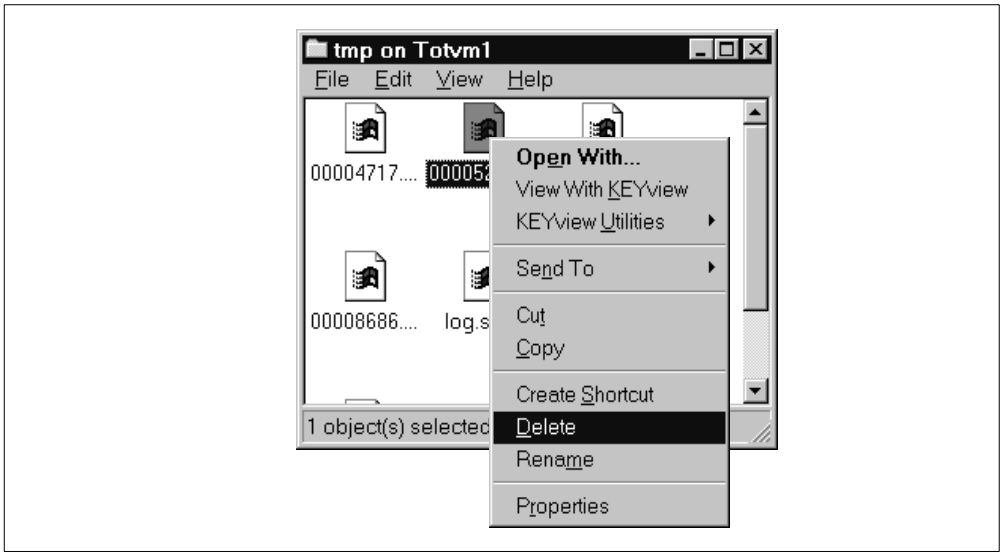


Figure 106. Using NT to delete a file on Samba

On the SAMBA user we can see what is going on by issuing the `smbstatus` command.

```
smbstatus

Samba version 1.9.18p10
Service   id      gid     pid     machine
----------------------------------------------
tmp       nobody  nobody  9635    tot54  (9.12.2.135) Tue May 04 15:56:43 1999

No locked files

Share mode memory usage (bytes):
   102088(99%) free + 256(0%) used + 56(0%) overhead = 102400(100%) total
```

Figure 107. Using smbstatus to display Samba activity

After deleting one file, we created another using the “Notepad” application of WinNT. Note the `dir` command issued on the VM system to display the file details.

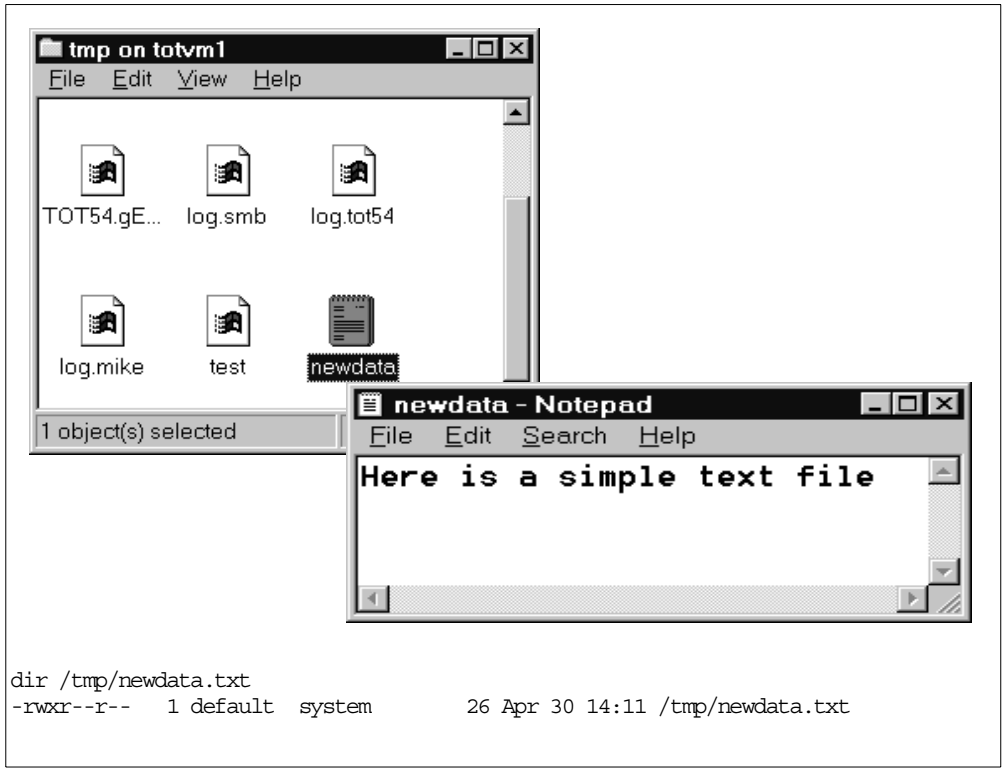


Figure 108. Creating a simple file within a Samba controlled folder

We then wanted to use the printer managed by the Samba server. We used Freelance to create a drawing and then asked it to print on the “poke3130” printer managed by our Samba server. As it was our first use of the printer, WinNT requested us to configure the printer.

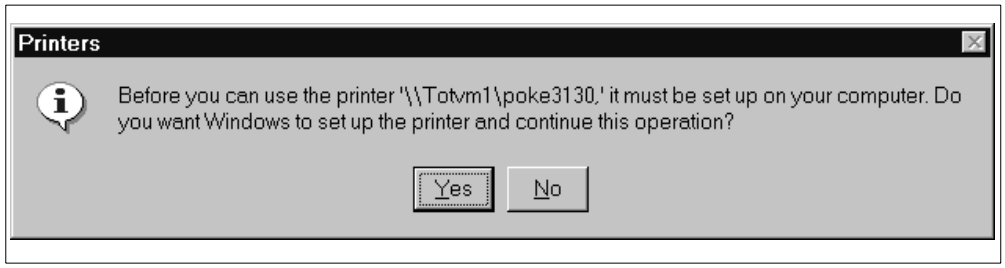


Figure 109. Accessing a Samba printer

We confirmed this by clicking the Yes button. WinNT then did some checking for us.

Figure 110. Connecting to Samba printer

We clicked Yes and WinNT installed the NULL driver for us. It then used the Printer Wizard to locate the correct printer driver for us.

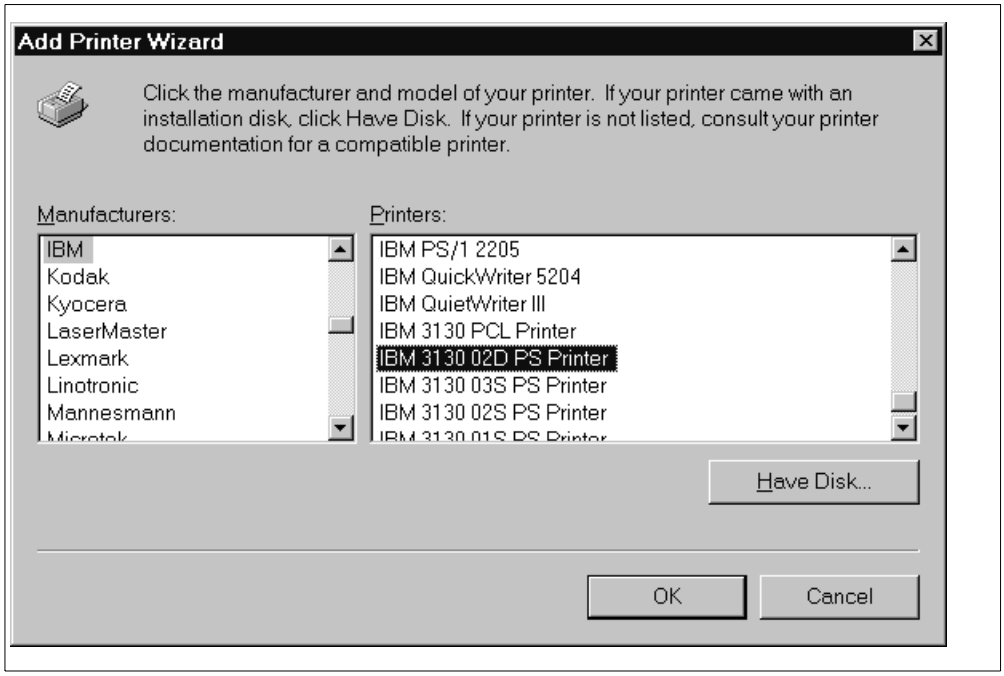


Figure 111. Configuring Samba printer

We selected the IBM 3130 PS Printer driver. WinNT then completed the install process for us and then allowed us to print the document.

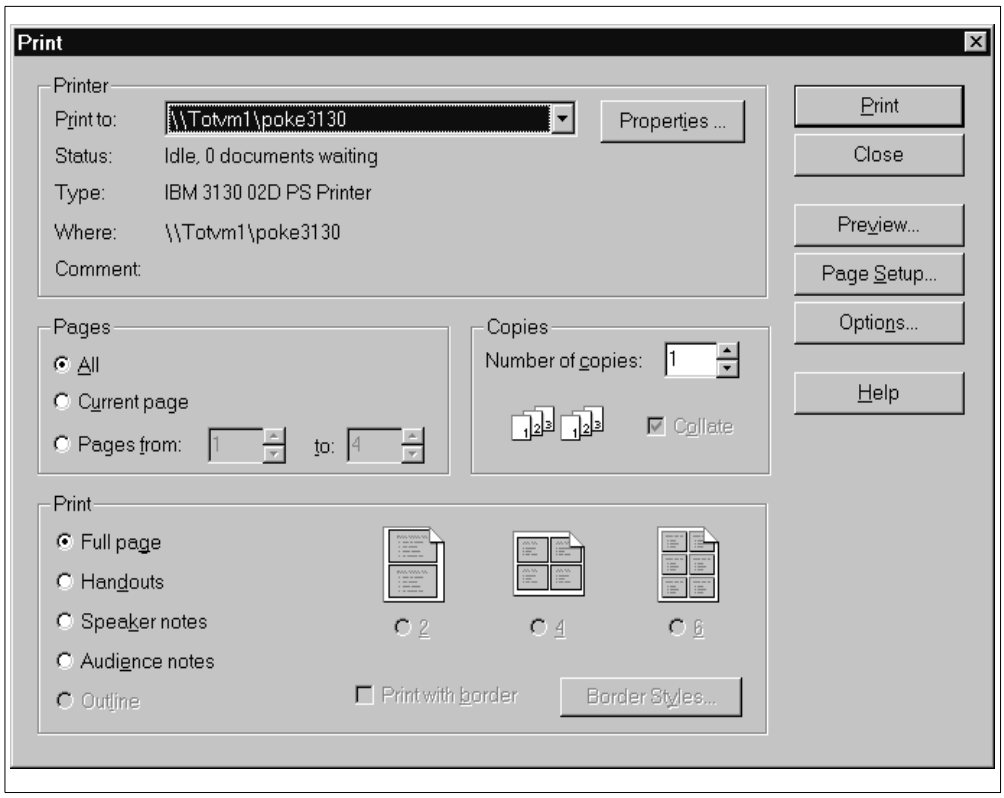


Figure 112. Printing to the Samba printer

WinNT then created the spool file and sent it to Samba for printing. Samba uses RSCS as its print server and forwarded the file to it via the `lp` command as we had configured in smb.conf:

```
[poke3130]
   comment = Atlantis Laser Printer
   path = /tmp
   printer = poke3130
   public = yes
   writable = no
   lpq command = lpq poke3130
   print command = lp -d .KOLP,A,P+ASCII %s
   printable = yes
```

The corresponding RSCS definition of the printer “kolp” can be seen in Appendix G.1, “RSCS configuration” on page 259. See *OpenEdition for VM/ESA Command Reference Version 2 Release 3.0*, SC24-5728 for a description of the `lp` command.

One of the shortcomings with our server was the lack of an `lpq` command that it could use to interrogate the status of the print job it sent to RSCS. In our configuration Samba simply assumes the job was sent immediately and without errors.

6.2 LDAP

The text of the following two sections is extracted from “The SLAPD and SLURPD Administrators’ Guide” which comes as part of the LDAP distribution. The entire document can be found in the doc/guides/guide.pdf subdirectory of the package.

6.2.1 What is a directory service

A directory is like a database, but tends to contain more descriptive, attribute-based information. The information in a directory is generally read much more often than it is written. As a consequence, directories don't usually implement the complicated transaction or roll-back schemes regular databases use for doing high-volume complex updates. Directory updates are typically simple all-or-nothing changes, if they are allowed at all. Directories are tuned to give quick response to high-volume lookup or search operations. They may have the ability to replicate information widely in order to increase availability and reliability, while reducing response time.

When directory information is replicated, temporary inconsistencies between the replicas may be OK, as long as they get in sync eventually. There are many different ways to provide a directory service. Different methods allow different kinds of information to be stored in the directory, place different requirements on how that information can be referenced, queried and updated, how it is protected from unauthorized access, and so forth. Some directory services are local, providing service to a restricted context (for example, the finger service on a single machine). Other services are global, providing service to a much broader context (such as the entire Internet). Global services are usually distributed, meaning that the data they contain is spread across many machines, all of which cooperate to provide the directory service. Typically, a global service defines a uniform namespace which gives the same view of the data no matter where you are in relation to the data itself.

6.2.2 What is LDAP

Slapd's model for directory service is based on a global directory model called LDAP, which stands for the Lightweight Directory Access Protocol. LDAP is a directory service protocol that runs over TCP/IP. The nitty-gritty details of LDAP are defined in RFC 1777 “The Lightweight Directory Access Protocol.” This section gives an overview of LDAP from a user's perspective.

What kind of information can be stored in the directory? The LDAP directory service model is based on entries. An entry is a collection of attributes that has a name, called a distinguished name (DN). The DN is used to refer to the entry unambiguously. Each of the entry’s attributes has a type and one or more values. The types are typically mnemonic strings, like “cn” for common name, or “mail” for E-mail address. The values depend on what type of attribute it is. For example, a mail attribute might contain the value “babs@umich.edu”. A jpegPhoto attribute would contain a photograph in binary JPEG/JFIF format.

How is the information arranged? In LDAP, directory entries are arranged in a hierarchical tree-like structure that reflects political, geographic and/or organizational boundaries. Entries representing countries appear at the top of the tree. Below them are entries representing states or national organizations. Below them might be entries representing people, organizational units, printers,

documents, or just about anything else you can think of. Figure 113 shows an example LDAP directory tree, which should help make things clear.

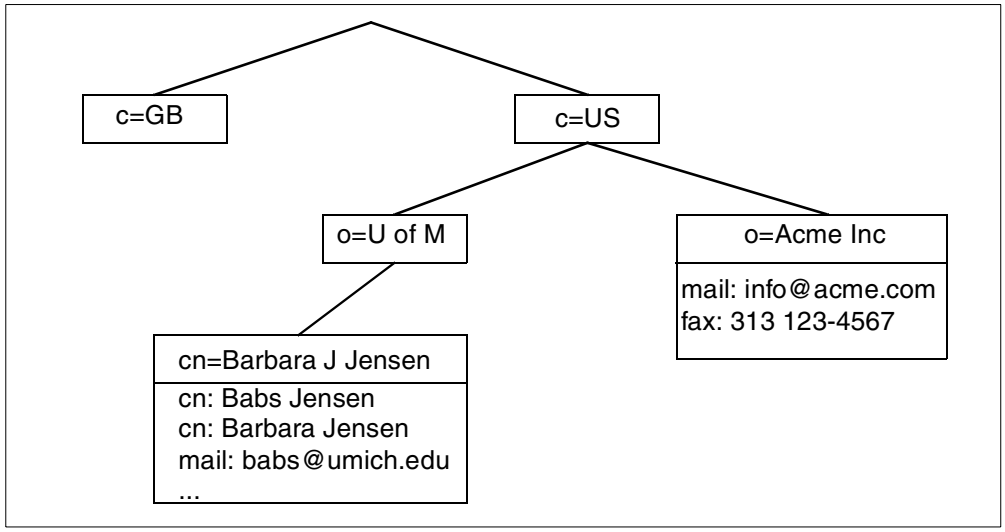


Figure 113. An example LDAP directory tree

In addition, LDAP allows you to control which attributes are required and allowed in an entry through the use of a special attribute called objectclass. The values of the objectclass attribute determine the schema rules the entry must obey.

How is the information referenced? An entry is referenced by its distinguished name, which is constructed by taking the name of the entry itself (called the relative distinguished name, or RDN) and concatenating the names of its ancestor entries.

For example, the entry for Barbara Jensen in the example above has an RDN of "cn=Barbara J Jensen" and a DN of "cn=Barbara J Jensen, o=U of M, c=US". The full DN format is described in RFC 1779, ":A String Representation of distinguished Names."

How is the information accessed? LDAP defines operations for interrogating and updating the directory. Operations are provided for adding and deleting an entry from the directory, changing an existing entry, and changing the name of an entry. Most of the time, though, LDAP is used to search for information in the directory. The LDAP search operation allows some portion of the directory to be searched for entries that match some criteria specified by a search filter. Information can be requested from each entry that matches the criteria. For example, you might want to search the entire directory subtree below the University of Michigan for people with the name Barbara Jensen, retrieving the E-mail address of each entry found. LDAP lets you do this easily. Or you might want to search the entries directly below the c=US entry for organizations with the string "Acme" in their name, and that have a fax number. LDAP lets you do this too.

How is the information protected from unauthorized access? Some directory services provide no protection, allowing anyone to see the information. LDAP provides a method for a client to authenticate, or prove its identity to a directory server, paving the way for rich access control to protect the information the server contains.

6.2.3 The OpenLDAP distribution

The package described in this section is the University of Michigan 3.3 distribution. The OpenLDAP organization has recently taken control of the public domain LDAP suite. The changes made for the 3.3 distribution have been fitted to the latest OpenLDAP package (1.3) and are being incorporated into the product. This means that all subsequent releases and modifications of the package will incorporate the VM/ESA changes and requirements.

6.2.4 LDAP CP directory entry

Following is the LDAP CP directory entry we used during the project.

```
   USER LDAP DVM 32M 64M DG 75
1    POSIXINFO UID 0 GNAME system
     SCR INA PIN STA YEL CPO TUR INR RED
2    OPTION QUICKDSP
     MACH XC
     SHARE RELATIVE 200 200
3    ACCOUNT TAB_TCP LDAP
     IPL CMS PARM AUTOCR FILEPOOL SFSTEST
     IUCV ALLOW
     IUCV ANY
     CONSOLE 0009 3215
     SPOOL   000C 2540 READER A
     SPOOL   000D 2540 PUNCH A
     SPOOL   000E 1403
     LINK MAINT    0190 0190 RR
     LINK MAINT    019D 019D RR
     LINK MAINT   019E 019E RR
```

Figure 114. LDAP CP Directory Entry

Figure 114 notes:

1. LDAP requires root authority.
2. For good daemon performance we set quick dispatch.
3. For accounting records we group our daemons under a single account.

6.2.5 LDAP BFS requirements

Consistent with the convention we decided upon, the user is given its own filespace, which is mounted as its home directory:

1. Enroll the user in the filepool and give it more than our default of 500 blocks so it can handle large LDIF files:

   ```
   ENROLL USER LDAP BFSTEST (BFS BLOCKS 1000
   ```

2. Create a mount point by creating a mount external link:

   ```
   OPENVM CREATE EXTLINK /home/ldap MOUNT /../VMBFS:BFSTEST:LDAP/
   ```

6.2.6 Install GDBM prerequisite

GNU dbm is a set of database routines that use extendible hashing and work similarly to the standard UNIX dbm routines. It is one of the database access methods supported by the LDAP package.

Install GDBM using the following steps:

1. Follow the FTP instructions in Appendix K, “General instructions for downloading packages” on page 311.
2. Enter the shell environment.
3. Change to the gdbm directory:

   ```
   cd gdbm-1.7.3
   ```

4. Verify that the directory `/usr/local/man/man3` exists.
5. Install the package:

   ```
   make install
   cp libgdbm.a /usr/local/lib/libgdbm.a
   cp gdbm.h /usr/local/lib/gdbm.h
   cp ./gdbm.3 /usr/local/man/man3/gdbm.3
   cp ./gdbm.info /usr/local/info/gdbm.info
   ```

6.2.7 LDAP installation

Install LDAP using the following steps:

1. Follow the FTP instructions in Appendix K, “General instructions for downloading packages” on page 311.
2. Enter VM/ESA build directory:

   ```
   cd ldap-3.3/build/platforms/openvm
   ```

3. Install using `make` (see F.3 “LDAP installation log” on page 228 for a detailed console log of the install process):

   ```
   make install
   ```

4. Create a directory for the database:

   ```
   mkdir -p /usr/tmp
   ```

6.2.8 LDAP configuration

Update the /usr/local/etc/slapd.conf file (the `;` is the tab character):

```
    include;;/usr/local/etc/slapd.at.conf
    include;;/usr/local/etc/slapd.oc.conf
    schemacheck;off
    referral;ldap://ldap.itd.umich.edu
    ########################################################################
    # ldbm database definitions
    ########################################################################
1   database;ldbm
2   suffix;;"o=ITSO,c=US"
3   directory;/usr/tmp
4   rootdn;;"cn=root, o=ITSO, c=US"
5   rootpw;;secret
```

Figure 115. slapd.conf - LDAP configuration file

Figure 115 notes:

1. We are using a database method to retrieve LDAP information (as opposed to shell scripts).
2. Our organization is known as ITSO and we are in the country USA (we used the 2 character ISO standard for the country code).
3. The database is located in the /usr/tmp directory.
4. The root user is able to modify the database online.
5. This is the password associated with this privileged user.

6.2.9 Building the database

6.2.9.1 Creating the LDIF data file

LDIF is short for the LDAP Data Interchange Format. LDIF is used to represent LDAP entries in text form. The tools can be used to convert from LDIF format to the LDBM format used by the LDAP server. The tool can also be used to do the reverse conversion. See the SLAPD and SLURPD Administrator's Guide (part of the package) for more information on this format and the conversion tools. The basic form of an LDIF entry is:

```
[<id>]
dn: <distinguished name>
<attrtype>: <attrvalue>
<attrtype>: <attrvalue>
...
```

where <id> is the optional entry ID (a positive decimal number).

Normally, you would not supply the <id>, allowing the database creation tools to do that for you. The program, however, produces an LDIF format that includes <id> so that new indexes created will be consistent with the existing database. A line may be continued by starting the next line with a single space or tab character, for example:

```
dn: cn=Barbara J Jensen, o=University of Michi
 gan, c=US
```

Multiple attribute values are specified on separate lines, for example:

```
cn: Barbara J Jensen
cn: Babs Jensen
```

If an <attrvalue> contains a non-printing character, or begins with a space or a colon ':', the <attrtype> is followed by a double colon and the value is encoded in base 64 notation. For example, the value "begins with a space" would be encoded like this:

```
cn:: IGJlZ2lucyB3aXRoIGEgc3BhY2U=
```

Multiple entries within the same LDIF file are separated by blank lines. Here is an example of an LDIF file containing three entries.

```
dn: cn=Barbara J Jensen, o=University of Michigan, c=US
cn: Barbara J Jensen
cn: Babs Jensen
objectclass: person
sn: Jensen

dn: cn=Bjorn J Jensen, o=University of Michigan, c=US
cn: Bjorn J Jensen
cn: Bjorn Jensen
objectclass: person
sn: Jensen

dn: cn=Jennifer J Jensen, o=University of Michigan, c=US
cn: Bjorn Jensen
objectclass: person
sn: Jensen

dn: cn=Jennifer J Jensen, o=University of Michigan, c=US
```

```
cn: Jennifer J Jensen
cn: Jennifer Jensen
objectclass: person
sn: Jensen
jpegPhoto:: /9j/4AAQSkZJRgABAAAAAQABAAD/2wBDABALD
A4MChAODQ4SERATGCgaGBYWGDEjJR0oOjM9PDkzODdASFxOQ
ERXRTc4UG1RV19iZ2hnPk1xeXBkeFxlZ2P/2wBDARESEhgVG
...
```

Notice that the jpegPhoto in Jennifer Jensen's entry is encoded using base 64.

To create the LDIF data file from which the LDAP database will be created, we wrote an EXEC that processed a CMS NAMES file which contained information about the organization, the divisions and sections within the organization, and the people who worked in each section. An excerpt from the names file follows.

```
:nick.DIVISION :list.ITSO VMVSE

:nick.ITSO     :list.VM Process_Services Administration EBusiness
                     IT_Operations S390_High_End_Servers
               :name.International Technical Support Organization

:nick.VMVSE    :list.OE RSCS TCPIP
               :name.VM/VSE Independent Business Unit

:nick.ADMINISTRATION
               :name.Poughkeepsie ITSO Administration

:nick.PROCESS_SERVICES
               :name.Poughkeepsie ITSO Process Services

:nick.S390_HIGH_END_SERVERS
               :name.S/390 High End Servers

:nick.NEALE    :userid.Neale    :node.totvm1.itso.ibm.com
               :name.Neale Ferguson
               :TYPE.Person
               :DIVISION.ITSO
               :SECTION.VM
               :TITLE.Systems Programmer
               :CITY.Poughkeepsie

:nick.GNORMAN  :userid.gnorman  :node.totvm1.itso.ibm.com
               :name.Gnorman Gnome
               :TYPE.Person
               :DIVISION.ITSO
               :SECTION.VM
               :TITLE.Marketing
               :CITY.Poughkeepsie
```

Figure 116. Excerpt from CMS NAMES file used as input to LDIF build EXEC

The resulting LDIF file looks like the following figure.

```
dn: cn=Gnorman Gnome (gnorman@totvm1.itso.ibm.com),o=ITSO,c=US
cn: Gnorman Gnome
givenname: Gnorman
sn: Gnome
uid: GNORMAN
labeleduri: http://totvm1.itso.ibm.com/~gnorman/
mail: gnorman@totvm1.itso.ibm.com
mailhost: totvm1.itso.ibm.com
telephonenumber: 555-0011
city: Poughkeepsie
nick: GNORMAN
title: Marketing
objectclass: person
ou: VM
ou: ITSO
organization: ITSO
```

Figure 117. Extract of LDIF file created by BLDLDIF EXEC.

Important

Be careful when making changes to this file via XEDIT. Null lines in the file are defined as consecutive newline characters (X'1515'). If you change this file and save it, XEDIT will convert these null lines to X'154015'. You will then need to run a pipe to fix the file.

6.2.9.2 Creating the GDBM database

To create a database usable by GDBM, the LDIF statements are converted into the GDBM representation via the `ldif2ldbm` command. To automate this procedure we can write a simple Regina EXEC.

```
    /* Create GDBM database from LDIF input */
1   Month.Jan = '01'; Month.Feb = '02'; Month.Mar = '03'; Month.Apr = '04'
    Month.May = '05'; Month.Jun = '06'; Month.Jul = '07'; Month.Aug = '08'
    Month.Sep = '09'; Month.Oct = '10'; Month.Nov = '11'; Month.Dec = '12'
2   x = 'ls'('-l itsoldif')
3   if Rc = 0 then
    do
4       parse var x . . . . . Month Day TTYY .
5       if INDEX(TTYY,':') > 0 then
            YYYY = LEFT(DATE('S'),4)
        else
            YYYY = TTYY
6       S_YYYYMMDD = YYYY || VALUE(Month.Month) || RIGHT(Day,2,'0')
        say S_YYYYMMDD
7       x = 'ls'('-l /usr/tmp/dn.gdbm')
        parse var x . . . . . Month Day TTYY .
        if INDEX(TTYY,':') > 0 then
            YYYY = LEFT(DATE('S'),4)
        else
            YYYY = TTYY
        D_YYYYMMDD = YYYY || VALUE(Month.Month) || RIGHT(Day,2,'0')
        say D_YYYYMMDD
8       if (S_YYYYMMDD > D_YYYYMMDD) then
        do
9           'rm /usr/tmp/*.gdbm'
            'rm /usr/tmp/NEXTID'
10          'ldif2ldbm -i itsoldif'
        end
    end
```

Figure 118. Regina EXEC that builds the GDBM database

Figure 118 notes:

1. Set up an array that will do month name to number conversion.
2. Check for the existence of and get details of the source file containing the ldif statements.
3. If the file exists, proceed.
4. Extract the month, day, and (possibly) year data.
5. If the year is in fact a timestamp, then default to current year.
6. Construct a datestamp from the directory information.
7. Get the directory details of one of the database files (construct datestamp as for steps 4-6).
8. If the source file is later than the database file.
9. Remove the existing database and id files.
10. Invoke the ldif-to-gdbm conversion utility.

6.2.10 Starting the LDAP server

Start the LDAP server using the following steps:

1. Logon to LDAP.
2. Create a PROFILE EXEC as shown in Section 5.1, “Standard PROFILE EXEC” on page 87.
3. Re-IPL CMS to enter the shell.

4. Create a /home/ldap/.profile.

```
# Build database if necessary and start LDAP server
rexx bldldif.rexx
slapd &
```

Figure 119. .profile for LDAP server

5. Create bldldif.rexx as shown in Figure 118 on page 128.
6. Re-IPL CMS to start the server.
7. Use the `getps` command (part of the XPG4 package) to check that the server is running:

```
getps
 pid       user     state        command
------ --------- ----------- --------------------
10276       ldap    Running
11806       ldap    Running       sh -L
 8932       ldap    Running       slapd
 7691       ldap    Running       getps
```

Figure 120. Use the `getps` command

6.2.11 Configuring an LDAP client

Once the LDAP server was up and running, we needed a client to interrogate the server. We used Netscape 4.51 in the following examples, but any LDAP-capable client would suffice.

We started the Address Book component of Netscape to define a new directory.

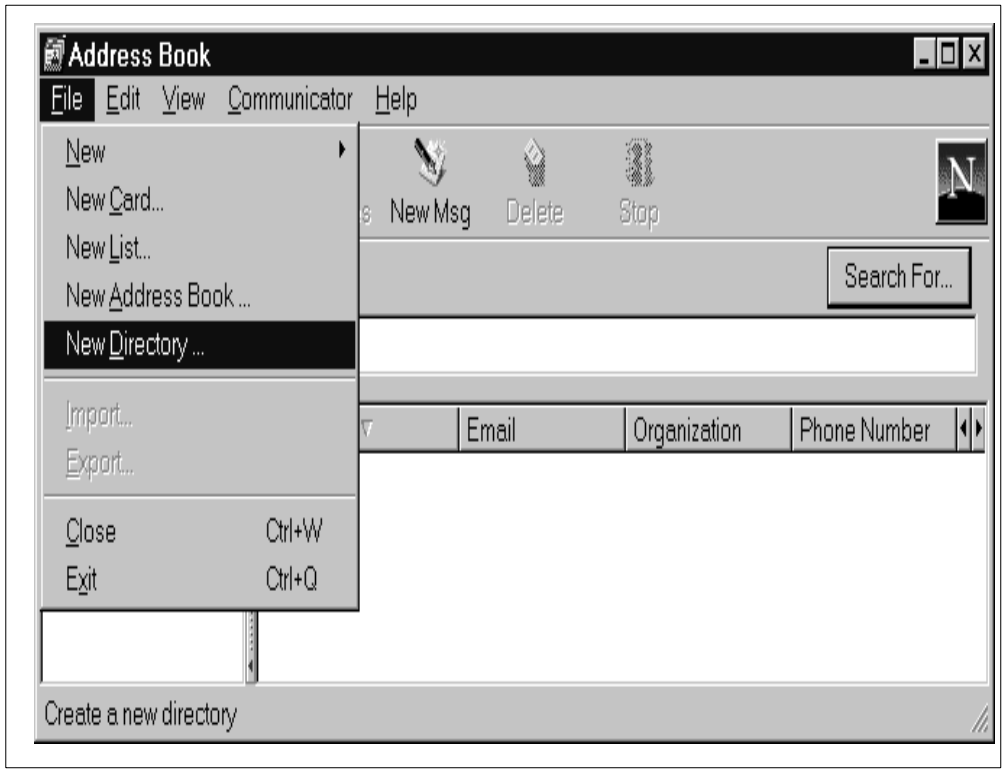


Figure 121. Configuring Netscape: Create a new directory

We then specified the properties of our server.

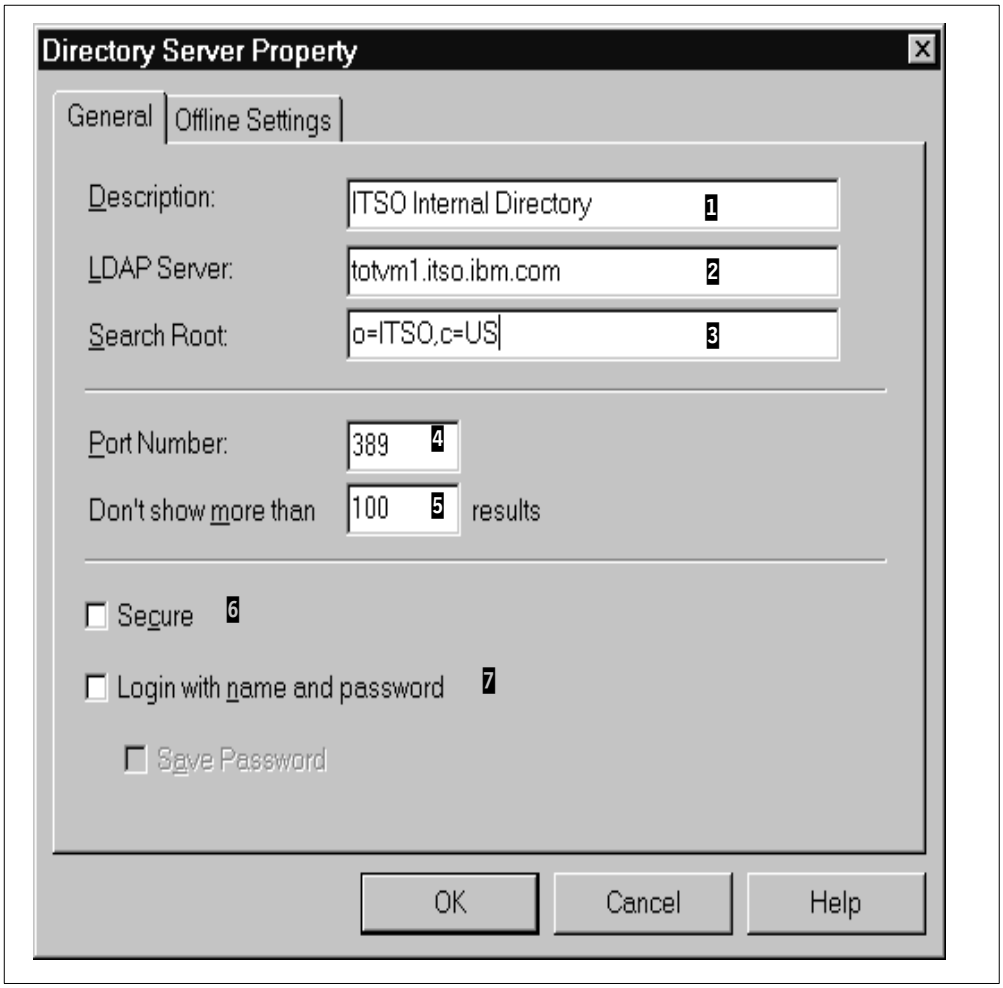


Figure 122. Configuring Netscape: Specifying directory properties

Figure 122 notes:

1. Enter any meaningful text that describes the server.
2. Specify the IP name or address of the LDAP server.
3. The search root will be o=<organization>,c=<ISO country code>.
4. LDAP defaults to port 389.
5. The default is 100 results.
6. This port of LDAP does not support SSL (yet).
7. No need to login with name and password.

Following this we called up the preferences window to change Netscape's behavior.

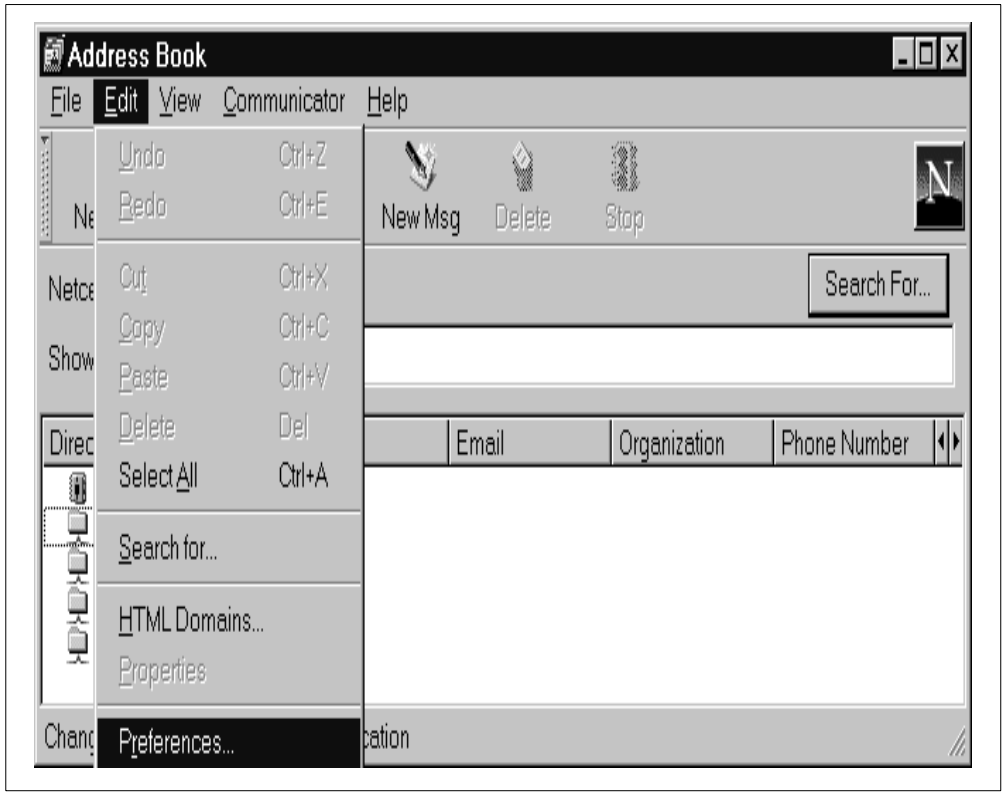


Figure 123. Configuring Netscape: Updating preferences

The addressing section of the preferences window allows us to get Netscape to use LDAP and to use our VM LDAP server as its directory server.

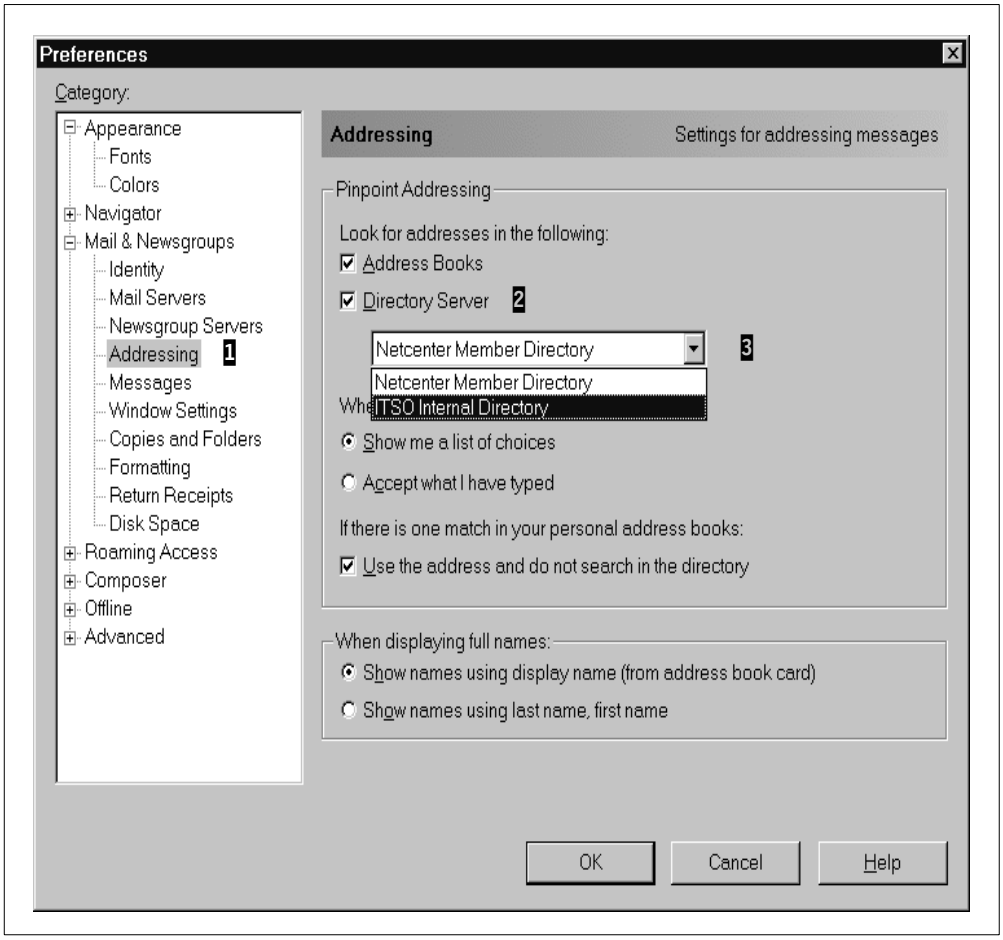


Figure 124. Configuring Netscape: Selecting default directory server

Figure 124 notes:

1. Select the Addressing menu item from the Mail & Newsgroups list.
2. Check the box indicating you want to use a directory server.
3. Choose the directory server you just configured.

The Netscape client is now ready to interrogate the VM LDAP server.

6.2.12 Using Netscape to search directory

Now that Netscape has been configured you can search the LDAP directory. Select the directory server just configured and enter your search parameters.

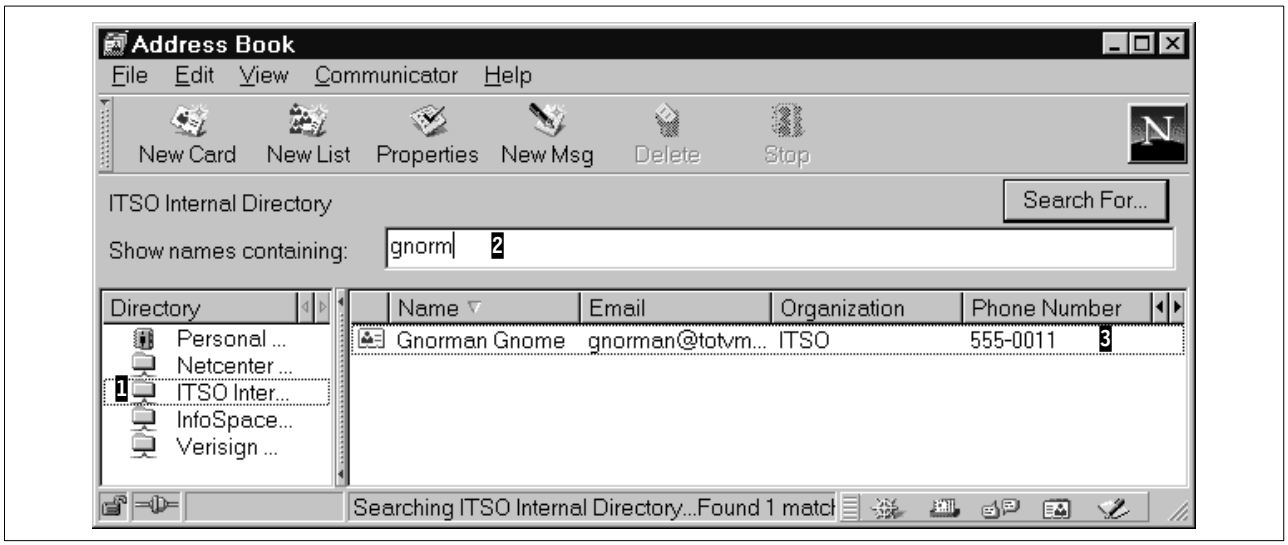


Figure 125. Using Netscape to search directory: Locate a name

Figure 125 notes:

1. Select the server just configured.
2. Enter the search parameter(s). As you type, the server will be interrogated so that continual partial searches will be conducted and results displayed.
3. The result of the search is displayed in this window.

When the search results are returned, you can double click any of the entries and the type of information shown in the following figure will be displayed.

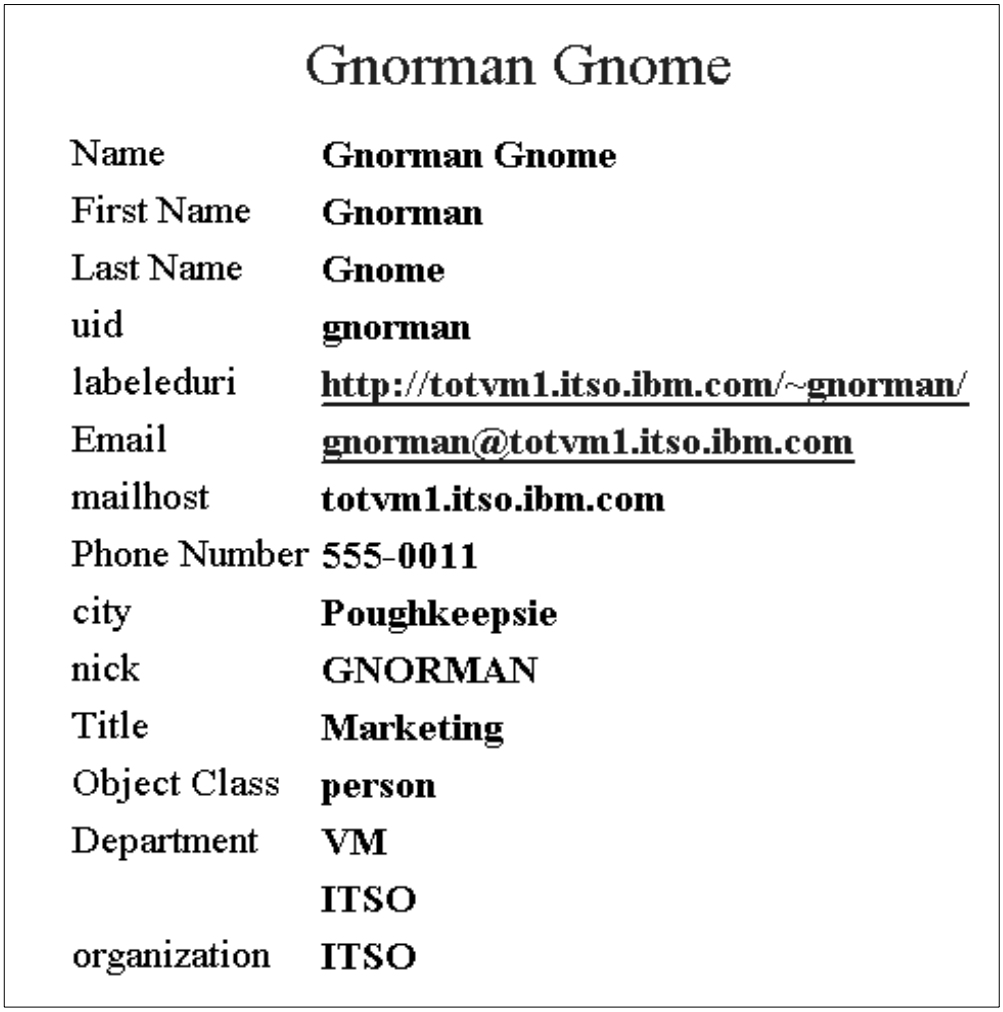


Figure 126. Using Netscape to search directory: Displaying entry details

6.3 Apache

Apache was originally based on code and ideas found in the most popular HTTP server of the time: NCSA httpd 1.3 (early 1995). It has since evolved into a far superior system which can rival (and probably surpass) almost any other UNIX-based HTTP server in terms of functionality, efficiency and speed.

Since it began, it has been completely rewritten, and includes many new features. Apache is, as of April 1999, the most popular Web server on the Internet, according to the Netcraft Survey. You can view these survey results at:

`http://www.netcraft.com/survey`

Table 3. Netcraft Survey of top WWW Server Systems

Server	April 1999	%
Apache	2832119	56.19
Microsoft-IIS	1164132	23.09
Netscape-Enterprise	253660	5.03
Rapidsite	94808	1.88
WebSitePro	79615	1.58
thttpd	66211	1.31
Stronghold	64491	1.28
WebStar	53985	1.07
Zeus	51969	1.03
NCSA	40823	0.81

Apache was created to address the concerns of a group of Web server providers and part-time httpd programmers who felt that httpd did not behave as they wanted it to behave. Apache is an entirely volunteer effort, completely funded by its members, not by commercial sales.

6.3.1 Apache CP directory entry

Figure 126 shows the Apache CP directory entry we used.

```
1  USER APACHE TOTVM1PW 128M 256M BDG
2   ACCOUNT DAEMON APACHE
    IPL CMS PARM AUTOCR FILEPOOL SFSTEST
3   IUCV ALLOW
4   IUCV ANY
    MACHINE XC
5   POSIXINFO UID 60 GNAME httpd IWDIR /home/apache
6   POSIXOPT SETIDS ALLOW QUERYDB ALLOW EXEC_SETIDS ALLOW
7   POSIXGLIST GNAMES system
    CONSOLE 0009 3215
    SPOOL 000C 3505 A
    SPOOL 000D 3525 A
    SPOOL 000E 1403 A
    LINK MAINT 0190 0190 RR
    LINK MAINT 019E 019E RR
    LINK MAINT 019D 019D RR
```

Figure 127. Apache CP directory entry

Figure 127 notes:

1. The server could have quite a number of connections, so give it a larger allocation of storage than normal.
2. We kept all our daemons under a single accounting group.
3. Needed to connect to UNIX sockets (SYSLOG).
4. Needed to allow connection from the outside world.
5. Apache does not require root access and has a group defined for web operations.
6. Special access allowed to user/group database.
7. Make it also a member of the system group.

6.3.2 Apache BFS requirements

Consistent with the convention we decided upon, the user is given its own filespace, which is mounted as its home directory:

1. Enroll the user in the filepool:

   ```
   ENROLL USER APACHE BFSTEST (BFS BLOCKS 500
   ```

2. Create a mount point by creating an external link:

   ```
   OPENVM CREATE EXTLINK /home/apache MOUNT /../VMBFS:BFSTEST:APACHE/
   ```

6.3.3 Apache installation and configuration

Install and configure Apache with the following steps:

1. Follow the FTP instructions in Appendix K, “General instructions for downloading packages” on page 311.
2. Enter the Apache directory:

   ```
   cd apache_1.3.6
   ```

3. Install the product (a detailed console log can be found in F.5.1 “Apache installation log” on page 240):

   ```
   make install
   ```

```
+-------------------------------------------------------+
| You now have successfully built and installed the     |
| Apache 1.3 HTTP server. To verify that Apache actually |
| works correctly you now should first check the        |
| (initially created or preserved) configuration files  |
|                                                       |
|   /usr/local/apache/bin/etc/{httpd,access,srm}.conf   |
|                                                       |
| and then you should be able to immediately fire up    |
| Apache the first time by running:                     |
|                                                       |
|   /usr/local/apache/bin/sbin/apachectl start          |
|                                                       |
| Thanks for using Apache.       The Apache Group       |
|                                http://www.apache.org/ |
+-------------------------------------------------------+
```

Figure 128. A successful install of Apache

4. Update the Apache configuration file in /usr/local/apache/conf/httpd.conf (see F.5.2 “HTTPD.CONF” on page 241).

   For a detailed look at Apache configuration and operation see *Apache Server Bible* Mohammed J. Kabir, ISBN 0-7645-3218-9**.**

5. Give ownership of the Apache production directories and files:

   ```
   chown -R apache:httpd /usr/local/apache
   ```

6. Logon to the user APACHE.
7. Create a PROFILE EXEC as shown in 5.1 “Standard PROFILE EXEC” on page 87.
8. Enter the shell by invoking the PROFILE EXEC.
9. Create /home/apache/inetd.apache:

   ```
   www-http   stream tcp nowait apache /usr/local/apache/bin/httpd
   ```

10. Create /home/apache/.profile

    ```
    # Start the Apache inetd
    inetd inetd.apache &
    ```

Figure 129. .profile for Apache

11. Configure any user home directories:

    - Make a subdirectory within a user’s home directory called public_html (this corresponds to the UserDir directive in the HTTP.CONF file; see F.5.2 “HTTPD.CONF” on page 241):

      ```
      mkdir /home/<user>/public_html
      ```

    - Allow the Apache server to access this directory by giving ownership to the user httpd:

      ```
      chown -R <user>:httpd /home/<user>/public_html
      ```

    - Give read/write access to the group and set the "set-group-id" flag on for objects within the directory:

      ```
      chmod -R 2770 /home/<user>/public_html
      ```

    - Ensure the home directory can be accessed:

```
chmod +x /home/<user>
```

If the user's home directory is an external link, you will need to do steps 3 and 4 for the external link:

```
chown -R <user>:httpd /../VMBFS:<filepool>:<user>/
chmod -R 2770 /../VMBFS:<filepool>:<user>/
```

12.Re-IPL CMS to start the Apache INET daemon.

6.3.4 Using the Apache server

Once the server has been started the browser can be pointed at the server.

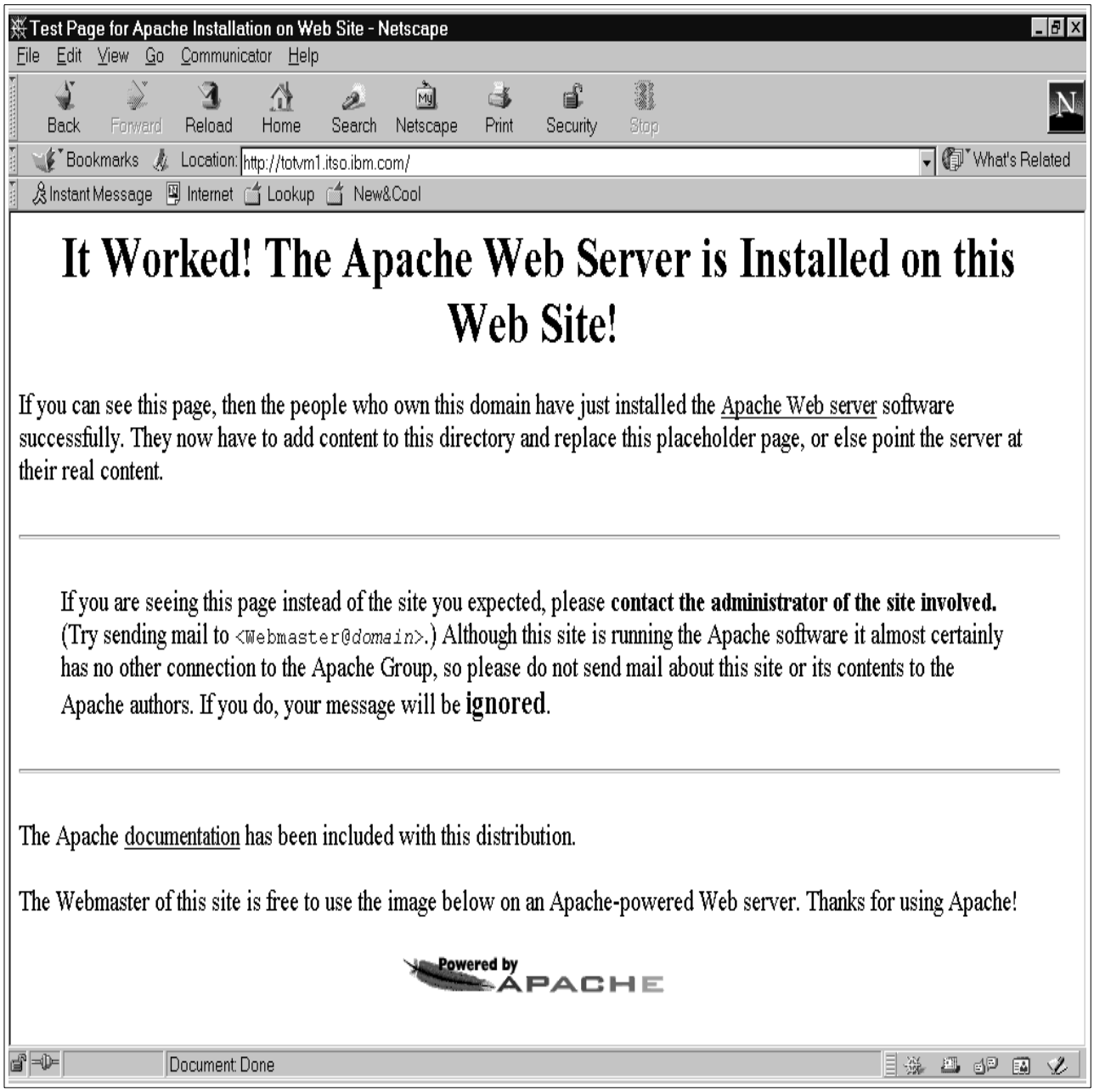


Figure 130. Our first Apache/VM page

Next, the ability to access personal web pages can be examined. Here is a fragment of HTML source in the /home/neale/public_html/index.html file.

```
<HTML>
<IMAGE SRC="aust_map.gif">
<H1>Hello</H1>
<FORM ACTION="/cgi-bin/test-cgi">
    <INPUT TYPE="submit" Value="Submit a new problem report">
</FORM>
<PRE>
#include &lt;stdio.h&gt;
#include &lt;syslog.h&gt;
#include &lt;fcntl.h&gt;
#include &lt;stat.h&gt;
#include &lt;string.h&gt;
#include &lt;unistd.h&gt;
#include &lt;spawn.h&gt;
#include &lt;pwd.h&gt;

int
```

Figure 131. Extract of a personal home page source file

The following screen is displayed by pointing the browser at:

`http://totvm1.itso.ibm.com/~neale`

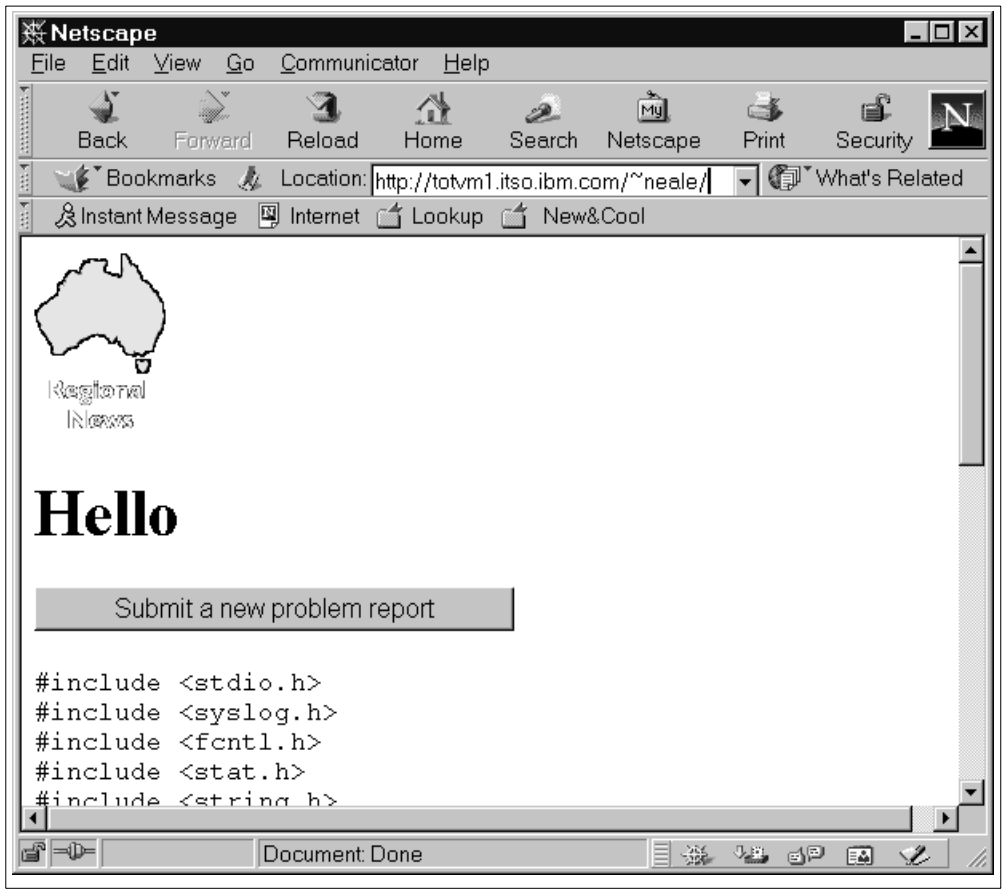


Figure 132. Accessing the personal home page

After this triumph, the ability to execute CGI programs can be tested. One of the nice features of Apache is that even if your operation does not support the

"magic-value" (also known as #! or hashbang: see 3.1.4.1 "The magic value" on page 40), Apache does.

The easiest test to perform is invoking test-cgi.

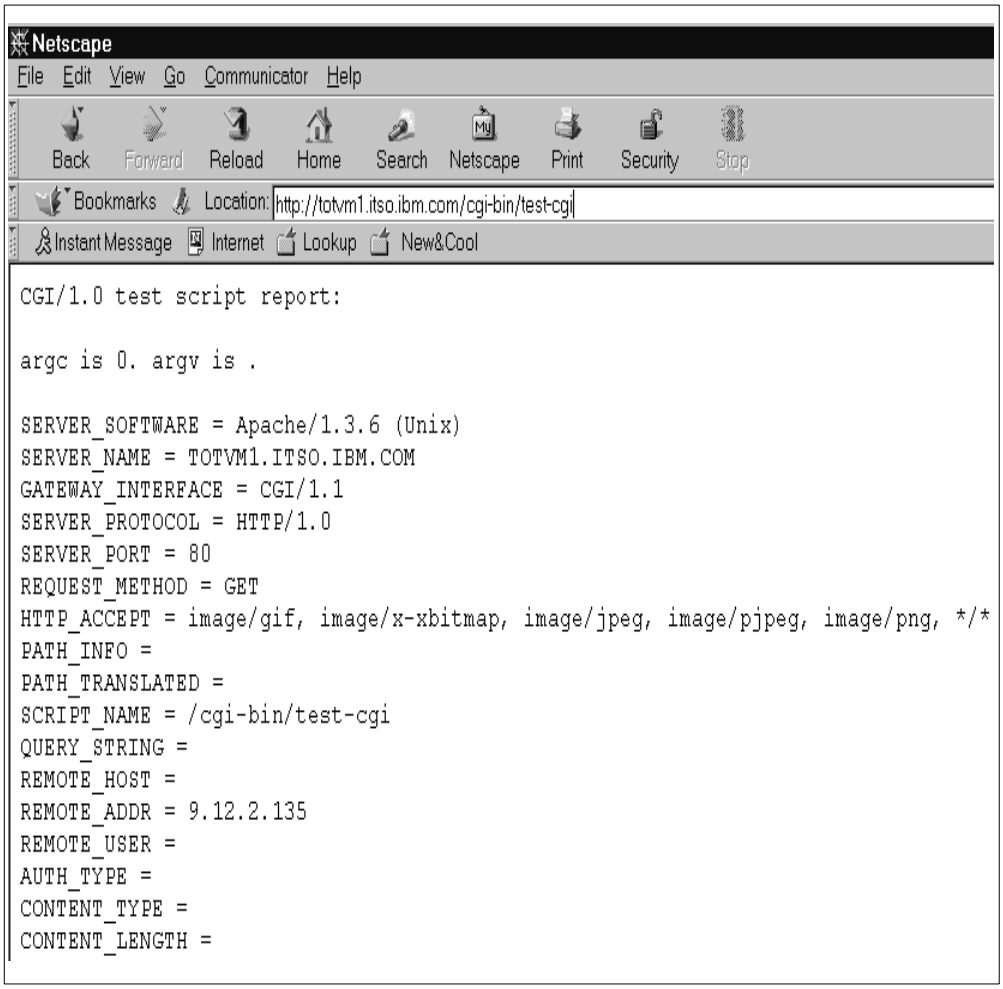


Figure 133. Invoking the test-cgi on Apache

Note: Since we have installed Regina, it is possible to create a CGI using REXX.

```
#!/usr/local/bin/rexx
say 'Content-type: text/html'
say
say '<H1>Hello</H1>'
exit
```

Figure 134. Sample CGI written in REXX

6.3.5 Exploiting Apache

There is a lot of data in the average VM installation that does not reside in the BFS, but that it would be nice for Apache to be able to access. One of the newest areas in web server technology is the emergence of the FastCGI standard. Time did not permit this technology to be examined in any detail but it does promise a means of merging the Apache with the terabytes of data living on or accessible to VM systems.

The following information has been obtained from the FastCGI website and has been modified to fit the intent of this section of the book. For more detailed information on FastCGI, see the FastCGI website:

`http://www.fastcgi.com`.

FastCGI is a language-independent, scalable, open extension to CGI that provides high performance and persistence without the limitations of server-specific APIs. FastCGI applications are not limited to a particular development language (the protocol is open). FastCGI application libraries currently exist for Perl, C, Java, Python, and TCL.

FastCGI applications use (TCP or Unix) sockets to communicate with the web server. This scalable architecture allows applications to run on the same platform as the web server or on many machines scattered across an enterprise network.

FastCGI applications are portable to other web server platforms.

FastCGI applications are fast because they are persistent. There is no per-request startup and initialization overhead. This makes possible the development of applications which would otherwise be impractical within the CGI paradigm (for example, a huge Perl script, or an application which requires a connection to one or more databases).

Theoretically, using the FastCGI approach, CGIs could be PIPEs or REXX EXECs running in other virtual machines (or other systems entirely), unaware of the actual type of web server they are connected to.

6.4 JacORB - a Java implementation of CORBA

JacORB is an object request broker written in Java. It is an implementation of OMG's CORBA 2.0 standard. JacORB is free, and easy to install and use.

JacORB features include:

- IDL compiler and stub generator.
- Both CORBA IDL and Java-only distributed programming are supported.
- Multithreaded clients and servers are supported (various thread models).
- Internet Inter-ORB Protocol IIOP.
- Interface Repository Implementation Repository/BOA.
- Dynamic Invocation Interface (DII) and Dynamic Skeleton Interface (DSI).
- Request-level and Message-level Interceptors (also known as Filters and Transformers).
- COSS compliant name service.
- Prototypical COSS Event service.
- 100% pure Java (portable).
- IDL and Java source for all CORBA/COSS interfaces.
- Examples, documentation and full source code are included.
- Free (published under the GNU Public Library License).

A future version of JacORB will feature:

- An implementation of the Java 2.3 language mapping and the POA.
- A request gateway for applets.
- A partial security service implementation, in particular:
  - User authentication with SPKI certificates.
  - View-based access control.

6.4.1 JacORB CP directory entry

We used the following directory entry for JacORB.

```
1  USER JACORB TOTVM1PW 64M 256M G
2    ACCOUNT DAEMON JACORB
     IPL CMS PARM AUTOCR PARM SFSTEST
     MACHINE XC
3    IUCV ALLOW
     IUCV ANY
4    POSIXINFO UID 58
     CONSOLE 0009 3215
     SPOOL 000C 3505 A
     SPOOL 000D 3525 A
     SPOOL 000E 1403 A
     LINK MAINT 0190 0190 RR
     LINK MAINT 019E 019E RR
```

Figure 135. JACORB CP directory entry

Figure 135 notes:

1. Java things take up more virtual storage than your average application. 64M is a lot, but if you have it then use it.
2. Again, we wanted to keep our daemons together.
3. The IUCV statements allow us to do TCP/IP things.

4. Any POSIX id will do.

6.4.2 JacORB BFS requirements

Consistent with the convention we decided upon, the user is given its own filespace, which is mounted as its home directory:

1. Enroll the user in the filepool and give it some space to fit the java class and source files:

   ```
   ENROLL USER JACORB BFSTEST (BFS BLOCKS 5000
   ```

2. Create a mount point by creating an external link:

   ```
   OPENVM CREATE EXTLINK /home/jacorb MOUNT /../VMBFS:BFSTEST:JACORB/
   ```

6.4.3 JacORB installation and configuration

Install and configure JacORB using the following steps:

1. Follow the FTP instructions in Appendix K, “General instructions for downloading packages” on page 311.
2. Logon to the user JACORB.
3. Create a PROFILE EXEC as shown in Section 5.1, “Standard PROFILE EXEC” on page 87.
4. Re-IPL CMS to enter the shell.
5. Create a .profile.

```
BASEDIR=$HOME
export CLASSPATH=$BASEDIR:$BASEDIR/classes:${CLASSPATH}
export PATH=$BASEDIR/bin:${PATH}
```

Figure 136. .profile for JacORB

6. Re-IPL CMS to enter the shell and invoke .profile.

7. Edit the JacORB configuration file jacorb/Orb/Config.java.

```
/*
 *        JacORB - a free Java ORB
 *
 *   Copyright (C) 1997-98  Gerald Brose.
 *
 *   This library is free software; you can redistribute it and/or
 *   modify it under the terms of the GNU Library General Public
 *   License as published by the Free Software Foundation; either
 *   version 2 of the License, or (at your option) any later version.
 *
 *   This library is distributed in the hope that it will be useful,
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 *   Library General Public License for more details.
 *
 *   You should have received a copy of the GNU Library General Public
 *   License along with this library; if not, write to the Free
 *   Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
 */

package jacorb.Orb;

/**
 * JacORB configuration constants
**/

public final class Config
{

    // how often do we retry to connect, and how long do we wait
    // before trying again (in msecs)?

    public static final int         retry           = 10;
    public static final long        retry_intvl     = 700;

    // network buffer size

    public static final int OUTBUFSIZE = 4096;

    // the following constants are strings in URL format pointing
    // to a file resource. The first identifies the default naming
    // context to be used, the second is the location for the
    // interface repository's IOR

    public static final String      default_context =
1       "file:/usr/local/ORB/NS_Ref";

    public static final String      IR_server =
2       "file:/usr/local/ORB/IR_Ref";

    public static final int IT_Port = 27000;

}
```

Figure 137. JacORB configuration file

Figure 137 notes:

1. Location of the Nameserver file.
2. Location of the Interface Repository file.

8. Build the package using this configuration:

```
make configure
javac jacorb/Orb/Config.java
```

```
javac jacorb/Naming/NameServer.java
javac org/omg/CORBA/ORB.java
javac jacorb/Orb/Connection.java
javac jacorb/Orb/IR/Repository.java
```

9. Create a directory for the Nameserver and Interface Repository files:

   mkdir /usr/local/ORB

6.4.4 Testing the ORB

1. Start the Nameserver:

   ns /usr/local/ORB/NS_Ref &

```
[ New connection to 9.12.13.62:2792 ]
[ Accepted connection from 9.12.13.62:2793 ]
```

2. Enter the sample program directory:

   jacorb/demo/example1

3. Start the server portion of the application:

   server &

```
[2]     9750
[ New connection to 9.12.13.62:2794 ]
[ Accepted connection from 9.12.13.62:2795 ]
[ New connection to 9.12.13.62:2792 ]
[ Accepted connection from 9.12.13.62:2796 ]
[ New connection to 9.12.13.62:2794 ]
[ Accepted connection from 9.12.13.62:2797 ]
[ Bound name: /server.service]
[ BOA: Connection timed out ]
[ Closing connection to 9.12.13.62:2793 ]
[ Closing connection to 9.12.13.62:2792 ]
```

4. Start the client portion of the application:

   client "Massage"

```
[ Accepted connection from 9.12.13.62:2799 ]
[ New connection to 9.12.13.62:2792 ]
[ New connection to 9.12.13.62:2794 ]
[ Accepted connection from 9.12.13.62:2800 ]
Message: Massage
Massage written
Message: Massage
Message: Massage
Message: Massage
Message: Massage
Message: Massage
string array written
From example1: Massage 0
From example1: Massage 1
From example1: Massage 2
From example1: Massage 3
From example1: Massage 4
[ Closing connection to 9.12.13.62:2799 ]
[ Closing connection to 9.12.13.62:2800 ]
[ BOA: Connection timed out ]
[ Closing connection to 9.12.13.62:2796 ]
[ Closing connection to 9.12.13.62:2792 ]
```

Chapter 7. Optional extras

We have now established an infrastructure and are running important applications within our VM system. There are several other applications that may be of benefit to the organization. They are described in this chapter.

7.1 INND - USENET

USENET is a highly used and valued feature of the Internet. The ability to conduct discussions and disseminate them to interested parties is a desirable facility within an enterprise as well. We have ported INN 1.4 and subsequently 1.5.1 to OpenEdition for VM/ESA and have managed to get the NNRPD (news server) portion running. The news server is now being used to conduct residency-specific fora. Users are able to subscribe, read, and append to news groups dealing with various parts of the residency operations.

Work still to be done includes:

- Fixing the `ctlinnd`-to-innd connection. `ctlinnd` is the means by which the operator controls the actions of innd.
- Getting newsfeeds from public servers. There are a lot of *nasty* newsgroups out there and a whole lot of useless ones. We would like to provide users with a selection of newsgroups that are not offensive and are, hopefully, relevant to their jobs.

7.1.1 News CP directory entry

To install INND - USENET the following CP directory entry is needed.

```
   USER NEWS DVM 32M 128M BDG
1   POSIXINFO UID 5 GNAME news
    SCR INA PIN STA YEL CPO TUR INR RED
2   ACCOUNT DAEMON INND
3   OPTION QUICKDSP
    MACH XC
    SHARE RELATIVE 200 200
    IPL CMS PARM AUTOCR FILEPOOL SFSTEST
    IUCV ALLOW
    IUCV ANY
    CONSOLE 009 3215
    SPOOL   00C 2540 READER A
    SPOOL   00D 2540 PUNCH A
    SPOOL   00E 1403
    LINK MAINT    190 190 RR
    LINK MAINT    19D 19D RR
    LINK MAINT    19E 19E RR
```

Figure 138. Usenet server CP directory entry

Figure 138 notes:

1. The user has been assigned the UID of 5 and belongs to the news group.
2. We keep our daemons in a single accounting group.
3. As a server, we give it preferential treatment at dispatch time.

7.1.2 News BFS requirements

Consistent with the convention we decided upon, the user is given his own filespace which is mounted as his home directory:

1. Enroll the user in the filepool:

   ```
   ENROLL USER NEWS BFSTEST (BFS BLOCKS 500
   ```

2. Create a mount point by creating an external link:

   ```
   OPENVM CREATE EXTLINK /home/news MOUNT /../VMBFS:BFSTEST:NEWS/
   ```

7.1.3 News installation and configuration

1. Issue the following commands:

   ```
   mkdir -p /usr/local/etc/control
   mkdir -p /var/log
   ```

2. Follow the FTP instructions in Appendix K, “General instructions for downloading packages” on page 311.

3. Enter the News directory:

   ```
   cd inn-1.5.1
   ```

4. Install the package (see F.2, “INND installation log” on page 223 for a log of the installation process):

   ```
   make install
   ```

5. Issue the following commands to give news access to the directories and files it will be using:

   ```
   chown -R news:news /usr/local/news
   chmod 0644 /usr/local/news/newsfeeds
   ```

6. The following files are candidates for customizing. They reside in the /usr/local/news directory:

   ```
   cd /usr/local/news
   ```

Table 4. News Configuration files.

File	Description
expire.ctl	Controls expiration of articles within newsgroups
newsfeeds	Defines other news servers to feed articles to
hosts.nntp	Names and address of server that feed us news
nnrp.access	Hosts allowed to connect to this server
inn.conf	Main configuration file
passwd.nntp	Passwords for connecting to remote NNTP servers
moderators	The e-mail addresses of those responsible for various newsgroups

7. Update ./hosts.nntp with your requirements.

```
##  $Revision: 1.7 $
##  hosts.nntp - names and addresses that feed us news
##  Format
##      <host>:
##      <host>:<password>
##  <host> can be a name or IP address; no wildcards.  Any hosts
##  not listed here are handed off to nnrpd.
1 metro.ucc.su.oz.au:
```

Figure 139. Configuring the hosts.nntp file

Figure 139 note:

1. We will attempt to receive news from metro.ucc.su.oz.au (the owner of this newslist must allow such a feed to occur).

8. Update ./inn.conf.

```
##  $Revision: 1.6 $
##  inn.conf -- inn configuration data
##  Format:
##      <parameter>:<whitespace><value>
##  Used by various programs and libinn. The following parameters are
##  defined:
##      domain          Local domain, without leading period.
##      fromhost        What to put in the From line; default is FQDN
##                      of the local host.
##      moderatormailer Where to mail moderated postings, if not found
##                      in the moderators file; see moderators(5).
##      pathhost        What to put in the Path and Xref headers; default
##                      is FQDN of the local host.
##      organization    If $ORGANIZATION doesn't exist.  What to put in
##                      the Organization header if blank.
##      server          If $NNTPSERVER doesn't exist.  Local NNTP server
##                      host to connect to.
##
1 organization:   ITSO Poughkeepsie
2 server:         totvm1.itso.ibm.com
```

Figure 140. Configuring the inn.conf file

Figure 140 notes:

1. Identifies the organization to which the news server belongs.
2. Identifies the IP address or name of the news server.

9. Configure ./moderators.

```
##  $Revision: 1.7 $
##  Mailing addresses for moderators.
##  Format:
##      <newsgroup>:<pathname>
##  First match found is used.
##      <newsgroup>     Shell-style newsgroup pattern or specific newsgroup
##      <pathname>      Mail address, "%s" becomes newgroup name with dots
##                      changed to dashes.
gnu.*:%s@prep.ai.mit.edu
bionet.*:%s@net.bio.net
bln.*:%s@fu-berlin.de
cz.*:%s@moderator.vslib.cz
hun.*:%s@sztaki.hu
linux.act.*:linux-submit@yggdrasil.com
linux.*:submit-%s@yggdrasil.com
nz.*:%s@usenet.net.nz
pl.*:%s@usenet.pl
relcom.*:%s@news.ussr.eu.net
sk.*:%s@news.ke.sanet.sk
tnn.*:%s@news.iij.ad.jp
1 *:news@totvm1.itso.ibm.com
```

Figure 141. Configuring the moderators file

Figure 141 note:

1. All newsgroups that do not match any of the patterns shown in Figure 141 are the responsibility of our news server.

10.Configure ./nnrp.access.

```
##  $Revision: 1.7 $
##  nnrp.access - access file for on-campus NNTP sites
##  Format:
##      <host>:<perm>:<user>:<pass>:<groups>
##  Connecting host must be found in this file; the last match found is
##  used, so put defaults first.
##      <host>          Wildcard name or IP address
##      <perm>          R to read; P to post
##      <user>          Username for authentication before posting
##      <pass>          Password, for same reason
##      <groups>        Newsgroup patterns that can be read or not read
##  To disable posting put a space in the <user> and <pass> fields, since
##  there is no way for client to enter one.
##
## Default is no access, no way to authentication, and no groups.
*:: -no- : -no- :!*
##  Foo, Incorporated, hosts have no password, can read anything.
stdin:Read Post:::*
1 localhost:Read Post:::*
2 127.0.0.1:Read Post:::*
3 9.*:Read Post:::*
4 *.ibm.com:Read Post:::*
```

Figure 142. Configuring the nnrp.access file

Figure 142 note:

Allow read and post access to hosts identified as:

1. localhost
2. IP address 127.0.0.1 (also known as localhost)
3. Every IP address on the class A network 9
4. Every host belonging to the ibm.com domain

11. Update ./newsfeeds.

```
##  $Revision: 1.12 $
##  newsfeeds - determine where Usenet articles get sent
##  Format:
##site[/exclude,exclude...]\
##:pattern,pattern...[/distrib,distrib...]\
##:flag,flag...\
##:param
  ME\
  :*,!foo.*/world,usa,na,gnu,bionet,pubnet,u3b,eunet,vmsnet,inet,ddn,\
  k12\
  ::
# Feed all moderated source postings to an archiver
source-archive\
  :Tp,Nm:/usr/local/etc/archive %s
# Feed all local non-internal postings to nearnet; sent off-line via
# nntpsend or send-nntp.
totvm1.itso.ibm.com\
  :!junk/!itso\
  :Tf,Wnm:totvm1.itso.ibm.com
#
# A real-time nntplink feed
#uunet\
# :/!foo\
# :Tc,Wnm:/usr/local/etc/nntplink -i logfile totvm1.itso.ibm.com
#
# Capture all Foo, Incorporated, postings
capture\
  :*/itso\
  :Tp,H2:/usr/lib/news/capture %s
#
# A UUCP feed, where we try to keep the "batching" between 4 and 1K.
ihnp4\
  :!junk,!control/!itso\
  :Tf,Wfb,B4096/1024:
```

Figure 143. Configuring the newsfeeds file

12. Ensure the user news owns its home directory:

```
chown news:news /home/news
```

13. Create newsgroup(s) by editing ./active.

```
1 control      0000000000 0000000000 n
2 junk         0000000000 0000000000 n
3 itso.porting 0000000000 0000000000 y
```

Figure 144. Creating newsgroups in the active file

Figure 144 notes:

1. `control` is needed by the server. It is not appended to (n).
2. `junk` is also needed by the server and is not appended to (n).
3. `itso.porting` is a “real” news group to which articles can be posted (y).

14. Create a directory where the itso.porting articles will reside:

```
mkdir -p /var/spool/news/itso/porting
```

15. Logon to the user NEWS.

16. Create a PROFILE EXEC as shown in 5.1, “Standard PROFILE EXEC” on page 87.

17. Enter the shell by invoking the PROFILE EXEC.

18. Issue the makehistory command prior to going online:

```
/usr/local/news/bin/makehistory
```

19. Enter the following commands:

```
cd /usr/local/news
mv history.n.dir history.dir
mv history.n.pag history.pag
```

20. Create a /home/news/.profile:

```
# Start news daemon in background
PATH=$PATH:/usr/local/etc:/usr/local/news/bin
innd &
```

Figure 145. .profile for news server

21. Re-IPL CMS to invoke the daemon.

7.1.4 Configuring Netscape to use News server

To use the newly installed and running News server you will need to configure your favorite newsreader. In our case we used Netscape. From any of the Netscape application windows, choose edit preferences as shown in Figure 146.

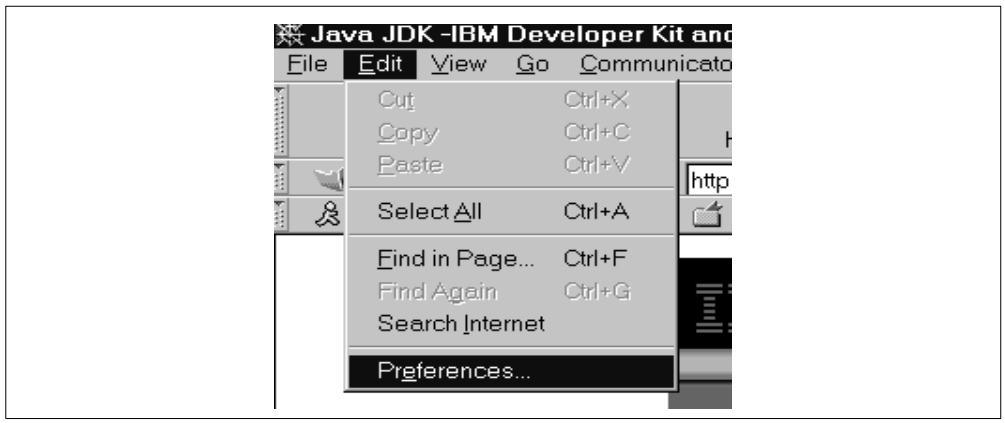


Figure 146. Configuring Netscape to use a News Server: Editing preferences

In the preferences window click **Newsgroup Servers** . Then click **Add**, as shown in Figure 147 on page 153.

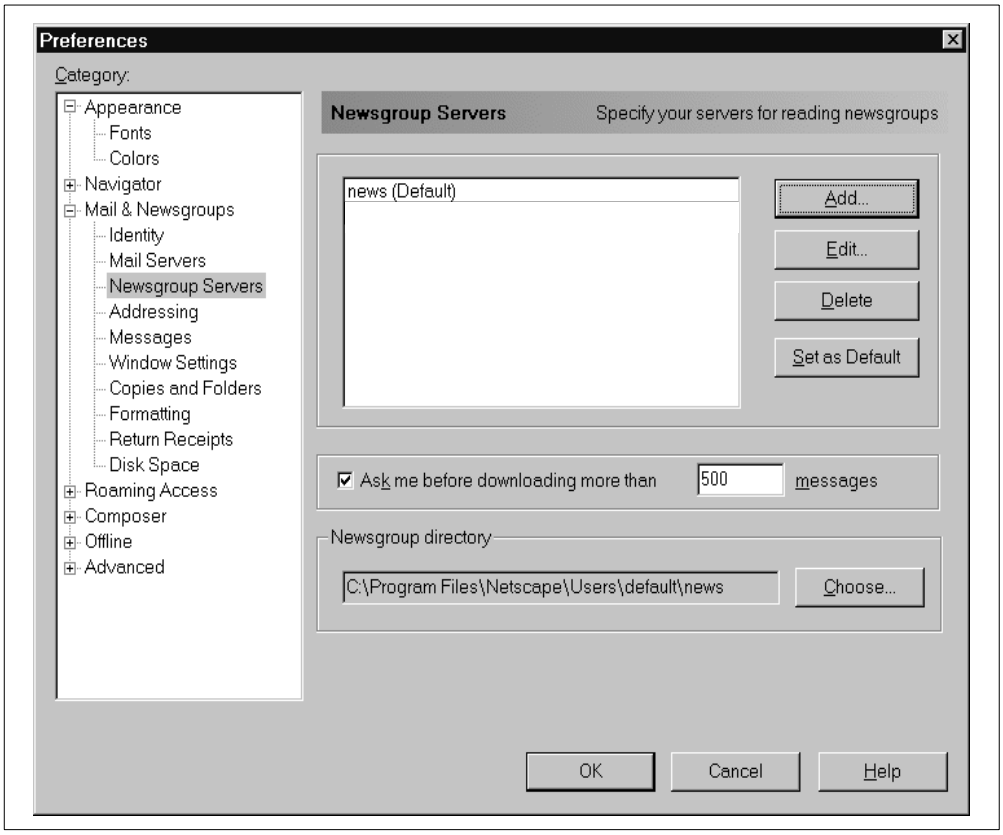


Figure 147. Configuring Netscape to use News Server: Adding the server

On the next window displayed, specify the server's properties such as IP address or IP name and the port on which it is listening (it defaults to 119; see Figure 148). Do not check the SSL option as this INND server has no support for encrypted connections.

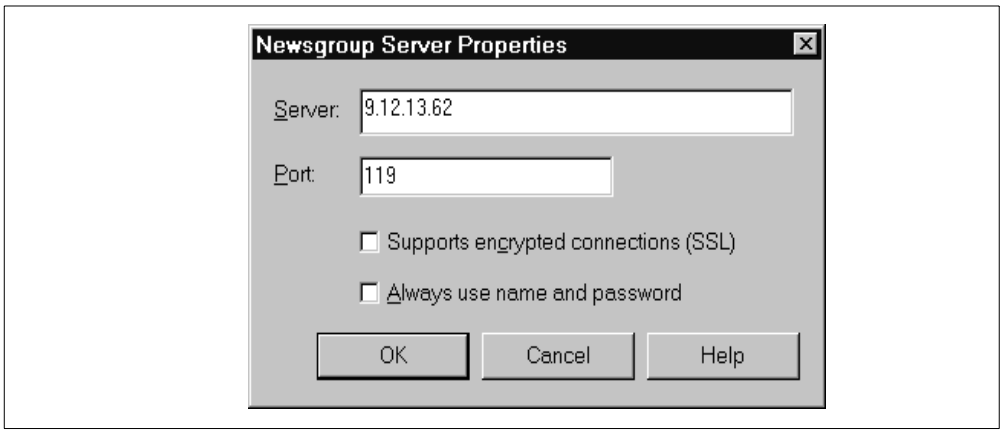


Figure 148. Configuring Netscape to use News Server: Specifying server parameters

7.1.5 Using Netscape with News Server

Once the server has been configured it can be accessed. Within Netscape Messenger, click the name of the server you have just configured (in this example

it is 9.12.13.62). Messenger will connect to the server. Right-click on the server name and you can subscribe to active newsgroups; see Figure 149.

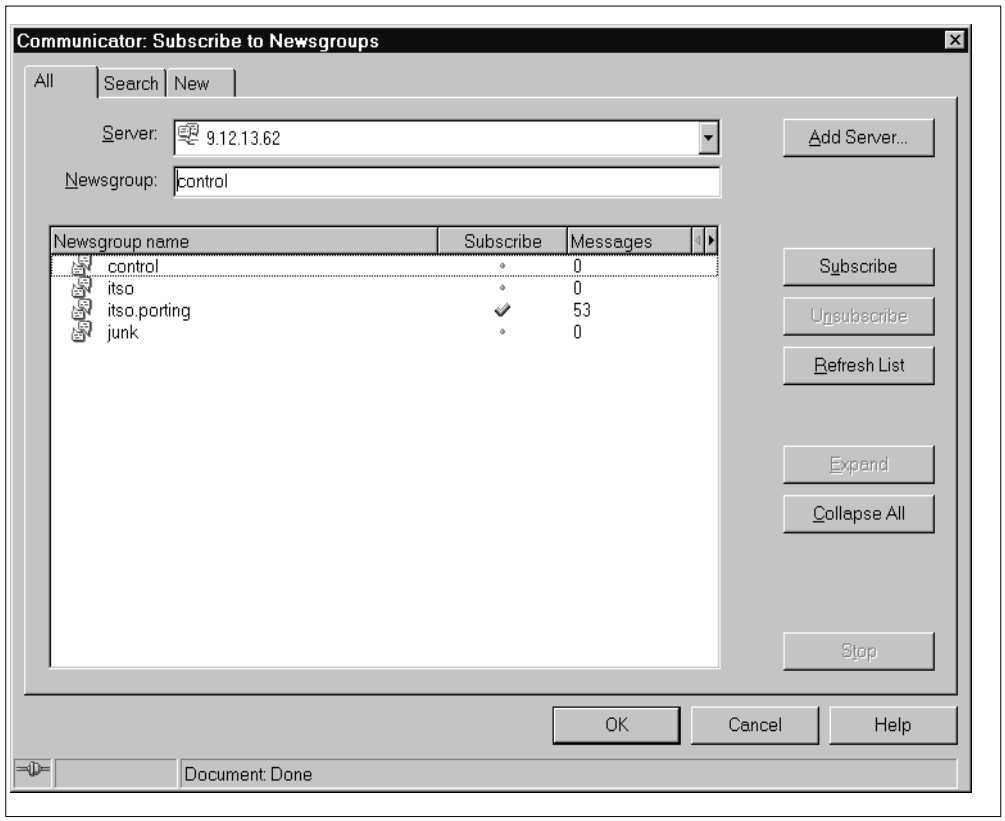


Figure 149. Using Netscape to subscribe to newsgroups

Once subscribed, you will be able to read messages appended to those newsgroups. Netscape will keep track of what messages you have read and will indicate whether there are any unread messages when you enter the news group. Figure 150 on page 155 illustrates Netscape in a session with a news group.

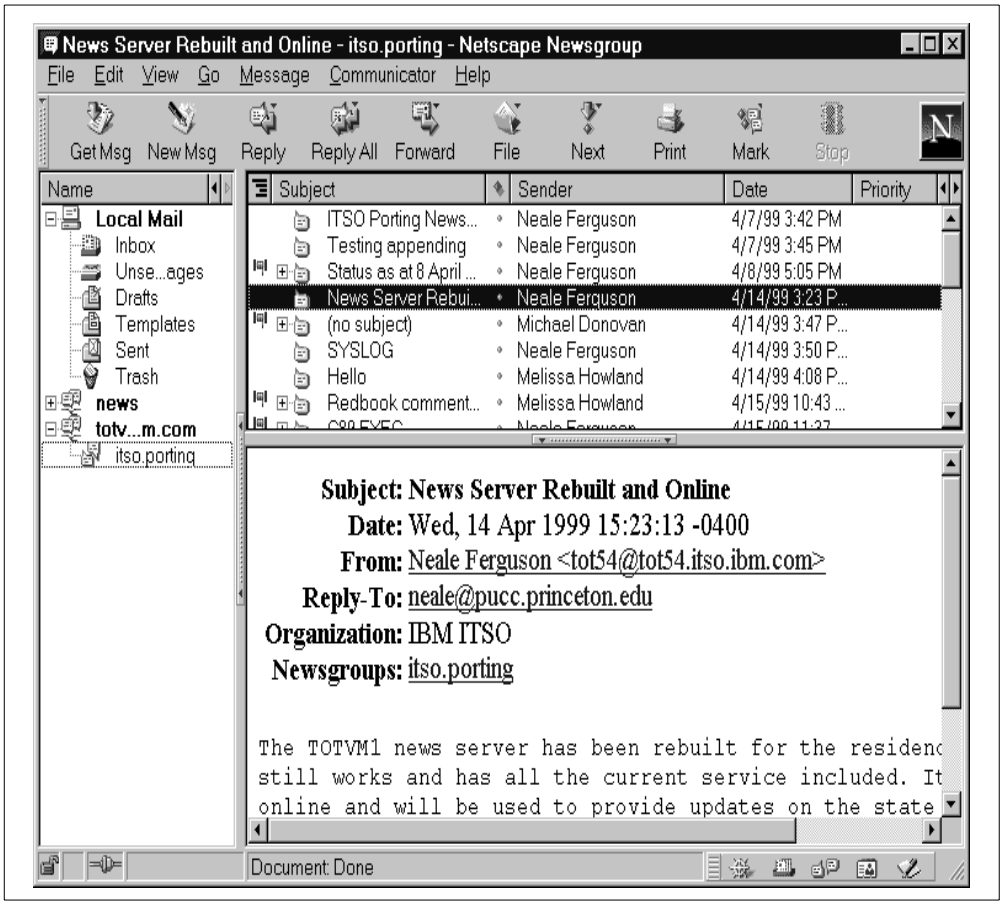


Figure 150. Internet News in action during the residency

7.2 Internet Relay Chat (IRC)

Internet Relay Chat was originally written by Jarkko Oikarinen in 1988. Since starting in Finland, it has been used in over 60 countries around the world. IRC is a multi-user chat system, where people meet on channels (rooms, virtual places, usually with a certain topic of conversation) to talk in groups, or privately. There is no restriction to the number of people that can participate in a given discussion, or the number of channels that can be formed on IRC. (Check `http://www.mirc.co.uk/help/jarkko.txt and jarkko2.txt` to find out more about how IRC started). As a user, you run a Client program which connects to a Server in an IRC network. All servers are interconnected and pass messages from user to user over the IRC network. One server can be connected to several other servers and up to hundreds of clients. Several larger and smaller IRC networks exist. The largest ones, called Eris Free net (EFnet), IRCnet, Undernet and Dalnet usually serve about 30,000 users at any given moment. Lots of other ones are a little less populated but often offer more stability and convenience.

An IRC client reads in the commands and text that you supply to it, and parses them. It filters them and performs the appropriate actions, and if necessary, passes them on to your IRC server. An IRC server can serve many other clients. The server holds information about the channels and people on IRC, as well as other pieces of information, and is also responsible for routing your messages to other users. The IRC Network itself consists of multiple servers which are all connected to each other.

7.2.1 IRCD CP directory entry

To install IRC the following CP directory entry is needed.

```
    USER IRCD DVM 32M 128M BDG
1    POSIXINFO UID 65 GNAME DEFAULT
     SCR INA PIN STA YEL CPO TUR INR RED
2    ACCOUNT DAEMON IRCD
3    OPTION QUICKDSP
     MACH XC
     SHARE RELATIVE 200 200
     IPL CMS PARM AUTOCR FILEPOOL SFSTEST
     CONSOLE 009 3215
     SPOOL   00C 2540 READER A
     SPOOL   00D 2540 PUNCH A
     SPOOL   00E 1403
     LINK MAINT    190 190 RR
     LINK MAINT    19D 19D RR
     LINK MAINT    19E 19E RR
```

Figure 151. IRCD CP directory entry

Figure 151 notes:

1. IRCD needs no special privileges.
2. IRCD belongs to the same accounting group as the other servers.
3. As a server we give IRCD preferential treatment.

7.2.2 IRC BFS requirements

Consistent with the convention we decided upon, the user is given its own filespace which is mounted as its home directory:

1. Enroll the user in the filepool:

   ```
   ENROLL USER IRCD BFSTEST (BFS BLOCKS 500
   ```

2. Create a mount point by creating an external link:

   ```
   OPENVM CREATE EXTLINK /home/ircd MOUNT /../VMBFS:BFSTEST:IRCD/
   ```

7.2.3 IRC installation and configuration

1. Follow the FTP instructions in Appendix K, “General instructions for downloading packages” on page 311.
2. Change to the IRC directory:

   ```
   cd irc2.8.21
   ```

3. Install the package (see F.4.1, “IRCD installation log” on page 234 for a log of the installation process):

   ```
   make install
   ```

4. Change directory to the configuration source files:

   ```
   cd /usr/local/src/ircd.
   ```

5. Create a workspace for ircd:

   ```
   mkdir /usr/local/ircd
   ```

6. Create a message of the day file; see Figure 152.

   ```
   x /usr/local/ircd.motd
   ```

```
   Welcome to the ITSO VM/ESA Internet Relay Chat
```

Figure 152. IRC message of the day file

7. Copy the sample configuration file:

   ```
   cp example.conf /usr/local/ircd/ircd.conf
   ```

8. Change to the configuration directory:

   ```
   cd /usr/local/ircd
   ```

9. XEDIT ircd.conf to configure the IRC daemon (see F.4.2, “IRCD configuration file” on page 235 for a full description of each field); see Figure 153.

```
1   M:totvm1.itso.ibm.com:*:IBM ITSO Poughkeepsie:6667
2   A:ITSO Poughkeepsie:LSA002R:Neale Ferguson <neale@totvm1.itso.ibm.com>
3   Y:1:90:0:20:100000
4   Y:2:90:300:1:600000
5   Y:10:90:0:3:100000
6   I:*@*:foobar:*@*::1
7   I:9.*::*.ibm.com::1
8   I:*@9.*::*@*.ibm.com::1
9   I:NOMATCH::neale@totvm1.itso.ibm.com::1
10  O:*.ibm.com:Neale:Trillian::10
11  O:neale@totvm1.itso.ibm.com:Neale:Trillian::10
12  o:*.ibm.com:ITChats:jhs::10
13  P:/tmp/.ircd:*:*:6666
```

Figure 153. Key statements from the IRC configuration file

Figure 153 notes:

1. Sets your server's name, description, and port number.
2. Lists your administrative information.
3. Defines a connection class for a "normal" client.
4. Defines a connection class for a "normal" server.
5. Defines a connection class for a client that does not time-out.
6. Client authorization line catching all connect attempts that do not match any of the following rules. Connection is permitted in this instance if the client can supply the password "foobar".
7. Allow all connections from the class A network 9 or the domain ibm.com.
8. Allow all connections from clients identifying themselves (that is, who want ident authentication and have an ident daemon running) who come from the class A 9 network or the ibm.com domain.
9. Those failing matching these rules will cause notification to be sent to the user neale.
10. Define Operator Access for people in the ibm.com domain supplying a user ID of Trillian and a password of Neale.
11. Same as for the previous rule except that ident is used (if available).
12. Define a local operator jhs with password ITChats.
13. Listen to the UNIX socket /tmp/.ircd on port 6666.

All remaining lines in the file can be commented out using the # character.

10. Give ownership of the workspace to ircd:

```
chown -R ircd /usr/local/ircd
```

11. Logon to the IRCD user.
12. Create a PROFILE EXEC as shown in 5.1, "Standard PROFILE EXEC" on page 87.
13. Re-IPL CMS to enter the shell.
14. Create .profile.

```
# Start the IRC daemon
PATH=/usr/local/src/ircd:$PATH
ircd -x 5 -t &
```

Figure 154. .profile for IRCD

15. Re-IPL CMS to start the server.

7.2.4 Choosing an IRC client

There are tens of IRC clients out there, so finding one that suits your particular needs (for example platform) should not be too hard. The two we chose to use were:

1. mIRC - a shareware product we obtained under an evaluation licence.
2. A java irc client from the IBM alphaworks site:
   `http://www.alphaWorks.ibm.com/tech/irc`

7.2.5 Configuring the mIRC client

As the mIRC client resides and runs on a PC, there was some configuration required.

The first thing we did when mIRC was started was to define the VM IRC server to the client:

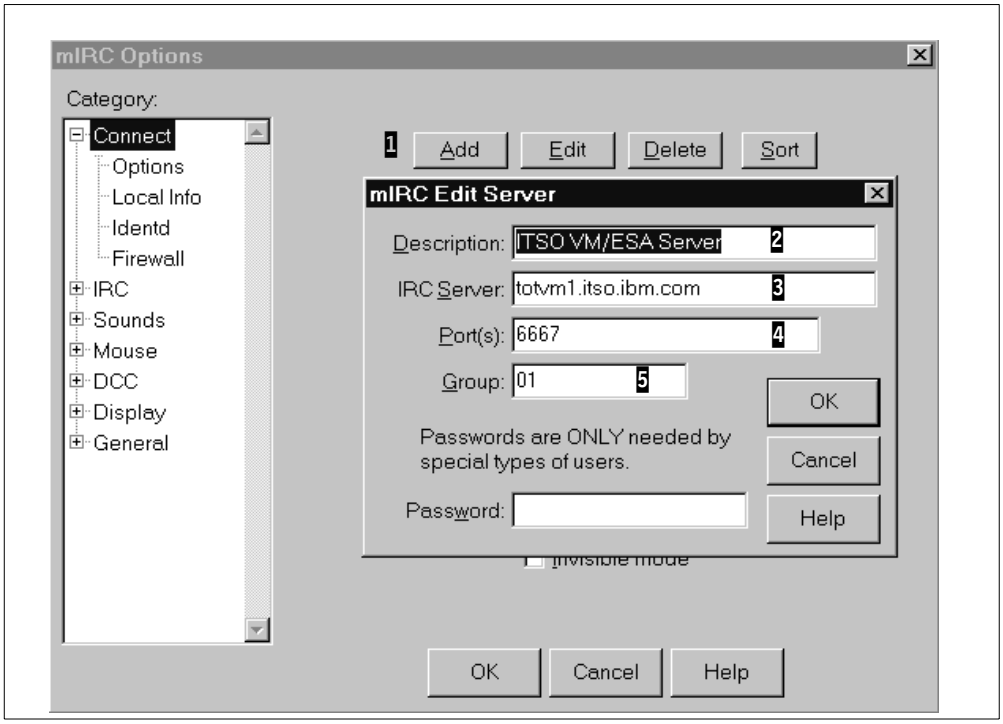


Figure 155. Configuring the IRC client

Figure 155 notes:

1. Press **Add** to add a new server.
2. Enter the description of the server.
3. Enter the IP address or name of the server.
4. Specify the port number (default is 6667).
5. This allows you to group servers together when they are sorted with the Sort button.

7.2.6 Using the mIRC client

Once the server was defined to the client, we connected to it by clicking on the Connect button.

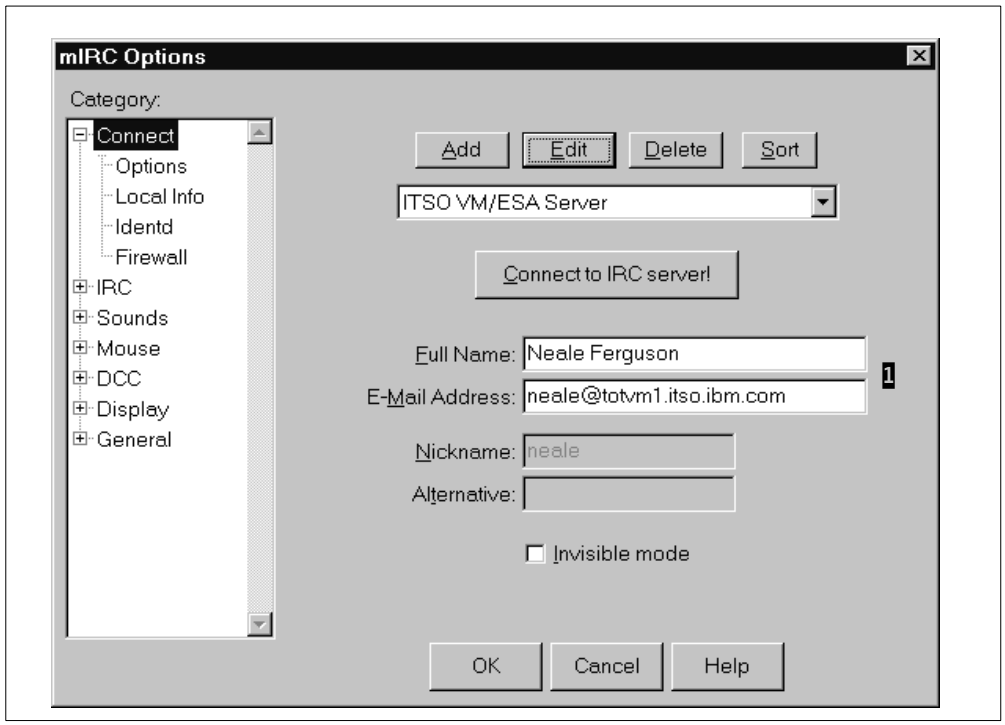


Figure 156. Connecting to IRC server

Figure 156 note:

1. This information was defined by the install wizard when the mIRC client was installed.

The mIRC client now attempted to connect to the VM server. Once connected, the server sent some connection messages.

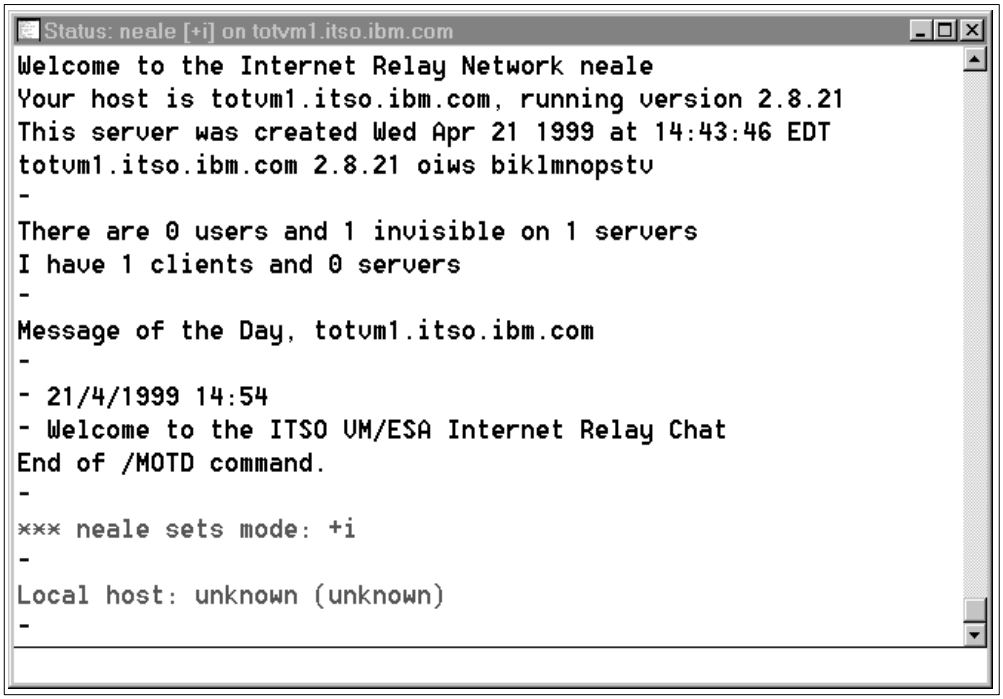


Figure 157. Connection made with IRC server

The client then asked (by way of pop-up window) what channels we wanted to join or define. As there were no channels on the newly started server, we added one called `#ITSO` using the Add function.

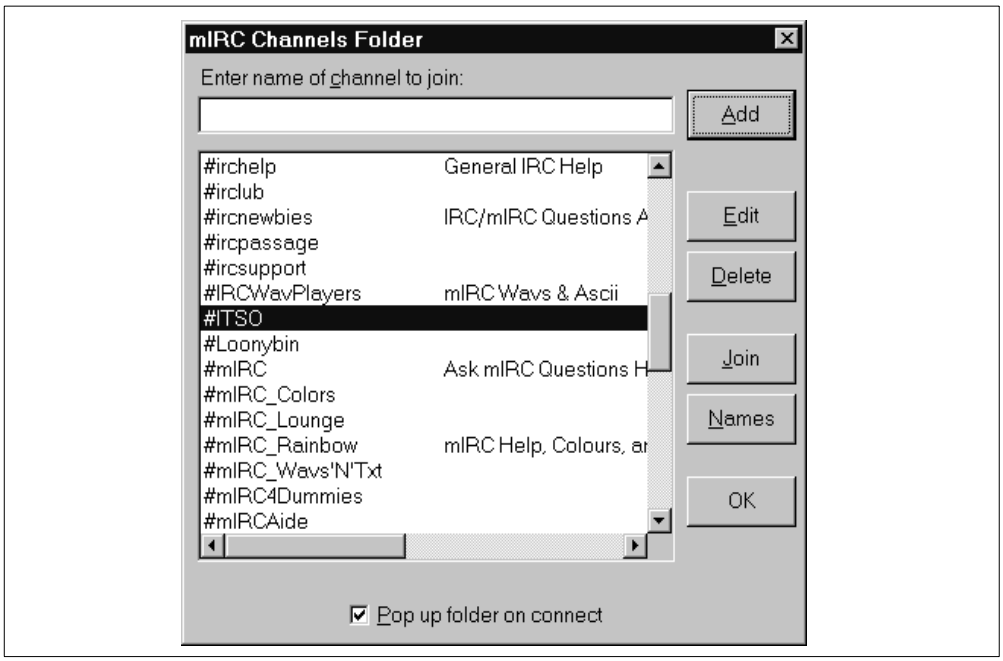


Figure 158. Adding and then joining an IRC channel

Once added, we could then Join the channel.

One of the frequently used features of IRC is to conduct private conversations. In the following example a user on another mIRC client wished to have such a conversation. To accomplish this, the user selected the **Chat** item from the **DCC** menu.

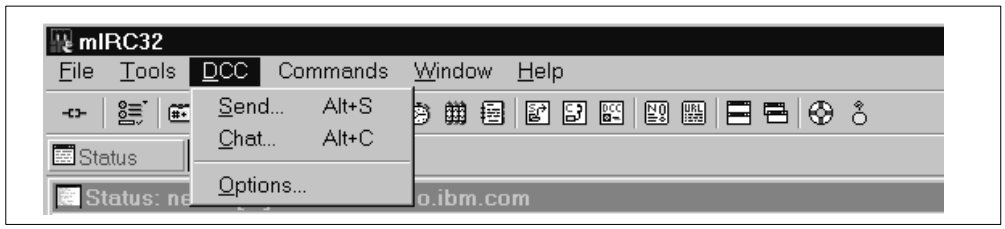


Figure 159. Establishing a private chat line: Request to chat

mIRC then asked which user was the target for such a request.

Figure 160. Establishing a private chat line: Selecting your partner

Once selected, a request is made to the other client via the IRC server. On the other user's PC, a message box appeared asking whether the private chat was to be accepted or ignored.

Figure 161. Establishing a private chat line: Your partner accepts

Once accepted, another window appears upon which all the private chat traffic is displayed and entered.

Figure 162. Establishing a private chat line: Start talking

7.2.7 Installing the Java IRC client

The client, as downloaded from IBM, had an expiration date of June 30, 1999. Therefore, reproduction of the exact steps involved in getting the client installed will not be useful. The broad steps involved are described here:

1. Download the code from the IBM Alphaworks site to a PC:
   `http://www.alphaWorks.ibm.com/tech/irc`
2. The client comes as a ZIP file which needs to be unzipped in its own directory on a PC (gzip is unable to process these files).
3. The directory structure was reproduced in the BFS.
4. Each of the files in each of the subdirectories was uploaded to the BFS.
5. A symbolic link was made from the directory to a personal home page area:
   `ln -s /usr/local/irc /home/neale/public_html/irc`

7.2.8 Tailoring the Java IRC client

Unlike the mIRC client where each PC user needs to perform the configuration, there only needs to be one update made. This change is made to the Web page that each PC client will request: `/usr/local/irc/irc.html`

```
    <applet code=room.IRC.class archive=irc.jar width=400 height=200>
    <!codebase="http"//venus.bocaraton.ibm.com/irc1_50/">
        <!frameName: the frame you want to display URLs>
        <param name=frameName value="URLs">
        <!server is your IRC server>
1       <param name=server value="totvm1.itso.ibm.com">
```

Figure 163. Updating the irc.htm with local details

Figure 163 note:

1. The IRC server's IP name or address is placed in this parameter.

When the browser was pointed at
`http://totvm1.itso.ibm.com/~neale/irc/irc.html`
the following screen was displayed:

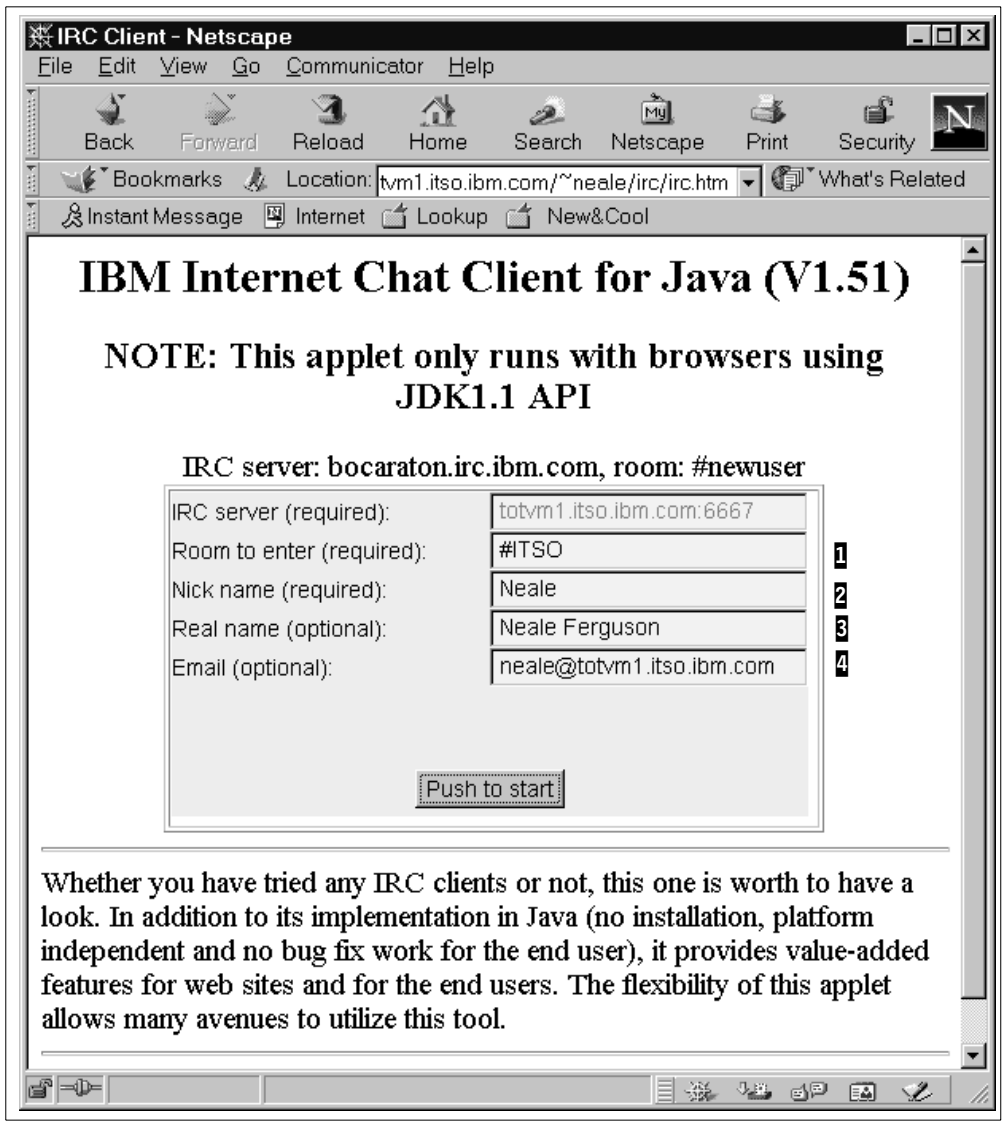


Figure 164. Using the IBM Alphaworks IRC Client via Apache Server

Figure 164 notes:

1. The channel we wished to join was specified here.
2. Any nickname will do.
3. This is used by the server when requests for user information are made.
4. The e-mail address is also used when requests for information about users are made.

When the fields were filled in, the **Push to start** button was pressed and the following was displayed:

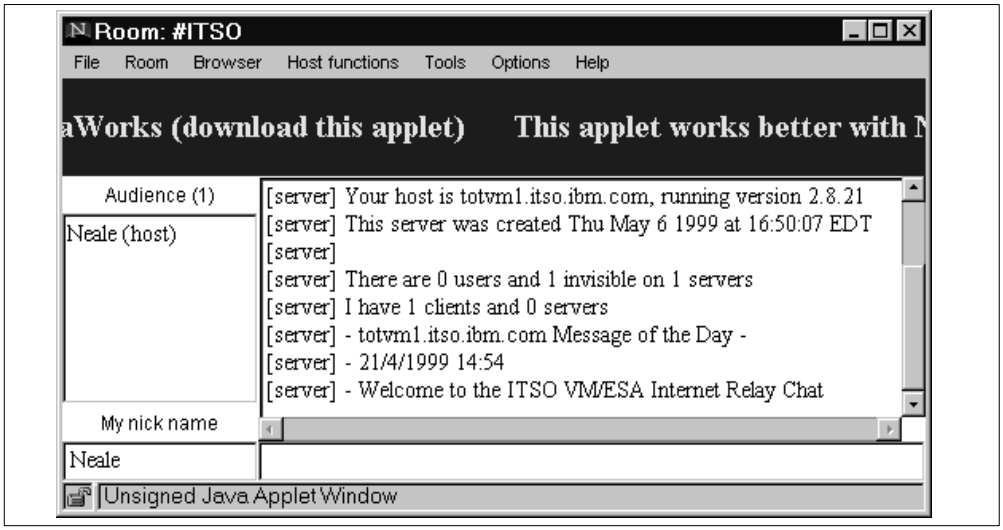


Figure 165. Using the IBM Alphaworks IRC client

7.3 MQM - MQSeries Client for Java

The MQSeries Client for Java provides a set of Java class libraries that permit Java applets on a Web browser, or stand-alone Java applets, to access MQSeries applications. MQSeries Client for Java is an MQSeries client written in the Java language for communicating via TCP/IP.

This package does not fit into the category of a ported application. It is another tool that is available to the VM programmer which can help leverage the investment the organization has made in VM. The MQ applets can now be served from the Apache Web server ported as part of this residency.

7.3.1 MQSeries Client for Java installation

The Java client requires no separate CP directory entry. To install:

1. Follow the FTP instructions in Appendix K, “General instructions for downloading packages” on page 311.
2. Move the mqm files to the production area:

   ```
   mv mqm /usr
   ```

7.3.2 Testing the MQSeries Client for Java

1. Define a server connection channel:
   a. Start your queue manager using the strmqm command.
   b. Type `runmsqsc` to start the runmqsc program.
   c. Define a channel by entering the following command:

   ```
   DEF CHL('JAVA.CHANNEL') CHLTYPE(SVRCONN) +
    TRPTYPE(TCP) MCAUSER(' ') +
    DESCRIPTION('Sample channel for MQSeries Client for Java')
   ```

2. Invoke the MQSeries Client for Java Installation and Verification Program and enter the parameters specified in bold type. (Note that, for some reason, the prompt appears only after you have entered the parameter!)

```
java MQIVP
MQSeries Client for Java Installation Verification Program
5639-C34 (C) Copyright IBM Corp. 1997 All Rights Reserved.
============================================================

WTSC58OE
Please enter the hostname of the machine running the MQServer :
1414
Please enter the port to connect to                           : (1414)
JAVA.CHANNEL
Please enter the server connection channel name               :
CSQ2
Please enter the queue manager name                           :

Testing MQSeries Client for Java v2.0


Verifying that host can be accessed.
If you are running a listener, the messages
  "Connection to host '<your host>' closed.",
  "The TCP/IP responder program could not be started."
may appear in its console during this test.

If the hostname is valid, but the host does not respond
to the connection request then you may have to wait for a
timeout to expire...

Success: Accessed host 'WTSC58OE'.

MQ testing will now begin...
If you are running this test over the web, it may take some time
to download the MQ classes, depending on the speed of your
link.

Success: Connected to queue manager.
Success: Opened SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Put a message to SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Got a message from SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Closed SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Disconnected from queue manager


Tests complete -
SUCCESS: Your MQSeries Client for Java is functioning correctly.
```

Figure 166. Testing the MQSeries Java Client

7.3.3 Using the MQSeries Client for Java with Apache

With Apache's inherent ability to access the BFS, the serving of Java applets was a simple task. The MQSeries Client for Java has a Web page defined in mqjavac.html that we used to initiate the applet on our browser. To do this:

1. Create a symbolic link from a user's personal Web area to the MQSeries code:

   ```
   ln -s /usr/mqm /home/neale/public_html/mqm
   ```

2. Point the browser to the following URL:

   ```
   http://totvm1.itso.ibm.com/~neale/mqm/mqjavac.html
   ```

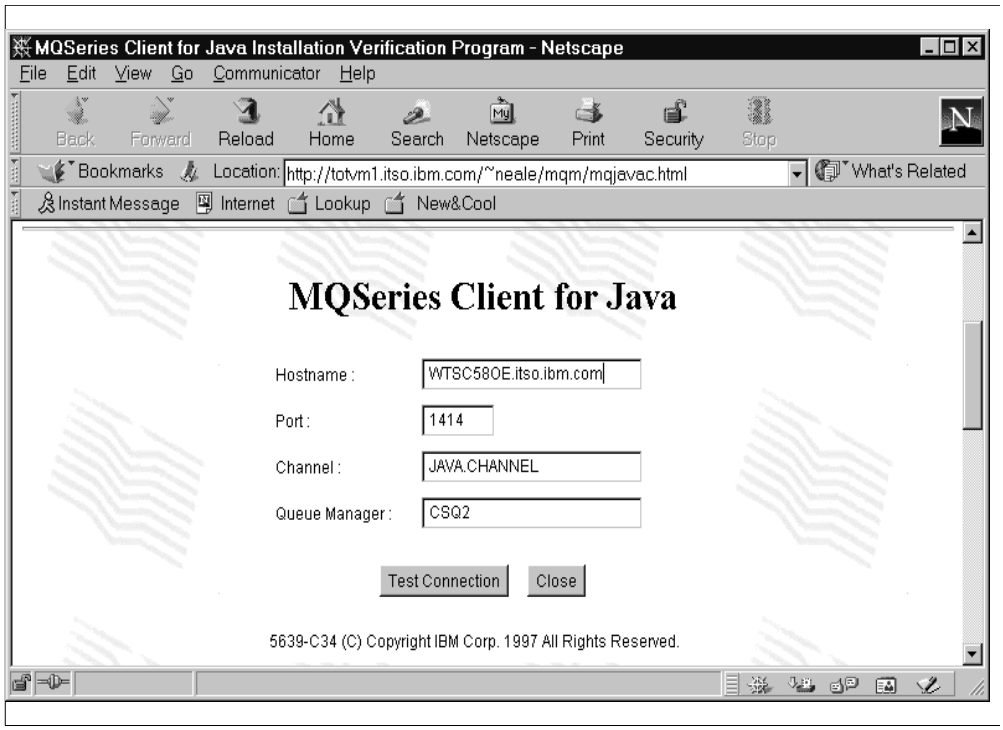


Figure 167. Using the MQSeries Java Client with Apache

There is a security restriction in the browser (both Netscape and Microsoft Internet Explorer) that requires that the queue manager must run on the same host as the Web server.

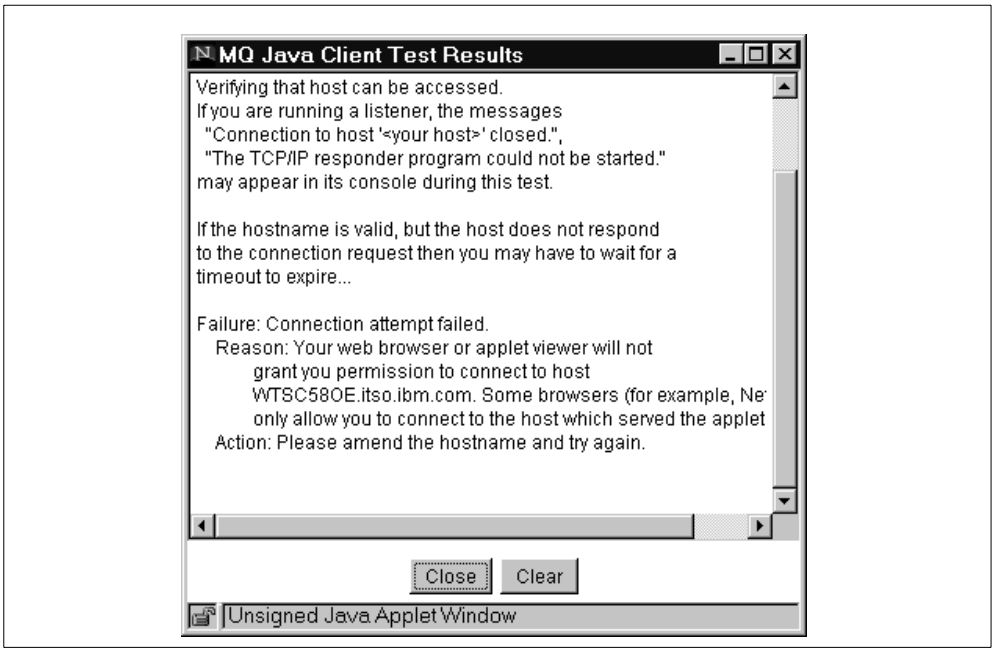


Figure 168. Error received when using MQSeries Java Client with Netscape

7.4 LIBASCII

The libascii package helps you port ASCII-based C applications to the EBCDIC-based S/390 UNIX environment. This package was originally written for OS/390, but works as well with VM/ESA. We have added several additional functions to the package and submitted these back to the package owner.

7.4.1 Overview

The C run-time library functions support EBCDIC characters. The libascii package provides an ASCII interface layer for some of the more commonly used C run-time library functions. libascii supports ASCII input and output characters by performing the necessary iconv() translations before and after invoking the C run-time library functions. Note that not all C functions are supported (see 7.4.1.7, “Limitations” on page 171 for additional information).

The C for VM/ESA V3.1 compiler predefined macro `__STRING_CODE_SET__="ISO8859-1"` generates ASCII characters rather than the default EBCDIC characters. Using this with the libascii code provides an ASCII-like environment.

The libascii package is as thread-safe as the run-time library except where stated under Limitations(see 7.4.1.7, “Limitations” on page 171).

You can download the package at:
`http://www.s390.ibm.com/products/oe/libascii.html`

The libascii source code is licensed by IBM; for more information, see the readme file in the package.

Note: If your browser does not work with the preceding link, use this URL instead:

`ftp://ftp.s390.ibm.com/u/ftp/os390/oe/toys/libascii.tar.Z`

To report problems or ask questions, send e-mail to libascii@vnet.ibm.com.

7.4.1.1 Floating point conversion

The libascii archive also contains four functions to convert between IEEE floating point and S/390 native hex floating point. The C compiler does not support IEEE floating point. Math operators such as + (add) and / (divide) and run-time library functions such as printf() and sin() using IEEE floating point numbers will produce undefined results. The main difference between the two floating point formats is IEEE supports a larger range of numbers, up to 10 to the 308 power. S/390 supports better precision but a smaller range, up to 10 to the 75 power.

The four floating point conversion routines included in libascii are:

- void ConvertFloatToIEEE(void *source, void *destination);
- void ConvertDoubleToIEEE(void *source, void *destination);
- void ConvertIEEEToFloat(void *source, void *destination);
- void ConvertIEEEToDouble(void *source, void *destination);

7.4.1.2 Installing the libascii code

The libascii TAR file contains all the parts required to create the libascii archive. To install:

1. Create a directory where you want libascii installed.
2. Copy the TAR file to that directory.
3. Unwind the file using this command:

   ```
   pax -rzf libascii.tar.Z
   ```

The libascii source files, makefile, and readme file should now be installed.

7.4.1.3 Building the libascii archive file (libascii.a)

You can use the makefile file provided to build libascii.a from the libascii source files.

To build libascii.a, cd into the directory with the libascii source files and issue the `make` command.

7.4.1.4 Building and running the samples

After you have made the libascii.a archive file, `cd` into the samples directory and issue the `make` command. This builds the samples.

To run the sample programs, issue these commands from the samples subdirectory:

```
sample1
sample2
```

7.4.1.5 Placing into Production

Once satisfied with the operation of the package, issue the `make install` command to place the objects (library and header files) into the production areas.

7.4.1.6 Using libascii

After you have installed libascii and built the archive file, to use the libascii functions you need to:

- Make minor modifications to your C source code and recompile, using the `__STRING_CODE_SET__="ISO8859-1"` predefined macro.
- Link-edit your application with the libascii.a archive file.

First, you need to determine which parts in your application will be compiled with `-D__STRING_CODE_SET__="ISO8859-1"` specified to generate ASCII strings. It is possible that part of the application will be compiled to generate ASCII strings and the other part will be compiled to generate EBCDIC strings.

For all parts compiled with the `-D__STRING_CODE_SET__="ISO8859-1"`, use libascii.

The following steps describe what to do with a program that will use the libascii functions:

1. Find all the C functions that require EBCDIC input [for example, fopen()] or produce EBCDIC output [for example, readdir()]. This grep command displays the file names that contain a libascii function followed by the character (:

   ```
   grep -l -f /the-libascii-directory/libascii.lst *.c
   ```

2. Determine which C functions you obtained are not supported by libascii. (See 7.4.1.7, “Limitations” on page 171 for additional information.) For any unsupported functions, you will need to either add functions to libascii or

change the code to add iconv() ASCII/EBCDIC conversions around the runtime library calls.

3. Make sure all the required header files are included, as specified in the *IBM C for VM/ESA Library Reference, Volume 2 Version 3 Release 1*, SC23-3908 . For example, strncasecmp() requires <strings.h> to be included.
4. Add the following statement to your source file, after all the existing #include statements:

   ```
   #include "_Ascii_a.h"
   ```

5. Recompile using the `-D__STRING_CODE_SET__="ISO8859-1"` option to cause the compiler to generate all strings defined in your program in ASCII rather than EBCDIC format.
6. Link-edit your application with the libascii.a archive file.

7.4.1.7 Limitations

The libascii interface package is code that we found useful in our IBM porting work, and it is offered *as is* for your use. It is intended to assist in getting your applications running as quickly as possible.

These are some of the known restrictions:

- Not all the C functions are supported. The file libascii.lst contains a list of supported ASCII run-time library routines.
- For some of the supported functions there are known restrictions, as follows:
  - **GETOPT function:**

    The libascii getopt() function is not thread-safe. The second argument is changed for a short period of time from EBCDIC to ASCII and then back to EBCDIC.

  - **PRINTF family of functions:**
    - The %$n specification is not supported in the format string.
    - These functions are limited to 2048 bytes of output. To increase the size of the strings and output supported, change #define MAXSTRING_a in global_a.h
  - **SCANF family of functions:**
    - The %$n specification is not supported in the format string.
    - The maximum number of arguments supported is 20.
    - These functions are limited to 2048 bytes of input. To increase the size of the strings and output supported, change #define MAXSTRING_a in global_a.h
- The interface layer is not NLS-enabled; it only supports conversions between the ISO8859-1 and IBM1047-1 character sets.

7.4.1.8 FAQs

Question 1: I compiled my code using `D__STRING_CODE_SET__="ISO8859-1"` and included the _Ascii_a.h. Why is the printf output to stdout unreadable?

Response: Not all the header files were included. For example, libascii requires <stdio.h> to be included to use the libascii version of printf(). If your application uses ctime() and you wish to use the libascii ctime(), then <time.h> must be included.

Another possible problem is _Ascii_a.h : it must be included after all the other header files.

Question 2: I compiled my code using `-D__STRING_CODE_SET__="ISO8859-1"` and included the _Ascii_a.h. Why does the link-edit of my application fail?

Response: The application's makefile needs to be modified to specify libascii.a on the link-edit.

Question 3: My link-edit shows libascii.a as not found, but I did install the libascii code. Why is this missing?

Response: Part of the installation of libascii includes compiling all the libascii code to create the libascii.a archive. Verify that the libascii.a archive has been created and it is in a directory that the linkage editor can find.

Chapter 8. Putting it all together

In this chapter we reflect on the process we have just completed and what we are left with when everything has been installed, configured, and is up and running. We believe this is not an end point, but rather a platform upon which complex and sophisticated environments can be built.

In addition we have brought together information about all the packages so that you can see how they relate to each other and what dependencies they have on other packages. This will enable you to better plan the way you want to implement the functions provided by the packages.

Finally, we look at what the next steps are and what work is involved to bring additional tools and applications to OpenEdition for VM/ESA.

8.1 A starting point

In this book we have attempted to guide you through the process of:

1. Establishing an OpenEdition environment for:
   - Program development and testing.
   - Running major applications.
2. "Bootstrapping" the porting process by installing the `gzip` and `make` tools.
3. Installing, configuring and using packages that form an infrastructure for program development and application support.
4. Installing, configuring and using important applications like LDAP and Samba.

We believe by following this process you will be able to provide your organization with a starting point from which you can add value to your business.

Table 5 shows the features provided by the major packages.

Table 5. Package features

Package	Description
APACHE	The world's most popular Web server, which provides access to the BFS and the serving of JAVA applets and the capability of supplementing the work performed by any existing Web server products.
DAYTIME	Clock synchronization and time-serving through an intranet. PC clients can now keep their clocks in synchronization with VM.
CROND	Job scheduler to help automate common tasks.
INETD	Super-server that automatically starts other servers on request (as it does for Samba and Apache).
INN	Internet (or intranet) News Server. This facilitates setting up discussion groups (private and public) within an organization.
IRC	Internet Relay Chat facilitates setting of a "party line," which is simpler than playing telephone tag and can supplement the work done on conference calls.
JACORB	This CORBA implementation facilitates the writing of distributed applications across the same or disparate platforms.

Package	Description
LDAP	This directory server will centralize an organization's directories and store configuration information in a central location.
LIBASCII	Simplifies the porting of heavily ASCII-dependent code.
MQM	The MQSeries Client for Java provides access to MQSeries servers via the Java language. This facility could be hooked up with Apache and JacORB to provide complex distributed applications.
RCS	Keeps source files under controlled access and provides the necessary audit trails to ensure the integrity of them.
REGINA	VM's favorite language with UNIX-specific functions for automation and applications.
SAMBA	LAN file-serving using high-performance and high-volume DASD that you already have backup, recovery and disaster recovery procedures for. Also, print serving via RSCS so that printers can live on different systems in different cities or countries.
SYSLOGD	Centralize console traffic from different applications and systems. Combine this with PROP to invoke action routines when certain messages are detected.

Using any, all, or a mixture of these applications should add significant value to your computing environment. We believe they meet the previously stated objective of leveraging you investment in skills, equipment, data, and applications.

These packages also serve as a starting point for building more extensive information systems. You will note that internet and Java technologies predominate the packages we have built, thus making VM/ESA an even better platform for electronic commerce.

8.2 Package dependencies

When installing the packages described in this redbook you can either:

1. Install all the packages by following the sequence of operations described in this book.
2. Choose specific application(s) and use the instructions provided in the book to install them.

For those who choose the second option, this section is designed to assist in the choice by describing the interrelationships (if any) of the packages. The relationship between the different packages and their dependencies (for example TCP/IP) are shown in Table 6.

Before installing one of the following packages, verify that the VMARC MODULE (on the CMS side), the GZIP package, the MAKE package, and the LIBXPG4 package are installed and ready to run.

Table 6. Package dependencies

Package	Description	Dependencies
APACHE	Web server	TCP/IP, INETD, browser
BYACC	A `yacc`-like utility	
DAYTIME	A time and date server installed with INETD or standalone	TCP/IP
CROND	Job scheduler	
FLEX	A fast lexical analyzer	
GDBM	GNU database manager	
GNUDIFF	Tools to compare two files	
GZIP	A compression tool	
INETD	Multi-threaded automatic server starter for INTERNET connection	TCP/IP, DAYTIME (optional)
INN	Internet News group server	TCP/IP, newsreader
IRC	Internet Relay Chat - a chat server	TCP/IP, IRC client
JACORB	A Java implementation of CORBA	TCP/IP, Java
LDAP	A directory service for your Web browser and applications	TCP/IP, browser, GDBM
LIBASCII	Library of routines to facilitate the porting of ASCII dependent code	
MAKE	The GNU make	
MQM	MQseries client for Java	TCP/IP, Web server, MQseries server, Java
RCS	A tool to manage multiple revisions of files	GNUDIFF
REGINA	A REXX under OpenEdition for VM/ESA	FLEX, BYACC
SAMBA	A file and print server	TCP/IP, INETD
SYSLOGD	A server to route messages, to consolidate logs from different platforms and to distribute specific messages.	PROP, TCP/IP

8.3 A view of the BFS directories

In Table 7 we give you a view of the existing directories in the BFS when all the ported packages are installed.

Table 7. View of BFS directories

Directories			Main files of the directory or explanations
level 1	level 2	level 3	
/bin			Directory containing the base commands, some of them are only a link to another command in another directory. compress, make, c89, cd, kill, paste, stty, x...
/dev			The driver directory null (driver equivalent to the pipe hole stage) tty (driver for the console)
/etc			The base config files startup.mk, profile...
	/samples		Some sample programs comics.lst, dc1.l, execute.c, dc1.y, hobbies, wc.l...
/lib			Link to directory /usr/lib
/opt			no files
/tmp			External link to /../VMBFS:BFSTEST:TMP
/u			Link to directory /home
/var			External link to /../VMBFS:BFSTEST:VAR

Directories			Main files of the directory or explanations
level 1	level 2	level 3	
/home			User's directory are under this directory
	/apache		Directory used by the Apache server
	/crond		Directory of crond server
	/inetd		Directory of inetd server
	/ircd		Directory of ircd server
	/jacorb		Directory used by jacorb server
	/ldap		Directory used by ldap server
	/mqm		Directory used by MQseries client
	/news		Directory used by news server
	/samba		Directory used by the Samba server
	/syslogd		Directory used by the syslogd server
	/porting		Directory used by porting userid to install `tar` files of the ported applications: gzip.tar make377.tar .profile ... source code is installed in the following subdirectories
		/apache_1.3.6	Apache package
		/byacc	Byacc package (used with Regina)
		/DAYTIME	Daytimed package

Directories			Main files of the directory or explanations
level 1	level 2	level 3	
/home	/porting	/dcron	Dcron package
		/flex-2.5.4	Flex package (used with Regina)
		/gdbm-1.7.3	Gdbm package (used with LDAP)
		/gnudiff-2.7	GNU diff package
		/gzip-1.2.4	GNU gzip package
		/INETD	Inetd package
		/inn-1.5.1	Inn package
		/irc2.8.21	Ircd package
		/JacORB_0.9e	JacOrb package
		/ldap-3.3	LDAP package
		/make-3.77	GNU make package
		/public_html	Personal page for Apache
		/RCS	RCS package
		/Regina-0.08c	Regina package
		/samba-1.9.18p10	Samba package
		/SYSLOGD	Syslogd package
		/XPG4	LIBXPG4 package
/usr			Contains the production code
	/bin		no files
	/include		The base header file ar.h, sysexits.h, syslog.h
		/sys	`time.h` is an external link to SYS_TIME H *
	/java		External link to java code /../VMBFS:VMSYS:_JAVA/./J1.1/
	/lib		`lex` library...
	/local		This directory and subdirectories are used by ported packages
		/apache	Directory for Apache code
		/bin	The new commands created by the ported package cmp, gunzip, ldapmodify, sdiff, zdiff, diff, gzexe, whoami...
		/etc	The different configuration files auth, ldapfilter.conf, rc.news, ldif2index, slapd.conf...
		/include	The package header files FlexLexer.h, gdbm.h, ldap.h, proto-ldap.h, rexxsaa.h...
		/info	Information files given with the packages ported
		/ircd	Configuration file used by ircd server

Directories			Main files of the directory or explanations
level 1	level 2	level 3	
/usr	/local	/irc	Java irc client code
		/lib	Libraries of LDAP, regina...
		/man	Directories for man files (help files)
		/news	News and log files
		/ORB	JacORB server code
		/rexx	REXX sample programs
		/samba	Directory containing Samba server code
		/src	no files
	/lpp		no files
	/mail		Link to /var/spool/mail
	/NetRexx		External link to /../VMBFS:VMSYS:_NETREXX/./NetRexx/
	/pub		no files
	/spool		Link to /var/spool
	/tmp		Directory used for temporary files
	/var		Directory for working files (news,logs...)

When you run `make install`, the objects are copied from your development environment (/home/porting) to:

- /usr/local/bin for the new command
- /usr/local/etc for the new config file
- /usr/local/lib for the new library
- /usr/local/man for the help files

8.4 The Way Ahead

In addition to the packages described in this document, a significant amount of work is being done on other major tools and applications. Once you establish your OpenEdition for VM/ESA environment and implement the infrastructure, you will be able to take advantage of the fruits of this work.

In the pipeline there are ports of the following under way:

- Perl 5. The modifications for VM/ESA are being incorporated into the base distribution. Once this is available, a significant number of Perl scripts used by Web servers will be usable by Apache under VM/ESA.
- OpenLDAP. The OpenLDAP organization is examining the modifications made for VM/ESA for incorporation into the base distribution.
- SSL: Secure Sockets Layer (port of SSLeay package).
- Socks Versions 4 and 5:
- PGP: Pretty Good Privacy.

- Samba Version 2.
- GNU C++.
- GNU utilities: file utilities, ld and gas.

Up to this point, you have followed our steps; the next steps will be yours. You have learned how to port an application on the VM platform, to install it and to get it running. It is now up to you to follow the VM Open way to your future.

Appendix A. Step-by-Step guide to porting Regina

This appendix provides a step-by-step guide to obtaining, changing, building, and installing the Regina package for VM/ESA.

A.1 Obtaining the package

1. FTP the archive from the PC:

```
ftp totvm1
Connected to totvm1.itso.ibm.com.
220-FTPSERVE IBM VM Level at TOTVM1.ITSO.IBM.COM, 11:26:45 EDT MONDAY
04/12/99
220 Connection will close if idle for more than 5 minutes.
User (totvm1.itso.ibm.com:(none)): porting
331 Send password please.
Password: ********
230 PORTING logged in; no working directory defined
ftp> cd /../VMBFS:BFSTEST:PORTING/
250 BFS working directory is /../VMBFS:BFSTEST:PORTING/
ftp> binary
200 Representation type is IMAGE.
ftp> put Regina-0.08c.tar.gz
200 Port request OK.
150 Storing file 'Regina-0.08c.tar.gz'
250 Transfer completed successfully.
1594080 bytes sent in 7.80 seconds (204.34 Kbytes/sec)
ftp> quit
221 Quit command received. Goodbye.
```

2. On VM enter the shell:

```
shell
```

3. Have a look for the file:

```
dir Regina-0.08c.tar.gz
-rw-rw-rw-   1 porting    system   1594080 Apr 14 10:17 Regina-0.08c.tar.gz
```

4. Unzip the archive:

```
gunzip Regina-0.08c.tar.gz
gunzip: Regina-0.08c.tar.gz: decompression OK, trailing garbage ignored
```

5. See what the unzip did:

```
dir Regina-0.08c.tar
-rw-rw-rw-   1 porting    system   6340096 Apr 14 10:17 Regina-0.08c.tar
```

6. Unarchive the files:

```
pax -rf Regina-0.08c.tar
```

7. See what the unarchive created:

```
ls -ld Regina-0.08c
drwxrwxrwx   1 porting    system         0 Apr 14 15:28 Regina-0.08c
```

8. Go to the directory just created:

```
cd Regina-0.08c
```

A.2 Making the changes

These sections examine the changes made to the Regina source code and why each of the changes was required.

A.2.1 Updating config.sub

This file contains all the valid platforms Regina can be built on. These changes add VM/ESA to that list.

1. Check in the config.sub file:

```
ci config.sub
config.sub,v  <--  config.sub
enter description, terminated with single '.' or end of file:
NOTE: This is NOT the log message!
>>
Original module
>>
.
initial revision: 1.1
done
```

2. Now check it out so we can make changes to it:

```
co config.sub
config.sub,v  -->  config.sub
revision 1.1
done
```

3. XEDIT the config.sub file and make the following changes as shown in the “diffs”:

```
***************
*** 689,695 ****
              | -ptx* | -coff* | -ecoff* | -winnt* | -domain* | -vsta* \
              | -udi* | -eabi* | -lites* | -ieee* | -go32* | -aux* \
              | -cygwin32* | -pe* | -psos* | -moss* | -proelf* | -rtems* \
!             | -linux-gnu* | -uxpv*)
        # Remember, each alternative MUST END IN *, to match a version number.
              ;;
        -linux*)
--- 689,695 ----
              | -ptx* | -coff* | -ecoff* | -winnt* | -domain* | -vsta* \
              | -udi* | -eabi* | -lites* | -ieee* | -go32* | -aux* \
              | -cygwin32* | -pe* | -psos* | -moss* | -proelf* | -rtems* \
!             | -linux-gnu* | -uxpv* | -vmesa*)
        # Remember, each alternative MUST END IN *, to match a version number.
              ;;
        -linux*)
***************
*** 908,913 ****
--- 908,916 ----
                                vendor=ns
                                ;;
                        -mvs*)
+                               vendor=ibm
+                               ;;
+                       -vmesa*)
                                vendor=ibm
                                ;;
                        -ptx*)
```

A.2.2 Modify the automatic configuration program

As with most UNIX packages, there is an automatic configuration script that will determine the APIs available on the platform and build appropriate header files

and C compiler flags. The configuration program for Regina assumes an ASCII character set so we had to make changes for our EBCDIC platform.

1. Check in the configuration program:

```
ci configure
configure,v  <--  configure
enter description, terminated with single '.' or end of file:
NOTE: This is NOT the log message!
>>
Original module
>>
.
initial revision: 1.1
done
```

2. Check out the configuration program:

```
co configure
configure,v  -->  configure
revision 1.1
done
```

3. Use XEDIT to make the changes to the program, as described by the following “diff” file:

```
***************
*** 1209,1215 ****
--- 1209,1221 ----
 #line 1210 "configure"
 #include "confdefs.h"
 #include <ctype.h>
+ #if 'M' == 0xd4
+ #define ISLOWER(c) (('a' <= (c) && (c) <= 'i') || \
+                     ('j' <= (c) && (c) <= 'r') || \
+                     ('s' <= (c) && (c) <= 'z'))
+ #else
 #define ISLOWER(c) ('a' <= (c) && (c) <= 'z')
+ #endif
 #define TOUPPER(c) (ISLOWER(c) ? 'A' + ((c) - 'a') : (c))
 #define XOR(e, f) (((e) && !(f)) || (!(e) && (f)))
 int main () { int i; for (i = 0; i < 256; i++)
```

Note:

1. This test determines if we are compiling on an EBCDIC or ASCII platform.

A.2.3 Run the automatic configuration program

1. The configuration command has many options and can be quite verbose. We suggest a file called config.vmesa be created and contain the following lines:

```
CPP="c89 -E -W c,nosource" \
CFLAGS="-W c,hwopts\\\(string\\\),langlvl\\\(ansi\\\) -D_OE_SOCKETS" \
LIBS="-l//posxsock -l//vmmtlib" \
./configure --host=i370-ibm-vmesa
```

2. Invoke the configuration program:

```
sh config.vmesa
creating cache ./config.cache
checking for one of the following C compilers: c89 acc gcc cc... using c89 specified in CC
env variable
checking for gcc... c89
checking whether the C compiler (c89 -W c,hwopts\\(string\\),langlvl\\(ansi\\) -
```

```
D_OE_SOCKETS ) works... yes
checking whether the C compiler (c89 -W c,hwopts\\(string\\),langlvl\\(ansi\\) -
D_OE_SOCKETS ) is a cross-compiler... no
checking whether we are using GNU C... no
checking for POSIXized ISC... no
checking if compiler supports ANSI prototypes... yes
checking host system type... i370-ibm-mvs
checking target system type... i370-ibm-mvs
checking build system type... i370-ibm-mvs
checking how to run the C preprocessor... c89 -E -W c,nosource
checking for ranlib... :
checking checking for gnu linker... not found
checking whether ln -s works... yes
checking whether make sets ${MAKE}... yes
checking for ANSI C header files... yes
checking for stdio.h... yes
checking for stdlib.h... yes
checking for setjmp.h... yes
checking for unistd.h... yes
checking for ctype.h... yes
checking for math.h... yes
checking for time.h... yes
checking for sys/time.h... yes
checking for assert.h... yes
checking for errno.h... yes
checking for stdarg.h... yes
checking for string.h... yes
checking for termios.h... yes
checking for signal.h... yes
checking for limits.h... yes
checking for fcntl.h... yes
checking for alloca.h... no
checking for malloc.h... no
checking for dlfcn.h... no
checking for dl.h... no
checking for putenv... yes
checking for usleep... yes
checking for random... yes
checking for ftruncate... yes
checking for memcpy... yes
checking for memmove... yes
checking for strerror... yes
checking for gettimeofday... yes
checking for working const... yes
checking for size_t... yes
checking whether time.h and sys/time.h may both be included... yes
checking whether struct tm is in sys/time.h or time.h... time.h
checking whether c89 understand -c and -o together... no
checking for 8-bit clean memcmp... yes
checking shared library support... done
checking for a BSD compatible install... ./install-sh -c
creating ./config.status
creating Makefile
creating config.h
```

A.2.4 Update the make file

Because we did not want to build all the “fancy” features of the Regina package, we do not need it to try (and fail) to install them. These modifications will simply comment out that part of the install procedure that refers to them.

1. Make a copy of the make file:

   cp Makefile makefile.vmesa

2. Check in the make file:

   ci makefile.vmesa

   ```
   makefile.vmesa,v  <--  makefile.vmesa
   enter description, terminated with single '.' or end of file:
   NOTE: This is NOT the log message!
   >>
   Original module
   >>
   ```

```
.
initial revision: 1.1
done
```

3. Check out the make file:

```
co makefile.vmesa
```

4. Use XEDIT to make the following changs to makefile.vmesa, as described in the following “diff” file:

```
***************
*** 584,596 ****
   ; $(INSTALL) -m 755 $(srcdir)/demo/newstr.rexx $(prefix)/rexx/newstr.rexx
   ; $(INSTALL) -m 755 $(srcdir)/demo/rexxcps.rexx $(prefix)/rexx/rexxcps.rexx
   ; $(INSTALL) -m 755 $(srcdir)/demo/timeconv.rexx $(prefix)/rexx/timeconv.rexx
!  ; $(INSTALL) -m 755 ./regina $(prefix)/bin/regina
!  ; $(INSTALL) -m 755 ./$(LIBPRE)$(SHLFILE).$(SHL).$(VERDOT) $(prefix)/lib/$
(LIBPRE)$(SHLFILE).$(SHL).$(VERDOT)
!  ; rm -f $(prefix)/lib/$(LIBPRE)$(SHLFILE).$(SHL).$(ABI)
!  ; $(LN_S) $(prefix)/lib/$(LIBPRE)$(SHLFILE).$(SHL).$(VERDOT) $(prefix)/lib
/$(LIBPRE)$(SHLFILE).$(SHL).$(ABI)
!  ; rm -f $(prefix)/lib/$(LIBPRE)$(SHLFILE).$(SHL)
!  ; $(LN_S) $(prefix)/lib/$(LIBPRE)$(SHLFILE).$(SHL).$(ABI)
$(prefix)/lib/$(LIBPRE)$(SHLFILE).$(SHL)
!  ; $(INSTALL) -m 755 ./curses.$(RXLIB) $(prefix)/lib/curses.$(RXLIB)
   ; $(INSTALL) -m 755 ./test1.$(RXLIB) $(prefix)/lib/test1.$(RXLIB)
   ; $(INSTALL) -m 755 ./test2.$(RXLIB) $(prefix)/lib/test2.$(RXLIB)
--- 584,596 ----
   ; $(INSTALL) -m 755 $(srcdir)/demo/newstr.rexx $(prefix)/rexx/newstr.rexx
   ; $(INSTALL) -m 755 $(srcdir)/demo/rexxcps.rexx $(prefix)/rexx/rexxcps.rexx
   ; $(INSTALL) -m 755 $(srcdir)/demo/timeconv.rexx $(prefix)/rexx/timeconv.rexx
! #; $(INSTALL) -m 755 ./regina $(prefix)/bin/regina
! #; $(INSTALL) -m 755 ./$(LIBPRE)$(SHLFILE).$(SHL).$(VERDOT)
$(prefix)/lib/$(LIBPRE)$(SHLFILE).$(SHL).$(VERDOT)
! #; rm -f $(prefix)/lib/$(LIBPRE)$(SHLFILE).$(SHL).$(ABI)
! #; $(LN_S) $(prefix)/lib/$(LIBPRE)$(SHLFILE).$(SHL).$(VERDOT)
$(prefix)/lib/$(LIBPRE)$(SHLFILE).$(SHL).$(ABI)
! #; rm -f $(prefix)/lib/$(LIBPRE)$(SHLFILE).$(SHL)
! #; $(LN_S) $(prefix)/lib/$(LIBPRE)$(SHLFILE).$(SHL).$(ABI)
$(prefix)/lib/$(LIBPRE)$(SHLFILE).$(SHL)
! #; $(INSTALL) -m 755 ./curses.$(RXLIB) $(prefix)/lib/curses.$(RXLIB)
  #; $(INSTALL) -m 755 ./test1.$(RXLIB) $(prefix)/lib/test1.$(RXLIB)
  #; $(INSTALL) -m 755 ./test2.$(RXLIB) $(prefix)/lib/test2.$(RXLIB)
```

5. Copy the modified makefile.vmesa over Makefile:

```
cp makefile.vmesa Makefile
```

A.2.5 Create yaccsrc.c & symbols.h from yaccsrc.y

As supplied, yaccsrc.c was prepared for an ASCII environment. By running the byacc command against yaccsrc.y, the resulting C source file is “EBCDIC-enabled”.

1. Invoke the byacc command:

```
yacc -ld yaccsrc.y
```

2. Copy yacc output:

```
cp y.tab.c yaccsrc.c
cp y.tab.h symbols.h
```

The file symbols.h is used by lexsrc.c to build the lexical analyser.

A.2.6 Update builtin.c

- This change removes an ASCII dependency. The initial test assumed a contiguous range and that “anding” a character with 0xdf would make it uppercase.

```
***************
*** 34,40 ****
--- 34,44 ----
  double pow( double, double ) ;
  #endif

+ #ifndef __OPEN_VM
  # define UPPERLETTER(a) ((((a)&0xdf)>='A')&&(((a)&0xdf)<='Z'))
+ #else
+ # define UPPERLETTER(a) (isupper((a)))
+ #endif
  #define NUMERIC(a) (((a)>='0')&&((a)<='9'))

  int contained_in( char *first, char *fend, char *second, char *send )
```

- This next change is purely cosmetic; we prefer to document what we are comparing with.

```
***************
*** 401,410 ****
          ptr = ptr ;
  #else
          char *val = getenv( string->value ) ;
!         if (val)
             ptr = Str_cre( val ) ;

!         if (value)
  #if defined(HAVE_PUTENV)
          {
  #if defined(FIX_PROTOS) && defined(ultrix)
--- 405,414 ----
          ptr = ptr ;
  #else
          char *val = getenv( string->value ) ;
!         if (val != NULL)
             ptr = Str_cre( val ) ;

!         if (value != NULL)
  #if defined(HAVE_PUTENV)
          {
  #if defined(FIX_PROTOS) && defined(ultrix)
```

- The next set of changes fix ASCII dependencies (note the use of 0xdf to convert to uppercase):

```
***************
*** 1072,1078 ****
--- 1076,1086 ----

     checkparam( parms, 1, 3 ) ;
     if ((parms->next)&&(parms->next->value))
+ #ifndef __OPEN_VM
        option = (getonechar(parms->next->value)) & (0xdf) ;
+ #else
+       option = toupper(getonechar(parms->next->value)) ;
+ #endif

     if ((parms->next)&&(parms->next->next)&&(parms->next->next->value))
        padch = getonechar(parms->next->next->value) ;
***************
*** 1481,1487 ****
--- 1489,1499 ----
     ref = parms->next->value ;
     if (parms->next->next) {
        if (parms->next->next->value) {
+ #ifndef __OPEN_VM
          ch = (*(parms->next->next->value->value)&0xdf) ;
+ #else
+         ch = toupper(*(parms->next->next->value->value)) ;
+ #endif
          if (ch=='M')
             inv = 1 ;
          else if (ch!='N')
***************
*** 1881,1887 ****
--- 1893,1903 ----
             for (; cptr<Str_end(string); cptr++)
             {
```

```
                ch = *cptr ;
+ #ifndef __OPEN_VM
                res &= ( ((ch<='z')&&(ch>='a')) || ((ch<='Z')&&(ch>='A'))
+ #else
+               res &= ( (isalpha(ch))
+ #endif
                           || ((ch<='9')&&(ch>='0')) || (ch=='.')
                           || (ch=='@') || (ch=='#') || (ch=='$')
                           || (ch=='?') || (ch=='_') || (ch=='!')) ;
```

A.2.7 Update files.c

- This change reflects the differences in the newline character in ASCII and EBCDIC.

```
***************
*** 317,323 ****
--- 317,327 ----
   * EOL-marker is a bit Unix-ish. We have to change this before porting
   * to MSDOS.
   */
+ #ifndef __OPEN_VM
  # define EOL (0x0a)
+ #else
+ # define EOL (0x15)
+ #endif

  /*
   * Regina truncates a file when repositioning by the use of a line
```

- This change is for POSIX compliance: if a programmer is attempting to use the LINEOUT() builtin function with a line number option, then a file needs to be opened in update mode (or it can be opened in write mode).

```
***************
*** 2846,2853 ****
     else
        lineno = 0 ;  /* illegal value */

!    if (string || lineno)
        ptr = get_file_ptr( file, OPER_WRITE, ACCESS_WRITE ) ;

     /*
      * First, let's reposition the file if necessary.
--- 2850,2859 ----
     else
        lineno = 0 ;  /* illegal value */

!    if (string && !lineno)
        ptr = get_file_ptr( file, OPER_WRITE, ACCESS_WRITE ) ;
+    else if (string && lineno)
+         ptr = get_file_ptr( file, OPER_WRITE, ACCESS_UPDATE ) ;

     /*
      * First, let's reposition the file if necessary.
```

- This change uses the definition of _CR made in the first modification to this module:

```
***************
*** 3281,3287 ****
     {
        if (i<BUFFERSIZE)
           source->value[i++] = ch = getc(stdin) ;
!    } while((ch!='\012')&&(ch!=EOF)) ;

     got_eof = (ch==EOF) ;
     source->len = i-1 ;
--- 3287,3293 ----
     {
        if (i<BUFFERSIZE)
           source->value[i++] = ch = getc(stdin) ;
!    } while((ch != _CR ) && (ch!=EOF)) ;

     got_eof = (ch==EOF) ;
     source->len = i-1 ;
```

A.2.8 Update funcs.c

- We added a new builtin function to Regina called getppid:

```
***************
*** 189,194 ****
--- 189,195 ----
  streng *unx_eof( paramboxptr parms ) ;
  streng *unx_popen( paramboxptr parms ) ;
  streng *unx_getpid( paramboxptr parms ) ;
+ streng *unx_getppid( paramboxptr parms ) ;
  streng *unx_fork( paramboxptr parms ) ;
  streng *unx_uname( paramboxptr parms ) ;
  #ifdef OLD_REGINA_FEATURES
```

- Owing to collating sequence differences, functions like B2X come after the other B*xx* functions and not before them (as they do in ASCII):

```
***************
*** 211,223 ****
--- 212,231 ----
        { 0,                    dbg_allocated, "ALLOCATED" },
  #endif
        { 0,                    std_arg, "ARG" },
+ #ifndef __OPEN_VM
        { 0,                    std_b2x, "B2X" },
+ #endif
        { 0,                    std_bitand, "BITAND" },
        { 0,                    std_bitor, "BITOR" },
        { 0,                    std_bitxor, "BITXOR" },
        { EXT_BUFTYPE_BIF,      cms_buftype, "BUFTYPE" },
+ #ifdef __OPEN_VM
+       { 0,                    std_b2x, "B2X" },
+ #endif
+ #ifndef __OPEN_VM
        { 0,                    std_c2d, "C2D" },
        { 0,                    std_c2x, "C2X" },
+ #endif
        { 0,                    unx_chdir, "CD" },
        { 0,                    std_center, "CENTER" },
        { 0,                    std_center, "CENTRE" },
***************
*** 233,240 ****
--- 241,254 ----
        { 0,                    std_condition, "CONDITION" },
        { 0,                    std_copies, "COPIES" },
       { 0,                  std_countstr, "COUNTSTR" },   /* ANSI Std 1996 - MH 10-06-96
*/
+ #ifdef __OPEN_VM
+       { 0,                    std_c2d, "C2D" },
+       { 0,                    std_c2x, "C2X" },
+ #endif
+ #ifndef __OPEN_VM
        { 0,                    std_d2c, "D2C" },
        { 0,                    std_d2x, "D2X" },
+ #endif
        { 0,                    std_datatype, "DATATYPE" },
        { 0,                    std_date, "DATE" },
        { 0,                    std_delstr, "DELSTR" },
***************
*** 247,252 ****
--- 261,270 ----
        { 0,                    dbg_dumptree, "DUMPTREE" },
        { 0,                    dbg_dumpvars, "DUMPVARS" },
  #endif
+ #ifdef __OPEN_VM
+       { 0,                    std_d2c, "D2C" },
+       { 0,                    std_d2x, "D2X" },
+ #endif
        { 0,                    unx_eof, "EOF" },
        { 0,                    std_errortext, "ERRORTEXT" },
  #ifdef VAXC
```

- Here is the definition of our new function getppid:

```
***************
*** 293,298 ****
--- 311,317 ----
```

```
        { 0,                    unx_getenv, "GETENV" },
        { 0,                    unx_getpath, "GETPATH" },
        { 0,                    unx_getpid, "GETPID" },
+       { 0,                    unx_getppid, "GETPPID" },
        { 0,                    cms_index, "INDEX" },
        { 0,                    std_insert, "INSERT" },
        { 0,                    cms_justify, "JUSTIFY" },
```

- More changes owing to the collating sequence differences between ASCII and EBCDIC.

```
***************
*** 354,363 ****
--- 373,389 ----
        { 0,                    std_wordlength, "WORDLENGTH" },
        { 0,                    std_wordpos, "WORDPOS" },
        { 0,                    std_words, "WORDS" },
+ #ifndef __OPEN_VM
        { 0,                    std_x2b, "X2B" },
        { 0,                    std_x2c, "X2C" },
        { 0,                    std_x2d, "X2D" },
+ #endif
        { 0,                    std_xrange, "XRANGE" },
+ #ifdef __OPEN_VM
+       { 0,                    std_x2b, "X2B" },
+       { 0,                    std_x2c, "X2C" },
+       { 0,                    std_x2d, "X2D" },
+ #endif
        { 0,                    NULL, NULL }
  } ;
```

A.2.9 Update interprt.c

- Fix the carriage-return differences:

```
***************
*** 541,547 ****
              if (stringen)
                 write( fileno(stdout), stringen->value, Str_len(stringen) ) ;

!             putchar( 0x0a ) ;
              fflush( stdout ) ;
           }

--- 541,547 ----
              if (stringen)
                 write( fileno(stdout), stringen->value, Str_len(stringen) ) ;

!             putchar( _CR ) ;
              fflush( stdout ) ;
           }
```

A.2.10 Update lexsc.l

- Fix the ASCII dependencies of contiguity and converting to uppercase:

```
***************
*** 328,335 ****

     j = 0 ;
     for (i=0;yytext[i];i++)
!       if ('a' <= yytext[i] && yytext[i] <= 'z')
!          retvalue[j++] = yytext[i] & 0xDF ;
        else if (yytext[i]!='=' && yytext[i]!='\t' && yytext[i]!='\n' && yytext[i]!=' ' &&
yytext[i]!='`')
           retvalue[j++] = yytext[i] ;
     retvalue[j] = 0x00 ;
--- 327,334 ----

     j = 0 ;
     for (i=0;yytext[i];i++)
!       if (islower(yytext[i]))
!          retvalue[j++] = toupper(yytext[i]);
        else if (yytext[i]!='=' && yytext[i]!='\t' && yytext[i]!='\n' && yytext[i]!=' ' &&
yytext[i]!='`')
           retvalue[j++] = yytext[i] ;
     retvalue[j] = 0x00 ;
```

```
***************
*** 531,538 ****
     BEGIN comm ;

     for (i=j=0;ch=yytext[i];i++) {
!       if ('a' <= ch && ch <= 'z')
!          retvalue[j++] = ch & 0xDF ;
        else if ((ch!=' ')&&(ch!=',')&&(ch!='\t')&&(ch!='\n')&&(ch!=':')&&(ch!='`'))
           retvalue[j++] = ch ; }
     retvalue[j] = 0x00 ;
--- 530,537 ----
     BEGIN comm ;

     for (i=j=0;ch=yytext[i];i++) {
!       if (islower(ch))
!          retvalue[j++] = toupper(ch);
        else if ((ch!=' ')&&(ch!=',')&&(ch!='\t')&&(ch!='\n')&&(ch!=':')&&(ch!='`'))
           retvalue[j++] = ch ; }
     retvalue[j] = 0x00 ;
More carriage return changes:
***************
*** 677,683 ****
     char1 = yytext[0] ;
     for (i=1; yytext[i+1]; i++)
     {
!       if (yytext[i]==0x0a)
           exiterror( ERR_UNMATCHED_QUOTE ) ;

        if (yytext[i]==char1 && yytext[i+1]==char1)
--- 676,682 ----
     char1 = yytext[0] ;
     for (i=1; yytext[i+1]; i++)
     {
!       if (yytext[i]==_CR)
           exiterror( ERR_UNMATCHED_QUOTE ) ;

        if (yytext[i]==char1 && yytext[i+1]==char1)
```

- More contiguity and uppercase changes:

```
***************
*** 709,716 ****
  ((([0-9]+\.|\.?[0-9])[0-9]*{e}(\-|\+)[0-9]+))|([.0-9][a-zA-Z0-9.$!?@#_]*) {
     BEGIN other ;
     for (i=0;yytext[i];i++)
!       if ('a' <= yytext[i] && yytext[i] <= 'z')
!          retvalue[i] = yytext[i] & 0xDF ;
        else
           retvalue[i] = yytext[i] ;
     retvalue[i] = 0x00 ;
--- 708,715 ----
  ((([0-9]+\.|\.?[0-9])[0-9]*{e}(\-|\+)[0-9]+))|([.0-9][a-zA-Z0-9.$!?@#_]*) {
     BEGIN other ;
     for (i=0;yytext[i];i++)
!       if (islower(yytext[i]))
!          retvalue[i] = toupper(yytext[i]);
        else
           retvalue[i] = yytext[i] ;
     retvalue[i] = 0x00 ;
```

- VM needs the meta-character ‘%’ escaped:

```
***************
*** 823,829 ****
  {bl}\/{bl} {
     return '/' ; }

! {bl}%{bl} {
     return '%' ; }

  {bl}\*{bl} {
--- 822,828 ----
  {bl}\/{bl} {
     return '/' ; }

! {bl}\%{bl} {
     return '%' ; }

  {bl}\*{bl} {
```

- Yet more contiguity and uppercase changes:

```
***************
*** 921,928 ****

     for (i=0;instr[i];i++) {
        ch = instr[i] ;
!       if ('a' <= ch && ch <= 'z')
!          instr[j++] = ch & 0xDF ;
        else if ((ch!=' ') && (ch!=',') && (ch!='\t') && (ch!='\n'))
           instr[j++] = ch ; }
     instr[j] = 0x00 ;
--- 920,927 ----

     for (i=0;instr[i];i++) {
        ch = instr[i] ;
!       if (islower(ch))
!          instr[j++] = toupper(ch);
        else if ((ch!=' ') && (ch!=',') && (ch!='\t') && (ch!='\n'))
           instr[j++] = ch ; }
     instr[j] = 0x00 ;
```

- A fix for the recursive interpretations (not sure it fixes a lot, though):

```
***************
*** 991,996 ****
--- 990,996 ----
     /* BUG. Fix this, or be hanged. Will not handle recursive
        interpretations. */
     bufptr=0;
+    YY_NEW_FILE;
  #ifdef FLEX_SCANNER
     yy_init = 1 ;
  #else
```

- More carriage return fixes:

```
***************
*** 1109,1115 ****
     extern int thischar ;
     static char thisline[BUFFERSIZE] ;
     int nextchar ;
!    static int  previous=0x0a ;
     extern int ipretflag ;
     extern char *interptr ;

--- 1109,1115 ----
     extern int thischar ;
     static char thisline[BUFFERSIZE] ;
     int nextchar ;
!    static int  previous=_CR ;
     extern int ipretflag ;
     extern char *interptr ;

***************
*** 1123,1129 ****
        if (interptr>=interptrmax)
        {
           myungetc(EOF,yyin) ;
!          nextchar = 0x0a ;
           cch++ ;
        }
        else
--- 1123,1129 ----
        if (interptr>=interptrmax)
        {
           myungetc(EOF,yyin) ;
!          nextchar = _CR ;
           cch++ ;
        }
        else
***************
*** 1132,1141 ****
     else
     {
        thisline[cch++] = nextchar = getc(str) ;
!       if ((previous!=0x0a) && (nextchar==EOF))
        {
           myungetc(EOF,yyin) ;
```

```
!          nextchar = 0x0a ;
        }
        previous = nextchar ;
     }
--- 1132,1141 ----
     else
     {
        thisline[cch++] = nextchar = getc(str) ;
!       if ((previous!=_CR) && (nextchar==EOF))
        {
           myungetc(EOF,yyin) ;
!          nextchar = _CR ;
        }
        previous = nextchar ;
     }
***************
*** 1218,1229 ****
        firstln = 1 ;
        if (this=='#')
        {
!          for (;(this!=0x0a) && (this!=EOF); this=mygetc(yyin) ) ;
           this = mygetc(yyin) ;
        }
     }

!    if ((this==0x0a) && (doblequote || singlequote))
        exiterror( ERR_UNMATCHED_QUOTE ) ;

     if ((this=='\'')&&(doblequote==0))
--- 1218,1229 ----
        firstln = 1 ;
        if (this=='#')
        {
!          for (;(this!=_CR) && (this!=EOF); this=mygetc(yyin) ) ;
           this = mygetc(yyin) ;
        }
     }

!    if ((this==_CR) && (doblequote || singlequote))
        exiterror( ERR_UNMATCHED_QUOTE ) ;

     if ((this=='\'')&&(doblequote==0))
```

- Still more contiguity and uppercase changes:

```
***************
*** 1277,1285 ****
     int i, j=0 ;

     for (i=0; text[i]; i++) {
!       if ('a' <= text[i] && text[i] <= 'z')
!          text[i] &= 0xDF ;
!       if (('A'<=text[i] && text[i]<='Z'))
           text[j++] = text[i] ; }

     text[j] = 0x00 ;
--- 1277,1285 ----
     int i, j=0 ;

     for (i=0; text[i]; i++) {
!       if (islower(text[i]))
!          text[i] = toupper(text[i]);
!       if (isupper(text[i]))
           text[j++] = text[i] ; }

     text[j] = 0x00 ;
```

A.2.11 Create lexsrc.c from lexsrc.l

```
flex -v -l8 -olexsrc.c lexsrc.l
flex version 2.5.4 usage statistics:
  scanner options: -lvI8 -Cem -olexsrc.c
  2741/3000 NFA states
  826/1000 DFA states (21166 words)
  108 rules
  Compressed tables always back-up
  15/40 start conditions
  1777 epsilon states, 1536 double epsilon states
```

```
44/100 character classes needed 715/750 words of storage, 334 reused
23500 state/nextstate pairs created
5429/18071 unique/duplicate transitions
840/1000 base-def entries created
3486/4000 (peak 4071) nxt-chk entries created
98/2500 (peak 770) template nxt-chk entries created
356 empty table entries
31 protos created
14 templates created, 285 uses
55/256 equivalence classes created
7/256 meta-equivalence classes created
16 (16 saved) hash collisions, 4634 DFAs equal
4 sets of reallocations needed
8963 total table entries needed
```

A.2.12 Update misc.c

- Character set dependency fixes:

```
***************
*** 70,75 ****
--- 70,76 ----

  unsigned char u_to_l[256] =
  {
+ #ifndef __OPEN_VM
      0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07,
      0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f,
      0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17,
***************
*** 102,107 ****
--- 103,142 ----
      0xe8, 0xe9, 0xea, 0xeb, 0xec, 0xed, 0xee, 0xef,
      0xf0, 0xf1, 0xf2, 0xf3, 0xf4, 0xf5, 0xf6, 0xf7,
      0xf8, 0xf9, 0xfa, 0xfb, 0xfc, 0xfd, 0xfe, 0xff
+ #else
+     0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07,
+     0x08, 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F,
+     0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17,
+     0x18, 0x19, 0x1A, 0x1B, 0x1C, 0x1D, 0x1E, 0x1F,
+     0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27,
+     0x28, 0x29, 0x2A, 0x2B, 0x2C, 0x2D, 0x2E, 0x2F,
+     0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37,
+     0x38, 0x39, 0x3A, 0x3B, 0x3C, 0x3D, 0x3E, 0x3F,
+     0x40, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47,
+     0x48, 0x49, 0x4A, 0x4B, 0x4C, 0x4D, 0x4E, 0x4F,
+     0x50, 0x51, 0x52, 0x53, 0x54, 0x55, 0x56, 0x57,
+     0x58, 0x59, 0x5A, 0x5B, 0x5C, 0x5D, 0x5E, 0x5F,
+     0x60, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67,
+     0x68, 0x69, 0x6A, 0x6B, 0x6C, 0x6D, 0x6E, 0x6F,
+     0x70, 0x71, 0x72, 0x73, 0x74, 0x75, 0x76, 0x77,
+     0x78, 0x79, 0x7A, 0x7B, 0x7C, 0x7D, 0x7E, 0x7F,
+     0x80, 0x81, 0x82, 0x83, 0x84, 0x85, 0x86, 0x87,
+     0x88, 0x89, 0x8A, 0x8B, 0x8C, 0x8D, 0x8E, 0x8F,
+     0x90, 0x91, 0x92, 0x93, 0x94, 0x95, 0x96, 0x97,
+     0x98, 0x99, 0x9A, 0x9B, 0x9C, 0x9D, 0x9E, 0x9F,
+     0xA0, 0xA1, 0xA2, 0xA3, 0xA4, 0xA5, 0xA6, 0xA7,
+     0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xAD, 0xAE, 0xAF,
+     0xB0, 0xB1, 0xB2, 0xB3, 0xB4, 0xB5, 0xB6, 0xB7,
+     0xB8, 0xB9, 0xBA, 0xBB, 0xBC, 0xBD, 0xBE, 0xBF,
+     0xC0, 0x81, 0x82, 0x83, 0x84, 0x85, 0x86, 0x87,
+     0x88, 0x89, 0xCA, 0xCB, 0xCC, 0xCD, 0xCE, 0xCF,
+     0xD0, 0x91, 0x92, 0x93, 0x94, 0x95, 0x96, 0x97,
+     0x98, 0x99, 0xDA, 0xDB, 0xDC, 0xDD, 0xDE, 0xDF,
+     0xE0, 0xE1, 0xA2, 0xA3, 0xA4, 0xA5, 0xA6, 0xA7,
+     0xA8, 0xA9, 0xEA, 0xEB, 0xEC, 0xED, 0xEE, 0xEF,
+     0xF0, 0xF1, 0xF2, 0xF3, 0xF4, 0xF5, 0xF6, 0xF7,
+     0xF8, 0xF9, 0xFA, 0xFB, 0xFC, 0xFD, 0xFE, 0xFF
+ #endif
  } ;
```

A.2.13 Update rexx.h

- Carriage-return fix:

```
***************
*** 42,47 ****
```

```
--- 42,53 ----
  #endif


+ #ifndef __OPEN_VM
+ # define _CR 0x0a
+ #else
+ # define _CR 0x15
+ # include <string.h>
+ #endif

  /*
   * define PATTERN_MEMORY to initiate newly allocated dynamic memory to
```

- Contiguity fix (0 to 9 comes before a to z in ASCII):

```
***************
*** 104,110 ****
--- 110,120 ----
  #define CHAR_OFFSET   32

  #define HEXNUM(c) (((c>='0')&&(c<='9'))||((c>='a')&&(c<='f')))
+ #ifndef __OPEN_VM
  # define HEXVAL(c) (((c)>'9')?(RXTOLOW(c)-'a'+10):((c)-'0'))
+ #else
+ # define HEXVAL(c) (isdigit((c))?((c)-'0'):(RXTOLOW(c)-'a'+10))
+ #endif

  #define PARSE_VERSION_STRING  "REXX-Regina_0_08c 4.50 23 Jul 1997" /* MH 10-03-97 */
```

A.2.14 Update shell.c

- Groundwork required to use spawn() API and POSIX threads:

```
***************
*** 45,50 ****
--- 45,55 ----
  #include <errno.h>
  #include <assert.h>
  #include <signal.h>
+ #ifdef __OPEN_VM
+ # include <spawn.h>
+ # include <pthread.h>
+   extern const char **environ;
+ #endif

  #if defined(VMS)
  # define fork() vfork()
***************
*** 132,137 ****
--- 137,147 ----
     streng *result=NULL ;
     char flags=' ' ;
     int count=0 ;
+ #ifdef __OPEN_VM
+    int fdMap[3];
+    struct inheritance inherit;
+    char *cmdArgs[4] = {"/bin/sh","-c",NULL,NULL};
+ #endif

     in = io_flags & REDIR_INPUT ;
     out = io_flags & REDIR_OUTLIFO ;
```

- Build the command arguments we will use when spawning (Str_ify is a Regina internal function used to create normal C strings from internal representation used by Regina):

```
***************
*** 152,158 ****
--- 162,171 ----
        cmd = *args ;
     }
     else
+    {
        args = NULL ;
+       cmdArgs[2] = Str_ify(command)->value;
+    }
```

```
     fflush( stdout ) ;
     fflush( stderr ) ;
```

- Avoid the fork() and set up for spawn() and call the appropriate API with the appropriate arguments depending on the environment Regina has current:

```
***************
*** 163,170 ****
--- 176,211 ----
     if (((out)||(fout))&&(pipe( fdout )))   /* for info from child to stack */
        perror("While opening output pipe") ;

+ #ifndef __OPEN_VM
     if ((child=fork()))
     {
+ #else
+    if (in)
+       fdMap[STDIN_FILENO] = fdin[0];
+    else
+       fdMap[STDIN_FILENO] = STDIN_FILENO;
+    if ((out) || (fout))
+       fdMap[STDOUT_FILENO] = fdout[1];
+    else
+       fdMap[STDOUT_FILENO] = STDOUT_FILENO;
+    fdMap[STDERR_FILENO] = STDERR_FILENO;
+    inherit.flags = 0;
+    if (envir == SUBENVIR_PATH)
+       child = spawnp(cmd, 3, fdMap, &inherit,
+                      (const char **) args, environ);
+    else
+       if (envir == SUBENVIR_COMMAND)
+          child = spawn(cmd, 3, fdMap, &inherit,
+                        (const char **) args, environ);
+       else
+          child = spawnp(cmdArgs[0], 3, fdMap, &inherit,
+                        (const char **) cmdArgs, environ);
+    if (child <= 0)
+       exiterror( ERR_SYSTEM_FAILURE ) ;
+    else
+    {
+       pthread_yield(0);
+ #endif
        signal( SIGPIPE, SIG_IGN ) ;

        if (out || fout)
```

- Avoid the OS/2 kludge:

```
***************
*** 212,218 ****
                in = 0 ;
                if ((out)||(fout))
                {
! #if !defined(VMS) && !defined(DOS) && !defined(__WATCOMC__) && !defined(_MSC_VER)  /* MH
10-06-96 */
  /*
   * Klugde for OS/2 ... see below
   */
--- 253,259 ----
                in = 0 ;
                if ((out)||(fout))
                {
! #if !defined(VMS) && !defined(DOS) && !defined(__WATCOMC__) && !defined(_MSC_VER)  &&
!defined(__OPEN_VM)
  /*
   * Klugde for OS/2 ... see below
   */
```

- Fix carriage return dependency:

```
***************
*** 227,233 ****
                if (!string)
                {
                   rstring = string = Str_ify(popline()) ;
!                  string->value[length=Str_len(string)] = 0x0a ;
                   string->len++ ;
                   length++ ;
```

```
                }
--- 268,274 ----
                if (!string)
                {
                   rstring = string = Str_ify(popline()) ;
!                  string->value[length=Str_len(string)] = _CR ;
                   string->len++ ;
                   length++ ;
                }
```

- Avoid any OS/2 kludges (again):

```
***************
*** 297,303 ****
                close( fdout[0] ) ;
                if (in)
                {
! #if !defined(VMS) && !defined(DOS) && !defined(__WATCOMC__) && !defined(_MSC_VER)  /* MH
10-06-96 */
  /*
   * OS/2 haven't defined fcntl() correctly, so we try to make it
   * work, by adding an extra parameter ... sigh!
--- 338,344 ----
                close( fdout[0] ) ;
                if (in)
                {
! #if !defined(VMS) && !defined(DOS) && !defined(__WATCOMC__) && !defined(_MSC_VER)  &&
!defined(__OPEN_VM)
  /*
   * OS/2 haven't defined fcntl() correctly, so we try to make it
   * work, by adding an extra parameter ... sigh!
```

- More carriage return-related processing:

```
***************
*** 316,322 ****
                if (cbuff[i]==0x0d)
                   cbuff[i] = ' ' ;

!               if (cbuff[i]!=0x0a)
                   continue ;

                if (istring)
--- 357,363 ----
                if (cbuff[i]==0x0d)
                   cbuff[i] = ' ' ;

!               if (cbuff[i]!=_CR)
                   continue ;

                if (istring)
```

- fork()-related code avoidance:

```
***************
*** 407,413 ****
  #endif
        signal( SIGPIPE, SIG_DFL ) ;
      }
!
     else
     {
       if (in)
--- 448,454 ----
  #endif
        signal( SIGPIPE, SIG_DFL ) ;
      }
! #ifndef __OPEN_VM
     else
     {
       if (in)
***************
*** 457,462 ****
--- 498,504 ----

       exit( -2 ) ;
    }
+ #endif /* OPEN_VM */
```

```
    if (args)
       destroyargs( args ) ;
```

- Does this carriage return stuff ever end?

```
***************
*** 546,552 ****
        }

      for (i=0; (cbuffer[i])&&(i<chars_read); i++)
!         if ((cbuffer[i]==0x0a)||(cbuffer[i]==0x0d))
            cbuffer[i] = '\0' ;

      nresult = Str_make((length=((result)?(Str_len(result)):0))+chars_read+1) ;
--- 588,594 ----
        }

      for (i=0; (cbuffer[i])&&(i<chars_read); i++)
!         if ((cbuffer[i]==_CR)||(cbuffer[i]==0x0d))
            cbuffer[i] = '\0' ;

      nresult = Str_make((length=((result)?(Str_len(result)):0))+chars_read+1) ;
```

- More spawn() setup (file descriptor maps, inheritance structures, and command to spawn):

```
***************
*** 614,619 ****
--- 656,666 ----
     int length=0 ;
     int envirno=0 ;
     int fd[2], print() ;
+ #ifdef __OPEN_VM
+    int fdMap[3];
+    struct inheritance inherit;
+    char *cmdArgs[4] = {"/bin/sh","-c",NULL,NULL};
+ #endif

     fflush(stdout) ;
     fflush(stderr) ;
```

- Spawn appropriate command with appropriate parameters depending on Regina environment:

```
***************
*** 641,647 ****
--- 688,728 ----
     }

     pipe(fd) ;
+ #ifndef __OPEN_VM
     if ((child=fork())) {
+ #else
+    cmd = NULL;
+    if (envirno != SUBENVIR_SYSTEM)
+    {
+       command = Str_ify(command);
+       args    = makeargs(command);
+       cmd     = *args;
+    }
+    else
+    {
+       args       = NULL ;
+       cmdArgs[2] = Str_ify(command)->value;
+    }
+    fdMap[STDIN_FILENO]  = STDIN_FILENO;
+    fdMap[STDOUT_FILENO] = fd[1];
+    fdMap[STDERR_FILENO] = STDERR_FILENO;
+    inherit.flags = 0;
+    if (envirno == SUBENVIR_PATH)
+       child = spawnp(cmd, 3, fdMap, &inherit,
+                      (const char **) args, environ);
+    else
+       if (envirno == SUBENVIR_COMMAND)
+          child = spawn(cmd, 3, fdMap, &inherit,
+                        (const char **) args, environ);
+       else
+          child = spawnp(cmdArgs[0], 3, fdMap, &inherit,
```

```
+                          (const char **) cmdArgs, environ);
+    if (child <= 0)
+       rc = -1;
+    else
+    {
+       pthread_yield(0);
+ #endif
        close(fd[1]) ;
        rsize = 0 ;
  #if defined(PIPE_BUF)
```

- More carriage return fixing:

```
***************
*** 657,666 ****
        cbuffer = Malloc(pipe_buf+1) ;

        for (;;) {
           rcode = read( fd[0], cbuffer, pipe_buf ) ;
-
           for (i=0; (cbuffer[i])&&(i<rcode); i++)
!             if ((cbuffer[i]==0x0a)||(cbuffer[i]==0x0d))
                 cbuffer[i] = ' ' ;

           if ((rcode>=0) || ((rcode==(-1)) && (errno==EINTR))) /* MH 10-06-96 */
--- 738,747 ----
        cbuffer = Malloc(pipe_buf+1) ;

        for (;;) {
+          pthread_yield(0);
           rcode = read( fd[0], cbuffer, pipe_buf ) ;
           for (i=0; (cbuffer[i])&&(i<rcode); i++)
!             if ((cbuffer[i]==_CR)||(cbuffer[i]==0x0d))
                 cbuffer[i] = ' ' ;

           if ((rcode>=0) || ((rcode==(-1)) && (errno==EINTR))) /* MH 10-06-96 */
```

- fork()-related processing avoidance:

```
***************
*** 720,725 ****
--- 801,807 ----
           rc = -1 ;
  #endif
     }
+ #ifndef __OPEN_VM
     else
     {
        dup2(fd[1],1) ;
***************
*** 770,775 ****
--- 852,858 ----

       exit( -2 ) ;
     }
+ #endif /* OPEN_VM */

  /*  should this be set ?  */
     {
```

A.2.15 Update stack.c

- Carriage return fix:

```
***************
*** 450,456 ****
              putc( (isprint(*cptr) ? (*cptr) : '?'), stddump ) ;

           putc( '"', stddump ) ;
!          putc( 0x0a, stddump ) ;
        }
        else
           fprintf(stddump,"==> Buffer: %d\n",--counter) ; }
--- 450,456 ----
              putc( (isprint(*cptr) ? (*cptr) : '?'), stddump ) ;

           putc( '"', stddump ) ;
!          putc( _CR, stddump ) ;
```

```
        }
        else
           fprintf(stddump,"==> Buffer: %d\n",--counter) ; }
```

A.2.16 Update strmath.c

- VM C compiler is very particular about unsigned and signed characters:

```
***************
*** 377,383 ****
   */
  void string_add( num_descr *f, num_descr *s, num_descr *r )
  {
!    static char *out=NULL ;
     static int outsize= 0 ;
     int count=0, carry=0, tmp=0, sum=0, neg=0 ;
     int lsd=0 ; /* least significant digit */
--- 377,383 ----
   */
  void string_add( num_descr *f, num_descr *s, num_descr *r )
  {
!    static unsigned char *out=NULL ;
     static int outsize= 0 ;
     int count=0, carry=0, tmp=0, sum=0, neg=0 ;
     int lsd=0 ; /* least significant digit */
```

- Temporary variable required for later use:

```
***************
*** 1559,1564 ****
--- 1559,1567 ----
     static int outsize= 0 ;
     char *outp=NULL, *answer=NULL ;
     int i=0, sskip=0, fskip=0, sstart=0, fstart=0, base=0, offset=0, carry=0, j=0 ;
+ #ifdef __OPEN_VM
+    short outData;
+ #endif

     IS_AT_LEAST( out, outsize, 2*(currlevel->currnumsize+1) ) ;
  #ifdef TRACEMEM
```

- This was to avoid a bug with the C compiler that has subsequently been APARed (PTF UM29038 unavailable at time of writing):

```
***************
*** 1584,1602 ****
        {
           answer = mult[f->num[j]-'0'][s->num[i]-'0'] ;
           assert( base-offset >= 0 ) ;
           out[base-offset] += answer[1] - '0' + carry ;
           carry = answer[0] - '0' ;
           for (; out[base-offset]>'9'; )
           {
              out[base-offset] -= 10 ;
              carry++ ;
           }
           offset++ ;
        }
        if (base-offset >= 0)
- {
           out[base-offset++] = carry + '0' ;
- }
        else
           exiterror( ERR_INTERPRETER_FAILURE ) ;

--- 1587,1616 ----
        {
           answer = mult[f->num[j]-'0'][s->num[i]-'0'] ;
           assert( base-offset >= 0 ) ;
+ #ifndef __OPEN_VM
           out[base-offset] += answer[1] - '0' + carry ;
+ #else
+          outData = out[base-offset] + answer[1] - '0' + carry ;
+ #endif
           carry = answer[0] - '0' ;
+ #ifndef __OPEN_VM
           for (; out[base-offset]>'9'; )
           {
```

```
               out[base-offset] -= 10 ;
               carry++ ;
            }
+ #else
+           for (; outData > '9'; )
+           {
+              outData -= 10 ;
+              carry++ ;
+           }
+           out[base-offset] = outData;
+ #endif
            offset++ ;
         }
         if (base-offset >= 0)
            out[base-offset++] = carry + '0' ;
         else
            exiterror( ERR_INTERPRETER_FAILURE ) ;
Update tracing.c
Another carriage return related fix:
***************
*** 94,100 ****
     if (rc==HOOK_GO_ON)
     {
        write( fileno(stderr), message->value, message->len ) ;
!       putc( 0x0a, stderr ) ;
     }
  }

--- 94,100 ----
     if (rc==HOOK_GO_ON)
     {
        write( fileno(stderr), message->value, message->len ) ;
!       putc( _CR, stderr ) ;
     }
  }
```

A.2.17 Update unxfuncs.c

- The code that performs the GETPPID() builtin function we had added:

```
***************
*** 124,129 ****
--- 124,136 ----
  }


+ streng *unx_getppid( paramboxptr parms )
+ {
+    checkparam( parms, 0, 0 ) ;
+    return int_to_streng( getppid() ) ;
+ }
+
+
  streng *unx_eof( paramboxptr parms )
  {
     checkparam( parms, 0, 0 ) ;
fork() avoidance (that is, Regina will not support the FORK() builtin function under VM):
***************
*** 182,188 ****
     int i=0 ;

     checkparam( parms, 0, 0 ) ;
! #if !defined(__WATCOMC__) && !defined(_MSC_VER)         /* MH 10-06-96 */
     i = fork() ;
  #endif                                                 /* MH 10-06-96 */
     return int_to_streng( i ) ;
--- 189,195 ----
     int i=0 ;

     checkparam( parms, 0, 0 ) ;
! #if !defined(__WATCOMC__) && !defined(_MSC_VER) && !defined(__OPEN_VM)
     i = fork() ;
  #endif                                                 /* MH 10-06-96 */
     return int_to_streng( i ) ;
```

A.2.18 Update variable.c

- Character set dependency fix:

```
***************
*** 291,296 ****
--- 291,297 ----
  #define RXISCOMMA(a) (char_types[(unsigned char)(a)]&0x10)

  char char_types[256] = {
+ #ifndef __OPEN_VM
     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,   /* nul - bel */
     0x00, 0x20, 0x20, 0x00, 0x20, 0x00, 0x00, 0x20,   /* bs  - si  */
     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,   /* dle - etb */
***************
*** 299,305 ****
     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x10, 0x00,   /*  (  -  /  */
     0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01,   /*  0  -  7  */
     0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x08,   /*  8  -  ?  */
!    0x08, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02,   /*  @  -  G  */
     0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02,   /*  H  -  O  */
     0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02,   /*  P  -  W  */
     0x02, 0x02, 0x02, 0x00, 0x00, 0x00, 0x00, 0x08,   /*  X  -  _  */
--- 300,306 ----
     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x10, 0x00,   /*  (  -  /  */
     0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01,   /*  0  -  7  */
     0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x08,   /*  8  -  ?  */
!    0x08, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02,   /*  @@  -  G  */
     0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02,   /*  H  -  O  */
     0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02,   /*  P  -  W  */
     0x02, 0x02, 0x02, 0x00, 0x00, 0x00, 0x00, 0x08,   /*  X  -  _  */
***************
*** 323,328 ****
--- 324,363 ----
     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
+ #else
+    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
+    0x00, 0x00, 0x20, 0x00, 0x20, 0x00, 0x00, 0x20,
+    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
+    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
+    0x00, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00,
+    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
+    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
+    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
+    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
+    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
+    0x00, 0x20, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00,
+    0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
+    0x00, 0x00, 0x00, 0x08, 0x00, 0x00, 0x00, 0x00,
+    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
+    0x08, 0x01, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00,
+    0x00, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
+    0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
+    0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
+    0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
+    0x00, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
+    0x04, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00,
+    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
+    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04,
+    0x08, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02,
+    0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
+    0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02,
+    0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
+    0x00, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02,
+    0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
+    0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01,
+    0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
+ #endif
  } ;

  /*
Carriage return related fix:
***************
*** 334,340 ****
     cptr = name->value ;
```

```
     eptr = cptr + name->len ;

!    if (cptr==eptr || (char_types[*cptr++] & (~0x0a)))
        return 0 ;

     stem = 0 ;
--- 369,375 ----
     cptr = name->value ;
     eptr = cptr + name->len ;

!    if (cptr==eptr || (char_types[*cptr++] & (~_CR)))
        return 0 ;

     stem = 0 ;
```

A.2.19 Update wrappers.c

- DLL support:

```
***************
*** 75,80 ****
--- 75,85 ----
  #  include <sys/ldr.h>
     typedef void *handle_type ;

+ # elif defined(__OPEN_VM)
+ #  include <sys/types.h>
+ #  include <dll.h>
+    typedef void *handle_type ;
+
  # elif defined(DYNAMIC_HPSHLOAD)
  #  include <dl.h>
     typedef shl_t handle_type ;
***************
*** 163,168 ****
--- 168,179 ----
       {
        set_err_message("LoadLibrary failed");
       }
+ #elif defined(__OPEN_VM)
+    handle = dllload( new_module->value ) ;
+    if (handle==NULL)
+      {
+       set_err_message( strerror (errno) );
+      }
  #endif

     if (new_module != module)
***************
*** 241,246 ****
--- 252,261 ----
        set_err_message( "DosQueryProcAddr() failed") ;
  #elif defined(DYNAMIC_WIN32)
     addr = (PFN)GetProcAddress(handle,(HMODULE)newname->value);
+    if (addr == NULL)
+       set_err_message( strerror(errno)) ;
+ #elif defined(__OPEN_VM)
+    addr = dllqueryfn(handle, newname->value);
     if (addr == NULL)
        set_err_message( strerror(errno)) ;
  #endif
```

A.2.20 Build the package

Building is as simple as entering the command `make`:

```
make
```

A.2.21 Install the binaries

```
make install
./install-sh -c -m 644 -m 755 -d /usr/local/bin
./install-sh -c -m 644 -m 755 -d /usr/local/lib
./install-sh -c -m 644 -m 755 -d /usr/local/include
```

```
./install-sh -c -m 644 -m 755 -d /usr/local/rexx
./install-sh -c -m 755 ./rexx /usr/local/bin/rexx
./install-sh -c -m 755 ./libregina.a /usr/local/lib/libregina.a
: /usr/local/lib/libregina.a
./install-sh -c -m 744 ./rexx.1 /usr/local/man/man1/rexx.1
./install-sh -c -m 744 ./rexxsaa.h /usr/local/include/rexxsaa.h
./install-sh -c -m 755 ./demo/rexxcps.rexx  /usr/local/rexx/rexxcps.rexx
./install-sh -c -m 755 ./demo/animal.rexx   /usr/local/rexx/animal.rexx
./install-sh -c -m 755 ./demo/block.rexx    /usr/local/rexx/block.rexx
./install-sh -c -m 755 ./demo/dateconv.rexx /usr/local/rexx/dateconv.rexx
./install-sh -c -m 755 ./demo/dynfunc.rexx  /usr/local/rexx/dynfunc.rexx
./install-sh -c -m 755 ./demo/hanoi.rexx    /usr/local/rexx/hanoi.rexx
./install-sh -c -m 755 ./demo/newstr.rexx   /usr/local/rexx/newstr.rexx
./install-sh -c -m 755 ./demo/rexxcps.rexx  /usr/local/rexx/rexxcps.rexx
./install-sh -c -m 755 ./demo/timeconv.rexx /usr/local/rexx/timeconv.rexx
./install-sh -c -m 755 ./test1.rxlib /usr/local/lib/test1.rxlib
install:  ./test1.rxlib does not exist
make: *** [install] Error 1
```

Error for test1.rxlib is acceptable.

Appendix B. BFS LOADBFS and SHELL LOADBFS

The files used to build the BFS and the Shell and Utilities are shown in this appendix.

B.1 BFS LOADBFS

This section contains the file used to load the Byte File System. It is the Byte File System described in 2.3, “Creating the BFS server” on page 16.

```
*********************************************************************
* LOADBFS file for base BFS structure
* COPYRIGHT -
*
*   5654-030 (C) COPYRIGHT IBM CORP. - 1995
*   LICENSED MATERIALS - PROPERTY OF IBM
*   SEE COPYRIGHT INSTRUCTIONS, G120-2083
*   ALL RIGHTS RESERVED.
*
* STATUS - VM/ESA Version 2, Release 1.0
*********************************************************************

*********************************************************************
* set defaults for owner, group, and permission bits
*********************************************************************

DEFAULT_OWNER           bin
DEFAULT_GROUP           bin
DEFAULT_PERMISSION      -rwxr-xr-x

*********************************************************************
* BFS filespaces to create                                          *
* ENROLL filespace owner group perm (filepool storagegroup blocks   *
*********************************************************************

UNMOUNT                 /

ENROLL ROOT       root     system -rwxrwxrwx ( BFSTEST 2 20000

ENROLL TMP        root     system -rwxrwxrwx ( BFSTEST 2 5000

ENROLL VAR        root     system -rwxrwxrwx ( BFSTEST 2 5000

*********************************************************************
* Mount the filespace to work in
* syntax:  MOUNT pathname1 pathname2                                *
*********************************************************************

MOUNT             /../VMBFS:BFSTEST:ROOT/ /

*****************************************************
* Directories to create                             *
* syntax:  DIR path owner group permission          *
* use '.' to mean 'take the default'                *
*****************************************************

DIR /dev                    root     system   .

DIR /etc                    root     system   .

DIR /home                   root     system   .

DIR /opt                    root     system   .

DIR /usr                    .        .        .
DIR /usr/bin                .        .        .
DIR /usr/include            .        .        .
DIR /usr/include/sys        .        .        .
DIR /usr/lib                .        .        .
DIR /usr/lib/nls            .        .        .
DIR /usr/lib/nls/msg        .        .        .
DIR /usr/lib/nls/msg/En_US.IBM-1047  . .      r-xr-xr-x
DIR /usr/lib/nls/msg/C.IBM-1047      . .      r-xr-xr-x
```

```
DIR /usr/lpp                .        .        .
DIR /usr/pub                root     system   .

*************************************************************************
* mount external links for these directories.we put them in separate *
* filespaces.                                                           *
* syntax:  EXTLINK path type owner group permission (string          *
*************************************************************************

EXTLINK /tmp  MOUNT    root     system   .   ( /../VMBFS:BFSTEST:TMP
EXTLINK /var  MOUNT    root     system   .   ( /../VMBFS:BFSTEST:VAR

*************************************************************************
* now that we have the MEL for /var, create what's under it          *
*************************************************************************

DIR /var/spool         root     system   .
DIR /var/spool/mail    root     system   -rwxrwxrwx

*************************************************************************
* Create symbolic links.                                                *
* syntax:  SLINK path1 owner group permission (path2                 *
*************************************************************************

SLINK /lib                  root     system   .   ( /usr/lib
SLINK /u                    root     system   .   ( /home
SLINK /usr/mail             root     system   .   ( /var/spool/mail
SLINK /usr/spool            root     system   .   ( /var/spool
SLINK /usr/lib/nls/msg/En_US  .      .        .   ( /usr/lib/nls/msg/En
SLINK /usr/lib/nls/msg/C      .      .        .   ( /usr/lib/nls/msg/C.

*************************************************************************
* special files                                                         *
* MKNODE pathname owner group permis '(' type major_node minor_node  *
* see BPXYFTYP for values for 'type'                                    *
*************************************************************************

MKNODE /dev/tty             root     system   crw-rw-rw-   ( c 3 0
MKNODE /dev/null            root     system   crw-rw-rw-   ( c 4 0

*************************************************************************
* external links to certain things on minidisks                       *
*************************************************************************

* sockets header file
EXTLINK /usr/include/sys/time.h CMSDATA bin  bin -rw-r--r--  ( SYS_T
*************************************************************************
* external links to certain things on minidisks                       *
*************************************************************************

* sockets header file
EXTLINK /usr/include/sys/time.h CMSDATA bin  bin -rw-r--r--  ( SYS_T

*************************************************************************
* Unmount the filespace                                                 *
* syntax:  UNMOUNT pathname                                             *
*************************************************************************
UNMOUNT                 /
```

B.2 SHELL LOADBFS

This section contains the file used to load BFS with Shell and Utilities objects. This file is discussed in 2.3, “Creating the BFS server” on page 16.

```
*******************************************************
*                                                     *
*  Copyright -                                        *
*                                                     *
*    THIS MODULE IS "RESTRICTED MATERIALS OF IBM"     *
*    5654-030 (C) COPYRIGHT IBM CORP. - 1995, 1998    *
*    LICENSED MATERIALS - PROPERTY OF IBM             *
*    ALL RIGHTS RESERVED.                             *
*                                                     *
*  Status - VM/ESA Version 2, Release 3.0             *
*                                                     *
```

```
*  what goes in the Shell & Utilities directories     *
*                                                     *
*  this file is processed by LOADBFS EXEC             *
*                                                     *
*  this file assumes that the structure set up by     *
*  BFS LOADBFS is already in place.  BFS LOADBFS      *
*  sets up the structure for a brand-new BFS.         *
*                                                     *
*******************************************************

* set defaults for owner, group, and permission bits

DEFAULT_OWNER           bin
DEFAULT_GROUP           bin
DEFAULT_PERMISSION      -rwxr-xr-x

*******************************************************
* Mount the filespace to work in                      *
* syntax:  MOUNT pathname1 pathname2                  *
*******************************************************

MOUNT                   /../VMBFS:BFSTEST:ROOT/ /

*******************************************************
* directories to create                               *
* syntax:  DIR path owner group permission            *
* use '.' to mean 'take the default'                  *
*******************************************************

DIR /bin                    root     system   .
DIR /etc/samples            root     system   .
DIR /usr/lib/nls/charmap    .        .        .
DIR /usr/lib/nls/locale     .        .        .
DIR /usr/lib/nls/locale/IBM .        .        .

*************************************************************
* files to create                                           *
* FILE path owner group permission '(' fn ft putbfs_opts    *
* use '.' to mean 'take the default'                        *
*************************************************************

* here's /bin
FILE /bin/ar             .        .        .    ( AR       MODULE
FILE /bin/awk            .        .        .    ( AWK      MODULE
FILE /bin/basename       .        .        .    ( BASENAME MODULE
FILE /bin/bc             .        .        .    ( BC       MODULE
FILE /bin/c89            .        .        .    ( C89      MODULE
FILE /bin/cat            .        .        .    ( CAT      MODULE
FILE /bin/chgrp          .        .        .    ( CHGRP    MODULE
FILE /bin/chmod          .        .        .    ( CHMOD    MODULE
FILE /bin/chown          .        .        .    ( CHOWN    MODULE
FILE /bin/cksum          .        .        .    ( CKSUM    MODULE
FILE /bin/cmp            .        .        .    ( CMP      MODULE
FILE /bin/comm           .        .        .    ( COMM     MODULE
FILE /bin/compress       .        .        .    ( COMPRESS MODULE
FILE /bin/cp             .        .        .    ( CP       MODULE
FILE /bin/cpio           .        .        .    ( CPIO     MODULE
FILE /bin/cut            .        .        .    ( CUT      MODULE
FILE /bin/date           .        .        .    ( DATE     MODULE
FILE /bin/dd             .        .        .    ( DD       MODULE
FILE /bin/diff           .        .        .    ( DIFF     MODULE
FILE /bin/dirname        .        .        .    ( DIRNAME  MODULE
FILE /bin/echo           .        .        .    ( ECHO     MODULE
FILE /bin/ed             .        .        .    ( ED       MODULE
FILE /bin/env            .        .        .    ( ENV      MODULE
FILE /bin/expr           .        .        .    ( EXPR     MODULE
FILE /bin/false          .        .        .    ( FALSE    MODULE
FILE /bin/find           .        .        .    ( FIND     MODULE
FILE /bin/fold           .        .        .    ( FOLD     MODULE
FILE /bin/getconf        .        .        .    ( GETCONF  MODULE
FILE /bin/grep           .        .        .    ( GREP     MODULE
FILE /bin/head           .        .        .    ( HEAD     MODULE
FILE /bin/iconv          .        .        .    ( ICONV    MODULE
FILE /bin/id             .        .        .    ( ID       MODULE
FILE /bin/join           .        .        .    ( JOIN     MODULE
FILE /bin/lex            .        .        .    ( LEX      MODULE
FILE /bin/ln             .        .        .    ( LN       MODULE
FILE /bin/locale         .        .        .    ( LOCALE   MODULE
```

```
FILE /bin/logger         .        .        .    ( LOGGER   MODULE
FILE /bin/logname        .        .        .    ( LOGNAME  MODULE
FILE /bin/lp             .        .        .    ( LP       MODULE
FILE /bin/ls             .        .        .    ( LS       MODULE
FILE /bin/mailx          .        .        .    ( MAILX    MODULE
FILE /bin/make           .        .        .    ( MAKE     MODULE
FILE /bin/mkdir          .        .        .    ( MKDIR    MODULE
FILE /bin/mkfifo         .        .        .    ( MKFIFO   MODULE
FILE /bin/mknod          .        .        .    ( MKNOD    MODULE
FILE /bin/mv             .        .        .    ( MV       MODULE
FILE /bin/newgrp         .        .        .    ( NEWGRP   MODULE
FILE /bin/nohup          .        .        .    ( NOHUP    MODULE
FILE /bin/od             .        .        .    ( OD       MODULE
FILE /bin/paste          .        .        .    ( PASTE    MODULE
FILE /bin/pathchk        .        .        .    ( PATHCHK  MODULE
FILE /bin/pax            .        .        .    ( PAX      MODULE
FILE /bin/pr             .        .        .    ( PR       MODULE
FILE /bin/printf         .        .        .    ( PRINTF   MODULE
FILE /bin/ps             .        .        .    ( PS       MODULE
FILE /bin/pwd            .        .        .    ( PWD      MODULE
FILE /bin/rm             .        .        .    ( RM       MODULE
FILE /bin/rmdir          .        .        .    ( RMDIR    MODULE
FILE /bin/sed            .        .        .    ( SED      MODULE
FILE /bin/sh             .        .        .    ( SH       MODULE
FILE /bin/shbltin        .        .        .    ( SHBLTIN  MODULE
FILE /bin/sleep          .        .        .    ( SLEEP    MODULE
FILE /bin/sort           .        .        .    ( SORT MODULE
FILE /bin/strip          .        .        .    ( STRIP MODULE
FILE /bin/stty           .        .        .    ( STTY MODULE
FILE /bin/su     root     system   -rwsr-xr--    ( SU   MODULE
FILE /bin/tail           .        .        .    ( TAIL MODULE
FILE /bin/tar            .        .        .    ( TAR   MODULE
FILE /bin/tee            .        .        .    ( TEE   MODULE
FILE /bin/time           .        .        .    ( TIME MODULE
FILE /bin/touch          .        .        .    ( TOUCH MODULE
FILE /bin/tr             .        .        .    ( TR   MODULE
FILE /bin/true           .        .        .    ( TRUE MODULE
FILE /bin/tty            .        .        .    ( TTY  MODULE
FILE /bin/uname          .        .        .    ( UNAME MODULE
FILE /bin/uncompress     .        .        .    ( UNCOMPRE MODULE
FILE /bin/uniq           .        .        .    ( UNIQ  MODULE
FILE /bin/wc             .        .        .    ( WC    MODULE
FILE /bin/xargs          .        .        .    ( XARGS MODULE
FILE /bin/yacc           .        .        .    ( YACC  MODULE
FILE /bin/zcat           .        .        .    ( ZCAT  MODULE

* stuff for /etc
FILE /etc/startup.mk     .        .      .            ( STARTUP  MK
FILE /etc/yylex.c        .        .      -rw-r--r--   ( YYLEX    C_XMP
FILE /etc/mailx.rc       .        .      -rw-r--r--   ( MAILX    RC
FILE /etc/yyparse.c      .        .      -rw-r--r--   ( YYPARSE  C_XMP
FILE /etc/profile.sample .        .      -rw-r--r--   ( PROFILE  SAMPLE

* STUFF FOR /ETC/SAMPLES.EXAMPLES "ADVANCED APPLICATION DEVELOPMENT
FILE /etc/samples/wc.l   .        .      -rw-r--r--   ( WC      L_XMP
FILE /etc/samples/wc.c   .        .      -rw-r--r--   ( WC      C_XMP
FILE /etc/samples/dc1.l  .        .      -rw-r--r--   ( DC1     L_XMP
FILE /etc/samples/dc1.y  .        .      -rw-r--r--   ( DC1     Y_XMP
FILE /etc/samples/dc2.l  .        .      -rw-r--r--   ( DC2     L_XMP
FILE /etc/samples/dc2.y  .        .      -rw-r--r--   ( DC2     Y_XMP
FILE /etc/samples/dc3.l  .        .      -rw-r--r--   ( DC3     L_XMP
FILE /etc/samples/dc3.y  .        .      -rw-r--r--   ( DC3     Y_XMP
FILE /etc/samples/execute.c .      .     -rw-r--r--   ( EXECUTE C_XMP
FILE /etc/samples/compile.c .      .     -rw-r--r--   ( COMPILE C_XMP
FILE /etc/samples/comics.lst .     .     -rw-r--r--   ( COMICS  LST
FILE /etc/samples/hobbies   .      .     -rw-r--r--   ( HOBBIES LST

* stuff for /usr/lib
FILE /usr/lib/libc.a     .     .     -rw-r--r--   ( LIBC A BFSLINE NONE
FILE /usr/lib/libm.a     .     .     -rw-r--r--   ( LIBM A BFSLINE NONE
FILE /usr/lib/libl.a     .     .    -rw-r--r--   ( LIBL A BFSLINE NONE FILE /usr/lib/liby.a
.     .     -rw-r--r--   ( LIBY A BFSLINE NONE FILE /usr/lib/lib.b     .     .     -rw-r--r--
( LIBB
FILE /usr/lib/tsmail   root  system  -rwsr-xr-x   ( TSMAIL MODULE

* and the default charmap
FILE /usr/lib/nls/charmap/IBM-1047 .   .   -rw-r--r--   ( IBM@104
```

```
* message repositories
FILE /usr/lib/nls/msg/C/cmsg1047.cat . .   -rw-r--r--   ( CMSG104

* these are all the dummy locale files for locale -a

DEFAULT_PERMISSION        PUSH
DEFAULT_PERMISSION        -rw-r--r--

FILE /usr/lib/nls/locale/Da_DK.IBM-1047 . . .         ( DA_DK IBM-1047
FILE /usr/lib/nls/locale/De_CH.IBM-1047 . . .         ( DE_CH IBM-1047
FILE /usr/lib/nls/locale/De_DE.IBM-1047 . . .         ( DE_DE IBM-1047
FILE /usr/lib/nls/locale/En_GB.IBM-1047 . . .         ( EN_GB IBM-1047
FILE /usr/lib/nls/locale/En_JP.IBM-1027 . . .         ( EN_JP IBM-1027
FILE /usr/lib/nls/locale/En_US.IBM-1047 . . .         ( EN_US IBM-1047
FILE /usr/lib/nls/locale/Es_ES.IBM-1047 . . .         ( ES_ES IBM-1047
FILE /usr/lib/nls/locale/Fi_FI.IBM-1047 . . .         ( FI_FI IBM-1047
FILE /usr/lib/nls/locale/Fr_BE.IBM-1047 . . .         ( FR_BE IBM-1047
FILE /usr/lib/nls/locale/Fr_CA.IBM-1047 . . .         ( FR_CA IBM-1047
FILE /usr/lib/nls/locale/Fr_CH.IBM-1047 . . .         ( FR_CH IBM-1047
FILE /usr/lib/nls/locale/Fr_FR.IBM-1047 . . .         ( FR_FR IBM-1047
FILE /usr/lib/nls/locale/Is_IS.IBM-1047 . . .         ( IS_IS IBM-1047
FILE /usr/lib/nls/locale/It_IT.IBM-1047 . . .         ( IT_IT IBM-1047
FILE /usr/lib/nls/locale/Ja_JP.IBM-1027 . . .         ( JA_JP IBM-1027
FILE /usr/lib/nls/locale/Ja_JP.IBM-939 . . .          ( JA_JP IBM-939
FILE /usr/lib/nls/locale/No_NO.IBM-1047 . . .         ( NO_NO IBM-1047
FILE /usr/lib/nls/locale/N1_BE.IBM-1047 . . .         ( N1_BE IBM-1047
FILE /usr/lib/nls/locale/N1_NL.IBM-1047 . . .         ( N1_NL IBM-1047
FILE /usr/lib/nls/locale/Pt_PT.IBM-1047 . . .         ( PT_PT IBM-1047
FILE /usr/lib/nls/locale/Sv_SE.IBM-1047 . . .         ( SV_SE IBM-1047
FILE /usr/lib/nls/locale/IBM/FSUMK010 . . .           ( FSUMK010 LOCALE
FILE /usr/lib/nls/locale/IBM/FSUMK011 . . .           ( FSUMK011 LOCALE
FILE /usr/lib/nls/locale/IBM/FSUMK012 . . .           ( FSUMK012 LOCALE
FILE /usr/lib/nls/locale/IBM/FSUMK013 . . .           ( FSUMK013 LOCALE
FILE /usr/lib/nls/locale/IBM/FSUMK014 . . .           ( FSUMK014 LOCALE
FILE /usr/lib/nls/locale/IBM/FSUMK015 . . .           ( FSUMK015 LOCALE
FILE /usr/lib/nls/locale/IBM/FSUMK016 . . .           ( FSUMK016 LOCALE
FILE /usr/lib/nls/locale/IBM/FSUMK017 . . .           ( FSUMK017 LOCALE
FILE /usr/lib/nls/locale/IBM/FSUMK018 . . .           ( FSUMK018 LOCALE
FILE /usr/lib/nls/locale/IBM/FSUMK019 . . .           ( FSUMK019 LOCALE
FILE /usr/lib/nls/locale/IBM/FSUMK020 . . .           ( FSUMK020 LOCALE
FILE /usr/lib/nls/locale/IBM/FSUMK021 . . .           ( FSUMK021 LOCALE
FILE /usr/lib/nls/locale/IBM/FSUMK022 . . .           ( FSUMK022 LOCALE
FILE /usr/lib/nls/locale/IBM/FSUMK023 . . .           ( FSUMK023 LOCALE
FILE /usr/lib/nls/locale/IBM/FSUMK024 . . .           ( FSUMK024 LOCALE
FILE /usr/lib/nls/locale/IBM/FSUMK025 . . .           ( FSUMK025 LOCALE
FILE /usr/lib/nls/locale/IBM/FSUMK026 . . .           ( FSUMK026 LOCALE
FILE /usr/lib/nls/locale/IBM/FSUMK027 . . .           ( FSUMK027 LOCALE
FILE /usr/lib/nls/locale/IBM/FSUMK028 . . .           ( FSUMK028 LOCALE
FILE /usr/lib/nls/locale/IBM/FSUMK029 . . .           ( FSUMK029 LOCALE
FILE /usr/lib/nls/locale/IBM/FSUMK029 . . .           ( FSUMK029 LOCALE
FILE /usr/lib/nls/locale/IBM/FSUMK030 . . .           ( FSUMK030 LOCALE

DEFAULT_PERMISSION POP

************************************************************
* List of symbolic links to be created by LOADBFS          *
* SLINK new_path  new_own new_grp new_perm '(' existing_path  *
************************************************************

* shell built-ins you're supposed to be able to exec() to...
* these are all implemented by shbltin
SLINK /bin/alias                .        .        .     ( /bin/shbltin
SLINK /bin/bg                   .        .        .     ( /bin/shbltin
SLINK /bin/cd                   .        .        .     ( /bin/shbltin
SLINK /bin/command              .        .        .     ( /bin/shbltin
SLINK /bin/fc                   .        .        .     ( /bin/shbltin
SLINK /bin/fg                   .        .        .     ( /bin/shbltin
SLINK /bin/getopts              .        .        .     ( /bin/shbltin
SLINK /bin/jobs                 .        .        .     ( /bin/shbltin
SLINK /bin/kill                 .        .        .     ( /bin/shbltin
SLINK /bin/read                 .        .        .     ( /bin/shbltin
SLINK /bin/test                 .        .        .     ( /bin/shbltin
SLINK /bin/umask                .        .        .     ( /bin/shbltin
SLINK /bin/unalias              .        .        .     ( /bin/shbltin
SLINK /bin/wait                 .        .        .     ( /bin/shbltin

* links called for by MKS installation book
SLINK /bin/bdiff                .        .        .     ( /bin/diff
```

```
SLINK /bin/diffh               .        .        .     ( /bin/diff
SLINK /bin/red                 .        .        .     ( /bin/ed
SLINK /bin/egrep               .        .        .     ( /bin/grep
SLINK /bin/fgrep               .        .        .     ( /bin/grep
SLINK /bin/rsh                 .        .        .     ( /bin/sh
SLINK /bin/[                   .        .        .     ( /bin/test

********************************************************************
* Unmount the filespace                                            *
* syntax:  UNMOUNT pathname                                        *
********************************************************************
UNMOUNT                /
```

Appendix C. ar.h

Sample file of `ar.h` used to build `make`. This file is included in LIBXPG4 PACKAGE.

```
#ifndef _AR_H
#define  _AR_H
#ifdef   __cplusplus
extern "C" {
#endif
/*
*    COMMON ARCHIVE FORMAT
*
* ARCHIVE File Organization:
* _____________________________________________
* |__________ARCHIVE_MAGIC_STRING_____________|
* |__________ARCHIVE_FILE_MEMBER_1_____________|
* |                                           |
* |  Archive File Header "ar_hdr"             |
* |...........................................|
* |  Member Contents                          |
* |     1. External symbol directory          |
* |     2. Text file                          |
* |___________________________________________|
* |________ARCHIVE_FILE_MEMBER_2_____________|
* |     "ar_hdr"                              |
* |...........................................|
* |  Member Contents (.o or text file)        |
* |___________________________________________|
* |  .     .     .                            |
* |  .     .     .                            |
* |  .     .     .                            |
* |___________________________________________|
* |________ARCHIVE_FILE_MEMBER_n_____________|
* |     "ar_hdr"                              |
* |...........................................|
* |     Member Contents                       |
* |___________________________________________|
*/
#define  ARMAG "!<arch>\n"
#define  SARMAG   8
#define  ARFMAG   "`\n"

struct ar_hdr     /* archive file member header - printable ascii */
{
   char  ar_name[16];   /* file member name - `/' terminated */
   char  ar_date[12];   /* file member date - decimal */
   char  ar_uid[6];  /* file member user id - decimal */
   char  ar_gid[6];  /* file member group id - decimal */
   char  ar_mode[8]; /* file member mode - octal */
   char  ar_size[10];   /* file member size - decimal */
   char  ar_fmag[2]; /* ARFMAG - string to end header */
};
#ifdef   __cplusplus
}
#endif

#endif   /* _AR_H */
```

Appendix D. Keyboard setting for Enetwork Personal Communication

This appendix provides a step-by-step guide on how to remap keyboard settings for Enetwork Personal Communication.

D.1 Remapping your keyboard

With Personal Communication, you can select different keyboard mappings. The keyboard name is the name of the country (United States, France, Germany, and so on).

When you have selected your country keyboard, you can modify the key arrangement of this chosen keyboard.

Get to the menu shown in Figure 169 to do your modifications

Figure 169. Go to keyboard setup

Go to the **Assist** menu and select **Keyboard Setup**.

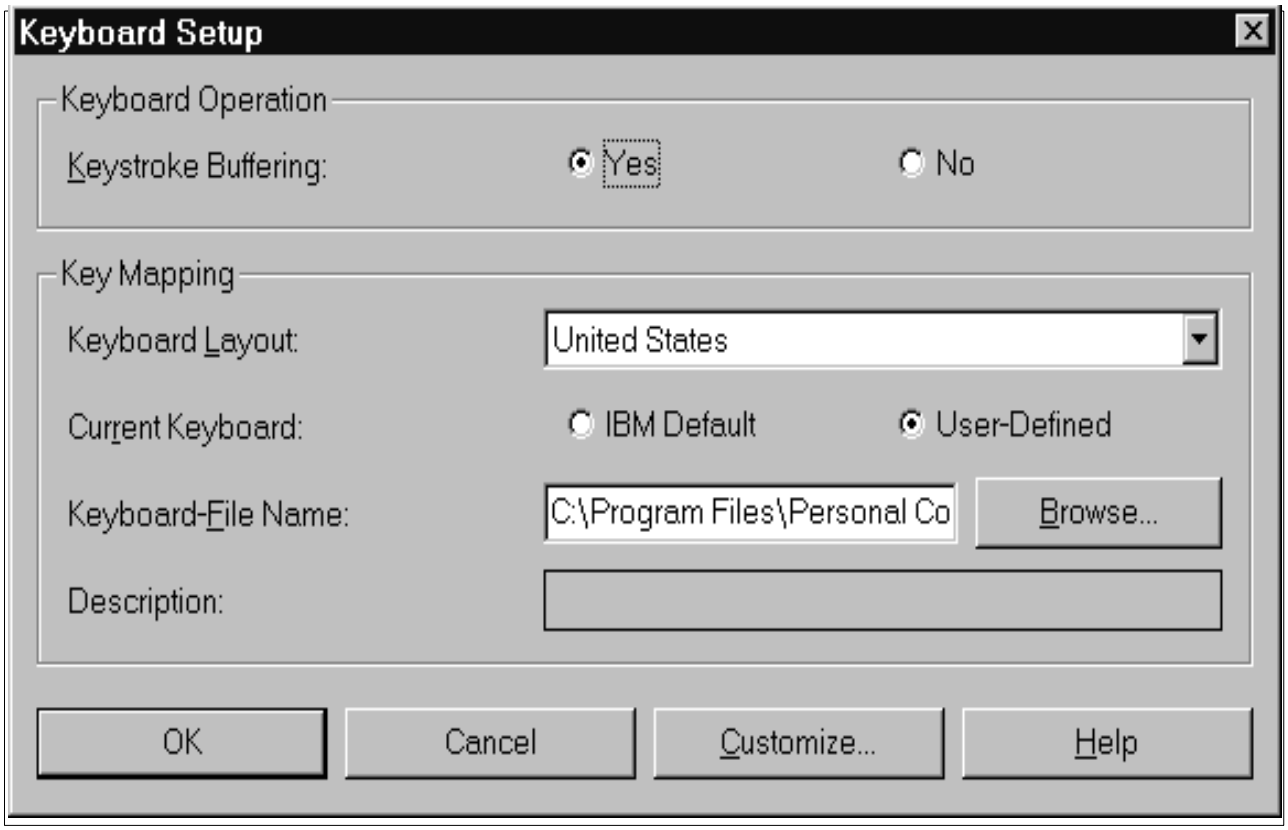


Figure 170. Choosing your keyboard

Choose your keyboard layout and select **Customize**.

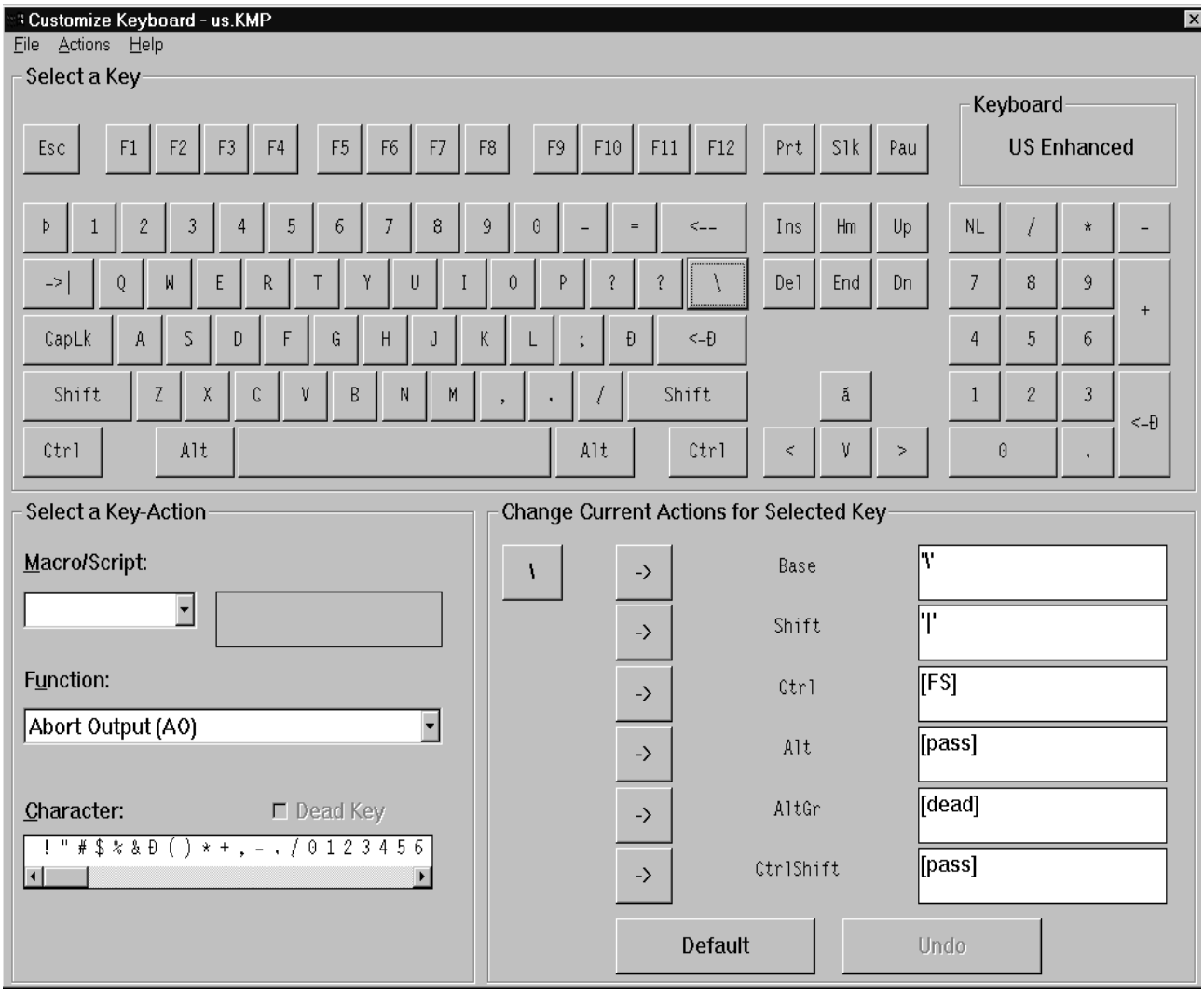


Figure 171. Modify your keyboard keys

1. Click the key that you want to modify in this panel.
2. Move your cursor to the menu element `Select a Key Action` and select a function or a character to assign to the key selected in step 1.
3. Move to **Change Current Actions for Selected Key** and select if you want this key action with Base,Shift,Ctrl, and so on.

When you have completed your change, go to the top of the screen and select File-Save-Exit.

Now your keyboard is ready to work with VM Open Edition.

D.2 SCREEN EXEC

This EXEC will allow you to verify that you are able to display the correct characters for each hexadecimal values between ‘40x’ and ‘FFx’.

1. Create SCREEN EXEC and run it on CMS:

```
/* TEST YOUR SCREEN CHAR*/
```

```
DO I=X2D(40) TO X2D(FF)
  B=I-X2D(3F)
  A.B=D2C(I) D2X(I)
END
A.0=B
"PIPE STEM A. | > SCREEN FILE A"
```

2. Run the EXEC.
3. XEDIT the resulting file named `SCREEN FILE A`.
4. Check with the code page table given in Appendix D of *OpenEdition for VM/ESA User's Guide Version 2 Release 1.0*, SC24-5727. You are able to see what characters are not correctly displayed.
5. To correct each wrong character, we have to create a statement

   `SET OUTPUT xx yy` and include them in SHELL EXEC.

   xx represents the hexadecimal value of the character read.

   yy represents the hexadecimal value that you send to the terminal.

   So for each xx incorrectly displayed, you set a yy matching value found in the IBM-1047 code page.

Now check the input characters received from your keyboard.

1. Start an XEDIT session by typing:

   ```
   XEDIT A A A
   ```

2. Type the letter `i` to get into INPUT mode.
3. Type all the special characters you have on your keyboard, doing a new record each ten characters.
4. On the command line, type:

   ```
   V H 1 20
   ```

5. Verify that for each key pressed, you get the correct hexadecimal value by referring to the code page IBM-1047 in Appendix D of *OpenEdition for VM/ESA User's Guide Version 2 Release 1.0*, SC24-5727.

   For each invalid value, you have to create a `SET INPUT xx yy` statement. These statements will be included in the SHELL EXEC.

   xx is the hexadecimal value received from the keyboard.

   yy is the hexadecimal value that will be set in your file.

Appendix E. Circumventing the Socket File Inheritance Limitation

The following code fragments show how to overcome the current limitation of not being able to inherit socket file descriptors. These examples are for TCP sockets (not UDP), and assume that file descriptor 0 is a socket file to be passed to the child.

E.1 Givesocket example

In this example, the parent program wishes to pass file descriptor 0 to the child. Because it is a socket file, simply using spawn() to inherit the file will not work.

```
int    fdMap[3], pFd[2], maxFd;
struct inheritance inherit;
pid_t  pid;
struct clientid myClient, sockClient;
fd_set eFd;
struct stat info;

1   FD_ZERO(&eFd);
2   getclientid(AF_INET, &myClient);
3   memcpy(&sockClient, &myClient, sizeof(sockClient));
    memset(sockClient.subtaskname, ' ',
    sizeof(sockClient.subtaskname));
4   fstat(cntrl, &info);

5   pipe(pFd);
6   fdMap[STDIN_FILENO]  = pFd[0];
    fdMap[STDOUT_FILENO] = STDOUT_FILENO;
    fdMap[STDERR_FILENO] = STDERR_FILENO;
7   write(pFd[1],&info.st_ino,sizeof(info.st_ino));
8   write(pFd[1],&myClient,sizeof(myClient));
9   if (givesocket(cntrl, &sockClient) == 0)
    {
10      FD_SET(cntrl, &eFd);

        inherit.flags   = SPAWN_SETGROUP;
        inherit.pgroup  = SPAWN_NEWPGROUP;
        if (setgid(sep->se_gid) == -1)
        syslog(LOG_ERR, "%s cant setgid in %s %m",
               sep->se_service, av[0]);
        if (setuid(sep->se_uid) == -1)
        syslog(LOG_ERR, "%s cant setuid in %s %m",
               sep->se_service, av[0]);
11      pid = spawnp(av[0], 3, fdMap, &inherit,
                     (const char **) av, environ);
        if (pid < 0)
        {
12          syslog(LOG_ERR, "%s cant spawn %s %m",
                   sep->se_service, av[0]);
            close (pFd[1]);
            FD_CLR(cntrl, &allsock);
        }
```

Figure 172. Givesocket example (1 of 2): Passing socket file descriptor to child

```
        else
        {
13          close (pFd[1]);
            maxFd = cntrl + 1;
            select(maxFd, NULL, NULL, &eFd, NULL);
        }
    }
    else
    {
14      syslog(LOG_ERR, "Error giving socket to %s - %m",
                sep->se_service, av[0]);
        close (pFd[1]);
        FD_CLR(cntrl, &allsock);
    }
15  close (pFd[0]);
16  close (cntrl);
    setgid(myGID);
    setuid(myUID);
    return(pid);
```

Figure 173. Givesocket example (2 of 2): Passing socket file descriptor to child

Figure 173 notes:

1. The file descriptor set is cleared.
2. The clientid structure myClient is built by a call to getclientid().
3. Copy this information for givesocket() to use.
4. cntrl is the file descritor to be passed; we get its details.
5. Create a pipe to talk to child with.
6. Map file descriptor 0 to the read-end of the pipe.
7. Write the socket number to the pipe.
8. Write the clientid structure itself to the pipe.
9. Give the socket away.
10. Indicate our interest in the socket file descriptor.
11. Spawn the child process.
12. If it fails, then report the error and close the pipe.
13. Close the write-end of the pipe and wait for the socket to be taken.
14. Report the failure of the give and close the write-end of the pipe.
15. Close the read-end of the pipe.
16. Close the socket.

E.2 Takesocket example

This initial part of the child process will effectively reverse the process of the parent to obtain file descriptor 0 from the data passed on the pipe.

```
   /*-------------------------------------------------------*/
   /*                                                       */
   /* If this program has been spawned then it is likely    */
   /* that the file descriptor for stdin, was a socket      */
   /* file descriptor originally. As OE/VM is unable        */
   /* to pass socket descriptors using spawn() or exec()    */
   /* then the necessary information is passed on a pipe.   */
   /* The file descriptor of this pipe is 0. Therefore,     */
   /* if this file exists then we will get and dup2 the     */
   /* stdxxx file descriptors, the socket number, and the   */
   /* clientid structure.                                   */
   /*-------------------------------------------------------*/

1  if (fstat(0, &info) != -1)
   {
2      if (S_ISFIFO(info.st_mode))
       {
3          read(0, &sd0, sizeof(sd0));
4          read(0, &client, sizeof(client));
5          fd0 = takesocket(&client, sd0);
6          dup2(fd0, STDIN_FILENO);
7          dup2(STDIN_FILENO, STDOUT_FILENO);
       }
   }
```

Figure 174. Takesocket example: Obtain socket file descriptor from parent process

Figure 174 notes:

1. Check if file descriptor 0 is available; if not, there is nothing to do.
2. Check that it is a pipe; if not, we have inherited a usable file descriptor.
3. Read the socket number we are to take.
4. Get the clientid structure from the pipe.
5. Take the socket from the parent.
6. Close the existing file descriptor 0 and then open it to the same as the file socket file we have just taken.
7. Duplicate this to file descriptor 1 (that is, STDOUT will refer to the same file as STDIN).

Appendix F. Detailed install logs and configuration files

When you are first installing the OE applications, it can be intimidating. You may wonder, did the install work or not? This section aims to alleviate this feeling by providing you with detailed console logs of the installation and of some of the products described in this book.

F.1 GNU make installation log

This output has been generated by `configure` of GNU `make`:

```
[porting - /home/porting/make-3.77]>
configure
creating cache ./config.cache
checking for a BSD compatible install... ./install-sh -c
checking whether build environment is sane... yes
checking whether make sets ${MAKE}... yes
checking for working aclocal... missing
checking for working autoconf... missing
checking for working automake... missing
checking for working autoheader... missing
checking for working makeinfo... missing
checking whether make sets ${MAKE}... (cached) yes
checking for gcc... c89 -W c,nosource
checking whether the C compiler (c89 -W c,nosource  ) works... yes
checking whether the C compiler (c89 -W c,nosource  ) is a cross-compiler... no
checking whether we are using GNU C... no
checking for a BSD compatible install... ./install-sh -c
checking how to run the C preprocessor... c89 -W c,nosource -E
checking for AIX... no
checking for POSIXized ISC... no
checking for minix/config.h... no
checking whether large file support needs explicit enabling... no
checking for ANSI C header files... no
checking for dirent.h that defines DIR... yes
checking for opendir in -ldir... no
checking for uid_t in sys/types.h... yes
checking for pid_t... yes
checking return type of signal handlers... void
checking for unistd.h... yes
checking for limits.h... yes
checking for sys/param.h... no
checking for fcntl.h... yes
checking for string.h... yes
checking for memory.h... yes
checking for sys/timeb.h... yes
checking whether c89 -W c,nosource and cc understand -c and -o together... yes
checking for working const... yes
checking whether stat file-mode macros are broken... no
checking for memmove... yes
checking for psignal... yes
checking for mktemp... yes
checking for pstat_getdynamic... yes
checking for dup2... yes
checking for getcwd... yes
checking for sigsetmask... yes
checking for getgroups... yes
checking for setlinebuf... yes
checking for seteuid... yes
checking for setegid... yes
checking for setreuid... yes
checking for setregid... yes
checking for strerror... yes
checking for strsignal... yes
checking for sys_siglist... yes
checking for working alloca.h... yes
checking for alloca... yes
checking for vfork.h... no
checking for working vfork... no
checking whether setvbuf arguments are reversed... no
checking for elf_begin in -lelf... no
```

```
checking for kvm_open in -lkvm... no
checking for getloadavg in -lutil... no
checking for getloadavg in -lgetloadavg... no
checking for getloadavg... yes
checking whether getloadavg requires setgid... no
checking for kstat_open in -lkstat... no
checking for working strcoll... yes
checking for sys/wait.h... yes
checking for waitpid... yes
checking for wait3... yes
checking for union wait... no
checking for sys_siglist declaration in signal.h or unistd.h... yes
checking for getpwnam in -lsun... no
checking for location of SCCS get command
updating cache ./config.cache
creating ./config.status
creating Makefile
creating build.sh
creating config.h
configuring in glob
running /bin/sh ./configure  --cache-file=.././config.cache --srcdir=.
loading cache .././config.cache
checking for a BSD compatible install... ./../install-sh -c
checking whether build environment is sane... yes
checking whether make sets ${MAKE}... (cached) yes
checking for working aclocal... missing
checking for working autoconf... missing
checking for working automake... missing
checking for working autoheader... missing
checking for working makeinfo... missing
checking for gcc... (cached) c89 -W c,nosource
checking whether the C compiler (c89 -W c,nosource  ) works... yes
checking whether the C compiler (c89 -W c,nosource  ) is a cross-compiler... no
checking whether we are using GNU C... (cached) no
checking for ar... ar
checking for ranlib... :
checking how to run the C preprocessor... (cached) c89 -W c,nosource -E
checking for AIX... no
checking for minix/config.h... (cached) no
checking for POSIXized ISC... no
checking for working const... (cached) yes
checking for ANSI C header files... (cached) no
checking for memory.h... (cached) yes
checking for unistd.h... (cached) yes
checking for string.h... (cached) yes
checking for dirent.h that defines DIR... (cached) yes
checking for opendir in -ldir... (cached) no
checking whether closedir returns void... no
checking for working alloca.h... (cached) yes
checking for alloca... (cached) yes
checking for working strcoll... (cached) yes
updating cache .././config.cache
creating ./config.status
creating Makefile
creating config.h
```

F.2 INND installation log

This is the (voluminous) installation log created by INN-1.5.1 (see 7.1, “INND - USENET” on page 147).

```
make install
chmod +x ./makedirs.sh
DESTDIR= ./makedirs.sh;
+ [ ! -d /usr/local/man ]
+ [ ! -d /usr/local/man/man1 ]
+ [ ! -d /usr/local/man/man3 ]
+ mkdir /usr/local/man/man3
+ chown news /usr/local/man/man3
+ chgrp news /usr/local/man/man3
+ chmod 0775 /usr/local/man/man3
+ [ ! -d /usr/local/man/man5 ]
+ mkdir /usr/local/man/man5
+ chown news /usr/local/man/man5
+ chgrp news /usr/local/man/man5
+ chmod 0775 /usr/local/man/man5
+ [ ! -d /usr/local/man/man8 ]
+ mkdir /usr/local/man/man8
+ chown news /usr/local/man/man8
+ chgrp news /usr/local/man/man8
+ chmod 0775 /usr/local/man/man8
+ [ ! -d /home/news ]
+ [ ! -d /usr/local/etc ]
+ mkdir /usr/local/etc
+ chown news /usr/local/etc
+ chgrp news /usr/local/etc
+ chmod 0775 /usr/local/etc
+ [ ! -d /usr/local/news ]
+ mkdir /usr/local/news
+ chown news /usr/local/news
+ chgrp news /usr/local/news
+ chmod 0775 /usr/local/news
+ [ ! -d /tmp ]
+ [ ! -d /home/news/local ]
+ mkdir /home/news/local
+ chown news /home/news/local
+ chgrp news /home/news/local
+ chmod 0775 /home/news/local
+ [ ! -d /var/spool/news ]
+ mkdir /var/spool/news
+ chown news /var/spool/news
+ chgrp news /var/spool/news
+ chmod 0775 /var/spool/news
+ [ ! -d /var/spool/news ]
+ [ ! -d /var/spool/news/archive ]
+ mkdir /var/spool/news/archive
+ chown news /var/spool/news/archive
+ chgrp news /var/spool/news/archive
+ chmod 0775 /var/spool/news/archive
+ [ ! -d /var/spool/news/out.going ]
+ mkdir /var/spool/news/out.going
+ chown news /var/spool/news/out.going
+ chgrp news /var/spool/news/out.going
+ chmod 0775 /var/spool/news/out.going
+ [ ! -d /usr/local/news ]
+ [ ! -d /var/log/news ]
+ mkdir /var/log/news
+ chown news /var/log/news
+ chgrp news /var/log/news
+ chmod 0775 /var/log/news
+ [ ! -d /var/log/news/OLD ]
+ mkdir /var/log/news/OLD
+ chown news /var/log/news/OLD
+ chgrp news /var/log/news/OLD
+ chmod 0775 /var/log/news/OLD
+ [ ! -d /var/spool/news/in.coming ]
+ mkdir /var/spool/news/in.coming
+ chown news /var/spool/news/in.coming
+ chgrp news /var/spool/news/in.coming
+ chmod 0775 /var/spool/news/in.coming
+ [ ! -d /var/spool/news/in.coming/bad ]
+ mkdir /var/spool/news/in.coming/bad
```

```
+ chown news /var/spool/news/in.coming/bad
+ chgrp news /var/spool/news/in.coming/bad
+ chmod 0775 /var/spool/news/in.coming/bad
+ [ ! -d /var/spool/news/in.coming/tmp ]
+ mkdir /var/spool/news/in.coming/tmp
+ chown news /var/spool/news/in.coming/tmp
+ chgrp news /var/spool/news/in.coming/tmp
+ chmod 0775 /var/spool/news/in.coming/tmp
+ [ ! -d /usr/local/news ]
+ [ ! -d /usr/local/news/innd ]
+ mkdir /usr/local/news/innd
+ chown news /usr/local/news/innd
+ chgrp news /usr/local/news/innd
+ chmod 0770 /usr/local/news/innd
+ [ ! -d /usr/local/news ]
+ [ ! -d /usr/local/news/bin ]
+ mkdir /usr/local/news/bin
+ chown news /usr/local/news/bin
+ chgrp news /usr/local/news/bin
+ chmod 0775 /usr/local/news/bin
+ [ ! -d /usr/local/news/bin/control ]
+ mkdir /usr/local/news/bin/control
+ chown news /usr/local/news/bin/control
+ chgrp news /usr/local/news/bin/control
+ chmod 0775 /usr/local/news/bin/control
+ [ ! -d /usr/local/news/bin/rnews ]
+ mkdir /usr/local/news/bin/rnews
+ chown news /usr/local/news/bin/rnews
+ chgrp news /usr/local/news/bin/rnews
+ chmod 0775 /usr/local/news/bin/rnews
+ [ ! -d /usr/local/etc/auth ]
+ mkdir /usr/local/etc/auth
+ chown news /usr/local/etc/auth
+ chgrp news /usr/local/etc/auth
+ chmod 0775 /usr/local/etc/auth
+ exit 0
make  WHAT_TO_MAKE=install DESTDIR= common
make[1]: Entering directory `/home/porting/inn-1.5.1'

cd config ; make  install ; cd ..
make[2]: Entering directory `/home/porting/inn-1.5.1/config'
make[2]: Nothing to be done for `install'.
make[2]: Leaving directory `/home/porting/inn-1.5.1/config'

cd lib ; make  install ; cd ..
make[2]: Entering directory `/home/porting/inn-1.5.1/lib'
lint -u -b -h -z -I../include -D_OE_SOCKETS -D_XOPEN_SOURCE_EXTENDED  strcasecmp.c getdtab.c
dbzalt.c checkart.c cleanfrom.c clienta
ctive.c clientlib.c closeonexec.c dbz.c defdist.c findheader.c genid.c getconfig.c getdtab.c
getfqdn.c getmodaddr.c gettime.c inndco
mm.c innvers.c localopen.c lockfile
lint: not found
grep -v yaccpar <lint.all >lint
make[2]: [lint] Error 1 (ignored)
/bin/sh ./makellib.sh NONE "-b -h -z -I../include -D_OE_SOCKETS -D_XOPEN_SOURCE_EXTENDED "
strcasecmp.c getdtab.c dbzalt.c checkart.
c cleanfrom.c clientactive.c clientlib.c closeonexec.c dbz.c defdist.c findheader.c genid.c
getconfig.c getdtab.c getfqdn.c getmodad
dr.c gettime.c inndcomm.c innvers.c
cp llib-linn.ln ../llib-linn.ln
make[2]: Leaving directory `/home/porting/inn-1.5.1/lib'

cd frontends ; make  install ; cd ..
make[2]: Entering directory `/home/porting/inn-1.5.1/frontends'
/bin/sh ../installit.sh -O news -G news -m 02555 -b .OLD inews /usr/local/etc/inews
/bin/sh ../installit.sh -O news -G news -m 04550 -b .OLD rnews /usr/local/etc/rnews
/bin/sh ../installit.sh -O news -G news -m  0550 -b .OLD ctlinnd /usr/local/news/bin/ctlinnd
/bin/sh ../installit.sh -O news -G news -m  0555 -b .OLD getlist /usr/local/news/bin/getlist
/bin/sh ../installit.sh -O news -G news -m  0555 -b .OLD innconfval
/usr/local/news/bin/innconfval
/bin/sh ../installit.sh -O news -G news -m  0555 -b .OLD decode
/usr/local/news/bin/rnews/decode
/bin/sh ../installit.sh -O news -G news -m  0555 -b .OLD encode
/usr/local/news/bin/rnews/encode
make[2]: Leaving directory `/home/porting/inn-1.5.1/frontends'

cd innd ; make  install ; cd ..
make[2]: Entering directory `/home/porting/inn-1.5.1/innd'
```

```
/bin/sh ../installit.sh -O news -G news -m 0550 -b .OLD innd /usr/local/etc/innd
/bin/sh ../installit.sh -O root -G news -m 4550 -b .OLD inndstart /usr/local/etc/inndstart

==============================
NOTE NOTE NOTE NOTE NOTE
-r-sr-x---   1 porting    news      222211 Apr 14 11:31 /usr/local/etc/inndstart
/usr/local/etc/inndstart needs to be installed setuid root


make[2]: Leaving directory `/home/porting/inn-1.5.1/innd'

cd nnrpd ; make  install ; cd ..
make[2]: Entering directory `/home/porting/inn-1.5.1/nnrpd'
/bin/sh ../installit.sh -O news -G news -m 0555 -b .OLD nnrpd /usr/local/etc/nnrpd
make[2]: Leaving directory `/home/porting/inn-1.5.1/nnrpd'

cd backends ; make  install ; cd ..
make[2]: Entering directory `/home/porting/inn-1.5.1/backends'
/bin/sh ../installit.sh -O news -G news -m 0555 -b .OLD archive /usr/local/news/bin/archive
/bin/sh ../installit.sh -O news -G news -m 0555 -b .OLD batcher /usr/local/news/bin/batcher
/bin/sh ../installit.sh -O news -G news -m 0555 -b .OLD buffchan /usr/local/news/bin/buffchan
/bin/sh ../installit.sh -O news -G news -m 0555 -b .OLD cvtbatch /usr/local/news/bin/cvtbatch
/bin/sh ../installit.sh -O news -G news -m 0555 -b .OLD filechan /usr/local/news/bin/filechan
/bin/sh ../installit.sh -O news -G news -m 0555 -b .OLD nntpget /usr/local/news/bin/nntpget
/bin/sh ../installit.sh -O news -G news -m 0555 -b .OLD innxmit /usr/local/news/bin/innxmit
/bin/sh ../installit.sh -O news -G news -m 0555 -b .OLD innxbatch
/usr/local/news/bin/innxbatch
/bin/sh ../installit.sh -O news -G news -m 0555 -b .OLD overchan /usr/local/news/bin/overchan
/bin/sh ../installit.sh -O news -G news -m 0555 -b .OLD shlock /usr/local/news/bin/shlock
/bin/sh ../installit.sh -O news -G news -m 0555 -b .OLD shrinkfile
/usr/local/news/bin/shrinkfile
/bin/sh ../installit.sh -O news -G news -m 0555 -b .OLD crosspost
/usr/local/news/bin/crosspost
/bin/sh ../installit.sh -O news -G news -m 0555 -b .OLD sendxbatches
/usr/local/news/bin/sendxbatches
/bin/sh ../installit.sh -O news -G news -m 0550 -b .OLD actsync /usr/local/news/bin/actsync
/bin/sh ../installit.sh -O news -G news -m 0550 -b .OLD actsyncd /usr/local/news/bin/actsyncd
/bin/sh ../installit.sh -O news -G news -m 0550 -b .OLD actmerge /usr/local/news/bin/actmerge
make[2]: Leaving directory `/home/porting/inn-1.5.1/backends'

cd expire ; make  install ; cd ..
make[2]: Entering directory `/home/porting/inn-1.5.1/expire'
/bin/sh ../installit.sh -O news -G news -m 0555 -b .OLD convdate /usr/local/news/bin/convdate
/bin/sh ../installit.sh -O news -G news -m 0555 -b .OLD expire /usr/local/news/bin/expire
/bin/sh ../installit.sh -O news -G news -m 0555 -b .OLD expireover
/usr/local/news/bin/expireover
/bin/sh ../installit.sh -O news -G news -m 0555 -b .OLD fastrm /usr/local/news/bin/fastrm
/bin/sh ../installit.sh -O news -G news -m 0555 -b .OLD grephistory
/usr/local/news/bin/grephistory
/bin/sh ../installit.sh -O news -G news -m 0555 -b .OLD makeactive
/usr/local/news/bin/makeactive
/bin/sh ../installit.sh -O news -G news -m 0555 -b .OLD makehistory
/usr/local/news/bin/makehistory
/bin/sh ../installit.sh -O news -G news -m 0555 -b .OLD makehistory
/usr/local/news/bin/makehistory
/bin/sh ../installit.sh -O news -G news -m 0555 -b .OLD newsrequeue
/usr/local/news/bin/newsrequeue
/bin/sh ../installit.sh -O news -G news -m 0555 -b .OLD prunehistory
/usr/local/news/bin/prunehistory
make[2]: Leaving directory `/home/porting/inn-1.5.1/expire'

cd doc ; make  install ; cd ..
make[2]: Entering directory `/home/porting/inn-1.5.1/doc'
/bin/sh ./putman.sh SOURCE " -m 0444" convdate.1 /usr/local/man/man1/convdate.1
/bin/sh ./putman.sh SOURCE " -m 0444" getlist.1 /usr/local/man/man1/getlist.1
/bin/sh ./putman.sh SOURCE " -m 0444" grephistory.1 /usr/local/man/man1/grephistory.1
/bin/sh ./putman.sh SOURCE " -m 0444" inews.1 /usr/local/man/man1/inews.1
/bin/sh ./putman.sh SOURCE " -m 0444" innconfval.1 /usr/local/man/man1/innconfval.1
/bin/sh ./putman.sh SOURCE " -m 0444" installit.1 /usr/local/man/man1/installit.1
/bin/sh ./putman.sh SOURCE " -m 0444" nntpget.1 /usr/local/man/man1/nntpget.1
/bin/sh ./putman.sh SOURCE " -m 0444" rnews.1 /usr/local/man/man1/rnews.1
/bin/sh ./putman.sh SOURCE " -m 0444" shlock.1 /usr/local/man/man1/shlock.1
/bin/sh ./putman.sh SOURCE " -m 0444" shrinkfile.1 /usr/local/man/man1/shrinkfile.1
/bin/sh ./putman.sh SOURCE " -m 0444" subst.1 /usr/local/man/man1/subst.1
/bin/sh ./putman.sh SOURCE " -m 0444" clientlib.3 /usr/local/man/man3/clientlib.3
/bin/sh ./putman.sh SOURCE " -m 0444" dbz.3 /usr/local/man/man3/dbz.3
/bin/sh ./putman.sh SOURCE " -m 0444" inndcomm.3 /usr/local/man/man3/inndcomm.3
/bin/sh ./putman.sh SOURCE " -m 0444" libinn.3 /usr/local/man/man3/libinn.3
```

```
/bin/sh ./putman.sh SOURCE " -m 0444" parsedate.3 /usr/local/man/man3/parsedate.3
/bin/sh ./putman.sh SOURCE " -m 0444" qio.3 /usr/local/man/man3/qio.3
/bin/sh ./putman.sh SOURCE " -m 0444" wildmat.3 /usr/local/man/man3/wildmat.3
/bin/sh ./putman.sh SOURCE " -m 0444" active.5 /usr/local/man/man5/active.5
/bin/sh ./putman.sh SOURCE " -m 0444" control.ctl.5 /usr/local/man/man5/control.ctl.5
/bin/sh ./putman.sh SOURCE " -m 0444" distrib.pats.5 /usr/local/man/man5/distrib.pats.5
/bin/sh ./putman.sh SOURCE " -m 0444" expire.ctl.5 /usr/local/man/man5/expire.ctl.5
/bin/sh ./putman.sh SOURCE " -m 0444" history.5 /usr/local/man/man5/history.5
/bin/sh ./putman.sh SOURCE " -m 0444" hosts.nntp.5 /usr/local/man/man5/hosts.nntp.5
/bin/sh ./putman.sh SOURCE " -m 0444" inn.conf.5 /usr/local/man/man5/inn.conf.5
/bin/sh ./putman.sh SOURCE " -m 0444" innwatch.ctl.5 /usr/local/man/man5/innwatch.ctl.5
/bin/sh ./putman.sh SOURCE " -m 0444" moderators.5 /usr/local/man/man5/moderators.5
/bin/sh ./putman.sh SOURCE " -m 0444" newsfeeds.5 /usr/local/man/man5/newsfeeds.5
/bin/sh ./putman.sh SOURCE " -m 0444" newslog.5 /usr/local/man/man5/newslog.5
/bin/sh ./putman.sh SOURCE " -m 0444" nnrp.access.5 /usr/local/man/man5/nnrp.access.5
/bin/sh ./putman.sh SOURCE " -m 0444" nntpsend.ctl.5 /usr/local/man/man5/nntpsend.ctl.5
/bin/sh ./putman.sh SOURCE " -m 0444" overview.fmt.5 /usr/local/man/man5/overview.fmt.5
/bin/sh ./putman.sh SOURCE " -m 0444" passwd.nntp.5 /usr/local/man/man5/passwd.nntp.5
/bin/sh ./putman.sh SOURCE " -m 0444" actsync.8 /usr/local/man/man8/actsync.8
/bin/sh ./putman.sh SOURCE " -m 0444" archive.8 /usr/local/man/man8/archive.8
/bin/sh ./putman.sh SOURCE " -m 0444" batcher.8 /usr/local/man/man8/batcher.8
/bin/sh ./putman.sh SOURCE " -m 0444" buffchan.8 /usr/local/man/man8/buffchan.8
/bin/sh ./putman.sh SOURCE " -m 0444" crosspost.8 /usr/local/man/man8/crosspost.8
/bin/sh ./putman.sh SOURCE " -m 0444" ctlinnd.8 /usr/local/man/man8/ctlinnd.8
/bin/sh ./putman.sh SOURCE " -m 0444" cvtbatch.8 /usr/local/man/man8/cvtbatch.8
/bin/sh ./putman.sh SOURCE " -m 0444" expire.8 /usr/local/man/man8/expire.8
/bin/sh ./putman.sh SOURCE " -m 0444" expireover.8 /usr/local/man/man8/expireover.8
/bin/sh ./putman.sh SOURCE " -m 0444" expirerm.8 /usr/local/man/man8/expirerm.8
/bin/sh ./putman.sh SOURCE " -m 0444" fastrm.8 /usr/local/man/man8/fastrm.8
/bin/sh ./putman.sh SOURCE " -m 0444" filechan.8 /usr/local/man/man8/filechan.8
/bin/sh ./putman.sh SOURCE " -m 0444" inncheck.8 /usr/local/man/man8/inncheck.8
/bin/sh ./putman.sh SOURCE " -m 0444" innd.8 /usr/local/man/man8/innd.8
/bin/sh ./putman.sh SOURCE " -m 0444" innlog.pl.8 /usr/local/man/man8/innlog.pl.8
/bin/sh ./putman.sh SOURCE " -m 0444" innstat.8 /usr/local/man/man8/innstat.8
/bin/sh ./putman.sh SOURCE " -m 0444" innwatch.8 /usr/local/man/man8/innwatch.8
/bin/sh ./putman.sh SOURCE " -m 0444" innxbatch.8 /usr/local/man/man8/innxbatch.8
/bin/sh ./putman.sh SOURCE " -m 0444" innxmit.8 /usr/local/man/man8/innxmit.8
/bin/sh ./putman.sh SOURCE " -m 0444" makeactive.8 /usr/local/man/man8/makeactive.8
/bin/sh ./putman.sh SOURCE " -m 0444" makehistory.8 /usr/local/man/man8/makehistory.8
/bin/sh ./putman.sh SOURCE " -m 0444" news-recovery.8 /usr/local/man/man8/news-recovery.8
/bin/sh ./putman.sh SOURCE " -m 0444" news.daily.8 /usr/local/man/man8/news.daily.8
/bin/sh ./putman.sh SOURCE " -m 0444" newslog.8 /usr/local/man/man8/newslog.8
/bin/sh ./putman.sh SOURCE " -m 0444" newsrequeue.8 /usr/local/man/man8/newsrequeue.8
/bin/sh ./putman.sh SOURCE " -m 0444" nnrpd.8 /usr/local/man/man8/nnrpd.8
/bin/sh ./putman.sh SOURCE " -m 0444" nntpsend.8 /usr/local/man/man8/nntpsend.8
/bin/sh ./putman.sh SOURCE " -m 0444" overchan.8 /usr/local/man/man8/overchan.8
/bin/sh ./putman.sh SOURCE " -m 0444" prunehistory.8 /usr/local/man/man8/prunehistory.8
/bin/sh ./putman.sh SOURCE " -m 0444" scanlogs.8 /usr/local/man/man8/scanlogs.8
/bin/sh ./putman.sh SOURCE " -m 0444" tally.control.8 /usr/local/man/man8/tally.control.8
/bin/sh ./putman.sh SOURCE " -m 0444" tally.unwanted.8 /usr/local/man/man8/tally.unwanted.8
/bin/sh ./putman.sh SOURCE " -m 0444" writelog.8 /usr/local/man/man8/writelog.8
make[2]: Leaving directory `/home/porting/inn-1.5.1/doc'

cd site ; make  install ; cd ..
make[2]: Entering directory `/home/porting/inn-1.5.1/site'
/bin/sh ../installit.sh -O news -G news -m 0550 -b .OLD rc.news /usr/local/etc/rc.news
/bin/sh ../installit.sh -O news -G news -m 0444 -b .OLD overview.fmt
/usr/local/news/overview.fmt
/bin/sh ../installit.sh -O news -G news -m 0550 -b .OLD checkgroups
/usr/local/news/bin/control/checkgroups
/bin/sh ../installit.sh -O news -G news -m 0550 -b .OLD default
/usr/local/news/bin/control/default
/bin/sh ../installit.sh -O news -G news -m 0550 -b .OLD ihave
/usr/local/news/bin/control/ihave
/bin/sh ../installit.sh -O news -G news -m 0550 -b .OLD newgroup
/usr/local/news/bin/control/newgroup
/bin/sh ../installit.sh -O news -G news -m 0550 -b .OLD rmgroup
/usr/local/news/bin/control/rmgroup
/bin/sh ../installit.sh -O news -G news -m 0550 -b .OLD sendme
/usr/local/news/bin/control/sendme
/bin/sh ../installit.sh -O news -G news -m 0550 -b .OLD sendsys
/usr/local/news/bin/control/sendsys
/bin/sh ../installit.sh -O news -G news -m 0550 -b .OLD senduuname
/usr/local/news/bin/control/senduuname
/bin/sh ../installit.sh -O news -G news -m 0550 -b .OLD version
/usr/local/news/bin/control/version
/bin/sh ../installit.sh -O news -G news -m 0550 -b .OLD parsecontrol
/usr/local/news/parsecontrol
```

```
/bin/sh ../installit.sh -O news -G news -m 0550 -b .OLD writelog /usr/local/news/bin/writelog
/bin/sh ../installit.sh -O news -G news -m 0550 -b .OLD docheckgroups
/usr/local/news/bin/control/docheckgroups
/bin/sh ../installit.sh -O news -G news -m 0550 -b .OLD news.daily
/usr/local/news/bin/news.daily
/bin/sh ../installit.sh -O news -G news -m 0550 -b .OLD scanlogs /usr/local/news/bin/scanlogs
/bin/sh ../installit.sh -O news -G news -m 0550 -b .OLD expirerm /usr/local/news/bin/expirerm
/bin/sh ../installit.sh -O news -G news -m 0550 -b .OLD tally.control
/usr/local/news/bin/tally.control
/bin/sh ../installit.sh -O news -G news -m 0550 -b .OLD tally.unwanted
/usr/local/news/bin/tally.unwanted
/bin/sh ../installit.sh -O news -G news -m 0550 -b .OLD nntpsend /usr/local/news/bin/nntpsend
/bin/sh ../installit.sh -O news -G news -m 0550 -b .OLD send-ihave /usr/local/news/send-ihave
/bin/sh ../installit.sh -O news -G news -m 0550 -b .OLD send-nntp /usr/local/news/send-nntp
/bin/sh ../installit.sh -O news -G news -m 0550 -b .OLD send-uucp /usr/local/news/send-uucp
/bin/sh ../installit.sh -O news -G news -m 0550 -b .OLD sendbatch
/usr/local/news/bin/sendbatch
/bin/sh ../installit.sh -O news -G news -m 0550 -b .OLD inncheck /usr/local/news/bin/inncheck
/bin/sh ../installit.sh -O news -G news -m 0550 -b .OLD innstat /usr/local/news/bin/innstat
/bin/sh ../installit.sh -O news -G news -m 0550 -b .OLD pgpverify
/usr/local/news/bin/pgpverify
/bin/sh ../installit.sh -O news -G news -m 0550 -b .OLD innwatch /usr/local/news/bin/innwatch
/bin/sh ../installit.sh -O news -G news -m 0440 -b .OLD innlog.pl /usr/local/news/innlog.pl
/bin/sh ../installit.sh -O news -G news -m 0440 -b .OLD innshellvars
/usr/local/news/innshellvars
/bin/sh ../installit.sh -O news -G news -m 0440 -b .OLD innshellvars.pl
/usr/local/news/innshellvars.pl
/bin/sh ../installit.sh -O news -G news -m 0440 -b .OLD startup_innd.pl
/usr/local/etc/control/startup_innd.pl
/bin/sh ../installit.sh -O news -G news -m 0440 -b .OLD startup_innd.pl
/usr/local/etc/control/startup_innd.pl
/bin/sh ../installit.sh -O news -G news -m 0440 -b .OLD filter_innd.pl
/usr/local/etc/control/filter_innd.pl
/bin/sh ../installit.sh -O news -G news -m 0440 -b .OLD filter_nnrpd.pl
/usr/local/etc/control/filter_nnrpd.pl
/bin/sh ../installit.sh -O news -G news -m 0440 -b .OLD innshellvars.tcl
/usr/local/news/innshellvars.tcl
/bin/sh ../installit.sh -O news -G news -m 0440 -b .OLD startup.tcl
/usr/local/etc/control/startup.tcl
/bin/sh ../installit.sh -O news -G news -m 0440 -b .OLD innshellvars.csh
/usr/local/news/innshellvars.csh
/bin/sh ../installit.sh -O news -G news -m 0440 -b .OLD filter.tcl
/usr/local/etc/control/filter.tcl
/bin/sh ../installit.sh -O news -G news -m 0550 -b .OLD scanspool
/usr/local/news/bin/scanspool
date >update
/bin/sh ../installit.sh -O news -G news -m 0444 -b .OLD newsfeeds /usr/local/news/newsfeeds
/bin/sh ../installit.sh -O news -G news -m 0440 -b .OLD hosts.nntp /usr/local/news/hosts.nntp
/bin/sh ../installit.sh -O news -G news -m 0440 -b .OLD hosts.nntp.nolimit
/usr/local/news/hosts.nntp.nolimit
/bin/sh ../installit.sh -O news -G news -m 0440 -b .OLD nnrp.access
/usr/local/news/nnrp.access
/bin/sh ../installit.sh -O news -G news -m 0440 -b .OLD passwd.nntp
/usr/local/news/passwd.nntp
/bin/sh ../installit.sh -O news -G news -m 0444 -b .OLD inn.conf /usr/local/news/inn.conf
/bin/sh ../installit.sh -O news -G news -m 0444 -b .OLD moderators /usr/local/news/moderators
/bin/sh ../installit.sh -O news -G news -m 0440 -b .OLD control.ctl
/usr/local/news/control.ctl
/bin/sh ../installit.sh -O news -G news -m 0440 -b .OLD expire.ctl /usr/local/news/expire.ctl
/bin/sh ../installit.sh -O news -G news -m 0440 -b .OLD nntpsend.ctl
/usr/local/news/nntpsend.ctl
/bin/sh ../installit.sh -O news -G news -m 0440 -b .OLD innwatch.ctl
/usr/local/news/innwatch.ctl
/bin/sh ../installit.sh -O news -G news -m 0444 -b .OLD distrib.pats
/usr/local/news/distrib.pats
/bin/sh ../installit.sh -O news -G news -m 0440 -b .OLD actsync.cfg
/usr/local/news/actsync.cfg
/bin/sh ../installit.sh -O news -G news -m 0440 -b .OLD actsync.ign
/usr/local/news/actsync.ign
/bin/sh ../installit.sh -O news -G news -m 0440 -b .OLD sample.control
/usr/local/news/bin/control/sample.control
make[2]: Leaving directory `/home/porting/inn-1.5.1/site'
make[1]: Leaving directory `/home/porting/inn-1.5.1'

Do not forget to update your cron entries.
Also run makehistory if you have to.
```

F.3 LDAP installation log

This is the complete installation log of the LDAP package (see 6.2, “LDAP” on page 121).

```
cd ldap-3.3
[porting - /home/porting/ldap-3.3]>
cd build/platforms/openvm
[porting - /home/porting/ldap-3.3/build/platforms/openvm]>
make install
** Setting platform to openvm and compiler openvm
( cd ../../..; make  install )
make[1]: Entering directory `/home/porting/ldap-3.3'
making buildtools
  cd build; make -w
make[2]: Entering directory `/home/porting/ldap-3.3/build'
make[2]: Leaving directory `/home/porting/ldap-3.3/build'

cd include; make -w install
make[2]: Entering directory `/home/porting/ldap-3.3/include'
mkdir -p /usr/local/include
install-sh -c -m 644 ldap.h /usr/local/include
install-sh -c -m 644 lber.h /usr/local/include
install-sh -c -m 644 proto-lber.h /usr/local/include
install-sh -c -m 644 proto-ldap.h /usr/local/include
install-sh -c -m 644 disptmpl.h /usr/local/include
install-sh -c -m 644 srchpref.h /usr/local/include
make[2]: Leaving directory `/home/porting/ldap-3.3/include'

cd libraries; make -w install
make[2]: Entering directory `/home/porting/ldap-3.3/libraries'
making install in /home/porting/ldap-3.3/libraries

  cd libavl; make -w install
make[3]: Entering directory `/home/porting/ldap-3.3/libraries/libavl'
make[3]: Nothing to be done for `install'.
make[3]: Leaving directory `/home/porting/ldap-3.3/libraries/libavl'

  cd liblber; make -w install
make[3]: Entering directory `/home/porting/ldap-3.3/libraries/liblber'
mkdir -p /usr/local/lib
install-sh -c -m 644 liblber.a /usr/local/lib
make[3]: Leaving directory `/home/porting/ldap-3.3/libraries/liblber'

  cd libldap; make -w install
make[3]: Entering directory `/home/porting/ldap-3.3/libraries/libldap'
mkdir -p /usr/local/lib
install-sh -c -m 644 libldap.a /usr/local/lib
mkdir -p /usr/local/etc
mv -f /usr/local/etc/ldapfriendly /usr/local/etc/ldapfriendly-
mv: /usr/local/etc/ldapfriendly: EDC5129I No such file or directory.
make[3]: [install] Error 1 (ignored)
install-sh -c -m 644 ldapfriendly /usr/local/etc
mv -f /usr/local/etc/ldapfilter.conf /usr/local/etc/ldapfilter.conf-
mv: /usr/local/etc/ldapfilter.conf: EDC5129I No such file or directory.
make[3]: [install] Error 1 (ignored)
install-sh -c -m 644 ldapfilter.conf /usr/local/etc
mv -f /usr/local/etc/ldaptemplates.conf /usr/local/etc/ldaptemplates.conf-
mv: /usr/local/etc/ldaptemplates.conf: EDC5129I No such file or directory.
make[3]: [install] Error 1 (ignored)
install-sh -c -m 644 ldaptemplates.conf /usr/local/etc
mv -f /usr/local/etc/ldapsearchprefs.conf /usr/local/etc/ldapsearchprefs.conf-
mv: /usr/local/etc/ldapsearchprefs.conf: EDC5129I No such file or directory.
make[3]: [install] Error 1 (ignored)
install-sh -c -m 644 ldapsearchprefs.conf /usr/local/etc
make[3]: Leaving directory `/home/porting/ldap-3.3/libraries/libldap'

  cd libldbm; make -w install
make[3]: Entering directory `/home/porting/ldap-3.3/libraries/libldbm'
make[3]: Nothing to be done for `install'.
make[3]: Leaving directory `/home/porting/ldap-3.3/libraries/libldbm'

  cd libldif; make -w install
make[3]: Entering directory `/home/porting/ldap-3.3/libraries/libldif'
make[3]: Nothing to be done for `install'.
make[3]: Leaving directory `/home/porting/ldap-3.3/libraries/libldif'
```

```
  cd liblthread; make -w install
make[3]: Entering directory `/home/porting/ldap-3.3/libraries/liblthread'
make[3]: Nothing to be done for `install'.
make[3]: Leaving directory `/home/porting/ldap-3.3/libraries/liblthread'

  cd macintosh; make -w install
make[3]: Entering directory `/home/porting/ldap-3.3/libraries/macintosh'
make[3]: Nothing to be done for `install'.
make[3]: Leaving directory `/home/porting/ldap-3.3/libraries/macintosh'

  cd msdos; make -w install
make[3]: Entering directory `/home/porting/ldap-3.3/libraries/msdos'
make[3]: Nothing to be done for `install'.
make[3]: Leaving directory `/home/porting/ldap-3.3/libraries/msdos'

  cd vms; make -w install
make[3]: Entering directory `/home/porting/ldap-3.3/libraries/vms'
make[3]: Nothing to be done for `install'.
make[3]: Leaving directory `/home/porting/ldap-3.3/libraries/vms'
make[2]: Leaving directory `/home/porting/ldap-3.3/libraries'

cd clients; make -w install
make[2]: Entering directory `/home/porting/ldap-3.3/clients'
making install in /home/porting/ldap-3.3/clients

  cd fax500; make -w install
make[3]: Entering directory `/home/porting/ldap-3.3/clients/fax500'
mkdir -p /usr/local/etc /usr/local/bin
install-sh -c -m 755 rp500 /usr/local/etc
sed -e 's%ETCDIR%/usr/local/etc%' xrpcomp > /tmp/xrpcomp.tmp
install-sh -c -m 755 /tmp/xrpcomp.tmp /usr/local/bin/xrpcomp
rm -f /tmp/xrpcomp.tmp
install-sh -c -m 755 fax500 /usr/local/etc
make[3]: Leaving directory `/home/porting/ldap-3.3/clients/fax500'

  cd finger; make -w install
make[3]: Entering directory `/home/porting/ldap-3.3/clients/finger'
mkdir -p /usr/local/etc
install-sh -c -m 755 in.xfingerd /usr/local/etc
make[3]: Leaving directory `/home/porting/ldap-3.3/clients/finger'

  cd gopher; make -w install
make[3]: Entering directory `/home/porting/ldap-3.3/clients/gopher'
mkdir -p /usr/local/etc
install-sh -c -m 755 go500 /usr/local/etc
install-sh -c -m 755 go500gw /usr/local/etc
mv -f /usr/local/etc/go500gw.help /usr/local/etc/go500gw.help-
mv: /usr/local/etc/go500gw.help: EDC5129I No such file or directory.
make[3]: [install] Error 1 (ignored)
install-sh -c -m 644 go500gw.help /usr/local/etc
make[3]: Leaving directory `/home/porting/ldap-3.3/clients/gopher'

  cd mail500; make -w install
make[3]: Entering directory `/home/porting/ldap-3.3/clients/mail500'
mkdir -p /usr/local/etc
install-sh -c -m 755 mail500 /usr/local/etc
make[3]: Leaving directory `/home/porting/ldap-3.3/clients/mail500'

  cd rcpt500; make -w install
make[3]: Entering directory `/home/porting/ldap-3.3/clients/rcpt500'
mkdir -p /usr/local/etc
install-sh -c -m 755 rcpt500 /usr/local/etc
install-sh -c -m 644 rcpt500.help /usr/local/etc
make[3]: Leaving directory `/home/porting/ldap-3.3/clients/rcpt500'

  cd tools; make -w install
make[3]: Entering directory `/home/porting/ldap-3.3/clients/tools'
mkdir -p /usr/local/bin
install-sh -c -m 755 ldapsearch /usr/local/bin
install-sh -c -m 755 ldapmodify /usr/local/bin
install-sh -c -m 755 ldapdelete /usr/local/bin
install-sh -c -m 755 ldapmodrdn /usr/local/bin
rm -f /usr/local/bin/ldapadd
ln /usr/local/bin/ldapmodify /usr/local/bin/ldapadd
make[3]: Leaving directory `/home/porting/ldap-3.3/clients/tools'

  cd ud; make -w install
```

```
make[3]: Entering directory `/home/porting/ldap-3.3/clients/ud'
mkdir -p /usr/local/bin
mv -f /usr/local/bin/ud /usr/local/bin/ud-
mv: /usr/local/bin/ud: EDC5129I No such file or directory.
make[3]: [install] Error 1 (ignored)
install-sh -c -m 775 ud /usr/local/bin
make[3]: Leaving directory `/home/porting/ldap-3.3/clients/ud'
make[2]: Leaving directory `/home/porting/ldap-3.3/clients'

cd servers; make -w install
make[2]: Entering directory `/home/porting/ldap-3.3/servers'
making install in /home/porting/ldap-3.3/servers
  cd ldapd; make -w install
make[3]: Entering directory `/home/porting/ldap-3.3/servers/ldapd'
uncomment the HAVEISODE=yes line in the Make-common file to build and install ldapd
make[3]: Leaving directory `/home/porting/ldap-3.3/servers/ldapd'
  cd slapd; make -w install
make[3]: Entering directory `/home/porting/ldap-3.3/servers/slapd'

make[4]: Entering directory `/home/porting/ldap-3.3/servers/slapd'

  cd back-ldbm; make -w all
make[5]: Entering directory `/home/porting/ldap-3.3/servers/slapd/back-ldbm'
make[6]: Entering directory `/home/porting/ldap-3.3/servers/slapd/back-ldbm'
make[6]: `libback-ldbm.a' is up to date.
make[6]: Leaving directory `/home/porting/ldap-3.3/servers/slapd/back-ldbm'
make[5]: Leaving directory `/home/porting/ldap-3.3/servers/slapd/back-ldbm'

  cd back-passwd; make -w all
make[5]: Entering directory `/home/porting/ldap-3.3/servers/slapd/back-passwd'
Include -DLDAP_PASSWD in SLAPD_BACKENDS in the
Make-common file to build the passwd backend
make[5]: Leaving directory `/home/porting/ldap-3.3/servers/slapd/back-passwd'

  cd back-shell; make -w all
make[5]: Entering directory `/home/porting/ldap-3.3/servers/slapd/back-shell'
make[6]: Entering directory `/home/porting/ldap-3.3/servers/slapd/back-shell'
make[6]: `libback-shell.a' is up to date.
make[6]: Leaving directory `/home/porting/ldap-3.3/servers/slapd/back-shell'
make[5]: Leaving directory `/home/porting/ldap-3.3/servers/slapd/back-shell'

make[5]: Entering directory `/home/porting/ldap-3.3/servers/slapd'
make[5]: `libbackends.a' is up to date.
make[5]: Leaving directory `/home/porting/ldap-3.3/servers/slapd'
make[4]: Leaving directory `/home/porting/ldap-3.3/servers/slapd'
make[4]: Entering directory `/home/porting/ldap-3.3/servers/slapd'
make[4]: `slapd' is up to date.
make[4]: Leaving directory `/home/porting/ldap-3.3/servers/slapd'
make[4]: Entering directory `/home/porting/ldap-3.3/servers/slapd/tools'
uncomment the HAVEISODE=yes line in the Make-common file to build edb2ldif
make[4]: Leaving directory `/home/porting/ldap-3.3/servers/slapd/tools'
install-sh -c -m 755 slapd /usr/local/etc
sed -e 's;%ETCDIR%;/usr/local/etc;' slapd.conf > /tmp/slapd.$
mv -f /usr/local/etc/slapd.conf /usr/local/etc/slapd.conf-
mv: /usr/local/etc/slapd.conf: EDC5129I No such file or directory.
make[3]: [/slapd.conf] Error 1 (ignored)
install-sh -c -m 644 /tmp/slapd.$ /usr/local/etc/slapd.conf
rm -f -f /tmp/slapd.$
mv -f /usr/local/etc/slapd.at.conf /usr/local/etc/slapd.at.conf-
mv: /usr/local/etc/slapd.at.conf: EDC5129I No such file or directory.
make[3]: [/slapd.at.conf] Error 1 (ignored)
install-sh -c -m 644 slapd.at.conf /usr/local/etc
mv -f /usr/local/etc/slapd.oc.conf /usr/local/etc/slapd.oc.conf-
mv: /usr/local/etc/slapd.oc.conf: EDC5129I No such file or directory.
make[3]: [/slapd.oc.conf] Error 1 (ignored)
install-sh -c -m 644 slapd.oc.conf /usr/local/etc
(cd tools; make -w install)
make[4]: Entering directory `/home/porting/ldap-3.3/servers/slapd/tools'
uncomment the HAVEISODE=yes line in the Make-common file to build edb2ldif
install-sh -c -m 755 ldif2ldbm /usr/local/etc
install-sh -c -m 755 ldif2index /usr/local/etc
install-sh -c -m 755 ldif2id2entry /usr/local/etc
install-sh -c -m 755 ldif2id2children /usr/local/etc
install-sh -c -m 755 ldbmcat /usr/local/etc
install-sh -c -m 755 centipede /usr/local/etc
install-sh -c -m 755 ldbmtest /usr/local/etc
install-sh -c -m 755 ldif /usr/local/etc
make[4]: Leaving directory `/home/porting/ldap-3.3/servers/slapd/tools'
```

```
make[3]: Leaving directory `/home/porting/ldap-3.3/servers/slapd'
  cd slurpd; make -w install
make[3]: Entering directory `/home/porting/ldap-3.3/servers/slurpd'
install-sh -c -m 755 slurpd /usr/local/etc
make[3]: Leaving directory `/home/porting/ldap-3.3/servers/slurpd'
make[2]: Leaving directory `/home/porting/ldap-3.3/servers'

cd doc; make -w install
make[2]: Entering directory `/home/porting/ldap-3.3/doc'
making install in /home/porting/ldap-3.3/doc

  cd guides; make -w install
make[3]: Entering directory `/home/porting/ldap-3.3/doc/guides'
make[3]: Nothing to be done for `install'.
make[3]: Leaving directory `/home/porting/ldap-3.3/doc/guides'

  cd man; make -w install
make[3]: Entering directory `/home/porting/ldap-3.3/doc/man'
making install in /home/porting/ldap-3.3/doc/man

  cd man1; make -w install
make[4]: Entering directory `/home/porting/ldap-3.3/doc/man/man1'
mkdir -p /usr/local/man/man1
installing /usr/local/man/man1/ldapdelete.1
installing /usr/local/man/man1/ldapmodify.1
installing /usr/local/man/man1/ldapadd.1 as link to ldapmodify.1
installing /usr/local/man/man1/ldapmodrdn.1
installing /usr/local/man/man1/ldapsearch.1
installing /usr/local/man/man1/ud.1
make[4]: Leaving directory `/home/porting/ldap-3.3/doc/man/man1'

  cd man3; make -w install
make[4]: Entering directory `/home/porting/ldap-3.3/doc/man/man3'
mkdir -p /usr/local/man/man3
installing /usr/local/man/man3/cldap_close.3
installing /usr/local/man/man3/cldap_open.3
installing /usr/local/man/man3/cldap_search_s.3
installing /usr/local/man/man3/cldap_setretryinfo.3
installing /usr/local/man/man3/lber-decode.3
installing /usr/local/man/man3/lber-encode.3
installing /usr/local/man/man3/ldap.3
installing /usr/local/man/man3/cldap.3 as link to ldap.3
installing /usr/local/man/man3/ldap_abandon.3
installing /usr/local/man/man3/ldap_add.3
installing /usr/local/man/man3/ldap_add_s.3 as link to ldap_add.3
installing /usr/local/man/man3/ldap_bind.3
installing /usr/local/man/man3/ldap_bind_s.3 as link to ldap_bind.3
installing /usr/local/man/man3/ldap_simple_bind.3 as link to ldap_bind.3
installing /usr/local/man/man3/ldap_simple_bind_s.3 as link to ldap_bind.3
installing /usr/local/man/man3/ldap_kerberos_bind_s.3 as link to ldap_bind.3
installing /usr/local/man/man3/ldap_kerberos_bind1.3 as link to ldap_bind.3
installing /usr/local/man/man3/ldap_kerberos_bind1_s.3 as link to ldap_bind.3
installing /usr/local/man/man3/ldap_kerberos_bind2.3 as link to ldap_bind.3
installing /usr/local/man/man3/ldap_kerberos_bind2_s.3 as link to ldap_bind.3
installing /usr/local/man/man3/ldap_unbind.3 as link to ldap_bind.3
installing /usr/local/man/man3/ldap_unbind_s.3 as link to ldap_bind.3
installing /usr/local/man/man3/ldap_set_rebind_proc.3 as link to ldap_bind.3
installing /usr/local/man/man3/ldap_cache.3
installing /usr/local/man/man3/ldap_enable_cache.3 as link to ldap_cache.3
installing /usr/local/man/man3/ldap_disable_cache.3 as link to ldap_cache.3
installing /usr/local/man/man3/ldap_destroy_cache.3 as link to ldap_cache.3
installing /usr/local/man/man3/ldap_flush_cache.3 as link to ldap_cache.3
installing /usr/local/man/man3/ldap_uncache_entry.3 as link to ldap_cache.3
installing /usr/local/man/man3/ldap_uncache_request.3 as link to ldap_cache.3
installing /usr/local/man/man3/ldap_set_cache_options.3 as link to ldap_cache.3
installing /usr/local/man/man3/ldap_charset.3
installing /usr/local/man/man3/ldap_set_string_translators.3 as link to ldap_charset.3
installing /usr/local/man/man3/ldap_enable_translation.3 as link to ldap_charset.3
installing /usr/local/man/man3/ldap_translate_from_t61.3 as link to ldap_charset.3
installing /usr/local/man/man3/ldap_translate_to_t61.3 as link to ldap_charset.3
installing /usr/local/man/man3/ldap_t61_to_8859.3 as link to ldap_charset.3
installing /usr/local/man/man3/ldap_8859_to_t61.3 as link to ldap_charset.3
installing /usr/local/man/man3/ldap_compare.3
installing /usr/local/man/man3/ldap_compare_s.3 as link to ldap_compare.3
installing /usr/local/man/man3/ldap_delete.3
installing /usr/local/man/man3/ldap_delete_s.3 as link to ldap_delete.3
installing /usr/local/man/man3/ldap_disptmpl.3
installing /usr/local/man/man3/ldap_init_templates.3 as link to ldap_disptmpl.3
```

```
installing /usr/local/man/man3/ldap_init_templates_buf.3 as link to ldap_disptmpl.3
installing /usr/local/man/man3/ldap_free_templates.3 as link to ldap_disptmpl.3
installing /usr/local/man/man3/ldap_first_disptmpl.3 as link to ldap_disptmpl.3
installing /usr/local/man/man3/ldap_next_disptmpl.3 as link to ldap_disptmpl.3
installing /usr/local/man/man3/ldap_oc2template.3 as link to ldap_disptmpl.3
installing /usr/local/man/man3/ldap_tmplattrs.3 as link to ldap_disptmpl.3
installing /usr/local/man/man3/ldap_first_tmplrow.3 as link to ldap_disptmpl.3
installing /usr/local/man/man3/ldap_next_tmplrow.3 as link to ldap_disptmpl.3
installing /usr/local/man/man3/ldap_first_tmplcol.3 as link to ldap_disptmpl.3
installing /usr/local/man/man3/ldap_next_tmplcol.3 as link to ldap_disptmpl.3
installing /usr/local/man/man3/ldap_entry2text.3
installing /usr/local/man/man3/ldap_entry2text_search.3 as link to ldap_entry2text.3
installing /usr/local/man/man3/ldap_vals2text.3 as link to ldap_entry2text.3
installing /usr/local/man/man3/ldap_entry2html.3 as link to ldap_entry2text.3
installing /usr/local/man/man3/ldap_entry2html_search.3 as link to ldap_entry2text.3
installing /usr/local/man/man3/ldap_vals2html.3 as link to ldap_entry2text.3
installing /usr/local/man/man3/ldap_error.3
installing /usr/local/man/man3/ldap_perror.3 as link to ldap_error.3
installing /usr/local/man/man3/ld_errno.3 as link to ldap_error.3
installing /usr/local/man/man3/ldap_result2error.3 as link to ldap_error.3
installing /usr/local/man/man3/ldap_errlist.3 as link to ldap_error.3
installing /usr/local/man/man3/ldap_err2string.3 as link to ldap_error.3
installing /usr/local/man/man3/ldap_first_attribute.3
installing /usr/local/man/man3/ldap_next_attribute.3 as link to ldap_first_attribute.3
installing /usr/local/man/man3/ldap_first_entry.3
installing /usr/local/man/man3/ldap_next_entry.3 as link to ldap_first_entry.3
installing /usr/local/man/man3/ldap_count_entries.3 as link to ldap_first_entry.3
installing /usr/local/man/man3/ldap_friendly.3
installing /usr/local/man/man3/ldap_friendly_name.3 as link to ldap_friendly.3
installing /usr/local/man/man3/ldap_free_friendlymap.3 as link to ldap_friendly.3
installing /usr/local/man/man3/ldap_get_dn.3
installing /usr/local/man/man3/ldap_explode_dn.3 as link to ldap_get_dn.3
installing /usr/local/man/man3/ldap_explode_dns.3 as link to ldap_get_dn.3
installing /usr/local/man/man3/ldap_dn2ufn.3 as link to ldap_get_dn.3
installing /usr/local/man/man3/ldap_is_dns_dn.3 as link to ldap_get_dn.3
installing /usr/local/man/man3/ldap_get_values.3
installing /usr/local/man/man3/ldap_get_values_len.3 as link to ldap_get_values.3
installing /usr/local/man/man3/ldap_value_free.3 as link to ldap_get_values.3
installing /usr/local/man/man3/ldap_value_free_len.3 as link to ldap_get_values.3
installing /usr/local/man/man3/ldap_count_values.3 as link to ldap_get_values.3
installing /usr/local/man/man3/ldap_count_values_len.3 as link to ldap_get_values.3
installing /usr/local/man/man3/ldap_getfilter.3
installing /usr/local/man/man3/ldap_init_getfilter.3 as link to ldap_getfilter.3
installing /usr/local/man/man3/ldap_init_getfilter_buf.3 as link to ldap_getfilter.3
installing /usr/local/man/man3/ldap_getfilter_free.3 as link to ldap_getfilter.3
installing /usr/local/man/man3/ldap_getfirstfilter.3 as link to ldap_getfilter.3
installing /usr/local/man/man3/ldap_getnextfilter.3 as link to ldap_getfilter.3
installing /usr/local/man/man3/ldap_setfilteraffixes.3 as link to ldap_getfilter.3
installing /usr/local/man/man3/ldap_build_filter.3 as link to ldap_getfilter.3
installing /usr/local/man/man3/ldap_modify.3
installing /usr/local/man/man3/ldap_modify_s.3 as link to ldap_modify.3
installing /usr/local/man/man3/ldap_mods_free.3 as link to ldap_modify.3
installing /usr/local/man/man3/ldap_modrdn.3
installing /usr/local/man/man3/ldap_modrdn_s as link to ldap_modrdn.3
installing /usr/local/man/man3/ldap_modrdn2 as link to ldap_modrdn.3
installing /usr/local/man/man3/ldap_modrdn2_s as link to ldap_modrdn.3
installing /usr/local/man/man3/ldap_open.3
installing /usr/local/man/man3/ldap_init.3 as link to ldap_open.3
installing /usr/local/man/man3/ldap_result.3
installing /usr/local/man/man3/ldap_msgfree.3 as link to ldap_result.3
installing /usr/local/man/man3/ldap_search.3
installing /usr/local/man/man3/ldap_search_s.3 as link to ldap_search.3
installing /usr/local/man/man3/ldap_search_st.3 as link to ldap_search.3
installing /usr/local/man/man3/ldap_searchprefs.3
installing /usr/local/man/man3/ldap_init_searchprefs.3 as link to ldap_searchprefs.3
installing /usr/local/man/man3/ldap_init_searchprefs_buf.3 as link to ldap_searchprefs.3
installing /usr/local/man/man3/ldap_free_searchprefs.3 as link to ldap_searchprefs.3
installing /usr/local/man/man3/ldap_first_searchobj.3 as link to ldap_searchprefs.3
installing /usr/local/man/man3/ldap_next_searchobj.3 as link to ldap_searchprefs.3
installing /usr/local/man/man3/ldap_sort.3
installing /usr/local/man/man3/ldap_sort_entries.3 as link to ldap_sort.3
installing /usr/local/man/man3/ldap_sort_values.3 as link to ldap_sort.3
installing /usr/local/man/man3/ldap_sort_strcasecmp.3 as link to ldap_sort.3
installing /usr/local/man/man3/ldap_ufn.3
installing /usr/local/man/man3/ldap_ufn_search_s.3 as link to ldap_ufn.3
installing /usr/local/man/man3/ldap_ufn_search_c.3 as link to ldap_ufn.3
installing /usr/local/man/man3/ldap_ufn_search_ct.3 as link to ldap_ufn.3
installing /usr/local/man/man3/ldap_ufn_setprefix.3 as link to ldap_ufn.3
```

```
installing /usr/local/man/man3/ldap_ufn_setfilter.3 as link to ldap_ufn.3
installing /usr/local/man/man3/ldap_ufn_timeout.3 as link to ldap_ufn.3
installing /usr/local/man/man3/ldap_url.3
installing /usr/local/man/man3/ldap_is_ldap_url.3 as link to ldap_url.3
installing /usr/local/man/man3/ldap_url_parse.3 as link to ldap_url.3
installing /usr/local/man/man3/ldap_free_urldesc.3 as link to ldap_url.3
installing /usr/local/man/man3/ldap_url_search.3 as link to ldap_url.3
installing /usr/local/man/man3/ldap_url_search_s.3 as link to ldap_url.3
installing /usr/local/man/man3/ldap_url_search_st.3 as link to ldap_url.3
installing /usr/local/man/man3/regex.3
make[4]: Leaving directory `/home/porting/ldap-3.3/doc/man/man3'

  cd man5; make -w install
make[4]: Entering directory `/home/porting/ldap-3.3/doc/man/man5'
mkdir -p /usr/local/man/man5
installing /usr/local/man/man5/ldapfilter.conf.5
installing /usr/local/man/man5/ldapfriendly.5
installing /usr/local/man/man5/ldapsearchprefs.conf.5
installing /usr/local/man/man5/ldaptemplates.conf.5
installing /usr/local/man/man5/ldif.5
installing /usr/local/man/man5/slapd.conf.5
installing /usr/local/man/man5/slapd.replog.5
installing /usr/local/man/man5/ud.conf.5
make[4]: Leaving directory `/home/porting/ldap-3.3/doc/man/man5'

  cd man8; make -w install
make[4]: Entering directory `/home/porting/ldap-3.3/doc/man/man8'
mkdir -p /usr/local/man/man8
installing /usr/local/man/man8/centipede.8
installing /usr/local/man/man8/chlog2replog.8
installing /usr/local/man/man8/edb2ldif.8
installing /usr/local/man/man8/go500.8
installing /usr/local/man/man8/go500gw.8
installing /usr/local/man/man8/in.xfingerd.8
installing /usr/local/man/man8/ldapd.8
installing /usr/local/man/man8/ldbmcat.8
installing /usr/local/man/man8/ldif.8
installing /usr/local/man/man8/ldif2ldbm.8
installing /usr/local/man/man8/ldif2index.8 as link to ldif2ldbm.8
installing /usr/local/man/man8/ldif2id2entry.8 as link to ldif2ldbm.8
installing /usr/local/man/man8/ldif2id2children.8 as link to ldif2ldbm.8
installing /usr/local/man/man8/mail500.8
installing /usr/local/man/man8/fax500.8 as link to mail500.8
installing /usr/local/man/man8/rcpt500.8
installing /usr/local/man/man8/slapd.8
installing /usr/local/man/man8/slurpd.8
make[4]: Leaving directory `/home/porting/ldap-3.3/doc/man/man8'
make[3]: Leaving directory `/home/porting/ldap-3.3/doc/man'

  cd rfc; make -w install
make[3]: Entering directory `/home/porting/ldap-3.3/doc/rfc'
make[3]: Nothing to be done for `install'.
make[3]: Leaving directory `/home/porting/ldap-3.3/doc/rfc'
make[2]: Leaving directory `/home/porting/ldap-3.3/doc'
make[1]: Leaving directory `/home/porting/ldap-3.3'
[porting - /home/porting/ldap-3.3/build/platforms/openvm]>
```

F.4 IRCD installation log and configuration file

The install logs and complete IRC configuration file are reproduced here (see 7.2, “Internet Relay Chat (IRC)” on page 156).

F.4.1 IRCD installation log

```
gunzip ircd.tar.gz

gunzip: ircd.tar.gz: decompression OK, trailing garbage ignored
[porting - /home/porting]>
dir ircd.tar
-rw-r--r--   1 porting    system   3100672 Apr 12 07:11 ircd.tar
[porting - /home/porting]>
pax -rf ircd.tar
[porting - /home/porting]>
ls -ld irc*
drwxrwxrwx   1 porting    system         0 Apr 21 14:01 irc2.8.21
-rw-r--r--   1 porting    system   3100672 Apr 12 07:11 ircd.tar
[porting - /home/porting]>
cd irc2.8.21
[porting - /home/porting/irc2.8.21]>
[porting - /home/porting/irc2.8.21]>
make install
Building common
make[1]: Entering directory `/home/porting/irc2.8.21/common'
make[1]: Nothing to be done for `build'.
make[1]: Leaving directory `/home/porting/irc2.8.21/common'
Building ircd
make[1]: Entering directory `/home/porting/irc2.8.21/ircd'
(cd ../common; make 'CFLAGS=-I../include -D_OE_SOCKETS -W
c,hwopts\\\(string\\\),langlvl\\\(ansi\\\)' 'CC=c89' 'IRCDLIBS=-l//posxsoc k -l//vmmtlib
-lxpg4' 'LDFLAGS=' 'IRCDMODE=711' build);
make[2]: Entering directory `/home/porting/irc2.8.21/common'
make[2]: Nothing to be done for `build'.
make[2]: Leaving directory `/home/porting/irc2.8.21/common'
make[1]: Leaving directory `/home/porting/irc2.8.21/ircd'
Building irc
make[1]: Entering directory `/home/porting/irc2.8.21/irc'
c89 -o irc c_bsd.o c_msg.o c_numeric.o c_version.o edit.o help.o ignore.o irc.o screen.o str.o
c_debug.o ctcp.o bsd.o dbuf.o packet.o send.o match.o parse.o support.o
WARNING EDC4011: Unresolved writable static references are detected.
DMSLIO201W The following names are undefined:
@@SOCKT @@I@ADR @@GHBNA @@CONEC @@SELCT ECHO NOCRMODE CLEAR REFRESH TSTP SETLINEB @@SEND@
make[1]: Leaving directory `/home/porting/irc2.8.21/irc'
chmod +x ./bsdinstall
Installing ircd
make[1]: Entering directory `/home/porting/irc2.8.21/ircd'
(cd ../common; make 'CFLAGS=-I../include -D_OE_SOCKETS -W
c,hwopts\\\(string\\\),langlvl\\\(ansi\\\)' 'CC=c89' 'IRCDLIBS=-l//posxsock -l//vmmtlib
-lxpg4' 'LDFLAGS=' 'IRCDMODE=711' build);
make[2]: Entering directory `/home/porting/irc2.8.21/common'
make[2]: Nothing to be done for `build'.
make[2]: Leaving directory `/home/porting/irc2.8.21/common'
if [ ! -d /usr/local/src/ircd -a ! -f /usr/local/src/ircd ] ; then \
mkdir /usr/local/src/ircd; \
fi
../bsdinstall -c -s -m 711 ircd /usr/local/src/ircd
GSU6781 strip: file "/usr/local/src/ircd/ircd": Not an executable file ../bsdinstall -c -s -m
700 chkconf /usr/local/src/ircd
GSU6781 strip: file "/usr/local/src/ircd/chkconf": Not an executable file /bin/cp
../doc/example.conf /usr/local/src/ircd
touch /usr/local/src/ircd/ircd.motd
/bin/rm -f /usr/local/src/ircd/ircd.m4
touch /usr/local/src/ircd/ircd.m4
chmod +x buildm4
./buildm4 /usr/local/src/ircd
egrep: GSU6003 input file "/etc/resolv.conf": EDC5129I No such file or directory.
make[1]: Leaving directory `/home/porting/irc2.8.21/ircd'
Installing irc
make[1]: Entering directory `/home/porting/irc2.8.21/irc'
../bsdinstall -c -s -m 711 irc /usr/local/src/ircd
GSU6781 strip: file "/usr/local/src/ircd/irc": Not an executable file
make[1]: Leaving directory `/home/porting/irc2.8.21/irc'
Installing doc make[1]: Entering directory `/home/porting/irc2.8.21/doc'
```

```
 ../install -c -m 644 ircd.8 /usr/local/man/man8
 /etc/chown: ../install 81: not found
 ../install -c -m 644 irc.1 /usr/local/man/man1
 /etc/chown: ../install 81: not found
 make[1]: Leaving directory `/home/porting/irc2.8.21/doc'
[porting - /home/porting/irc2.8.21]>
```

F.4.2 IRCD configuration file

```
# IRC - Internet Relay Chat, doc/example.conf
# Copyright (C) 1994, Helen Rose
#
#   This program is free software; you can redistribute it and/or modify
#   it under the terms of the GNU General Public License as published by
#   the Free Software Foundation; either version 1, or (at your option)
#   any later version.
#
#   This program is distributed in the hope that it will be useful,
#   but WITHOUT ANY WARRANTY; without even the implied warranty of
#   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#   GNU General Public License for more details.
#
#   You should have received a copy of the GNU General Public License
#   along with this program; if not, write to the Free Software
#   Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
#
# This is an example configuration file for the IRC server
#
# You only need an ircd.conf (IRC server configuration file) if you are
# running an IRC server. If you are running a standalone client this file
# is not necessary.
#
# This file will explain the various lines in the IRC server
# configuration file. Not all lines are mandatory. You can check to make
# sure that your configuration file is correct by using the program
# "chkconf", provided in the server distribution (and when you do "make
# install" this program will be installed in the same directory as the irc
# server).
#
# The options for whether a line is needed or not are:
# MANDATORY: you absolutely MUST have this line
# NETWORKED: you must have this line if you are connecting this irc
#            server to any other server (servers can run standalone).
# SUGGESTED: it is highly suggested that you use this line
# OPTIONAL: it's completely up to you whether to define this or not
# DISCOURAGED: you really really should not use this line if at all
#              possible.
# NOT NECESSARY: an old or out of date line that isn't needed.
#
# MANDATORY lines are absolute *musts*, that is, if you do not have this
# line then your server will not work properly. SUGGESTED lines are
# close-to-mandatory (that is, the server will run without it, but you are
# highly encouraged to use these lines).
#
# Note that "*" in a field indicates an "unused" field.
#
#
# =========================================================================
# NOTE! this entire configuration file is read UPSIDE-DOWN! So if you have
# to put something in a specific order (for example, client-connection
# lines), put them in reverse order!
# =========================================================================
#
#
# M: [MANDATORY]. This line sets your server's name, description, and
# port number. Fields, in order, are:
#
# M:hostname:*:Description Of Your Server:6667
#
M:totvm1.itso.ibm.com:*:IBM ITSO Poughkeepsie:6667
#
# A: [MANDATORY]. This line lists your administrative information
# (contact address, etc). To view this information, /admin (server) will
# show it to you.
#
# The A: line has no set information, in fact, you can put arbitrary text
# in there if you wish (it is encouraged that you put at *least* a contact
```

```
# address for a person responsible for the irc server, however)
#
A:ITSO Poughkeepise:LSA002R:Neale Ferguson <neale@totvm1.itso.ibm.com>
#
# Y: [SUGGESTED]. These lines define connection classes. Connection
# classes allow you to fine-tune your client and server connections. It is
# suggested that clients and servers be placed in seperate classes, and if
# you have lots of server connections (if you do have lots of servers you
# shouldn't be reading this file :-) each set of servers (defined
# arbitrarily by you) should have its own class. If you have clients
# coming in from lots of different sites, you may want to seperate them
# out into classes. For instance, you may want to put local users in one
# class, with remote users in another class.
#
# The class numbers are not arbitrary. In auto-connecting servers -- that
# is, servers that you have a port number (e.g. 6667) on the end of the C:
# line (see below) the higher the number the higher the priority in
# auto-connecting.
#
# The fields in order are: class number, ping frequency (in seconds),
# connect frequency (in seconds), maximum number of links (used for
# auto-connecting, and for limiting the number of clients in that class),
# and sendq (this overrides any value set in include/config.h for #define
# MAXSENDQLENGTH).
#
# Note that it is a good idea to have ping frequency the same at both ends
# of the link.
#
# in this case, connect-frequency is 0 indicating that this is a client
# class (servers never connect to clients, it is the other way around).
Y:1:90:0:20:100000
#
# this is a normal server connection (normal as of March, 1994)
Y:2:90:300:1:600000
#
Y:10:90:0:3:100000
#
# I: [MANDATORY]. The I: lines are client-authorization lines. Without
# these lines, no clients will be able to connect to your server.
# Wildcards ("*") are permitted. Passwords are also permitted (clients can
# be configured to send passwords).
#
# Ident (for more information on this, see rfc1413) can also be used by
# placing a @ in the appropriate fields.
#
# Fields are as follows:
# I:IP-address-mask:optional password:domain-mask::connection class (opt)
#
# With a password..... This will allow anyone from anywhere to connect
# as long as they know the password ("foobar"). Note listing this I: line
# first, it will be read *last*, meaning it is the "fall-through". That
# is, anyone who doesn't match the I: lines listed below must know the
# password ("foobar") to connect.
#
I:*@*:foobar:*@*::1
# This is a standard vanilla I: line which will permit anyone with an IP
# address starting with 128.197 OR with a hostname ending in .bu.edu to
# connect to the server. NOTE, the ircd matches on the *right-most* match,
# so if I connect as hrose@csa.bu.edu (which is hrose@128.197.10.3) I will
# show up on irc as hrose@csa.bu.edu since that is the first match it
# found. (Even though the second match is valid).
I:9.*::*.ibm.com::1
#
# using ident
I:*@9.*::*@*.ibm.com::1
# and you can even specify just certain usernames running ident (as long
# as the client's site is running the ident daemon):
I:NOMATCH::neale@totvm1.itso.ibm.com::1
# putting NOMATCH in the first field will stop the ircd from matching
# automatically against the IP address and it will force the server to
# match against the hostname. (the "NOMATCH" string is not mandatory, you
# can use any arbitrary text in the first field).
#
#
# O: [OPTIONAL]. These lines define operator access. You do not need to
# have an operator to run a server. A well configured leaf site should not
# need an operator online, if it's connections are well defined, the irc
# administrator can use kill -HUP on the ircd to reload the configuration
```

```
# file.
# The fields are as follows:
# O:hostname (ident "@" permitted):password:NickName
# if the person in "NickName" is not coming from the hostname defined in
# the first field then the person will get the error message "No O: lines
# for your host".
# NOTE that since Crypted Passwords are defined by default in
# include/config.h this text probably will not be plaintext. See
# ircd/crypt/README for more information.
#
O:*.ibm.com:Neale:Trillian::10
#
# and this line forces ident:
O:neale@totvm1.itso.ibm.com:Neale:Trillian::10
#
# This line is a "local operator", it is specified with a lower-case "o"
# -- it is the only lower-case type in the ircd.conf file.
#
# this line permits the nickname "jhs" with the password of "ITBites" to
# be a local operator only (be able to issue commands locally -- can /kill
# and /squit and /connect -- but *only* locally)
#
o:*.ibm.com:ITBites:jhs::10
#
# a crypted password line (NOTE that if you have crypted passwords, *all*
# of you passwords must be crypted! In fact, if you are getting an error
# "Incorrect Password" it may well be because crypted passwords are
# defined and you have used plaintext.  So my example of plaintext and
# crypted strings in the same IRC server configuration file is an
# impossibility (but it is just theoretical, which is why I explained both).
#
# O:rocker@csa.bu.edu:T0eiVgHrqeKTQ:Rocker::10
#
## C: [NETWORKED]. These lines define what servers your server tries to
# connect to.
# N: [NETWORKED]. These lines define what servers your server permits
# connections to be initiated from.
# C/N lines MUST be used in pairs. You cannot have one without the other.
#
# C: lines contain the following fields:
# C:remote server's hostname:passwd:remote server's name:port:conn class
# (connection class)
# N: lines contain the following fields:
# N:remote server's hostname:passwd:remote server's name:host mask:conn class
# (connection class)
# "host mask" is the number of parts in *your* hostname to mask to. For
# instance, with my servername being "csa.bu.edu", if I wanted to present
# my servername to be "*.bu.edu" I would have a host-mask portion of "1".
#
# it is *strongly* advised that your C/N line passwords be different for
# security's sake.
#
# ident is allowed in the server's hostname part of the field.
# these lines tell the server to automatically (note the port number, that
# means automatic connection) connect to cs-ftp.bu.edu:
# C:hrose@cs-ftp.bu.edu:bigspark:cs-ftp.bu.edu:6667:2
# N:hrose@cs-ftp.bu.edu:bigalpha:cs-ftp.bu.edu::2
#
# This server's connection lines are more vanilla, masking the host to
# *.bu.edu (as described above):
# C:irc-2.mit.edu:camelsrk001:irc-2.mit.edu::2
# N:irc-2.mit.edu:andsoarellamas:irc-2.mit.edu:1:2
#
# K: [OPTIONAL]. These lines define user@host patterns to be banned from
# this particular server (with an optional time field). Note that K: lines
# are *not* global, and if you ban a user they can still use any other IRC
# server (unless they have specifically been banned there as well).
#
# the fields are defined as:
# K:hostmask:time field:username
# wildcards are permitted in any one of the fields, in other words, you can
# K:*::* if you wanted (but your server wouldn't be used much ;-)
#
# This K: line bans the username "FSSPR" (the wildcards are used to make
# sure that any ident-checking character will match) on any machine from
# the University of Alaska.
# K:*.alaska.edu::*FSSPR*
#
```

```
# This K: line bans any users from acs*.bu.edu between the hours of 8am
# and 12pm and 1pm and 5pm (the time is always the server's local time):
# K:acs*.bu.edu:0800-1200,1300-1700:*
# Note that 24 hour time is used (no "AM" or "PM").
#
# R: [DISCOURAGED]. These lines restrict user access based on a more
# stringent checking system than is available in the K: line. It looks for
# a match (based on hostname and username) and then runs an outside
# program (which MUST be specified using a full pathname). The output of
# the program should be a string in the form "Y <message>" (which permits
# access for the user) or "N <message>" (which denies access for the
# user). If "Y <message>" is received by the server, the server ignores
# the message and permits access for the user. If "N <message>" is
# returned, the server tells the user that he/she is not permitted to
# access that irc server, and gives the reason.
#
# Again, like K: lines, R: lines are local and thus not very effective in
# blocking certain machines from having IRC access.
#
# Use of R: requires that you have defined R_LINES in include/config.h
#
# The fields are as follows:
# R:hostmask:/full/path/to/program:username
# you can use wildcards in either the hostmask or username portion
#
# R:csl.bu.edu:/home/hrose/bin.sun3/sun3access:*
#
# Q: [DISCOURAGED]. These lines "quarantine" specified servers.  Because
# of the way they operates, the same Q: lines MUST be installed by
# everyone or the net will keep breaking. I CANNOT EMPHASIZE THIS ENOUGH.
# Do NOT use Q: lines lightly!
#
# The fields are as follows:
# Q:*:reason why quarantine is in place:servername
#
# Q::this server is too slow and lags the net:cm5.eng.umd.edu
#
# L: [OPTIONAL]. These lines "Leaf" specified servers. They are only
# useful if you are a non-leaf site yourself. There are two ways you can
# use L: lines. The first will limit one particular site to a particular
# tree depth (including 0, which would mean the server has to connect with
# no servers linked behind it otherwise the connection will fail). The
# second will allow you to be selective about which other servers you wish
# the connecting server to behave as a leaf towards.
#
# The fields are as follows:
# L:disallow connections to this hostmask::server name:depth
# For example, this will force kaja.gi.alaska.edu to connect only as a
# leaf (if it is not a leaf, the link will be dropped):
# L:::kaja.gi.alaska.edu
# This line will force cm5.eng.umd.edu to have a depth of only 1 below it
# (that is, it is allowed to have only leaves connected to it):
# L:::cm5.eng.umd.edu:1
#
# This line will prohibit anything matching *.edu to be connected behind
# any server matching *.au:
# L:*.edu::*.au
#
# H: [OPTIONAL]. These lines define who you permit to act as a "hub" to
# you (that is, who you permit to connect non-leafed servers to you).
#
# the first field may use wildcards, the third field *must* be an exact
# match for a server's name (NOT a server's hostname, if they differ, the
# server's name must be used). If the servername is a wildcard (e.g. *.au)
# that is an acceptable name for the third field.
#
# The fields are as follows:
# H:servers which are permitted entry::hub server
#
# Example, permit cs-ftp.bu.edu to allow any servers behind it to connect:
# H:*::cs-ftp.bu.edu
#
# Example, permit irc-2.mit.edu to allow any MIT servers behind it to
# connect:
# H:*.mit.edu::irc-2.mit.edu
#
# P: [OPTIONAL]. This field allows the server to listen on various ports
# (other than 6667) for connections. Any internet domain port that is
```

```
# below 1024 means the ircd has to be run from inetd. The server can
# listen to ports in the UNIX domain or the internet domain. If you wish
# to create a port in the UNIX domain you must compile with UNIXPORT
# defined in include/config.h. If you are permitting connections to a
# seperate port, you can control access to that port by the host field.
#
# The fields are as follows::
# P:hostmask or UNIX socket file:*:*:port number
# for example, an internet domain socket on port 6665 for South African
# users:
# P:*.za:*:*:6665
#
# This line is an example of a UNIX domain socket in /tmp
P:/tmp/.ircd:*:*:6666
```

F.5 Apache installation log and configuration file

The Apache installation logs and configuration files are reproduced in this section (see 6.3, “Apache” on page 136).

F.5.1 Apache installation log

```
gunzip apache_1.3.6.tar.gz
[porting - /home/porting]>
pax -rf apache_1.3.6.tar
[porting - /home/porting]>
ls -ld ap*
-rw-rw-rw-   1 porting     system   11018752 May 11 18:22 apache-1.3.6.tar
drwxrwxrwx   1 porting     system         0 May  7 13:06 apache_1.3.6
[porting - /home/porting]>
cd apache_1.3.6
[porting - /home/porting]>
make install
make[1]: Entering directory `/home/porting/apache_1.3.6'
===> [mktree: Creating Apache installation tree]
./src/helpers/mkdir.sh /usr/local/apache/bin
mkdir /usr/local/apache
mkdir /usr/local/apache/bin
./src/helpers/mkdir.sh /usr/local/apache/bin
./src/helpers/mkdir.sh /usr/local/apache/libexec
mkdir /usr/local/apache/libexec
./src/helpers/mkdir.sh /usr/local/apache/man/man1
mkdir /usr/local/apache/man
mkdir /usr/local/apache/man/man1
./src/helpers/mkdir.sh /usr/local/apache/man/man8
mkdir /usr/local/apache/man/man8
./src/helpers/mkdir.sh /usr/local/apache/conf
mkdir /usr/local/apache/conf
./src/helpers/mkdir.sh /usr/local/apache/htdocs
mkdir /usr/local/apache/htdocs
./src/helpers/mkdir.sh /usr/local/apache/icons
mkdir /usr/local/apache/icons
./src/helpers/mkdir.sh /usr/local/apache/cgi-bin
mkdir /usr/local/apache/cgi-bin
./src/helpers/mkdir.sh /usr/local/apache/include
mkdir /usr/local/apache/include
./src/helpers/mkdir.sh /usr/local/apache/logs
mkdir /usr/local/apache/logs
./src/helpers/mkdir.sh /usr/local/apache/logs
./src/helpers/mkdir.sh /usr/local/apache/proxy
mkdir /usr/local/apache/proxy
<=== [mktree]
===> [programs: Installing Apache httpd program and shared objects]
./src/helpers/install.sh -c -m 755 -s ./src/httpd /usr/local/apache/bin/httpd
GSU6781 strip: file "/usr/local/apache/bin/#inst.2727#": Not an executable file
./src/helpers/install.sh -c -m644 ./src/support/httpd.8 /usr/local/apache/man/man8/httpd.8
<=== [programs]
===> [support: Installing Apache support programs and scripts]
./src/helpers/install.sh -c -m 755 -s ./src/support/ab /usr/local/apache/bin/ab
GSU6781 strip: file "/usr/local/apache/bin/#inst.6833#": Not an executable file
./src/helpers/install.sh -c -m 644 ./src/support/ab.8 /usr/local/apache/man/man8/ab.8
./src/helpers/install.sh -c -m 755  ./src/support/apachectl[*]
/usr/local/apache/bin/apachectl
./src/helpers/install.sh -c -m 644 ./src/support/apachectl.8
/usr/local/apache/man/man8/apachectl.8
./src/helpers/install.sh -c -m755 -s ./src/support/htpasswd /usr/local/apache/bin/htpasswd
GSU6781 strip: file "/usr/local/apache/bin/#inst.6840#": Not an executable file
./src/helpers/install.sh -c -m 644 ./src/support/htpasswd.1
/usr/local/apache/man/man1/htpasswd.1
./src/helpers/install.sh -c -m755 -s ./src/support/htdigest /usr/local/apache/bin/htdigest
GSU6781 strip: file "/usr/local/apache/bin/#inst.6846#": Not an executable file
./src/helpers/install.sh -c -m 644 ./src/support/htdigest.1
/usr/local/apache/man/man1/htdigest.1
./src/helpers/install.sh -c -m 755  ./src/support/dbmmanage[*]
/usr/local/apache/bin/dbmmanage
./src/helpers/install.sh -c -m 644 ./src/support/dbmmanage.1
/usr/local/apache/man/man1/dbmmanage.1
./src/helpers/install.sh -c -m 755 -s ./src/support/logresolve
/usr/local/apache/bin/logresolve
GSU6781 strip: file "/usr/local/apache/bin/#inst.6856#": Not an executable file
```

```
./src/helpers/install.sh -c -m 644 ./src/support/logresolve.8
/usr/local/apache/man/man8/logresolve.8
./src/helpers/install.sh -c -m 755 -s ./src/support/rotatelogs
/usr/local/apache/bin/rotatelogs
GSU6781 strip: file "/usr/local/apache/bin/#inst.6862#": Not an executable file
./src/helpers/install.sh -c -m 644 ./src/support/rotatelogs.8
/usr/local/apache/man/man8/rotatelogs.8
./src/helpers/install.sh-c-m755 ./src/support/apxs[*] /usr/local/apache/bin/apxs
./src/helpers/install.sh-c-m644./src/support/apxs.8/usr/local/apache/man/man8/apxs.8
<=== [support]
===> [include: Installing Apache C header files]
cp ./src/include/*.h /usr/local/apache/include/
cp ./src/os/vmesa/os.h /usr/local/apache/include/
cp ./src/os/vmesa/os-inline.c /usr/local/apache/include/
chmod 644 /usr/local/apache/include/*.h
<=== [include]
===> [data: Installing initial data files]
Copying tree ./htdocs/ -> /usr/local/apache/htdocs/
tar: blocksize = 1
./src/helpers/install.sh-c-m644./conf/printenv[*] /usr/local/apache/cgi-bin/printenv
./src/helpers/install.sh-c-m644./conf/test-cgi[*] /usr/local/apache/cgi-bin/test-cgi
Copying tree ./icons/ -> /usr/local/apache/icons/
tar: blocksize = 1
<=== [data]
===> [config: Installing Apache configuration files]
./src/helpers/install.sh -c -m 644 ./conf/httpd.conf-dist[*]
/usr/local/apache/conf/httpd.conf.default
./src/helpers/install.sh -c -m 644 ./conf/httpd.conf-dist[*]
/usr/local/apache/conf/httpd.conf
./src/helpers/install.sh -c -m 644 ./conf/access.conf-dist[*]
/usr/local/apache/conf/access.conf.default
./src/helpers/install.sh -c -m 644 ./conf/access.conf-dist[*]
/usr/local/apache/conf/access.conf
./src/helpers/install.sh -c -m 644 ./conf/srm.conf-dist[*]
/usr/local/apache/conf/srm.conf.default
./src/helpers/install.sh-c-m644./conf/srm.conf-dist[*]/usr/local/apache/conf/srm.conf
./src/helpers/install.sh -c -m 644 ./conf/mime.types
/usr/local/apache/conf/mime.types.default
./src/helpers/install.sh-c-m644./conf/mime.types/usr/local/apache/conf/mime.types
./src/helpers/install.sh-c-m644./conf/magic/usr/local/apache/conf/magic.default
./src/helpers/install.sh-c-m644./conf/magic/usr/local/apache/conf/magic
<=== [config]
make[1]: Leaving directory `/home/porting/apache_1.3.6'
+--------------------------------------------------------+
| You now have successfully built and installed the      |
| Apache 1.3 HTTP server. To verify that Apache actually |
| works correctly you now should first check the         |
| (initially created or preserved) configuration files   |
|                                                        |
|  /usr/local/apache/conf/httpd.conf
|                                                        |
| and then you should be able to immediately fire up     |
| Apache the first time by running:                      |
|                                                        |
|  /usr/local/apache/bin/apachectl start
|                                                        |
| Thanks for using Apache.       The Apache Group        |
|                                http://www.apache.org/  |
+--------------------------------------------------------+
[porting - /home/porting/apache_1.3.6]>
```

F.5.2 HTTPD.CONF

```
##
## httpd.conf -- Apache HTTP server configuration file
##

#
# Based upon the NCSA server configuration files originally by Rob McCool.
#
# This is the main Apache server configuration file.  It contains the
# configuration directives that give the server its instructions.
# See <URL:http://www.apache.org/docs/> for detailed information about
# the directives.
#
# Do NOT simply read the instructions in here without understanding
# what they do.  They're here only as hints or reminders.  If you are unsure
```

```
# consult the online docs. You have been warned.
#
# After this file is processed, the server will look for and process
# /usr/local/apache/conf/srm.conf and then /usr/local/apache/conf/access.conf
# unless you have overridden these with ResourceConfig and/or
# AccessConfig directives here.
#
# The configuration directives are grouped into three basic sections:
#  1. Directives that control the operation of the Apache server process as a
#     whole (the 'global environment').
#  2. Directives that define the parameters of the 'main' or 'default' server,
#     which responds to requests that aren't handled by a virtual host.
#     These directives also provide default values for the settings
#     of all virtual hosts.
#  3. Settings for virtual hosts, which allow Web requests to be sent to
#     different IP addresses or hostnames and have them handled by the
#     same Apache server process.
#
# Configuration and logfile names: If the filenames you specify for many
# of the server's control files begin with "/" (or "drive:/" for Win32), the
# server will use that explicit path.  If the filenames do *not* begin
# with "/", the value of ServerRoot is prepended -- so "logs/foo.log"
# with ServerRoot set to "/usr/local/apache" will be interpreted by the
# server as "/usr/local/apache/logs/foo.log".
#

### Section 1: Global Environment
#
# The directives in this section affect the overall operation of Apache,
# such as the number of concurrent requests it can handle or where it
# can find its configuration files.
#

#
# ServerType is either inetd, or standalone.  Inetd mode is only supported on
# Unix platforms.
#
ServerType inetd

#
# ServerRoot: The top of the directory tree under which the server's
# configuration, error, and log files are kept.
#
# NOTE!  If you intend to place this on an NFS (or otherwise network)
# mounted filesystem then please read the LockFile documentation
# (available at <URL:http://www.apache.org/docs/mod/core.html#lockfile>);
# you will save yourself a lot of trouble.
#
# Do NOT add a slash at the end of the directory path.
#
ServerRoot "/usr/local/apache"

#
# The LockFile directive sets the path to the lockfile used when Apache
# is compiled with either USE_FCNTL_SERIALIZED_ACCEPT or
# USE_FLOCK_SERIALIZED_ACCEPT. This directive should normally be left at
# its default value. The main reason for changing it is if the logs
# directory is NFS mounted, since the lockfile MUST BE STORED ON A LOCAL
# DISK. The PID of the main server process is automatically appended to
# the filename.
#
#LockFile /usr/local/apache/logs/httpd.lock

#
# PidFile: The file in which the server should record its process
# identification number when it starts.
#
PidFile /usr/local/apache/logs/httpd.pid

#
# ScoreBoardFile: File used to store internal server process information.
# Not all architectures require this.  But if yours does (you'll know because
# this file will be  created when you run Apache) then you *must* ensure that
# no two invocations of Apache share the same scoreboard file.
#
ScoreBoardFile /usr/local/apache/logs/httpd.scoreboard

#
```

```
# In the standard configuration, the server will process this file,
# srm.conf, and access.conf in that order.  The latter two files are
# now distributed empty, as it is recommended that all directives
# be kept in a single file for simplicity.  The commented-out values
# below are the built-in defaults.  You can have the server ignore
# these files altogether by using "/dev/null" (for Unix) or
# "nul" (for Win32) for the arguments to the directives.
#
#ResourceConfig conf/srm.conf
#AccessConfig conf/access.conf

#
# Timeout: The number of seconds before receives and sends time out.
#
Timeout 300

#
# KeepAlive: Whether or not to allow persistent connections (more than
# one request per connection). Set to "Off" to deactivate.
#
KeepAlive On

#
# MaxKeepAliveRequests: The maximum number of requests to allow
# during a persistent connection. Set to 0 to allow an unlimited amount.
# We recommend you leave this number high, for maximum performance.
#
MaxKeepAliveRequests 100

#
# KeepAliveTimeout: Number of seconds to wait for the next request from the
# same client on the same connection.
#
KeepAliveTimeout 15

#
# Server-pool size regulation.  Rather than making you guess how many
# server processes you need, Apache dynamically adapts to the load it
# sees --- that is, it tries to maintain enough server processes to
# handle the current load, plus a few spare servers to handle transient
# load spikes (e.g., multiple simultaneous requests from a single
# Netscape browser).
#
# It does this by periodically checking how many servers are waiting
# for a request.  If there are fewer than MinSpareServers, it creates
# a new spare.  If there are more than MaxSpareServers, some of the
# spares die off.  The default values are probably OK for most sites.
#
MinSpareServers 5
MaxSpareServers 10

#
# Number of servers to start initially --- should be a reasonable ballpark
# figure.
#
StartServers 5

#
# Limit on total number of servers running, i.e., limit on the number
# of clients who can simultaneously connect --- if this limit is ever
# reached, clients will be LOCKED OUT, so it should NOT BE SET TOO LOW.
# It is intended mainly as a brake to keep a runaway server from taking
# the system with it as it spirals down...
#
MaxClients 150

#
# MaxRequestsPerChild: the number of requests each child process is
# allowed to process before the child dies.  The child will exit so
# as to avoid problems after prolonged use when Apache (and maybe the
# libraries it uses) leak memory or other resources.  On most systems, this
# isn't really needed, but a few (such as Solaris) do have notable leaks
# in the libraries.
#
MaxRequestsPerChild 30

#
# Listen: Allows you to bind Apache to specific IP addresses and/or
```

```
# ports, in addition to the default. See also the <VirtualHost>
# directive.
#
#Listen 3000
#Listen 12.34.56.78:80

#
# BindAddress: You can support virtual hosts with this option. This directive
# is used to tell the server which IP address to listen to. It can either
# contain "*", an IP address, or a fully qualified Internet domain name.
# See also the <VirtualHost> and Listen directives.
#
#BindAddress *

#
# Dynamic Shared Object (DSO) Support
#
# To be able to use the functionality of a module which was built as a DSO you
# have to place corresponding `LoadModule' lines at this location so the
# directives contained in it are actually available _before_ they are used.
# Please read the file README.DSO in the Apache 1.3 distribution for more
# details about the DSO mechanism and run `httpd -l' for the list of already
# built-in (statically linked and thus always available) modules in your httpd
# binary.
#
# Note: The order is which modules are loaded is important.  Don't change
# the order below without expert advice.
#
# Example:
# LoadModule foo_module libexec/mod_foo.so

#
# ExtendedStatus controls whether Apache will generate "full" status
# information (ExtendedStatus On) or just basic information (ExtendedStatus
# Off) when the "server-status" handler is called. The default is Off.
#
#ExtendedStatus On

### Section 2: 'Main' server configuration
#
# The directives in this section set up the values used by the 'main'
# server, which responds to any requests that aren't handled by a
# <VirtualHost> definition.  These values also provide defaults for
# any <VirtualHost> containers you may define later in the file.
#
# All of these directives may appear inside <VirtualHost> containers,
# in which case these default settings will be overridden for the
# virtual host being defined.
#

#
# If your ServerType directive (set earlier in the 'Global Environment'
# section) is set to "inetd", the next few directives don't have any
# effect since their settings are defined by the inetd configuration.
# Skip ahead to the ServerAdmin directive.
#

#
# Port: The port to which the standalone server listens. For
# ports < 1023, you will need httpd to be run as root initially.
#
Port 8080

#
# If you wish httpd to run as a different user or group, you must run
# httpd as root initially and it will switch.
#
# User/Group: The name (or #number) of the user/group to run httpd as.
#  . On SCO (ODT 3) use "User nouser" and "Group nogroup".
#  . On HPUX you may not be able to use shared memory as nobody, and the
#    suggested workaround is to create a user www and use that user.
#  NOTE that some kernels refuse to setgid(Group) or semctl(IPC_SET)
#  when the value of (unsigned)Group is above 60000;
#  don't use Group  on these systems!
#
User apache
Group httpd
```

```
#
# ServerAdmin: Your address, where problems with the server should be
# e-mailed.  This address appears on some server-generated pages, such
# as error documents.
#
ServerAdmin neale@TOTVM1.itso.ibm.com

#
# ServerName allows you to set a host name which is sent back to clients for
# your server if it's different than the one the program would get (i.e., use
# "www" instead of the host's real name).
#
# Note: You cannot just invent host names and hope they work. The name you
# define here must be a valid DNS name for your host. If you don't understand
# this, ask your network administrator.
# If your host doesn't have a registered DNS name, enter its IP address here.
# You will have to access it by its address (e.g., http://123.45.67.89/)
# anyway, and this will make redirections work in a sensible way.
#
#ServerName TOTVM1

#
# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.
#
DocumentRoot "/usr/local/apache/htdocs"

#
# Each directory to which Apache has access, can be configured with respect
# to which services and features are allowed and/or disabled in that
# directory (and its subdirectories).
#
# First, we configure the "default" to be a very restrictive set of
# permissions.
#
<Directory />
    Options FollowSymLinks
    AllowOverride None
</Directory>

#
# Note that from this point forward you must specifically allow
# particular features to be enabled - so if something's not working as
# you might expect, make sure that you have specifically enabled it
# below.
#

#
# This should be changed to whatever you set DocumentRoot to.
#
<Directory "/usr/local/apache/htdocs">

#
# This may also be "None", "All", or any combination of "Indexes",
# "Includes", "FollowSymLinks", "ExecCGI", or "MultiViews".
#
# Note that "MultiViews" must be named *explicitly* --- "Options All"
# doesn't give it to you.
#
    Options Indexes FollowSymLinks

#
# This controls which options the .htaccess files in directories can
# override. Can also be "All", or any combination of "Options", "FileInfo",
# "AuthConfig", and "Limit"
#
    AllowOverride None

#
# Controls who can get stuff from this server.
#
    Order allow,deny
    Allow from all
</Directory>

#
# UserDir: The name of the directory which is appended onto a user's home
```

```
# directory if a ~user request is received.
#
UserDir public_html
UserDir enable neale
#
# Control access to UserDir directories.  The following is an example
# for a site where these directories are restricted to read-only.
#
#<Directory /*/public_html>
#    AllowOverride FileInfo AuthConfig Limit
#    Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
#    <Limit GET POST OPTIONS PROPFIND>
#        Order allow,deny
#        Allow from all
#    </Limit>
#    <Limit PUT DELETE PATCH PROPPATCH MKCOL COPY MOVE LOCK UNLOCK>
#        Order deny,allow
#        Deny from all
#    </Limit>
#</Directory>

#
# DirectoryIndex: Name of the file or files to use as a pre-written HTML
# directory index.  Separate multiple entries with spaces.
#
DirectoryIndex index.html

#
# AccessFileName: The name of the file to look for in each directory
# for access control information.
#
AccessFileName .htaccess

#
# The following lines prevent .htaccess files from being viewed by
# Web clients.  Since .htaccess files often contain authorization
# information, access is disallowed for security reasons.  Comment
# these lines out if you want Web visitors to see the contents of
# .htaccess files.  If you change the AccessFileName directive above,
# be sure to make the corresponding changes here.
#
<Files .htaccess>
    Order allow,deny
    Deny from all
</Files>

#
# CacheNegotiatedDocs: By default, Apache sends "Pragma: no-cache" with each
# document that was negotiated on the basis of content. This asks proxy
# servers not to cache the document. Uncommenting the following line disables
# this behavior, and proxies will be allowed to cache the documents.
#
#CacheNegotiatedDocs

#
# UseCanonicalName:  (new for 1.3)  With this setting turned on, whenever
# Apache needs to construct a self-referencing URL (a URL that refers back
# to the server the response is coming from) it will use ServerName and
# Port to form a "canonical" name.  With this setting off, Apache will
# use the hostname:port that the client supplied, when possible.  This
# also affects SERVER_NAME and SERVER_PORT in CGI scripts.
#
UseCanonicalName On

#
# TypesConfig describes where the mime.types file (or equivalent) is
# to be found.
#
TypesConfig /usr/local/apache/conf/mime.types

#
# DefaultType is the default MIME type the server will use for a document
# if it cannot otherwise determine one, such as from filename extensions.
# If your server contains mostly text or HTML documents, "text/plain" is
# a good value.  If most of your content is binary, such as applications
# or images, you may want to use "application/octet-stream" instead to
# keep browsers from trying to display binary files as though they are
# text.
```

```
#
DefaultType text/plain

#
# The mod_mime_magic module allows the server to use various hints from the
# contents of the file itself to determine its type.  The MIMEMagicFile
# directive tells the module where the hint definitions are located.
# mod_mime_magic is not part of the default server (you have to add
# it yourself with a LoadModule [see the DSO paragraph in the 'Global
# Environment' section], or recompile the server and include mod_mime_magic
# as part of the configuration), so it's enclosed in an <IfModule> container.
# This means that the MIMEMagicFile directive will only be processed if the
# module is part of the server.
#
<IfModule mod_mime_magic.c>
    MIMEMagicFile /usr/local/apache/conf/magic
</IfModule>

#
# HostnameLookups: Log the names of clients or just their IP addresses
# e.g., www.apache.org (on) or 204.62.129.132 (off).
# The default is off because it'd be overall better for the net if people
# had to knowingly turn this feature on, since enabling it means that
# each client request will result in AT LEAST one lookup request to the
# nameserver.
#
HostnameLookups Off

#
# ErrorLog: The location of the error log file.
# If you do not specify an ErrorLog directive within a <VirtualHost>
# container, error messages relating to that virtual host will be
# logged here.  If you *do* define an error logfile for a <VirtualHost>
# container, that host's errors will be logged there and not here.
#
ErrorLog /usr/local/apache/logs/error_log

#
# LogLevel: Control the number of messages logged to the error_log.
# Possible values include: debug, info, notice, warn, error, crit,
# alert, emerg.
#
LogLevel warn

#
# The following directives define some format nicknames for use with
# a CustomLog directive (see below).
#
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

#
# The location and format of the access logfile (Common Logfile Format).
# If you do not define any access logfiles within a <VirtualHost>
# container, they will be logged here.  Contrariwise, if you *do*
# define per-<VirtualHost> access logfiles, transactions will be
# logged therein and *not* in this file.
#
CustomLog /usr/local/apache/logs/access_log common

#
# If you would like to have agent and referer logfiles, uncomment the
# following directives.
#
#CustomLog /usr/local/apache/logs/referer_log referer
#CustomLog /usr/local/apache/logs/agent_log agent

#
# If you prefer a single logfile with access, agent, and referer information
# (Combined Logfile Format) you can use the following directive.
#
#CustomLog /usr/local/apache/logs/access_log combined

#
# Optionally add a line containing the server version and virtual host
# name to server-generated pages (error documents, FTP directory listings,
```

```
# mod_status and mod_info output etc., but not CGI generated documents).
# Set to "EMail" to also include a mailto: link to the ServerAdmin.
# Set to one of:  On | Off | EMail
#
ServerSignature On

#
# Aliases: Add here as many aliases as you need (with no limit). The format is
# Alias fakename realname
#
# Note that if you include a trailing / on fakename then the server will
# require it to be present in the URL.  So "/icons" isn't aliased in this
# example, only "/icons/"..
#
Alias /icons/ "/usr/local/apache/icons/"

<Directory "/usr/local/apache/icons">
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>

#
# ScriptAlias: This controls which directories contain server scripts.
# ScriptAliases are essentially the same as Aliases, except that
# documents in the realname directory are treated as applications and
# run by the server when requested rather than as documents sent to the client.
# The same rules about trailing "/" apply to ScriptAlias directives as to
# Alias.
#
ScriptAlias /cgi-bin/ "/usr/local/apache/cgi-bin/"

#
# "/usr/local/apache/cgi-bin" should be changed to whatever your ScriptAliased
# CGI directory exists, if you have that configured.
#
<Directory "/usr/local/apache/cgi-bin">
    AllowOverride None
    Options All
    Order allow,deny
    Allow from all
</Directory>

#
# Redirect allows you to tell clients about documents which used to exist in
# your server's namespace, but do not anymore. This allows you to tell the
# clients where to look for the relocated document.
# Format: Redirect old-URI new-URL
#

#
# Directives controlling the display of server-generated directory listings.
#

#
# FancyIndexing is whether you want fancy directory indexing or standard
#
IndexOptions FancyIndexing

#
# AddIcon* directives tell the server which icon to show for different
# files or filename extensions.  These are only displayed for
# FancyIndexed directories.
#
AddIconByEncoding (CMP,/icons/compressed.gif) x-compress x-gzip

AddIconByType (TXT,/icons/text.gif) text/*
AddIconByType (IMG,/icons/image2.gif) image/*
AddIconByType (SND,/icons/sound2.gif) audio/*
AddIconByType (VID,/icons/movie.gif) video/*

AddIcon /icons/binary.gif .bin .exe
AddIcon /icons/binhex.gif .hqx
AddIcon /icons/tar.gif .tar
AddIcon /icons/world2.gif .wrl .wrl.gz .vrml .vrm .iv
AddIcon /icons/compressed.gif .Z .z .tgz .gz .zip
AddIcon /icons/a.gif .ps .ai .eps
```

```
AddIcon /icons/layout.gif .html .shtml .htm .pdf
AddIcon /icons/text.gif .txt
AddIcon /icons/c.gif .c
AddIcon /icons/p.gif .pl .py
AddIcon /icons/f.gif .for
AddIcon /icons/dvi.gif .dvi
AddIcon /icons/uuencoded.gif .uu
AddIcon /icons/script.gif .conf .sh .shar .csh .ksh .tcl
AddIcon /icons/tex.gif .tex
AddIcon /icons/bomb.gif core

AddIcon /icons/back.gif ..
AddIcon /icons/hand.right.gif README
AddIcon /icons/folder.gif ^^DIRECTORY^^
AddIcon /icons/blank.gif ^^BLANKICON^^

#
# DefaultIcon is which icon to show for files which do not have an icon
# explicitly set.
#
DefaultIcon /icons/unknown.gif

#
# AddDescription allows you to place a short description after a file in
# server-generated indexes.  These are only displayed for FancyIndexed
# directories.
# Format: AddDescription "description" filename
#
#AddDescription "GZIP compressed document" .gz
#AddDescription "tar archive" .tar
#AddDescription "GZIP compressed tar archive" .tgz

#
# ReadmeName is the name of the README file the server will look for by
# default, and append to directory listings.
#
# HeaderName is the name of a file which should be prepended to
# directory indexes.
#
# The server will first look for name.html and include it if found.
# If name.html doesn't exist, the server will then look for name.txt
# and include it as plaintext if found.
#
ReadmeName README
HeaderName HEADER

#
# IndexIgnore is a set of filenames which directory indexing should ignore
# and not include in the listing.  Shell-style wildcarding is permitted.
#
IndexIgnore .??* *~ *# HEADER* README* RCS CVS *,v *,t

#
# AddEncoding allows you to have certain browsers (Mosaic/X 2.1+) uncompress
# information on the fly. Note: Not all browsers support this.
# Despite the name similarity, the following Add* directives have nothing
# to do with the FancyIndexing customization directives above.
#
AddEncoding x-compress Z
AddEncoding x-gzip gz

#
# AddLanguage allows you to specify the language of a document. You can
# then use content negotiation to give a browser a file in a language
# it can understand.  Note that the suffix does not have to be the same
# as the language keyword --- those with documents in Polish (whose
# net-standard language code is pl) may wish to use "AddLanguage pl .po"
# to avoid the ambiguity with the common suffix for perl scripts.
#
AddLanguage en .en
AddLanguage fr .fr
AddLanguage de .de
AddLanguage da .da
AddLanguage el .el
AddLanguage it .it

#
# LanguagePriority allows you to give precedence to some languages
```

```
# in case of a tie during content negotiation.
# Just list the languages in decreasing order of preference.
#
LanguagePriority en fr de

#
# AddType allows you to tweak mime.types without actually editing it, or to
# make certain files to be certain types.
#
# For example, the PHP3 module (not part of the Apache distribution - see
# http://www.php.net) will typically use:
#
#AddType application/x-httpd-php3 .php3
#AddType application/x-httpd-php3-source .phps

#
# AddHandler allows you to map certain file extensions to "handlers",
# actions unrelated to filetype. These can be either built into the server
# or added with the Action command (see below)
#
# If you want to use server side includes, or CGI outside
# ScriptAliased directories, uncomment the following lines.
#
# To use CGI scripts:
#
#AddHandler cgi-script .cgi

#
# To use server-parsed HTML files
#
#AddType text/html .shtml
#AddHandler server-parsed .shtml

#
# Uncomment the following line to enable Apache's send-asis HTTP file
# feature
#
#AddHandler send-as-is asis

#
# If you wish to use server-parsed imagemap files, use
#
#AddHandler imap-file map

#
# To enable type maps, you might want to use
#
#AddHandler type-map var

#
# Action lets you define media types that will execute a script whenever
# a matching file is called. This eliminates the need for repeated URL
# pathnames for oft-used CGI file processors.
# Format: Action media/type /cgi-script/location
# Format: Action handler-name /cgi-script/location
#

#
# MetaDir: specifies the name of the directory in which Apache can find
# meta information files. These files contain additional HTTP headers
# to include when sending the document
#
#MetaDir .web

#
# MetaSuffix: specifies the file name suffix for the file containing the
# meta information.
#
#MetaSuffix .meta

#
# Customizable error response (Apache style)
#  these come in three flavors
#
#    1) plain text
#ErrorDocument 500 "The server made a boo boo.
#  n.b.  the (") marks it as text, it does not get output
#
```

```
#    2) local redirects
#ErrorDocument 404 /missing.html
#  to redirect to local URL /missing.html
#ErrorDocument 404 /cgi-bin/missing_handler.pl
#  N.B.: You can redirect to a script or a document using server-side-includes.
#
#    3) external redirects
#ErrorDocument 402 http://some.other_server.com/subscription_info.html
#  N.B.: Many of the environment variables associated with the original
#  request will *not* be available to such a script.

#
# The following directives modify normal HTTP response behavior.
# The first directive disables keepalive for Netscape 2.x and browsers that
# spoof it. There are known problems with these browser implementations.
# The second directive is for Microsoft Internet Explorer 4.0b2
# which has a broken HTTP/1.1 implementation and does not properly
# support keepalive when it is used on 301 or 302 (redirect) responses.
#
BrowserMatch "Mozilla/2" nokeepalive
BrowserMatch "MSIE 4\.0b2;" nokeepalive downgrade-1.0 force-response-1.0

#
# The following directive disables HTTP/1.1 responses to browsers which
# are in violation of the HTTP/1.0 spec by not being able to grok a
# basic 1.1 response.
#
BrowserMatch "RealPlayer 4\.0" force-response-1.0
BrowserMatch "Java/1\.0" force-response-1.0
BrowserMatch "JDK/1\.0" force-response-1.0

#
# Allow server status reports, with the URL of http://servername/server-status
# Change the ".your_domain.com" to match your domain to enable.
#
#<Location /server-status>
#    SetHandler server-status
#    Order deny,allow
#    Deny from all
#    Allow from .your_domain.com
#</Location>

#
# Allow remote server configuration reports, with the URL of
#  http://servername/server-info (requires that mod_info.c be loaded).
# Change the ".your_domain.com" to match your domain to enable.
#
#<Location /server-info>
#    SetHandler server-info
#    Order deny,allow
#    Deny from all
#    Allow from .your_domain.com
#</Location>

#
# There have been reports of people trying to abuse an old bug from pre-1.1
# days.  This bug involved a CGI script distributed as a part of Apache.
# By uncommenting these lines you can redirect these attacks to a logging
# script on phf.apache.org.  Or, you can record them yourself, using the script
# support/phf_abuse_log.cgi.
#
#<Location /cgi-bin/phf*>
#    Deny from all
#    ErrorDocument 403 http://phf.apache.org/phf_abuse_log.cgi
#</Location>

#
# Proxy Server directives. Uncomment the following lines to
# enable the proxy server:
#
#<IfModule mod_proxy.c>
#ProxyRequests On
#
#<Directory proxy:*>
#    Order deny,allow
#    Deny from all
#    Allow from .your_domain.com
#</Directory>
```

```
#
# Enable/disable the handling of HTTP/1.1 "Via:" headers.
# ("Full" adds the server version; "Block" removes all outgoing Via: headers)
# Set to one of: Off | On | Full | Block
#
#ProxyVia On

#
# To enable the cache as well, edit and uncomment the following lines:
# (no cacheing without CacheRoot)
#
#CacheRoot "/usr/local/apache/proxy"
#CacheSize 5
#CacheGcInterval 4
#CacheMaxExpire 24
#CacheLastModifiedFactor 0.1
#CacheDefaultExpire 1
#NoCache a_domain.com another_domain.edu joes.garage_sale.com

#</IfModule>
# End of proxy directives.

### Section 3: Virtual Hosts
#
# VirtualHost: If you want to maintain multiple domains/hostnames on your
# machine you can setup VirtualHost containers for them.
# Please see the documentation at <URL:http://www.apache.org/docs/vhosts/>
# for further details before you try to setup virtual hosts.
# You may use the command line option '-S' to verify your virtual host
# configuration.

#
# If you want to use name-based virtual hosts you need to define at
# least one IP address (and port number) for them.
#
#NameVirtualHost 12.34.56.78:80
#NameVirtualHost 12.34.56.78

#
# VirtualHost example:
# Almost any Apache directive may go into a VirtualHost container.
#
#<VirtualHost ip.address.of.host.some_domain.com>
#    ServerAdmin webmaster@host.some_domain.com
#    DocumentRoot /www/docs/host.some_domain.com
#    ServerName host.some_domain.com
#    ErrorLog logs/host.some_domain.com-error_log
#    CustomLog logs/host.some_domain.com-access_log common
#</VirtualHost>

#<VirtualHost _default_:*>
#</VirtualHost>
```

F.6 SAMBA installation log and configuration file

The Samba installation log and main configuration file are provided in detail (see 6.1, “Samba” on page 111).

F.6.1 Samba installation log

```
make install
Using CFLAGS = -DSYSLOG -W c,hwopts\(string\),langlvl\(ansi\),expmac
-DSMBLOGFILE="/usr/local/samba/var/log.smb" -DNMBLOGFILE="/usr/
local/samba/var/log.nmb" -DCONFIGFILE="/usr/local/samba/lib/smb.conf"
-DLMHOSTSFILE="/usr/local/samba/lib/lmhosts" -DWEB_ROOT="/usr/
local/samba" -DLOCKDIR="/usr/local/
Using LIBS =  -l//vmmtlib -l//posxsock -lxpg4
Installing smbd as /usr/local/samba/bin/smbd
Installing nmbd as /usr/local/samba/bin/nmbd
======================================================================
The binaries are installed. You may restore the old binaries (if there
were any) using the command "make revert". You may uninstall the binaries
using the command "make uninstallbin" or "make uninstall" to uninstall
binaries, man pages and shell scripts.
======================================================================
Installing smbclient as /usr/local/samba/bin/smbclient
Installing testparm as /usr/local/samba/bin/testparm
Installing testprns as /usr/local/samba/bin/testprns
Installing smbrun as /usr/local/samba/bin/smbrun
Installing smbstatus as /usr/local/samba/bin/smbstatus
Installing smbpasswd as /usr/local/samba/bin/smbpasswd
Installing make_smbcodepage as /usr/local/samba/bin/make_smbcodepage
Installing nmblookup as /usr/local/samba/bin/nmblookup
Installing make_printerdef as /usr/local/samba/bin/make_printerdef
======================================================================
The binaries are installed. You may restore the old binaries (if there
were any) using the command "make revert". You may uninstall the binaries
using the command "make uninstallbin" or "make uninstall" to uninstall
binaries, man pages and shell scripts.
======================================================================
Installing man pages in /usr/local/man
======================================================================
The man pages have been installed. You may uninstall them using the command
the command "make uninstallman" or make "uninstall" to uninstall binaries,
man pages and shell scripts.
======================================================================
Installing scripts in /usr/local/samba/bin
Installing /usr/local/samba/bin/smbtar
Setting permissions on /usr/local/samba/bin/smbtar
Installing /usr/local/samba/bin/addtosmbpass
Setting permissions on /usr/local/samba/bin/addtosmbpass
======================================================================
The scripts have been installed. You may uninstall them using
the command "make uninstallscripts" or "make install" to install binaries,
man pages and shell scripts. You may recover the previous version (if any
by "make revert".
======================================================================
Installing codepage files in /usr/local/samba/lib/codepages
Creating codepage file /usr/local/samba/lib/codepages/codepage.437 from codepage_def.437
Creating codepage file /usr/local/samba/lib/codepages/codepage.737 from codepage_def.737
Creating codepage file /usr/local/samba/lib/codepages/codepage.850 from codepage_def.850
Creating codepage file /usr/local/samba/lib/codepages/codepage.852 from codepage_def.852
Creating codepage file /usr/local/samba/lib/codepages/codepage.861 from codepage_def.861
Creating codepage file /usr/local/samba/lib/codepages/codepage.932 from codepage_def.932
Creating codepage file /usr/local/samba/lib/codepages/codepage.866 from codepage_def.866
Creating codepage file /usr/local/samba/lib/codepages/codepage.949 from codepage_def.949
Creating codepage file /usr/local/samba/lib/codepages/codepage.950 from codepage_def.950
Creating codepage file /usr/local/samba/lib/codepages/codepage.936 from codepage_def.936
======================================================================
The code pages have been installed. You may uninstall them using the
command "make uninstallcp" or make "uninstall" to uninstall binaries,
man pages, shell scripts and code pages.
======================================================================
```

F.6.2 Samba configuration file

```
# This is the main Samba configuration file. You should read the
# smb.conf(5) manual page in order to understand the options listed
```

```
# here. Samba has a huge number of configurable options (perhaps too
# many!) most of which are not shown in this example
#
# Any line which starts with a ; (semi-colon) or a # (hash)
# is a comment and is ignored. In this example we will use a #
# for commentry and a ; for parts of the config file that you
# may wish to enable
#
# NOTE: Whenever you modify this file you should run the command "testparm"
# to check that you have not many any basic syntactic errors.
#
#======================= Global Settings =====================================
[global]

# workgroup = NT-Domain-Name or Workgroup-Name, eg: REDHAT4
   workgroup = TOTVM1
   remote announce = 9.12.14.194/SCNF

# server string is the equivalent of the NT Description field
   server string = Samba Server for VM/ESA

# This option is important for security. It allows you to restrict
# connections to machines which are on your local network. The
# following example restricts access to two C class networks and
# the "loopback" interface. For more examples of the syntax see
# the smb.conf man page
;  hosts allow =

# If you want to automatically load your printer list rather
# than setting them up individually then you'll need this
   load printers = yes

# you may wish to override the location of the printcap file
   printcap name = /etc/printcap

# It should not be necessary to specify the print system type unless
# it is non-standard. Currently supported print systems include:
# bsd, sysv, plp, lprng, aix, hpux, qnx
   printing = aix

# Uncomment this if you want a guest account, you must add this to /etc/passwd
# otherwise the user "nobody" is used
  guest account = nobody

# this tells Samba to use a separate log file for each machine
# that connects
;  log file = /usr/local/samba/var/log.%m
   log file = /tmp/log.%m

# Put a capping on the size of the log files (in Kb).
   max log size = 0

# Security mode. Most people will want user level security. See
# security_level.txt for details.
   security = share
# Use password server option only with security = server
;  password server = 9.12.14.194

# You may wish to use password encryption. Please read
# ENCRYPTION.txt, Win95.txt and WinNT.txt in the Samba documentation.
# Do not enable this option unless you have read those documents
;  encrypt passwords = yes

# Using the following line enables you to customise your configuration
# on a per machine basis. The %m gets replaced with the netbios name
# of the machine that is connecting
;   include = /usr/local/samba/lib/smb.conf.%m

# Most people will find that this option gives better performance.
# See speed.txt and the manual pages for details
  socket options = TCP_NODELAY=1
# Configure Samba to use multiple interfaces
# If you have multiple network interfaces then you must list them
# here. See the man page for details.
;  interfaces =

# Browser Control Options:
# set local master to no if you don't want Samba to become a master
```

```
# browser on your network. Otherwise the normal election rules apply
   local master = yes

# OS Level determines the precedence of this server in master browser
# elections. The default value should be reasonable
   os level = 33

# Domain Master specifies Samba to be the Domain Master Browser. This
# allows Samba to collate browse lists between subnets. Don't use this
# if you already have a Windows NT domain controller doing this job
   domain master = yes

# Preferred Master causes Samba to force a local browser election on startup
# and gives it a slightly higher chance of winning the election
   preferred master = no

# Use only if you have an NT server on your network that has been
# configured at install time to be a primary domain controller.
; domain controller = sctd100f

# Enable this if you want Samba to be a domain logon server for
# Windows95 workstations.
   domain logons = yes

# if you enable domain logons then you may want a per-machine or
# per user logon script
# run a specific logon batch file per workstation (machine)
;   logon script = %m.bat
# run a specific logon batch file per username
;   logon script = %U.bat

# Where to store roving profiles (only for Win95 and WinNT)
#        %L substitutes for this servers netbios name, %U is username
#        You must uncomment the [Profiles] share below
   logon path = \\%L\Profiles\%U

# Windows Internet Name Serving Support Section:
# WINS Support - Tells the NMBD component of Samba to enable it's WINS Server
   wins support = yes

# WINS Server - Tells the NMBD components of Samba to be a WINS Client
#Note: Samba can be either a WINS Server, or a WINS Client, but NOT both
;  wins server = 10.1.4.201

# WINS Proxy - Tells Samba to answer name resolution queries on
# behalf of a non WINS capable client, for this to work there must be
# at least oneWINS Server on the network. The default is NO.
;   wins proxy = yes

# DNS Proxy - tells Samba whether or not to try to resolve NetBIOS names
# via DNS nslookups. The built-in default for versions 1.9.17 is yes,
# this has been changed in version 1.9.18 to no.
;  dns proxy = yes

text conv = filetype
translate map = *.txt TEXT

#============================ Share Definitions ==============================
[homes]
   comment = Home Directories
   browseable = no
   writable = yes
   path = /home/%u
   read only = no
   create mode = 0700

# Un-comment the following and create the netlogon directory for Domain Logons
[netlogon]
   comment = Network Logon Service
   path = /usr/local/samba/lib/netlogon
   guest ok = yes
   writable = no
   share modes = no

# Un-comment the following to provide a specific roving profile share
# the default is to use the user's home directory
[Profiles]
    path = /usr/local/samba/profiles
```

```
    browseable = no
    guest ok = yes


# NOTE: If you have a BSD-style print system there is no need to
# specifically define each individual printer
[printers]
   comment = All Printers
   path = /usr/spool/samba
   browseable = no
# Set public = yes to allow user 'guest account' to print
   guest ok = no
   writable = no
   printable = yes
;  lpq command = true
   print command = lp %2

# This one is useful for people to share files
[tmp]
   comment = Temporary file space
   path = /tmp
   read only = no
   public = yes


; A private printer, usable only by fred. Spool data will be placed in fred's
; home directory. Note that fred must have write access to the spool directory,
; wherever it is.
[poke3130]
   comment = Atlantis Laser Printer
   path = /tmp
   printer = poke3130
   public = yes
   writable = no
   lpq command = lpq poke3130
   print command = lp -d .KOLP,A,P+ASCII %s
   printable = yes

[pokd3130]
   comment = Atlantis Laser Printer
   path = /tmp
   printer = pokd3130
   public = yes
   writable = no
   lpq command = lpq poke3130
   print command = lp -d .LPR,A,STANDARD %s
   printable = yes

# A publicly accessible directory, but read only, except for people in
# the "staff" group
;[public]
;   comment = Public Stuff
;   path = /home/samba
;   public = yes
;   writable = yes
;   printable = no
;   write list = @staff

# Other examples.
#
# A private printer, usable only by fred. Spool data will be placed in fred's
# home directory. Note that fred must have write access to the spool directory,
# wherever it is.
;[fredsprn]
;   comment = Fred's Printer
;   valid users = fred
;   path = /homes/fred
;   printer = freds_printer
;   public = no
;   writable = no
;   printable = yes

# A private directory, usable only by fred. Note that fred requires write
# access to the directory.
;[fredsdir]
;   comment = Fred's Service
;   path = /usr/somewhere/private
;   valid users = fred
```

```
;   public = no
;   writable = yes
;   printable = no

# a service which has a different directory for each machine that connects
# this allows you to tailor configurations to incoming machines. You could
# also use the %U option to tailor it by user name.
# The %m gets replaced with the machine name that is connecting.
[pchome]
  comment = PC Directories
  path = /usr/pc/%m
  public = no
  writable = yes

# A publicly accessible directory, read/write to all users. Note that all files
# created in the directory by users will be owned by the default user, so
# any user with access can delete any other user's files. Obviously this
# directory must be writable by the default user. Another user could of course
# be specified, in which case all files would be owned by that user instead.
[public]
   path = /usr/somewhere/else/public
   public = yes
   only guest = yes
   writable = yes
   printable = no

# The following two entries demonstrate how to share a directory so that two
# users can place files there that will be owned by the specific users. In this
# setup, the directory should be writable by both users and should have the
# sticky bit set on it to prevent abuse. Obviously this could be extended to
# as many users as required.
;[myshare]
;   comment = Mary's and Fred's stuff
;   path = /usr/somewhere/shared
;   valid users = mary fred
;   public = no
;   writable = yes
;   printable = no
;   create mask = 0765
```

F.7 DAYTIMED installation log

The complete installation log of DAYTIMED is reproduced in the section (see 5.4, “DAYTIMED - daytime and time daemon” on page 95).

```
gunzip daytime.tar.gz

gunzip: daytime.tar.gz: decompression OK, trailing garbage ignored
[porting - /home/porting]>
pax -rf daytime.tar
[porting - /home/porting]>
cd DAYTIME
[porting - /home/porting/DAYTIME]>
make install
cp daytimed /usr/bin/daytimed
cp daytime /usr/bin/daytime
```

F.8 CROND linstallation log

This is the installation log of CROND (see 5.5.3, “CROND installation and configuration” on page 99).

```
make install
rm -f /usr/bin/crond /usr/bin/crontab /usr/bin/rplstream
cp crond crontab rplstream /usr/bin
chown root /usr/bin/crond
chown root /usr/bin/crontab
chown root /usr/bin/rplstream
chmod 700 /usr/bin/crond
chmod 4750 /usr/bin/crontab
chmod 4750 /usr/bin/rplstream
mkdirs /usr/local/man/man1
mkdirs: GSU6039 mkdirs: not found
make: [install] Error 127 (ignored)
mkdirs /usr/local/man/man8
mkdirs: GSU6039 mkdirs: not found
make: [install] Error 127 (ignored)
cp crontab.1 /usr/local/man/man1
cp crond.8 /usr/local/man/man8
chmod 444 /usr/local/man/man1/crontab.1
chmod 444 /usr/local/man/man8/crond.8
```

F.9 FLEX installation log

This is the installation log of the Flex package (see 5.6.1.2, “FLEX” on page 101).

```
make install
/bin/sh ./mkinstalldirs \
   /usr/local/bin /usr/local/lib /usr/local/include /usr/local/man/man1
rm -f /usr/local/man/man1/flexdoc.1
./install.sh -c -m 644 ./flex.1 /usr/local/man/man1/flex.1
./install.sh -c flex /usr/local/bin/flex
cd /usr/local/bin && ln -s flex flex++
./install.sh -c -m 644 libfl.a /usr/local/lib/libfl.a
cd /usr/local/lib && : libfl.a
./install.sh -c -m 644 ./FlexLexer.h /usr/local/include/FlexLexer.h
```

Appendix G. Product configuration files

This appendix contains the configuration files used by the program products essential to the operation of the residency.

G.1 RSCS configuration

```
*************************************************************************
*--< Update:  RSCSCHNG UPDT479   On: Tuesday,   12/03/91  16:22:43 >--*
*************************************************************************
*    Warning:  Configuration File statements and their operands must    *
*              be in UPPER case.  This file must NOT contain sequence   *
*              numbers.                                                 *
*                                                                       *
*************************************************************************
 LOCAL         TOTVM1   *
 CHANNELS      F
 DEST          3820D13 3820RF1 3820SLAB 3820E16 3820A21 3820RF2 3820C16
 DEST          3820SG02 3820BTH 3820BTH1 3827B12 3825DAL
 DUMP          CP    MAINT
 HIDECHARACTER \
 IMBED         AUTHS
 IMBED         EXITS
 IMBED         SSH
 IMBED         SFFCONF
*IMBED         SRMVCONF
 LANGUAGE      * =
 MSGNOH
 OPFORM        STANDARD
 OPTION        ENQMSG=NO SENTMSG=YES FINALMSG=YES LISTPROC=YES MAXDSH=10
 OPTION        SECORGID=YES LOOPING=ALL MAXHOPS=12
 RECOVERY      MSG DB Re-IPLing GCS#IPL GCS
 SAFCLASS      *
 TRACEDEST     DB WTSCPOK
*************************************************************************
* Our links.. (Not the sausage kind.)                                   *
*************************************************************************
* The following statement defines the HTA driver.
*
*
 LINKDEFINE WTSCVMXA TYPE NJE LINE 811 CLASS * QUEUE FIFO DP 5
 PARM       WTSCVMXA BUFF=3976 ST=1 TA=0
 LINKDEFINE LPR TYPE LPR FORM * AST
 PARM LPR EXIT=LPRXONE  HOST=9.12.2.3 PORT=515 PR=afccu2
 LINKDEFINE KOLP TYPE LPR FORM * AST
 PARM KOLP EXIT=LPRXONE  HOST=9.12.2.4 PORT=515 PR=afccu2
*
*************************************************************************
RSC00500
*                    LOCAL  ROUTE  SPECIFICATIONS                     * RSC00510
*************************************************************************
RSC00520
* Route everything else that isn't a link to WTSCPOK                   *
*************************************************************************
ROUTE    *         WTSCVMXA
*
```

G.2 TCPIP configuration

```
; ====================================================================
; File Name:   PROFILE STCPIP
; Description: Sample PROFILE TCPIP Configuration File
;
;  This configuration file is used to define system operation
;  parameters, Telnet and network configuration information for your
;  TCP/IP environment.  The definitions within this file are used by
;  the TCPIP virtual machine to establish your TCP/IP services during
;  initialization, although some statements within this file can be
;  changed dynamically by using the OBEYFILE command.
;
;  For detailed information about the use and syntax of configuration
;  statements used within this file, see "Chapter 6. Configuring the
;  TCPIP Virtual Machine" in the "IBM TCP/IP for VM Planning and
;  Customization" manual.
;
;  Note: Comments within this file must begin with a semicolon (;)
;        followed by at least one blank.
;
; ====================================================================

; ====================================================================
; Free Pool Statements
;
; Use the statements below to define the allocation of control blocks
; and data buffers used by the TCP/IP stack (TCPIP) and other TCP/IP
; service virtual machines.  Ensure the Free Pool statements precede
; all other statements within this file.
; ====================================================================

ACBPOOLSIZE                 1000
ADDRESSTRANSLATIONPOOLSIZE  1500
CCBPOOLSIZE                 150
DATABUFFERPOOLSIZE          160  8192
ENVELOPEPOOLSIZE            750
IPROUTEPOOLSIZE             600
LARGEENVELOPEPOOLSIZE       50   8192
RCBPOOLSIZE                 50
SCBPOOLSIZE                 256
SKCBPOOLSIZE                256
SMALLDATABUFFERPOOLSIZE     0
TCBPOOLSIZE                 256
TINYDATABUFFERPOOLSIZE      0
UCBPOOLSIZE                 100

; --------------------------------------------------------------------
; Monitorrecords statement
;
; Use the MONITORRECORDS statement to select the monitor data records
; that are produced by TCPIP, if any.
; --------------------------------------------------------------------
MONITORRECORDS

; --------------------------------------------------------------------
; Tracing Statements
```

```
;
; Define tracing statements below, to log events for diagnostic
; purposes.  Tracing statements defined within this file will be
; enabled during TCPIP server initialization.
;
; Note:  For most situations that require tracing, use of the OBEYFILE
; command to dynamically enable and disable tracing may be more
; advantageous than defining trace statements within this file.
;
; NOTRACE disables all TCP/IP tracing.  For detailed information about
; tracing, see the description of the TRACE statement in the
; "IBM TCP/IP for VM Planning and Customization" manual.
; ---------------------------------------------------------------------
NOTRACE ALL

;  TRACE     MOST
;  LESSTRACE ALL
;  MORETRACE MOST
;  SCREEN
;  NOSCREEN
;  FILE TCPIP PROB0001 A

; ---------------------------------------------------------------------
; Control trace file timestamps.
; ---------------------------------------------------------------------
TIMESTAMP 0

; ---------------------------------------------------------------------
; Define user IDs to receive messages when serious errors occur.
; ---------------------------------------------------------------------
INFORM
  OPERATOR TCPMAINT
ENDINFORM

; =====================================================================
; Various TCPIP initialization, operation and authorization
; parameters.
; =====================================================================
; Assorted Parameters
; ---------------------------------------------------------------------
; ASSORTEDPARMS
;   NOFWD
;   VMDUMP
;   NOUDPQUEUELIMIT
;   RESTRICTLOWPORTS
;   PERMITTEDUSERSONLY
; ENDASSORTEDPARMS

; ---------------------------------------------------------------------
; Define user IDs that are allowed to issue OBEYFILE and other
; restricted commands.
; ---------------------------------------------------------------------
OBEY
  OPERATOR TCPMAINT SNMPD SNMPQE ROUTED REXECD DHCPD NETNAINT NEALE JEAN
ENDOBEY

; ---------------------------------------------------------------------
; Restrict specific user IDs (or a group of similarly named user IDs)
```

```
; from using TCP/IP services.
; ------------------------------------------------------------------------
; RESTRICT
;   VEND*
;   BADGUY
; ENDRESTRICT

; ------------------------------------------------------------------------
; Permit specific user IDs to use TCP/IP services.
; ------------------------------------------------------------------------
; PERMIT
;   NETMAINT
; ENDPERMIT

; ------------------------------------------------------------------------
; Keep-Alive Packet parameters
; ------------------------------------------------------------------------
KEEPALIVEOPTIONS
  INTERVAL     20
  SENDGARBAGE  FALSE
ENDKEEPALIVEOPTIONS

; ------------------------------------------------------------------------
; SNMP-related Statements
;
; SYSCONTACT defines the contact person for this managed node, as well
; as how to contact this person.  This definition supplies the value
; for the SNMP MIB variable, "sysContact".
;
; SYSLOCATION defines the physical location of this node.  This
; definition supplies the value for the SNMP MIB variable,
; "sysLocation".
; ------------------------------------------------------------------------
SYSCONTACT
  Main Operator (555-2150)
  Susquahanna Hat Company
ENDSYSCONTACT

SYSLOCATION
  First Floor Computer Room
ENDSYSLOCATION

; ========================================================================
; Server Virtual Machine-related Statements
; ========================================================================

; ------------------------------------------------------------------------
; Xautolog the server machines identified via the AUTOLOG statement.
; ------------------------------------------------------------------------
AUTOLOG
  FTPSERVE PASSWORD    ; FTP SERVER
  PORTMAP  PASSWORD    ; PORTMAP SERVER
  SMTP     PASSWORD    ; SMTP SERVER
  VMNFS    PASSWORD    ; NFS SERVER
  SAMBA    PASSWORD    ; SAMBA SERVER
  JACORB   PASSWORD    ; WEB SERVER (WEBSHARE)
  APACHE   PASSWORD    ; WEB SERVER (APACHE)
  INETD    PASSWORD    ; INET SUPER SERVER
```

```
  NEWS      PASSWORD    ; INN SERVER
  IRCD      PASSWORD    ; IRC SERVER
  LDAP      PASSWORD    ; LDAP SERVER
ENDAUTOLOG

; ---------------------------------------------------------------------
; Reserve ports for specific server machines.  Port values used are
; those defined in RFC 1060, "Assigned Numbers"
; ---------------------------------------------------------------------
PORT
  20   TCP FTPSERVE  NOAUTOLOG ; FTP SERVER
  21   TCP FTPSERVE            ; FTP SERVER
  23   TCP INTCLIEN            ; TELNET SERVER
  25   TCP SMTP                ; SMTP SERVER
; 53   TCP NAMESRV             ; DOMAIN NAME SERVER
; 53   UDP NAMESRV             ; DOMAIN NAME SERVER
; 67   UDP BOOTPD              ; BOOTP SERVER
; 67   UDP DHCPD               ; DHCP SERVER
; 69   UDP TFTPD               ; TFTPD (TRIVIAL FTP) SERVER
  80   TCP APACHE              ; WEB SERVER (APACHE)
  111  TCP PORTMAP             ; PORTMAP SERVER
  111  UDP PORTMAP             ; PORTMAP SERVER
  119  TCP NEWS                ; INND SERVER
  137  TCP SAMBA               ; SAMBA SERVER
  138  TCP SAMBA               ; SAMBA SERVER
  139  TCP SAMBA               ; SAMBA SERVER
  194  TCP IRCD                ; IRC SERVER
  389  TCP LDAP                ; LDAP SERVER
  514  TCP SYSLOGD             ; SYSLOG SERVER
  2049 UDP VMNFS               ; NFS Server
;
; (End of PORT reservations)

; ====================================================================
; TELNET Server Configuration Statements.
; (The TELNET Server is an "internal client" of the TCPIP server).
; ====================================================================
; INTERNALCLIENTPARMS
;   PORT         23
;   INACTIVE     0
;   TIMEMARK     600
;   CCSTERMNAME  TCPIP
;   CONNECTEXIT  TNEXIT1
; ENDINTERNALCLIENTPARMS

; ====================================================================
; DEVICE and LINK statements
;
; Define the network interfaces used in your environment.
; (Sample device and link statements are included below).
; ====================================================================

; CTC CONNECTION FROM WTSCVMT TO WTSCVMXA
DEVICE VMXACTC  CTC       812
LINK WTSCVMXA CTC   1 VMXACTC

; ====================================================================
; Routing Statements
```

```
;
; Define the routing information required for your environment via the
; HOME, GATEWAY, ARPAGE, BSDROUTINPARMS and PRIMARYINTERFACE statements.
; ====================================================================
;
; --------------------------------------------------------------------
; Flush the ARP tables every nn minutes (a 5 minute interval is used
; below).
; --------------------------------------------------------------------
ARPAGE 5

; --------------------------------------------------------------------
; Primary interface Definition
; --------------------------------------------------------------------
; PRIMARYINTERFACE  ETH1

; --------------------------------------------------------------------
; Define the internet (IP) address(es) for this VM host
; --------------------------------------------------------------------
HOME
; THESE ARE THE CURRENT HOME ADDRESSES THAT ARE BEING USED TODAY

  9.12.13.62   WTSCVMXA

; (End of HOME address information)

; --------------------------------------------------------------------
; Routing Information
; --------------------------------------------------------------------
; Note:
;  * Routes defined via the GATEWAY statement are STATIC routes.
;  * Routes defined via the BSDROUTINGPARMS statement are DYNAMIC
;    routes.
; --------------------------------------------------------------------
; Static Routing Information
; --------------------------------------------------------------------
GATEWAY

; (IP) Network  First        Link      Max. Packet  Subnet      Subnet
; Address       Hop          Name      Size (MTU)   Mask        Value
; -----------   ------------ -------   -----------  ----------- --------
;
; THESE ARE THE CURRENT GATEWAYS THAT ARE BEING USED TODAY IN PRODUCTION
  9.12.13.63    =            WTSCVMXA  1500         HOST
  9             =            WTSCVMXA  1500         0.255.255.0 0.12.14.0

; --------------------------------------------------------------------
; Define The DEFAULT route used for any network not explicitly routed
; via the previous entries.
; --------------------------------------------------------------------
;
; THIS THE CURRENT DEFAULT GATEWAY THAT IS BEING USED TODAY
  DEFAULTNET    =            WTSCVMXA  1500         0

; (End of GATEWAY Static Routing information)


; --------------------------------------------------------------------
```

```
; TRANSLATE statements indicate relationships between internet
; addresses and network addresses.  The TRANSLATE statement is most
; often used only in conjunction with HYPERchannel (HCH) and X.25
; network connections.
; ------------------------------------------------------------------------
TRANSLATE

; (End of Translate information)
;

; ------------------------------------------------------------------------
; Start all network interface devices used in this environment.
; ------------------------------------------------------------------------
; THESE ARE THE CURRENT DEVICES THAT SHOULD BE STARTED
START VMXACTC

; ========================================================================
; End of PROFILE STCPIP
; ========================================================================
```

Appendix H. JacORB installation and configuration

The following sections have been taken from “JacORB Programming Guide, v0.9” by Gerald Brose of the Institute fuer Informatik Freie Universitaet at Berlin, Germany June 26, 1998, by permission of its author. Note this chapter uses a convention of referring to items in the bibliography (see Section H.10.2, “Bibliography” on page 297) using the form [reference number].

H.1 Introduction

This document gives an introduction to programming distributed applications with JacORB, a free Java object request broker. JacORB comes with full source code and a number of example programs. Additionally, both IDL and Java code for all OMG object services defined in [3] is also included in this distribution. This document is not an introduction to CORBA in general. Please see [6, 7] for this purpose.

The IDL-Java language mapping provided by the JacORB IDL compiler is close to the OMG standard in [5]. Most of the differences between these mappings should not be important or even visible for application programmers. A detailed description of our language mapping will be part of future versions of this document which will hopefully be more complete than the present one.

The rest of this document is structured as follows. First, we briefly describe how to obtain and install JacORB. Appendix H.3, “Getting started” on page 269 gives a few examples on how to use JacORB to write distributed Java programs while starting at Appendix H.5, “The JacORB name service” on page 275 contains a description of the utilities that come with JacORB.

H.1.1 Document changes

Only minor changes have been made to this document since version 0.9. For 0.9d, a new paragraph in Appendix H.8, “The interface repository” on page 289 has been added which points out potential problems with complicated class paths for the IR. In Appendix H.7, “Interceptors” on page 280 the IDL definition for GIOP message headers has been added.

H.2 Installing JacORB

In this chapter we explain how to obtain and install JacORB and give an overview of the package contents.

H.2.1 Obtaining and installing JacORB

JacORB can be obtained as a gzipped tar archive or as a zip archive from the JacORB home page at http://www.inf.fu-berlin.de/~brose/jacorb/. It can also be downloaded via anonymous ftp from ftp.inf.fu-berlin.de from the directory pub/jacorb/.

To install JacORB, just gunzip and untar (or unzip) the archive somewhere. This will result in a new directory JacORB 0.9/. Make sure your CLASSPATH environment variable contains the path to the installation directory and,

additionally, the subdirectory classes/. Extend your search path with JacORB/bin, so that the shell scripts and batch files for the utilities in this directory are found.

H.2.2 Preprocessor, Make

Also make sure some C-preprocessor is in your search path and can be used by calling cpp. If your preprocessor is not called cpp you need to edit the file jacorb/bin/idl2j so that your local preprocessor is used.

Another word on non-Unix platforms: JacORB will run on any JavaVM, but working and developing software with JacORB (and recompiling examples) is much easier if a “make” utility (such as GNU's or Miscrosoft's nmake) is present on your system. If you do not have one installed, I recommend downloading one, for example, the UNIX tools from the gnuwin32 project (see www.cygnus.com).

H.2.3 Configuration

To configure a few JacORB options and defaults, have a look at the file jacorb/Orb/Config.java. Here is an example:

```
package jacorb.Orb;
public final class Config
{
        // how often do we retry to connect, and how long do we wait
        // before trying again (in msecs)?
    public static final int retry = 10;
    public static final long retry_intvl = 400;
        // network buffer size
    public static final int OUTBUFSIZE = 4096;
        // the "classpath" that makes up the Interface Repository
    public static final String IR_server =
            "http://www.inf.fu-berlin.de/~brose/IR_location";
        // identifies the default naming context to be used
        // a string in URL format pointing to a file resource
    public static final String default_context =
            "http://www.inf.fu-berlin.de/~brose/NS_Ref";
}
```

Figure 175. Sample JacORB Config.java.

Configurable options include the size of network buffers, the number of retries JacORB makes if a connection cannot be established, and how long it shall wait before retrying. The two string values for IR server and default context are URLs for file resources used to set up the JacORB name server and the server for the Interface Repository. These URLs will be used by the ORB to locate the servers and to provide object references to them.

After modifying this file, please type `make configure` in the JacORB installation directory. This will recompile all files in the distribution which depend on the configuration file.

Before testing the examples in jacorb/demo/ you should also have a look at the (simple) file Makefile.config: simply adjust the paths in this file to reflect your local installation. You can now test your installation by typing make in one of the subdirectories of the jacorb/demo/ directory which contains a number of

examples for using JacORB. Alternatively, you can make all examples in one go if you type make in the jacorb/demo/ directory. If everything compiles, you should be done with the installation.

H.3 Getting started

Before we go about explaining this example in detail, we will have a look at the general process of developing CORBA or distributed Java applications with JacORB. We will follow this road map when working through the example. This example can be found in jacorb/demo/example1 which also contains a Makefile so that the development steps do not have to be carried out manually every time. Still, you should know what is going on.

As this document gives only a short introduction to JacORB programming and does not cover all the details of CORBA IDL, we recommend that you also look at the other examples in the jacorb/demo/ directory. These are organized so as to show how the different aspects of CORBA IDL can be used with JacORB.

H.3.1 JacORB development

The steps we will have to take are:

1. Write an IDL interface specification for objects that are to be accessed as CORBA objects. (If you want to do only Java programming and have no need for IDL, this step can be skipped).
2. Generate a corresponding Java interface with the IDL compiler and compile it with `javac`. If step 1 was omitted, you have to come up with the Java interface yourself.
3. Write a Java class which implements the interface of step 2 and compile it.
4. Generate a stub and a skeleton class from the server interface with the stub generator `jgen` and compile them.
5. Write a “Main” class that instantiates the server implementation and its skeleton (and compile it).
6. Write a client class which instantiates a server stub and binds to it (and compile it).

H.3.2 Step 1: Interface design

This example uses a very simple server. Its interface is given in server.idl.

```
// File: server.idl
module example1
{
    typedef sequence < string > strings;
    interface server {
        string writeMessage(in string a1);
        string writeMessages(in strings a1);
        strings arryfy(in string a1, in long a2);
    };
};
```

Figure 176. Interface to a simple CORBA server.

The interface “server” offers four operations. Operation writeMessage() accepts a message string and returns a result string, whereas writeMessages() takes a sequence of strings as its argument. Operation arrfy1() takes a string and a number as is input arguments and returns a sequence of strings with length *n* in its out-parameter. The last operation, arryfy2() returns an array of strings in its out-parameter.

H.3.3 Step 2: Generating the Java interface

Feeding this file into the idl compiler:

```
$ idl2j -d ../../.. server.idl
```

produces a corresponding Java interface in server.java which can then be compiled with `javac`. Note that the IDL compiler will produce a directory structure corresponding to the module structure in the IDL file, so it would have produced a subdirectory jacorb with a subdirectory demo in the current directory had we not directed it to put this directory structure to ../../.. by using the -d switch with the compiler. As a result, all generated files that would have been in the jacorb/demo/example1 subdirectory are now put into the current directory. Also note that server inherits from org.omg.CORBA.CORBject. This is an empty interface which is used for the sole purpose of telling apart interfaces that were generated by the IDL compiler from those that were not. You will never need to write this line into interfaces developed by yourself.

```
// Automatically generated by idl2java from interface server
package example1;
public interface server
    extends org.omg.CORBA.CORBject
{
    java.lang.String writeMessage(java.lang.String a1);
    java.lang.String writeMessages(java.lang.String[] a1);
    void arryfy1(java.lang.String a1, int a2,
        jacorb.Orb.SequenceOutHolder/*out*/ s);
    void arryfy2(java.lang.String a1,
        jacorb.demo.example1.StringArray5OutHolder/*out*/ s);
}
```

Figure 177. Java code generated by compilation of server interface.

Not that all IDL types have been replaced by their Java equivalents, for example, t stringSeq has been replaced by java.lang.String[].

A client of this server relies only on the operation signatures in server.idl or server.java. What the client actually uses is not an instance of serverImpl, but a stub that is automatically generated by the IDL compiler. It is type-correct to use the stub in place of the server since the stub also implements the interface called server.

H.3.4 Step 3: Implementing the interface

We are now ready to actually provide an implementation of the functionality promised by the interface. The class which implements that interface is called serverImpl.java and is straightforward:

```
package example1;

public class serverImpl
    implements server
{
    public String writeMessage( String s )
    {
        System.out.println("Message: " + s );
        return s + " written";
    }

    public String writeMessages( String[] s )
    {
        for( int i = 0; i < s.length; i++)
            System.out.println("Message: " + s[i] );
        return "string array written";
    }

    public void arryfy1(java.lang.String a1, int a2,
                jacorb.Orb.SequenceOutHolder s)
    {
        String result [] = new String[a2];
        for( int j = 0; j < a2; j++ )
            result[j] = a1;
        s.set_value(result);
    }

    public void arryfy2(java.lang.String a1,
            jacorb.demo.example1.StringArray5OutHolder s)
    {
        int sz = s.size();
        String result [] = new String[sz];
        for( int j = 0; j < sz; j++ )
            result[j] = a1;
        s.set_value(new StringArray5(result));
    }
}
```

Figure 178. Class which implements the interface.

Note that this class has not a single line of ORB-related code in it (save for the use of Holder classes to mimic the IDL out parameter passing mode). JacORB supports only what is known as the TIE-mechanism for binding skeleton code to an application class, so there is no need to inherit from any ORB class. As Java supports only single implementation inheritance, you will more often than not need to inherit from application defined superclasses, so the other approach (BOAImpl in Orbix parlance) will frequently be ruled out. That is why it is not supported in JacORB.

H.3.5 Step 4: Generating stubs and skeletons

A client of this server relies only on the operation signatures in server.idl or server.java. What the client actually uses is not an instance of serverImpl, but a stub that is automatically generated from the Java interface (which in turn was

generated from the IDL interface). It is type-correct to use the stub in place of the server since the stub also implements the interface server.

In order to generate the stub (and the server side skeleton), all you have to do now is type:

```
$ jgen server.class
```

This produces the two files serverStub.java and serverSkeleton.java. Alternatively, you can do a `make all` to compile and generate everything. To build everything anew you can do a `make clean; make all`.

H.3.6 Step 5: Writing the server mainline

To actually start a serverImpl object which can be accessed remotely you have to instantiate it in a main method of some other class and register it with what is known as the Object Adapter. Here is the class remoteServer.java which does all that is necessary to activate a serverImpl object (the server code itself remains unchanged):

```
package example1;

public class remoteServer
{
    public static void main( String[] args )
    {
        try
        {
            // initialize ORB and BOA
            org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init();
            org.omg.CORBA.BOA boa = orb.BOA_init();
            // let the BOA create the server CORBA object
            org.omg.CORBA.Object server =
                boa.create( new serverImpl(),
                    "IDL:example1/server:1.0");
            // tell the BOA about it
            // (this named_obj_is_ready()-call is a proprietary
            // shortcut for calling boa.obj_is_ready() and
            // telling the name server to bind a name to
            // this object.
            boa.named_obj_is_ready( server, "server" );

            // accept incoming requests
            boa.impl_is_ready();

        } catch (Exception e ){
            e.printStackTrace();
        }
    }
}
```

Figure 179. Server mainline which activates the server object.

After initializing the ORB and obtaining a reference to the object adapter (the BOA) a CORBA object can be created from a Java object by calling the BOA's create() operation. Only now does a CORBA object exist. In order to make it

accessible, we have to make its object reference available. This is done using a publicly accessible directory service, the naming server. JacORB allows you to reference the default name server conveniently, in this case it is not even directly visible to the Java class but is only indirectly involved in the boa.named_obj_is_ready() operation.

H.3.7 Step 6: Writing a client

Finally, we looked at the client which invokes server operations. There are actually two versions of it: one uses only the standard CORBA interface, the other makes use of a few JacORB shortcuts. Here is the compliant one.

After initializing the ORB, it obtains a reference to the naming service by calling orb.resolve_initial_references("NameService"). It then builds a name structure (an array of name components) to query the name server for the "server" object by calling resolve(). The result is an object reference of type org.omg.CORBA.Object which has to be narrowed to the type we are expecting, that is, "server". Narrowing is done through a static method in a dedicated helper class, serverHelper.

```
package example1;

import org.omg.CORBA.ORB;
import org.omg.CosNaming.*;

public class CorbaClient
{
    public static void main( String[] args )
    {
        try
        { // initialize the ORB and find the name service
            ORB orb = ORB.init();
            NamingContext nc = NamingContextHelper.narrow(
                orb.resolve_initial_references("NameService") );

            // find an object (standard COSS naming interface)
            NameComponent[] components = new NameComponent[1];
            components[0] = new NameComponent("server","service");
            server s = serverHelper.narrow(nc.resolve(components));

            // make the call
            System.out.println( s.writeMessage( "hello World" ));

        } catch (jacorb.Orb.SystemException se){
            se.printStackTrace();
        }
    }
}
```

Figure 180. Client using standard CORBA interface.

Now here is the one which uses the more convenient, but proprietary interface to the naming service to locate an object. Rather than going through all the individual steps of the first example, we simply call a static method resolve() and

supply a string argument for the name we wish to look up. Narrowing has to be done all the same, however.

```
package example1;

import jacorb.Naming.NameServer;
public class Client
{
      public static void main( String[] args )
      {
          if( args.length != 1 )
          {
              System.out.println("Usage: client <message>");
              System.exit(1);
          }

          try
          {
              // locate the server object using the Name Service
              server s = serverHelper.narrow( NameServer.locate("server"));

              // do something with it
              System.out.println( s.writeMessage( args[0] ));
              StringArray5OutHolder sh5 = new StringArray5OutHolder();
              s.arryfy2( args[0], sh5 );
              String a[] = sh5.value();
              System.out.println( s.writeMessages( a ));
              for( int i = 0; i < a.length; i++)
                  System.out.println("From example1: " + a[i] + " " + i );

          } catch (jacorb.Orb.SystemException se){
              se.printStackTrace();
          } // catch (Exception e){}
      }
}
```

Figure 181. Client using proprietary interface.

After compiling everything we are now ready to actually run the server and the client on different (virtual) machines. Make sure the name server is running before starting either the server or the client. If it is not, type something like:

```
$ ns ~/public_html/NS_Ref
```

where ~/public_html/NS_Ref is the name of a locally writable file which can be read by using the URL given in both the remote client and server code. (This is to avoid using a well-known address for the name server, so both client and server look up the location of the name server via the URL and later communicate with it directly.)

You can now launch the server:

```
$ java jacorb.demo.example1.remoteServer
```

The client can be invoked on any machine you like:

```
$ java jacorb.demo.example1.Client hello
```

Running the client using the server stub after starting the server remotely produces the following output, which is exactly the same as if the server object was local:

```
Message: hello
hello written
From example1: hello 0
From example1: hello 1
From example1: hello 2
From example1: hello 3
From example1: hello 4
```

H.4 The IDL/Java language mapping

The language mapping employed by JacORB (and realized by the IDL compiler) is standard compliant with respects to IDL base classes. The classes generated for IDL interfaces, however, differ in a number of respects. This is due to both design and technical reasons. Helper classes generally correspond to helpers generated by standard compliers though.

(As an aside: JacORB will converge with the standard mapping in the nearer future as it seems that the standard has settled a little. Still, the POA mapping appears to be incomplete.)

H.5 The JacORB name service

JacORB provides an implementation of the standard CORBA name service (specified as a CORBA Service) which allows to bind names to object references and to lookup object references using these names. The JacORB name service comprises two components: the name server program, and a set of interfaces and classes used to access the service.

H.5.1 Running the name server

The JacORB name server is a process that needs to be started before the name service can be accessed by programs. Starting the name server is done by typing on the command line either simply

```
ns <filename>
```

to run a shell script or

```
ns.bat <filename>
```

for a DOS batch file. You can also start the Java interpreter explicitly by typing

```
java jacorb.Naming.NameServer <filename>
```

In the example

```
ns ~/public_html/NS_Ref
```

We direct the name server process to write location information and logging information to the file ~/public_html/NS_Ref. Clients of the name service use this file to locate the server process. These clients do not, however, access the file through a local or shared file system. Rather, the file is read as a WWW resource by using a URL pointing to it. This implies that the name server log file is

accessible through a URL in the first place, that is, that you know of a web server in your domain which can answer HTTP request to read the file.

The advantage of this approach is that clients do not need to rely on a hard-coded well known port and that the name server is immediately available world-wide if the URL uses HTTP. If you want to restrict name server visibility to your domain (assuming that the log file is on a shared file system accessible throughout your domain) or you do not have access to a web server, you can use file URLs rather than HTTP URLs, that is the URL pointing to your name server log file would look like

```
file:/home/brose/public html/NS Ref
```

rather than

```
http://www.inf.fu-berlin.de/~brose/NS_Ref
```

In order to restrict access to the name server even further, you can use an access control interceptor (see Section H.7, “Interceptors” on page 280).

Please note that the overhead of using HTTP is only incurred once: when the clients first locate the name server. Subsequent requests will use standard CORBA operation invocations which means they will be IIOP requests (over TCP).

H.5.2 Configuring a default context

Configuring a naming context (that is, a name server) as the ORB's default or root context is done by simply writing the URL that points to this server's boostrap file to the file jacorb/Orb/Config.java. You need to type `make configure` in the JacORB directory for this change to take effect. After the default context has thus been configured, all requests through the API given in the class jacorb.Naming.NameServer will go to that server (provided it is running).

H.5.3 Accessing the name service

Name servers are used to locate objects using a human-readable reference (their name) rather than a machine or network address. If objects providing a certain service are looked up using the service name, their clients are decoupled from the actual locations of the objects that provide this service. The binding from name to service can be changed without the clients needing to know.

The JacORB name service can be accessed using the standard CORBA defined interface, but there are also a number of proprietary extensions to the name service which make it easier to use. We will go through a few examples and first use this simplified API.

H.5.3.1 The Easy Way

A very simple way of locating objects is by just calling locate on the default naming context as in:

```
server s = serverHelper.narrow(NameServer.locate("server"));
```

The locate operation is a static method of jacorb.Naming.NameServer which contacts the default name server, so we do not need to obtain a reference to a particular naming context.

The interface to the root context in jacorb.Naming.NameServer also allows to bind and unbind service names using registerService() and unregisterService() and names of other naming contexts using registerContext() and unregisterContext().

The server holding the object that provides the “server” service, which was looked up in the previous example, can create and register the object like this:

```
org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init();
org.omg.CORBA.BOA boa = orb.BOA_init();
org.omg.CORBA.Object server =
    boa.create(new serverImpl(),"IDL:jacorb/demo/example1/server:1.0");
boa.obj_is_ready( server );
jacorb.Naming.NameServer.registerService(server, "server");
```

Figure 182. The easy way to create and register an object.

The two calls boa.obj is ready(server) and jacorb.Naming.NameServer.registerService(serv"server") can even be condensed into one: boa.named obj is ready(server, "server").This shortcut is intended for service objects only, not for contexts.

H.5.3.2 The Compliant Way

In the previous section we used a proprietary interface to the name service for convenience. If you want to make the name service replaceable, that is, you want to take care that another vendor's name service can be used with your programs without rewriting them, you might prefer to use the standardized interface to the name service specified in [3].

In a simple case like the one of the previous example, this would look like this:

```
// get a reference to the naming service
ORB orb = ORB.init();
org.omg.CORBA.Object o = orb.resolve_initial_references("NameService")
NamingContext nc = NamingContextHelper.narrow( o );

// look up an object
NameComponent[] components = new NameComponent[1];
components[0] = new NameComponent("server","service");
server s = serverHelper.narrow( nc.resolve(components) );
```

Figure 183. The compliant way to create and register an object.

Before an object can be looked up, you need a reference to the ORB's name service. The standard way of obtaining this reference is to call orb.resolve initial references("NameService"). In calls using the standard name service interface, object names are represented as arrays of NameComponents rather than as strings in order to allow for structured names. Therefore, you have to construct such an array and specify that the name's name is “server” and that it is of kind “service” (rather than “context”). Now, we can look up the object by calling resolve() on the naming context, supplying the array as an argument.

H.6 The Server Side: BOA, implementation repository, threads

This section describes the facilities offered by JacORB for controlling how servers are started and executed. These include an activation daemon, the Basic Object Adapter (BOA) and its interface and threading.

H.6.1 BOA

The BOA is that part of CORBA that is responsible for creating CORBA objects and object references, and, with a little help from skeletons, dispatching operation requests to actual object implementations. In cooperation with the Implementation Repository it can also activate objects, that is, start processes with programs that provide implementations for CORBA objects.

Since the BOA is actually being deprecated by the OMG and is to be replaced by the more flexible and far better specified Portable Object Adapter (POA), JacORB implements only a subset of the full BOA functionality. Our aim is to come up with a POA implementation in the long run. In particular, only two out of four different activation modes for servers are supported: *persistent server* and *unshared server*.

Persistent servers are not registered with the Implementation Repository but rather started manually. Unshared servers, on the other hand, are registered with the Implementation Repository and started on demand when operation requests arrive. They area called unshared because they only host one registered object, in contrast to shared servers.

H.6.2 The implementation repository

The JacORB Implementation Repository allows you to register Java classes as unshared servers. Basically, it is a daemon process that you will need to run on any machine on which you want to be able to automatically activate servers.

H.6.3 Thread models

JacORB allows you to set the server-side thread model for your object implementations. Three different thread models are provided: single-threading (which is actually not a thread model at all), thread-pool and thread-per-session. These are explained in the following sections.

H.6.3.1 Single Threading

This is the most trivial thread model and suppresses any kind of multi-threading on the server side. It should only be selected in cases where servers will not tolerate any concurrent accesses at all. If more than one client accesses an object implementation using this model, performance will degrade severely.

H.6.3.2 Thread Pool

In the thread-pool model, skeletons manage a set or pool of (pre-created) worker threads. Upon invocation, an idle thread is selected from the pool and asked to carry out the invocation. This thread rejoins the pool upon completion of its task.

Concurrency in this model is only constrained by the maximum number of threads in the pool. If no idle thread is available from the pool, new threads will be created on demand if the thread pool is below some maximum, which is Configurable. If

you have multi-threaded clients, this should be your default choice. (Actually, it is the default if no thread model is explicitly selected)

H.6.3.3 Thread per Session

This model can be regarded as a specialization of the thread-pool model. Skeleton will provide one thread per client session, that is, a connection from a client. Since JacORB manages connection resources such that there will only ever be a single connection per client address space, thread-per-session equates to thread-per-client-process.

In this model, requests from multiple clients can be carried out concurrently, while multiple (potentially concurrent) requests from a single client process will not.

H.6.3.4 Setting the Thread Model

The thread model can be set per object with the setThreadModel() operation on the BOA. This operation must be called after creating the CORBA object with BOA.create(), but before calling BOA.obj is ready() for that particular object, as in the following example:

```
public class Server
{
    public static void main( String[] args )
    {
        try {
            org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init();
            org.omg.CORBA.BOA boa = orb.BOA_init();
            org.omg.CORBA.Object o = boa.create( new benchImpl(),
                "IDL:jacorb/demo/benchmark/bench:1.0");
            boa.setThreadModel( o, jacorb.Orb.ThreadModel.PER_SESSION, 0, 0 );
            boa.obj_is_ready( o );
            boa.impl_is_ready();
        } catch (Exception e ) {
            e.printStackTrace();
        }
    }
}
```

Figure 184. Setting the thread model.

The parameters for setThreadModel() in this example are:

- The CORBA object for which the thread model is to be set.
- jacorb.Orb.ThreadModel.PER SESSION: an integer denoting the thread model (one of POOLING, PER SESSION or SINGLE).
- The maximum pool size.
- The minimum pool size.

The last two parameters are ignored if the thread model is not jacorb.Orb.ThreadModel.POOLING. The default thread model is POOLING with the minimum and maximum pool size set to 10 and 20 respectively. It will be selected if the setThreadModel() operation is not called to set a different thread model or pool size.

H.7 Interceptors

Interceptors are a flexible means for having code called by the ORB at specific points during operation invocations. Using interceptors you can, for example, monitor and log method calls or define simple access control policies.

H.7.1 Introducing interceptors

Interceptors are objects of particular classes which are called at specific points during the processing of operation invocations. By defining subclasses of interceptor classes and registering instances with the ORB, you can “insert” your own code into the invocation path. This provides access to the request itself and thus a convenient and very powerful way of observing and/or modifying requests. On the other hand, it requires more detailed knowledge about ORB data structures than necessary for application level code.

Interceptors can be defined at two levels: at the request level or at the message level. Request-level interceptors are given access to a request object representing the current request and may access and modify this request object before and after it is actually invoked. Message-level interceptors have access to the actual message buffer before and after it is sent across the network and can be used, for example, to encrypt and decrypt the entire message.

In a CORBA invocation, there are eight points where interceptors are (potentially) given control. On the client side, these are points 1 and 2 (see Figure 185 on page 280): before the request is sent across the network, and points 7 and 8: after the result is received and before control is returned to the caller. At point 1, the interceptor has access to a structured request object, whereas at point 2, the message can be accessed in marshalled form in a buffer just before it is sent onto the network.

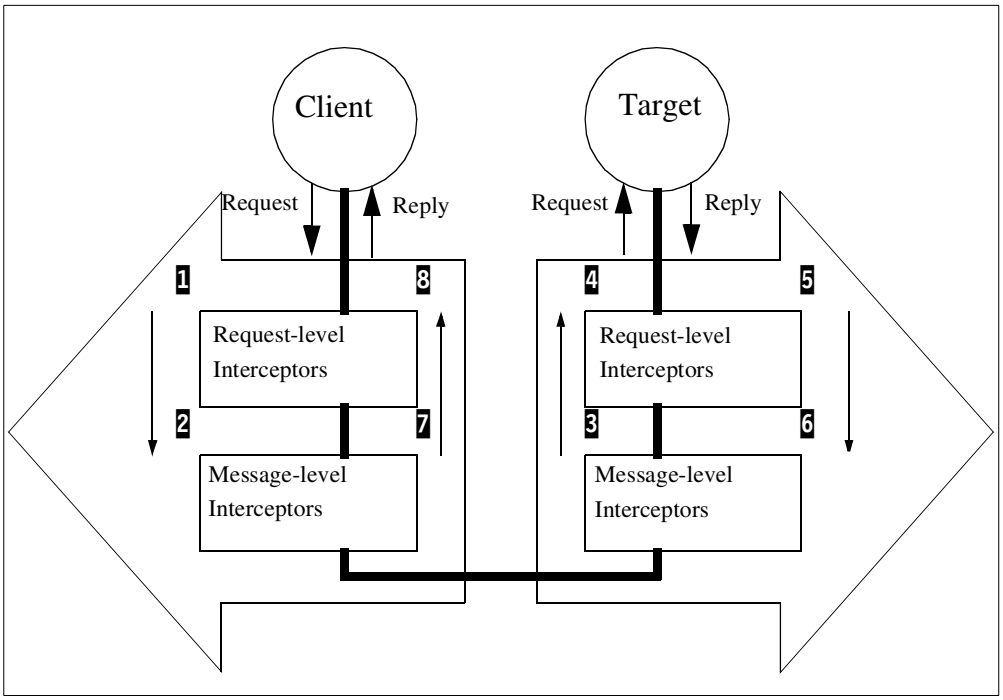


Figure 185. Eight points at which interceptors may gain control

Because of this distinction, interceptors come in two flavours: request-level and message-level interceptors. On the server side, intercepting requests can also be done at these different levels (points 3, 4 and 5, 6). For every single one of these eight points, a different method is called on different interceptor objects. We look first at request-level interceptors.

H.7.2 Request-level interceptors

Request-level Interceptors are objects of subclasses of ClientRequestInterceptor on the client side and of ServerRequestInterceptor on the server side. We will look at these two in turn.

H.7.2.1 Client-side interceptors

On the client side, methods pre_invoke() and post_invoke() of classes implementing jacorb.Orb.ClientRequestInterceptor are called at points 1) and 8), respectively. Below is this interface:

```
package jacorb.Orb;

public interface ClientRequestInterceptor
{
    /* for JacORB requests */
    public void pre_invoke( org.omg.CORBA.Object t,
                            String op, String argsig,
                            String returnsig, Object[] args
                          );

    public void post_invoke(org.omg.CORBA.Object t,
                            String op, String argsig,
                            String returnsig, Object[] args
                            Object result
                          );

    /* for DII requests */
    public void pre_invoke( org.omg.CORBA.Request r );
    public void post_invoke( org.omg.CORBA.Request r );
}
```

Figure 186. Client-side interceptor interface.

Note that different interceptor methods are called depending on whether the invocation was performed using statically compiled stubs or the DII. As a consequence, request information is available in different form when the DII is used as when the static invocation interface (stubs/skeletons) is used. As an example, consider the following interceptor which “traces” method calls:

```
package jacorb.Orb.util;

/**
* a simple request interceptor
**/

public class TraceClientRequestInterceptor
    implements jacorb.Orb.ClientRequestInterceptor
{
    private int indent = 0;

    public TraceClientRequestInterceptor(){}

    public void pre_invoke( org.omg.CORBA.Object t,
                            String op, String argsig,
                            String returnsig, Object[] args )
    {
        for( int i = 0; i < indent; i++ )
            System.out.print(" ");
        System.out.println("[ invoke <" + op + "> ]");
        indent += 2;
    }

    public void post_invoke( org.omg.CORBA.Object t,
                            String op, String argsig,
                            String returnsig, Object[] args,
                            Object result )
    {
        indent -= 2;
        for( int i = 0; i < indent; i++ )
            System.out.print(" ");
        System.out.println("[ <" + op + "> returns ]");
    }

    public void pre_invoke( org.omg.CORBA.Request r )
    {
        for( int i = 0; i < indent; i++ )
            System.out.print(" ");
        System.out.println("[ invoke " + r.operation() + " ]");
        indent += 2;
    }

    public void post_invoke( org.omg.CORBA.Request r )
    {
        indent -= 2;
        for( int i = 0; i < indent; i++ )
            System.out.print(" ");
        System.out.println("[ " + r.operation() + " returns ]");
    }
}
```

Figure 187. Sample use of client-site interceptors.

When the DII is used, request information is available as an object of class org.omg.CORBA.Request whereas it is available through a number of parameters such as org.omg.CORBA.Object and strings when stubs are used.

H.7.2.2 Server-side interceptors

Interceptors that are called at points 2) and 3) on the server side implement the interface jacorb.Orb.ServerRequestInterceptor:

```
package jacorb.Orb;

public interface ServerRequestInterceptor
{
    /* for JacORB requests */
    public void pre_invoke( jacorb.Orb.Request r);
    public void post_invoke( jacorb.Orb.Request r);

    /* for DII requests */
    public void pre_invoke( org.omg.CORBA.ServerRequest r );
    public void post_invoke( org.omg.CORBA.ServerRequest r );
}
```

Figure 188. Sever-side interceptor interface.

These interceptors gain access to the request through objects of type jacorb.Orb.Request and org.omg.CORBA.ServerRequest respectively.

H.7.2.3 Per-object versus per-process Interceptors

If the ORB is to use instances of the above class during invocation processing, these interceptors need to be registered with the ORB. There are two options for doing this. An interceptor can be registered for general use, i.e. it will be called during every single invocation from the current address space, regardless of which object is the target of the call. This is called a per-process interceptor.

Another, more fine-grained way of using interceptors is to register them as per-object interceptors, that is, to register them for use in invocations on particular objects only. As an example, consider the simple tracer again: You might want to trace method calls on all objects, or you might be interested in operations on one particular object only.

H.7.2.4 Registering Interceptors

A per-process interceptor is registered with the ORB as shown in the following code fragment:

```
org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init();
jacorb.Orb.util.TraceClientRequestInterceptor t =
       new jacorb.Orb.util.TraceClientRequestInterceptor();
...
orb.addInterceptor( t );
...
```

Figure 189. Registering a per-process interceptor.

When addInterceptor() is called, the ORB will append the argument as a per-process interceptor to the end of an internal interceptor chain. On any subsequent invocation, all interceptors in this chain will be called in the order in

which they were appended to the chain. Of course, interceptors can also be removed again:

```
orb.removeInterceptor(t);
```

Per-object interceptors are registered by calling addInterceptor() with the object reference as the first argument:

```
org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init();
jacorb.Orb.util.TraceClientRequestInterceptor t =
       new jacorb.Orb.util.TraceClientRequestInterceptor();
org.omg.CORBA.Object o = ...
...
orb.addInterceptor( o, t );
...
orb.removeInterceptor( o, t);
```

Figure 190. Registering per-object interceptors.

H.7.2.5 Example: An Access-Control Interceptor

As an example, consider the following server-side request interceptor which uses a simple ACL object to check authorization for operations using the principal information which is transmitted with every request.

```
package jacorb.Orb.Security;

public class AccessControlInterceptor
    implements jacorb.Orb.ServerRequestInterceptor
{
    private ACL acl;

    public AccessControlInterceptor( ACL _acl)
    {
        acl = _acl;
    }

    public void pre_invoke( jacorb.Orb.Request r )
    {
        if( !acl.is_allowed( r.op_name(),
               r.req_hdr.requesting_principal.toString() ))
        {
            System.out.println("[ Denied <" + r.op_name() +
                       "> to " + r.req_hdr.requesting_principal + " ]");
            throw new org.omg.CORBA.NO_PERMISSION( 3000,
                       org.omg.CORBA.completion_status.COMPLETED_NO);
        }
    }

    //...
    /* for DSI requests */
    public void pre_invoke( org.omg.CORBA.ServerRequest r )
    {
        org.omg.CORBA.Principal p =
            ((jacorb.Orb.DSI.ServerRequest)r).req_hdr.requesting_principal;
        if( !acl.is_allowed( r.op_name(), p.toString() ))
        {
            System.out.println("[ Denied <" + r.op_name() +
                       "> to " + p + " ]");
            throw new org.omg.CORBA.NO_PERMISSION( 3000,
                       org.omg.CORBA.completion_status.COMPLETED_NO);
        }
    }
    //...
}
```

Figure 191. Example server-side request interceptor.

A server could install this interceptor like this (in main()):

```
org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init();
org.omg.CORBA.BOA boa = orb.BOA_init();
NamingContextImpl n = new NamingContextImpl();

// create and set up an ACL for this implementation
jacorb.Orb.Security.ACL acl = new jacorb.Orb.Security.ACL( n );
acl.allow( "bind", "brose" );
acl.allow( "resolve", "brose" );

// create and install the access control interceptor
jacorb.Orb.Security.AccessControlInterceptor act =
       new jacorb.Orb.Security.AccessControlInterceptor(acl);
orb.addServerInterceptor( n, act );

// tell the BOA
org.omg.CORBA.Object ctx = boa.create( n,
       "IDL:omg.org/CosNaming/NamingContext:1.0" );
boa.obj_is_ready( ctx );

//...
boa.impl_is_ready();
```

Figure 192. Example installation of server-side interceptor.

In this example, the access control interceptor is registered as a per-object interceptor, that is, it will only be consulted for invocations on this particular implementation. If there were any other objects in this server, they would not be access-protected. In this particular case, a naming context is protected so as to allow only the bind and resolve operations, and these only to a principal with identity “brose”. All other requests will receive a NO PERMISSION exception.

H.7.3 Message-level interceptors

Message-level Interceptors implement either ClientMessageInterceptor or ServerMessageInterceptor and generally work like request-level interceptors. The main difference is that the request is passed as an unstructured buffer, so there is no need to distinguish between DII accesses and accesses through the static stub layer.

Also note that the client-side interceptor has an additional parameter indicating the target of the invocation. This parameter is not available at the server-side because it is only known after the request message has been interpreted by the transport layer software, which is only after the message-level interceptor has had access to the message, for example, for decryption purposes.

```
package jacorb.Orb;

public interface ClientMessageInterceptor
{
    public byte [] pre_invoke( org.omg.CORBA.Object t,
                               byte [] buf
                               );
    public byte [] post_invoke( org.omg.CORBA.Object t,
                                byte [] buf
                              );
}

package jacorb.Orb;
public interface ServerMessageInterceptor
{
    public byte [] pre_invoke( byte [] buf );
    public byte [] post_invoke( byte [] buf );
}
```

Figure 193. Message-level interceptor interface.

H.7.3.1 Example: A simple Encryption Interceptor

As an example, consider a simple interceptor which "encrypts" the message contents by reversing the message buffer. Note that the interceptor preserves the first 12 bytes of the message (this is the GIOP message header). The reason for this is that the message can only be correctly received if the receiving side is able to determine the message size. To this end, a GIOP message header with the correct message size must be sent across.

In the example, the transformation on the message buffer does not change the message size, so we can simply reuse the original message header. Many other transformations, however, will result in a different message length, so it is necessary to provide a correct message header. The format of the GIOP message header, as described in the CORBA specification, is as follows:

```
struct Version
{
    char major;
    char minor;
};

struct MessageHeader
{
    char magic [4];
    Version GIOP_version;
    boolean byte_order;
    octet message_type;
    unsigned long message_size;
};
```

Figure 194. GIOP message header.

Alternatively, you can examine the class org.omg.GIOP.MessageHeader which gives the Java equivalent of the IDL specification. Here comes the Java code for the example “encryption” interceptor:

```
public class ClientMessageInterceptor
    implements jacorb.Orb.ClientMessageInterceptor
{
    public byte [] pre_invoke( org.omg.CORBA.Object t,
                               byte [] buf )
    {
        byte buf2[] = new byte[buf.length];

        // copy GIOP message header
        System.arraycopy( buf, 0, buf2, 0, 12 );

        // reverse rest of message
        for( int i = 12; i < buf.length; i++ )
            buf2[i] = buf[ (buf.length+11) - i ];
        return buf2;
    }

    public byte [] post_invoke( org.omg.CORBA.Object t,
                                byte [] buf )
    {
        // don't do anything here
        return buf;
    }
}
```

Figure 195. Java code for example "encryption" interceptor.

On the receiving side, the effect of the buffer transformation must be undone before the message can be passed up to the ORB software and be interpreted as a request, so we must apply the same transformation again:

```
public class ServerMessageInterceptor
    implements jacorb.Orb.ServerMessageInterceptor
{
    public byte [] pre_invoke( byte [] buf )
    {
        byte buf2[] = new byte[buf.length];

        // copy GIOP message header
        System.arraycopy( buf, 0, buf2, 0, 12 );

        // reverse rest of message
        for( int i = 12; i < buf.length; i++ )
            buf2[i] = buf[ buf.length+11-i ];
        return buf2;
    }

    public byte [] post_invoke( byte [] buf )
    {
        // don't do anything here
        return buf;
    }
}
```

Figure 196. Undoing the effect of the buffer transformation.

H.7.3.2 Per-object versus per-process Interceptors

As with request-level interceptors, it is sometimes useful to determine whether to apply a particular interceptor depending on which object is actually called. Accordingly, message-level interceptors can be registered per-object or per-process. However, this is only possible on the client side:

```
orb.addInterceptor( (org.omg.CORBA.Object)s, new
ClientMessageInterceptor());
```

Recall that on the receiving side, a reference to the object is only available after the ORB software has interpreted the message, which it cannot do, for example, before the message has been decrypted in the above example. As a consequence, server-side message-level interceptors can only be registered per process. If an interceptor may only apply certain transformations on messages to particular objects, it is responsible for determining the target by itself.

```
org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init();
org.omg.CORBA.BOA boa = orb.BOA_init();
org.omg.CORBA.Object o = boa.create( new dynamicServer(),
                    "IDL:jacorb/demo/Interceptors/server:1.0");

// create and register a trace interceptor
orb.addServerInterceptor( new jacorb.Orb.util.TraceServerRequestInterceptor() );

boa.named_obj_is_ready( o, "Interceptors-Example Server");

orb.addServerInterceptor( new ServerMessageInterceptor() );
boa.impl_is_ready();
```

Figure 197. Sample interceptor registration.

H.8 The interface repository

This chapter introduces the JacORB Interface Repository and tells you how to run and use it.

H.8.1 Introduction

The Interface Repository (IR) is a runtime component in the ORB architecture and used to dynamically obtain information on IDL types, for example, object interfaces. This kind of information is necessary when a client wants to use the Dynamic Invocation Interface (DII) to construct requests dynamically, that is, without static knowledge about another object's type.

In this case, when there is no stub available for a CORBA object, the Interface Repository can be queried to return information about the object's operations and the parameters necessary to invoke them. Using this information, a client can determine how to dynamically construct a request to the object.

ORBs can also use the IR for runtime type checks when narrowing object references to a particular type, that is, to determine whether an object's type really matches a local type definition. An ORB might consider this necessary when receiving a reference to an object that is managed by another ORB.

H.8.2 The JacORB Interface Repository

The JacORB Interface Repository is just another CORBA object that can be accessed as soon as its reference is known, for example, after retrieving it from the ORB using the resolve initial references() call with an argument string “InterfaceRepository”. This reference can then be used to query the IR for type definitions. Write operations on the IR, that is, modifications, are not supported in JacORB.

There is also another way to contact the IR. Every CORBA object reference allows a client to call the get interface() operation, which will return an object of type InterfaceDef that describes the object most derived type. InterfaceDef objects are managed by the IR, and every get interface() call will actually contact the IR and retrieve the InterfaceDef from its contents.

H.8.3 Running the IR

As the IR is actually a CORBA object, a separate process must be started for the IR to be available. To start the JacORB Interface Repository, simply type

```
$ ir <IR location file> [ <classpath> ]
```

to run a shell script or

```
$ ir.bat <IR location file> [ <classpath> ]
```

for a DOS batch file. You can also start the Java interpreter explicitly by typing

```
java jacorb.Orb.IR.ir <IR location file> [ <classpath> ]
```

In these examples, <IR location file> is a file name and the optional <classpath> argument is a Java CLASSPATH. The usage of a file to store the Repository's object reference so that it can be retrieved by the ORB is analogous to the way the name server uses such files, that is, it must be able to read the file using the URL configured in Config.java.

The <classpath> argument determines the contents of the Repository, that is, the set of repository definitions retrievable from the IR. If it is not explicitly given, the class path is taken from the environment.

In JacORB, the IR does not keep any persistent data of its own. Rather, it uses the Java .class files accessible in the class path that was given as an argument to the IR process. The JacORB IR employs Java reflection to derive IDL meta data like InterfaceDef objects on demand, that is, when this information is needed.

This design has a number of consequences for users. First of all, you need not worry about IDL compiler switches or anything of that kind in order to make sure that the IR will know about the types you are working with. The IR will know as soon as there are java classes representing the IDL types and these classes are necessary anyway and therefore generated by the IDL compiler without any intervention from your side. So, as soon as you compile an IDL file and then compile the generated Java classes, this information is accessible to the IR provided its class path is properly configured.

In the future, the IR will be able to accept Java classes as input that were written manually, so it will be possible to derive IDL meta information from classes that never originated from or represent IDL types! This will support the development of CORBA objects in Java only without ever writing a single line of IDL!

H.8.3.1 Pragma Prefix

Another consequence of this IR design is the fact that the JacORB IDL compiler does not accept the #pragma prefix directive in IDL files. This directive can be used to set the repository identifier for IDL types. The JacORB IR relies on the equivalence of Java class names and repository identifiers, so naming these apart by using the #pragma prefix directive cannot not be allowed.

The only exception to this is the “built-in” prefix "omg.org" which is specified for the whole CORBA package. JacORB does of course manage repository IDs so that classes in the org.omg package have identifiers that begin with "omg.org" rather than "org/omg".

H.8.3.2 The Class Path

When providing a class path as an argument to the IR process, a few things need to be considered. First of all, the IR cannot presently handle class archives like .zip, .jar or .cab files. Secondly, providing a class path made up of directories where one is actually a subdirectory of another will lead to confusion. For example, a class path like /home/jim/JacORB:/home/jim/JacORB/classes will lead to such problems.

In particular, the examples in jacorb/demo/example1 and example2 will confuse the IR because the directory structure in these examples does not match the class path structure. You need delete the .class files in these directories (or do a make clean) to work around this limitation.

H.8.4 Accessing the Interface Repository

There are two ways to access the Interface Repository, as was already pointed out previously. Here are two examples for contacting the IR.

H.8.4.1 Calling get interface()

In the following code fragment a client wishes to obtain interface information about some object reference it has just received from the name server. It retrieves the InterfaceDef object and prints out its ID:

```
org.omg.CORBA.Object obj = NameServer.locate("SomeUnknownServer"));

// get the server object's interface definition
// in the Interface Repository

InterfaceDef i_def = obj._get_interface();
if( i_def != null )
    System.out.println("ID: " + i_def.id() );
```

Figure 198. Example of calling _get_interface().

The InterfaceDef object describing a particular IDL interface type contains enough information about the type to, for example, reconstruct the textual IDL definition, so the client could do the following:

```
// print it to the screen using the IdlWriter class
// (with an indentation of 2)
IdlWriter idlw = new IdlWriter(System.out);
idlw.printInterface( i_def, 2 );
```

Figure 199. Reconstructing the IDL definition.

This will print the IDL definition of the object's interface to the screen.

H.8.4.2 Accessing the IR directly

The IR can also be queried directly, using operations to lookup names and return collections describing the IR contents. This can be done from any client program that holds a reference to the IR.

JacORB also comes with a little utility program `qir` that allows you to lookup repository identifiers from the command line and display the corresponding IDL definition on the terminal:

```
$ qir "IDL:omg.org/GIOP/LocateStatusType:1.0"
```

will print:

```
enum LocateStatusType fUNKNOWN OBJECT,OBJECT HERE,OBJECT FORWARDg;
```

The Java code for this little program follows. The program retrieves the reference to the IR using a proprietary JacORB API. It could also have used the standard operation orb.resolve initial reference("InterfaceRepository"). It then looks up the ID given on the command line and prints the result using the JacORB IdlWriter class.

```
package jacorb.Orb.IR;
import java.io.*;
import org.omg.CORBA.Contained;
import org.omg.CORBA.Repository;

public class QueryIR
{
    public static void main( String[] args )
    {
        if( args.length != 1 )
        {
            System.err.println("Usage: qir <RepositoryID>");
            System.exit(1);
        }
        try
        {
            Repository ir = jacorb.Orb.IR.Repository.getRepository();
            Contained c = ir.lookup_id( args[0] );
            if( c != null )
            {
                IdlWriter idlw = new IdlWriter(System.out);
                idlw.printContained( c, 2 );
            } else
                System.out.println( args[0] + " not found in IR.");
        } catch ( Exception e){
            e.printStackTrace();
        }
    }
}
```

Figure 200. Example of accessing the IR directly.

H.9 JacORB utilities

In this chapter, we briefly explain the executable files that come with JacORB. These include the IDL compiler and stub generator, the JacORB name server and

the server for the interface repository. For users without C Shells installed, the command-line for starting the utilities directly by starting the Java interpreter has been given.

H.9.1 idl/idl2j

The IDL compiler parses IDL specifications and maps these to a set of Java classes. IDL interfaces are translated into Java interfaces, and typedefs, structs, const declarations and so forth are mapped onto “equivalent” Java classes. For a description of the Java language mapping please see [1] Using the idl utility, stubs and skeletons for all interface types in the IDL specification will automatically be generated.

If want only IDL mapping without stubs generation, use the `idl2j` utility instead. The syntax is the same for both commands.

idl/idl2 Usage

```
idl [-syntax] [-p package prefix ][-d <Output Directory>] file.idl
```

or

```
idl.bat [-syntax] [-p package prefix ][-d <Output Directory>] file.idl
```

Invoking `idl` with the -syntax option allows you to check your IDL specification for syntactic errors without producing code. Without -syntax, the compiler creates directories according to the Java package structure.

Compiling a file with a module ModuleX and an interface InterfaceY in it will result in a subdirectory ModuleX with InterfaceY.java in it (and possibly more .java files). By default, the root directory for all of the files created during the process is the current directory. You can, however, provide a different directory in which these files are to be placed by using the -d option. Using the -d option when you are only checking the syntax of an IDL file has no effect.

With the -p option it is also possible to specify a package pre_x for the generated Java classes. For example, if the IDL file contains a module Bank which defines an interface Account, you can direct the IDL compiler to generate Java classes example.Bank.Account by compiling with the package pre_x set to example, that is, by compiling with

```
idl -p example bank.idl
```

The IDL compiler will create the appropriate directories if necessary.

If you used `idl2j`, you will have to generate stubs and skeletons for selected interfaces in a separate step using `jgen`.

(The IDL parser was generated with Scott Hudson's CUP parser generator. The LALR grammar for the CORBA IDL is in the file jacorb/Idl/parser.cup.)

H.9.2 jgen

The generator takes a .class file (which can be compiled from a regular Java class or interface) and automatically constructs Java source code for a client and a server stub class from it.

It can also be used to derive a class's interface from its implementation or produce informative interface output about the class on stdout. It does more than the javap disassembler in that it traverses the whole inheritance graph of a class in order to list every method of your class the inherited ones as well as the ones defined in the class itself. There is one exception to this: methods inherited from class Object are not listed.

jgen Usage

```
jgen [ options ] <file.class>
```

or

```
jgen.bat [ options ] <file.class>
```

or

```
java jacorb.generator.Main [ options ] <file.class>
```

The JacORB generator, when used without command line options, produces two files containing the Java source code for a client stub and a server skeleton.

jgen Example

```
$ jgen Game.class
```

generates GameStub.java and GameSkeleton.java which you can then compile and use.

jgen Options

-i derive an interface file instead of stub and proxy class files from a class. This interface file can then be compiled with javac.

Example:

```
$ jgen -i Game.class
```

generates a file GameInterface.java. You can check that it is ok by adding implements GameInterface to your class definition Game in Game.java and recompile.

In the produced interface file, every inherited method has a comment to it which indicates where it was inherited from, for example:

```
public void repaint(int, int, int, int);
// from java.awt.Component
```

Any method which is not accessible to the class the interface is derived from is commented out, so that the interface will compile correctly.

-f print interface information on stdout.

Example:

```
$ jgen -f jacorb/demo/cardgame/PlayerApp.class
```

(The -f means " atten the inheritance structure").

H.9.3 ns

JacORB provides a service for mapping names to network references. The name server itself is written in Java like the rest of the package and is a straightforward implementation of the CORBA “Naming Service” from Common Object Services

Specification, Vol.1 [3]. The IDL interfaces are mapped to Java according to our Java mapping.

ns Usage

```
ns <filename>
```

or

```
ns.bat <filename>
```

or

```
java jacorb.Naming.NameServer <filename>
```

ns Example

```
ns ~/public_html/NS_Ref
```

The name server does not use a well known port for its service and there is no way to direct it to a specific port. Since clients cannot (and need not) know in advance where the name service will be provided, we use a bootstrap file in which the name server records an opaque object reference to itself (its Interoperable Object Reference or IOR). The name of this bootstrap file has to be given as an argument to the ns command. This bootstrap file has to be available to clients network-wide, so we demand that it be reachable via a URL:| that is, there must be an appropriately configured HTTP server in your network domain which allows read access to the bootstrap file over a HTTP connection. (This implies that the file must have its read permissions set appropriately. If the binding to the name service fails, please check that this is the case.) After locating the name service through this mechanism, clients will connect to the name server directly, so the only HTTP overhead is in the first lookup of the server.

The name bindings in the server's database are mirrored in the bootstrap file, so you have a network wide readable log of which names are already in use.

H.9.4 lsns

This utility lists the contents of the default naming context. Only currently active servers that have registered are listed. The -r option recursively lists the contents of naming contexts contained in the root context.

lsns Usage

```
lsns [ -r ]
```

or

```
lsns.bat [ -r ]
```

or

```
java jacorb.Naming.lsns [ -r ]
```

lsns Example

```
$ lsns
/grid.service
```

when the server for the grid example is running.

H.9.5 ir

In order to use the JacORB Interface Repository, a separate server process needs to be started “somewhere” in your network domain. Similar to the name server, the IR server takes a file name argument that will be used to store its IOR and make it accessible via a URL (as configured in Config.java) to the ORB so that it can find the IR server.

The second argument, the classpath, determines the contents of the Repository. The JacORB Interface Repository contains nothing but Java class files, and it will know only about the files in the particular class path given to the IR server at startup time.

ir Usage

```
ir <filename> [<classpath>]
```

or

```
ir.bat <filename> [<classpath>]
```

or

```
java jacorb.Orb.IR.Repository <filename> [<classpath>]
```

ir Example

```
ir ~/public_html/IR_location ~/classes
```

H.9.6 qir

This utility queries the IR for a particular repository identifier and prints the respective IDL definition to the screen.

qir Usage

```
qir <repository ID>
```

or

```
qir.bat <repository ID>
```

or

```
java jacorb.Orb.IR.QueryIR <repository ID>
```

qir Example

```
$ qir "IDL:omg.org/GIOP/LocateStatusType:1.0"
enum LocateStatusType UNKNOWN OBJECT,OBJECT HERE,OBJECT FORWARD;
```

H.9.7 dior

JacORB comes with a simple utility to decode an interoperable object reference (IOR) in string form in to a more readable representation. In the following example we use it to print out the contents of the IOR that the JacORB name server writes to the client bootstrap file:

dior Usage

```
dior <IOR-string>
```

or

```
dior <IOR-string>
```

or

```
java jacorb.Orb.util.PrintIOR <IOR-string>
```

dior Example

```
dior IOR:000000000000002849444c3a6f6d672e6f72672f436f7
34e616d696e672f4e616d696e67436f6e746578743a312e300000000001000000
000000002c000100000000000e3136302e34352e3131302e36370094590000001
0eee51bdce9d8222eded628696cf002c3
------IOR components-----
TypeId : IDL:omg.org/CosNaming/NamingContext:1.0
Profile Id : TAG_INTERNET_IOP
IIOP Version : 1.0
Host : 160.45.110.67
Port : 37977
Object key : ".(il
```

H.10 Limitations, Feedback

A few limitations and known bugs (list is incomplete):

- The IDL compiler does not support
  - The #pragma prefix
  - The context construct
- The Interface Repository
  - Cannot handle zip files in the class path properly
- There is no reference counting for stubs or object implementations yet
- There is no implementation repository yet servers have to be started manually
- The API documentation and this document are incomplete.

H.10.1 Feedback and bug reports

Please mail bug reports as well as criticism and experience reports to: brose@inf.fu-berlin.de

If you want to receive update notifications, please subscribe to the “jacorb-announce” mailing list by sending mail with “subscribe jacorb-announce” to Majordomo@inf.fu-berlin.de

H.10.2 Bibliography

- [1] Gerald Brose, *JacORB - A Java Object Request Broker*, http://www.inf.fu-berlin.de/~brose/jacorb/jacorb.ps.
- [2] Gerald Brose, *JacORB - Implementation and Design of a Java ORB*, Proceedings DAIS '97, Chapman & Hall 1997, pp. 143-154.
- [3] OMG, *Common Object Services Specification, Volume I*, Revision 1.0, March 1994, OMG Document 94-1-1.
- [4] OMG, *The Common Object Request Broker: Architecture and Specification*, Revision 2.0, July 1995.
- [5] OMG, *IDL/Java Language Mapping*, OMG TC document orbos/97-03-01.
- [6] Jon Siegel, *CORBA Fundamentals and Programming*, Wiley 1996.
- [7] Steve Vinoski, *CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments*, IEEE Communications, Vol. 14(2), February 1997.

Appendix I. LDAP Support EXEC

LDAP has two important EXECs: one to build the LDIF file (BLDLDIF EXEC) and the other to build the GDBM database from the LDIF file (bldldif.rexx). The latter is illustrated in Figure 118 on page 128, and will work regardless of from what source the LDIF was created. The EXEC that created the source file is specific to the residency, but can be used as a guide to writing others that meet the needs of different organizations.

I.0.1 BLDLDIF EXEC

```
/* Build LDIF from our names file */
parse source . . . Type .
if Type = 'EXEC' then
   call BLDLDIF_EXEC
else
   call BLDLDIF_REXX
exit Rc

BLDLDIF_EXEC:
   call PROLOG
   call GET_DIV_SECT
   call GET_ADDR_DATA
   call OUTPUT
   call EPILOG
return

PROLOG:
   'EXECLOAD BLDLDIF EXEC * BLDLDIF REXX'
   say TIME('N') 'Prolog'
   Prolog.1 = 'dn: o=ITSO,c=US'
   Prolog.2 = 'objectclass: top'
   Prolog.3 = 'objectclass: organization'
   Prolog.4 = 'o: ITSO'
   Prolog.5 = 'telephonenumber: +1-914-5556666^'
   Prolog.6 = 'dn: cn=root,o=ITSO,c=US'
   Prolog.7 = 'cn: root'
   Prolog.8 = 'objectclass: administration^'
   Prolog.0 = 9
   Epilog.0 = 0
   I_InRec  = 0
return

GET_DIV_SECT:
   say TIME('N') 'Getting division and section data'
   'NAMEF :nick DIVISION :list (STEM DIV. FILE ADDRESS'
   'PIPE (name SPLIT_DIVS)',
      '| stem Div.',
      '| split',
      '| stem Div.'
   do I_Div = 1 to Div.0
      'NAMEF :nick' Div.I_Div ':name (STEM DivName. FILE ADDRESS'
      DivName           = STRIP(DivName.1)
      I_InRec = I_InRec + 1
      InRec.I_InRec = 'dn: ou='DivName',o=ITSO,c=US^' ||,
                      'objectclass: top^' ||,
                      'objectclass: division^' ||,
```

```
                        'objectclass: organizationalUnit^' ||,
                        'ou:' DivName'^'
      'NAMEF :nick' Div.I_Div ':list (STEM Sect. FILE ADDRESS'
      'PIPE (name SPLIT_SECTS)',
         '| stem Sect.',
         '| split',
         '| stem Sect.'
      do I_Sect = 1 to Sect.0
         'NAMEF :nick' Sect.I_Sect ':name (STEM SectName. FILE ADDRESS'
         I_InRec = I_InRec + 1
         InRec.I_InRec = 'dn: ou='SectName.1',o=ITSO,c=US^' ||,
                         'objectclass: top^' ||,
                         'objectclass: section^' ||,
                         'objectclass: organizationalUnit^' ||,
                         'division:' DivName'^' ||,
                         'ou:' SectName.1'^'
      end
   end
   InRec.0 = I_InRec
return

GET_ADDR_DATA:
   say TIME('N') 'Retrieving address data'
   'NAMEF :TYPE Person (STEM Person. * FILE ADDRESS'
   'PIPE (name BLDPERSON)',
      '| stem Person.',
      '| bldldif',
      '| stem InRec. append'
return

OUTPUT:
   say TIME('N') 'Building ldif database'
   'OPENVM ERASE /home/ldap/itsoldif'
   'PIPE (name BUILD_OUTPUT)',
      '| stem Prolog.',
      '| append stem InRec.',
      '| append stem Epilog.',
      "| change /^/"'15'x"/",
      '| strip trailing',
      '| > /home/ldap/itsoldif'
return

EPILOG:
   say TIME('N') 'LDIF file built.'
   'EXECDROP BLDLDIF REXX'
   Rc = 0
return

BLDLDIF_REXX:
   'PEEKTO INPUT'
   do while Rc = 0
      'CALLPIPE (name GET_A_USER end ?)',
         '| *:',
         '| a: take 1',
         '| b: faninany',
         '| change w1 $:$/$',
         '| xlate w1',
         '| spec w1 1 $/$ n w2-* n',
```

```
        '| varload',
        '? a:',
        '| tolabel :nick',
        '| b:'
        OutRec = 'dn: cn='Name '(' ||,
                UserID'@'node'),o=ITSO,c=US^' ||,
                'cn:' Name'^' ||,
                'givenname:' WORD(Name,1) '^' ||,
                'sn:' WORD(Name,WORDS(Name))'^' ||,
                'uid:' TRANSLATE(UserID)'^' ||,
                HOMEPAGE(UserID) ||,
                'mail:' UserID'@'Node'^' ||,
                'mailhost:' Node '^' ||,
                'telephonenumber:' Phone'^' ||,
                'city:' City'^' ||,
                'nick:' Nick'^' ||,
                'title:' Title'^' ||,
                'objectclass: person^' ||,
                'ou:' Section'^' ||,
                'ou:' Division'^' ||,
                'organization:' Division'^'
      'OUTPUT' OutRec
      'PEEKTO INPUT'
   end
return Rc * (Rc <> 12)

HOMEPAGE:
   parse arg UserID .
   URL = 'labeleduri: http://'node'/~'UserID'/^'
return URL
```

Appendix J. Product service levels

This appendix contains the SRVAPPS files for the products used during this residency.

J.1 CP SRVAPPS

```
:PTF.UM28669 :STAT.APPLIED.02/07/98.14:39:00.MAINT
:PTF.UMRSU01 :STAT.SUPED.04/13/98.13:02:02.MAINT APPLIED.02/07/98.14:39:00.MAINT
:PTF.UM28691 :STAT.SUPED.04/13/98.13:02:02.MAINT
:PTF.UM28693 :STAT.APPLIED.04/13/98.13:02:02.MAINT
:PTF.UM28695 :STAT.APPLIED.04/13/98.13:02:02.MAINT
:PTF.UM28724 :STAT.APPLIED.04/13/98.13:02:02.MAINT
:PTF.UM28733 :STAT.APPLIED.04/13/98.13:02:02.MAINT
:PTF.UM28746 :STAT.APPLIED.04/13/98.13:02:02.MAINT
:PTF.UM28752 :STAT.APPLIED.04/13/98.13:02:02.MAINT
:PTF.UM28782 :STAT.APPLIED.04/13/98.13:02:02.MAINT
:PTF.UM28784 :STAT.SUPED.04/13/98.13:02:02.MAINT
:PTF.UM28814 :STAT.SUPED.09/25/98.15:22:55.MAINT APPLIED.04/13/98.13:02:02.MAINT
:PTF.UMRSU04 :STAT.SUPED.05/26/98.13:18:32.MAINT APPLIED.04/13/98.13:02:02.MAINT
:PTF.UM28728 :STAT.APPLIED.05/26/98.13:18:32.MAINT
:PTF.UMRSU07 :STAT.SUPED.07/20/98.17:50:55.MAINT APPLIED.05/26/98.13:18:32.MAINT
:PTF.UM28945 :STAT.APPLIED.07/20/98.17:50:55.MAINT
:PTF.UMRSU10 :STAT.SUPED.09/01/98.23:52:25.MAINT APPLIED.07/20/98.17:50:55.MAINT
:PTF.UM28835 :STAT.SUPED.09/01/98.23:52:25.MAINT
:PTF.UM28969 :STAT.APPLIED.09/01/98.23:52:25.MAINT
:PTF.UM28970 :STAT.APPLIED.09/01/98.23:52:25.MAINT
:PTF.UM28973 :STAT.APPLIED.09/01/98.23:52:25.MAINT
:PTF.UM28982 :STAT.APPLIED.09/01/98.23:52:25.MAINT
:PTF.UM28989 :STAT.APPLIED.09/01/98.23:52:25.MAINT
:PTF.UM28992 :STAT.APPLIED.09/01/98.23:52:25.MAINT
:PTF.UM29005 :STAT.APPLIED.09/01/98.23:52:25.MAINT
:PTF.UMRSU13 :STAT.SUPED.09/25/98.15:22:55.MAINT APPLIED.09/01/98.23:52:25.MAINT
:PTF.UM28994 :STAT.APPLIED.09/25/98.15:22:55.MAINT
:PTF.UM29020 :STAT.SUPED.11/03/98.21:27:10.MAINT APPLIED.09/25/98.15:22:55.MAINT
:PTF.UM29053 :STAT.APPLIED.09/25/98.15:22:55.MAINT
:PTF.UM29068 :STAT.APPLIED.09/25/98.15:22:55.MAINT
:PTF.UM29077 :STAT.APPLIED.09/25/98.15:22:55.MAINT
:PTF.UM29079 :STAT.APPLIED.09/25/98.15:22:55.MAINT
:PTF.UMRSU16 :STAT.SUPED.11/03/98.21:27:10.MAINT APPLIED.09/25/98.15:22:55.MAINT
:PTF.UM28829 :STAT.APPLIED.11/03/98.21:27:10.MAINT
:PTF.UM28968 :STAT.SUPED.11/03/98.21:27:10.MAINT
:PTF.UM28894 :STAT.SUPED.11/03/98.21:27:10.MAINT
:PTF.UM29044 :STAT.APPLIED.11/03/98.21:27:10.MAINT
:PTF.UM28833 :STAT.APPLIED.11/03/98.21:27:10.MAINT
:PTF.UM28847 :STAT.APPLIED.11/03/98.21:27:10.MAINT
:PTF.UM29062 :STAT.APPLIED.11/03/98.21:27:10.MAINT
:PTF.UM29081 :STAT.APPLIED.11/03/98.21:27:10.MAINT
:PTF.UM28794 :STAT.SUPED.11/03/98.21:27:10.MAINT
:PTF.UM29104 :STAT.APPLIED.11/03/98.21:27:10.MAINT
:PTF.UMRSU19 :STAT.SUPED.12/10/98.16:43:35.MAINT APPLIED.11/03/98.21:27:10.MAINT
:PTF.UM29049 :STAT.APPLIED.12/10/98.16:43:35.MAINT
:PTF.UM29098 :STAT.APPLIED.12/10/98.16:43:35.MAINT
:PTF.UM29107 :STAT.APPLIED.12/10/98.16:43:35.MAINT
:PTF.UMRSU22 :STAT.SUPED.01/26/99.14:40:04.MAINT APPLIED.12/10/98.16:43:35.MAINT
:PTF.UM29052 :STAT.SUPED.01/26/99.14:40:04.MAINT
:PTF.UM29121 :STAT.APPLIED.01/26/99.14:40:04.MAINT
:PTF.UM29126 :STAT.APPLIED.01/26/99.14:40:04.MAINT
:PTF.UM29142 :STAT.APPLIED.01/26/99.14:40:04.MAINT
:PTF.UMRSU25 :STAT.APPLIED.01/26/99.14:40:04.MAINT
```

J.2 CMS SRVAPPS

```
:PTF.UM28653 :STAT.SUPED.07/20/98.17:35:08.MAINT APPLIED.02/07/98.14:23:56.MAINT
:PTF.UM28667 :STAT.APPLIED.02/07/98.14:23:56.MAINT
:PTF.UM28661 :STAT.APPLIED.02/07/98.14:23:56.MAINT
:PTF.UM28663 :STAT.APPLIED.02/07/98.14:23:56.MAINT
:PTF.UM28662 :STAT.APPLIED.02/07/98.14:23:56.MAINT
:PTF.UM28664 :STAT.APPLIED.02/07/98.14:23:56.MAINT
:PTF.UM28659 :STAT.SUPED.09/25/98.15:03:16.MAINT APPLIED.02/07/98.14:23:56.MAINT
:PTF.UM28654 :STAT.APPLIED.02/07/98.14:23:56.MAINT
```

```
:PTF.UM28655 :STAT.SUPED.04/13/98.12:46:48.MAINT APPLIED.02/07/98.14:23:56.MAINT
:PTF.UM28668 :STAT.APPLIED.02/07/98.14:23:56.MAINT
:PTF.UM28660 :STAT.APPLIED.02/07/98.14:23:56.MAINT
:PTF.UM28658 :STAT.APPLIED.02/07/98.14:23:56.MAINT
:PTF.UMRSU00 :STAT.SUPED.04/13/98.12:46:48.MAINT APPLIED.02/07/98.14:23:56.MAINT
:PTF.UM28649 :STAT.SUPED.12/10/98.16:38:20.MAINT APPLIED.04/13/98.12:46:48.MAINT
:PTF.UM28672 :STAT.APPLIED.04/13/98.12:46:48.MAINT
:PTF.UM28675 :STAT.APPLIED.04/13/98.12:46:48.MAINT
:PTF.UM28679 :STAT.APPLIED.04/13/98.12:46:48.MAINT
:PTF.UM28689 :STAT.APPLIED.04/13/98.12:46:48.MAINT
:PTF.UM28678 :STAT.SUPED.04/13/98.12:46:48.MAINT
:PTF.UM28716 :STAT.APPLIED.04/13/98.12:46:48.MAINT
:PTF.UM28722 :STAT.APPLIED.04/13/98.12:46:48.MAINT
:PTF.UM28743 :STAT.APPLIED.04/13/98.12:46:48.MAINT
:PTF.UM28787 :STAT.APPLIED.04/13/98.12:46:48.MAINT
:PTF.UM28798 :STAT.APPLIED.04/13/98.12:46:48.MAINT
:PTF.UMRSU03 :STAT.SUPED.05/26/98.13:12:36.MAINT APPLIED.04/13/98.12:46:48.MAINT
:PTF.UM28687 :STAT.SUPED.01/26/99.14:34:31.MAINT APPLIED.05/26/98.13:12:36.MAINT
:PTF.UM28701 :STAT.APPLIED.05/26/98.13:12:36.MAINT
:PTF.UM28734 :STAT.APPLIED.05/26/98.13:12:36.MAINT
:PTF.UM28791 :STAT.APPLIED.05/26/98.13:12:36.MAINT
:PTF.UM28803 :STAT.SUPED.09/01/98.23:34:31.MAINT APPLIED.05/26/98.13:12:36.MAINT
:PTF.UM28808 :STAT.APPLIED.05/26/98.13:12:36.MAINT
:PTF.UM28813 :STAT.APPLIED.05/26/98.13:12:36.MAINT
:PTF.UM28824 :STAT.APPLIED.05/26/98.13:12:36.MAINT
:PTF.UM28852 :STAT.APPLIED.05/26/98.13:12:36.MAINT
:PTF.UMRSU06 :STAT.SUPED.07/20/98.17:35:08.MAINT APPLIED.05/26/98.13:12:36.MAINT
:PTF.UM28862 :STAT.SUPED.09/25/98.15:03:16.MAINT APPLIED.06/05/98.16:09:19.MAINT
:PTF.UM28838 :STAT.APPLIED.07/20/98.17:35:08.MAINT
:PTF.UM28849 :STAT.APPLIED.07/20/98.17:35:08.MAINT
:PTF.UM28854 :STAT.APPLIED.07/20/98.17:35:08.MAINT
:PTF.UM28822 :STAT.SUPED.07/20/98.17:35:08.MAINT
:PTF.UM28857 :STAT.APPLIED.07/20/98.17:35:08.MAINT
:PTF.UM28858 :STAT.APPLIED.07/20/98.17:35:08.MAINT
:PTF.UM28861 :STAT.APPLIED.07/20/98.17:35:08.MAINT
:PTF.UM28741 :STAT.APPLIED.07/20/98.17:35:08.MAINT
:PTF.UM28892 :STAT.SUPED.09/25/98.15:03:16.MAINT APPLIED.07/20/98.17:35:08.MAINT
:PTF.UM28912 :STAT.APPLIED.07/20/98.17:35:08.MAINT
:PTF.UM28930 :STAT.APPLIED.07/20/98.17:35:08.MAINT
:PTF.UM28935 :STAT.APPLIED.07/20/98.17:35:08.MAINT
:PTF.UM28941 :STAT.APPLIED.07/20/98.17:35:08.MAINT
:PTF.UMRSU09 :STAT.SUPED.09/01/98.23:34:31.MAINT APPLIED.07/20/98.17:35:08.MAINT
:PTF.UM28774 :STAT.APPLIED.09/01/98.23:34:31.MAINT
:PTF.UM28776 :STAT.APPLIED.09/01/98.23:34:31.MAINT
:PTF.UM28865 :STAT.APPLIED.09/01/98.23:34:31.MAINT
:PTF.UM29011 :STAT.APPLIED.09/01/98.23:34:31.MAINT
:PTF.UM29013 :STAT.APPLIED.09/01/98.23:34:31.MAINT
:PTF.UMRSU12 :STAT.SUPED.09/25/98.15:03:16.MAINT APPLIED.09/01/98.23:34:31.MAINT
:PTF.UM28947 :STAT.APPLIED.09/25/98.15:03:16.MAINT
:PTF.UM28999 :STAT.APPLIED.09/25/98.15:03:16.MAINT
:PTF.UM29016 :STAT.APPLIED.09/25/98.15:03:16.MAINT
:PTF.UM29018 :STAT.SUPED.01/26/99.14:34:31.MAINT APPLIED.09/25/98.15:03:16.MAINT
:PTF.UM29029 :STAT.APPLIED.09/25/98.15:03:16.MAINT
:PTF.UM28953 :STAT.SUPED.09/25/98.15:03:16.MAINT
:PTF.UM29036 :STAT.APPLIED.09/25/98.15:03:16.MAINT
:PTF.UM28937 :STAT.APPLIED.09/25/98.15:03:16.MAINT
:PTF.UM29046 :STAT.APPLIED.09/25/98.15:03:16.MAINT
:PTF.UM29069 :STAT.APPLIED.09/25/98.15:03:16.MAINT
:PTF.UM29075 :STAT.APPLIED.09/25/98.15:03:16.MAINT
:PTF.UMRSU15 :STAT.SUPED.11/03/98.21:15:50.MAINT APPLIED.09/25/98.15:03:16.MAINT
:PTF.UM28867 :STAT.APPLIED.11/03/98.21:15:50.MAINT
:PTF.UM28804 :STAT.SUPED.11/03/98.21:15:50.MAINT
:PTF.UM28737 :STAT.SUPED.11/03/98.21:15:50.MAINT
:PTF.UM28888 :STAT.APPLIED.11/03/98.21:15:50.MAINT
:PTF.UM28788 :STAT.SUPED.11/03/98.21:15:50.MAINT
:PTF.UM28908 :STAT.APPLIED.11/03/98.21:15:50.MAINT
:PTF.UM29074 :STAT.APPLIED.11/03/98.21:15:50.MAINT
:PTF.UM29022 :STAT.SUPED.11/03/98.21:15:50.MAINT
:PTF.UM29086 :STAT.APPLIED.11/03/98.21:15:50.MAINT
:PTF.UM29088 :STAT.SUPED.12/10/98.16:38:20.MAINT APPLIED.11/03/98.21:15:50.MAINT
:PTF.UMRSU18 :STAT.SUPED.12/10/98.16:38:20.MAINT APPLIED.11/03/98.21:15:50.MAINT
:PTF.UM29090 :STAT.APPLIED.12/10/98.16:38:20.MAINT
:PTF.UM29093 :STAT.APPLIED.12/10/98.16:38:20.MAINT
:PTF.UM29094 :STAT.APPLIED.12/10/98.16:38:20.MAINT
:PTF.UM29100 :STAT.APPLIED.12/10/98.16:38:20.MAINT
:PTF.UM29112 :STAT.SUPED.01/26/99.14:34:31.MAINT APPLIED.12/10/98.16:38:20.MAINT
:PTF.UM29120 :STAT.APPLIED.12/10/98.16:38:20.MAINT
:PTF.UMRSU21 :STAT.SUPED.01/26/99.14:34:31.MAINT APPLIED.12/10/98.16:38:20.MAINT
```

```
:PTF.UM29123 :STAT.APPLIED.01/26/99.14:34:31.MAINT
:PTF.UM29138 :STAT.APPLIED.01/26/99.14:34:31.MAINT
:PTF.UM29133 :STAT.SUPED.01/26/99.14:34:31.MAINT
:PTF.UM29116 :STAT.SUPED.01/26/99.14:34:31.MAINT
:PTF.UM29165 :STAT.APPLIED.01/26/99.14:34:31.MAINT
:PTF.UM29184 :STAT.APPLIED.01/26/99.14:34:31.MAINT
:PTF.UMRSU24 :STAT.APPLIED.01/26/99.14:34:31.MAINT
:PTF.UM28990 :STAT.SUPED.02/23/99.07:45:36.230MAINT
:PTF.UM29113 :STAT.APPLIED.02/23/99.07:45:36.230MAINT
:PTF.UM28942 :STAT.SUPED.04/14/99.09:43:07.230MAINT
:PTF.UM29177 :STAT.APPLIED.04/14/99.09:43:07.230MAINT
:PTF.UM29258 :STAT.SUPED.04/14/99.09:44:53.230MAINT
:PTF.UM29255 :STAT.SUPED.04/14/99.09:44:53.230MAINT
:PTF.UM29259 :STAT.APPLIED.04/14/99.09:44:53.230MAINT
```

J.3 Shell and Utilities SRVAPPS

```
:PTF.UM28810 :STAT.APPLIED.10/28/98.17:01:22.MAINT
:PTF.UM28916 :STAT.APPLIED.10/28/98.17:01:22.MAINT
:PTF.UM28914 :STAT.SUPED.10/28/98.17:01:22.MAINT
:PTF.UM29071 :STAT.APPLIED.10/28/98.17:01:22.MAINT
:PTF.UM29270 :STAT.APPLIED.04/09/99.11:59:23.2VMVMZ30
:PTF.UM29287 :STAT.APPLIED.04/09/99.12:02:39.2VMVMZ30
:PTF.UM29256 :STAT.APPLIED.04/09/99.12:04:17.2VMVMZ30
```

J.4 LE/370 SRVAPPS

```
:PTF.UQ04618 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ05023 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ05029 :STAT.SUPED.12/16/98.03:47:11.MAINT APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ05078 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ05167 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ05575 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ06210 :STAT.SUPED.05/22/98.12:34:24.MAINT APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ06776 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ06788 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ04381 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ07158 :STAT.SUPED.05/22/98.12:34:24.MAINT APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ07162 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ07148 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ06197 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ07732 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ07733 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ07746 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ08182 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ08337 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ08354 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ08360 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ08393 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ08565 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ08659 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ08689 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ09121 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ09192 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ09555 :STAT.SUPED.04/14/98.01:39:39.MAINT APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ09554 :STAT.SUPED.04/14/98.01:39:39.MAINT APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ09553 :STAT.SUPED.04/14/98.01:39:39.MAINT APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ09970 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ10072 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ10257 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ10509 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ10749 :STAT.SUPED.04/14/98.01:39:39.MAINT APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ10457 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ10591 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ10926 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ10953 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ11052 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ11408 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ05070 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ05165 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ05231 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ05325 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ05939 :STAT.SUPED.12/16/98.03:47:11.MAINT APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ07550 :STAT.APPLIED.01/02/98.11:26:34.P688198H
```

```
:PTF.UQ08186 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ04825 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ08427 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ05817 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ04821 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ04621 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ08426 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ08505 :STAT.SUPED.04/14/98.01:39:39.MAINT APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ08512 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ08342 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ08537 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ08569 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ09128 :STAT.SUPED.02/22/99.09:59:44.P688198H APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ08169 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ08786 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ08965 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ08989 :STAT.SUPED.04/14/98.01:39:39.MAINT APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ08993 :STAT.SUPED.04/14/98.01:39:39.MAINT APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ09038 :STAT.SUPED.05/22/98.12:34:24.MAINT APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ09130 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ09114 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ09048 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ08729 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ08362 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ07047 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ06206 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ05049 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ09247 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ09261 :STAT.SUPED.04/14/98.01:39:39.MAINT APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ09270 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ05083 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ09273 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ09325 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ08969 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ09282 :STAT.SUPED.04/14/98.01:39:39.MAINT APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ08979 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ09350 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ07430 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ07052 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ09499 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ10033 :STAT.SUPED.02/09/98.03:50:28.MAINT APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ08893 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ10093 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ10069 :STAT.SUPED.04/14/98.01:39:39.MAINT APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ10074 :STAT.SUPED.05/22/98.12:34:24.MAINT APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ10103 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ10412 :STAT.SUPED.04/14/98.01:39:39.MAINT APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ10424 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ10460 :STAT.SUPED.02/09/98.03:50:28.MAINT APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ04803 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ05021 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ06364 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ06618 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ10605 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ08375 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ10730 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ10054 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ09868 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ05829 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ04810 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ10806 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ10787 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ10807 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ11384 :STAT.SUPED.02/09/98.03:50:28.MAINT APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ10108 :STAT.SUPED.01/02/98.11:26:34.P688198H
:PTF.UQ11435 :STAT.APPLIED.01/02/98.11:26:34.P688198H
:PTF.UQ11552 :STAT.APPLIED.02/09/98.03:50:28.MAINT
:PTF.UQ11569 :STAT.SUPED.02/09/98.03:50:28.MAINT
:PTF.UQ11564 :STAT.APPLIED.02/09/98.03:50:28.MAINT
:PTF.UQ11850 :STAT.APPLIED.02/09/98.03:50:28.MAINT
:PTF.UQ11800 :STAT.SUPED.02/09/98.03:50:28.MAINT
:PTF.UQ12023 :STAT.APPLIED.02/09/98.03:50:28.MAINT
:PTF.UQ12675 :STAT.APPLIED.02/09/98.03:50:28.MAINT
:PTF.UQ12840 :STAT.SUPED.02/09/98.03:50:28.MAINT
:PTF.UQ12831 :STAT.SUPED.02/09/98.03:50:28.MAINT
:PTF.UQ11686 :STAT.SUPED.02/09/98.03:50:28.MAINT
:PTF.UQ13158 :STAT.APPLIED.02/09/98.03:50:28.MAINT
:PTF.UQ13617 :STAT.SUPED.09/29/98.08:45:22.MAINT APPLIED.02/09/98.03:50:28.MAINT
```

```
:PTF.UQ11697 :STAT.SUPED.02/09/98.03:50:28.MAINT
:PTF.UQ13890 :STAT.APPLIED.02/09/98.03:50:28.MAINT
:PTF.UQ13882 :STAT.APPLIED.02/09/98.03:50:28.MAINT
:PTF.UQ14115 :STAT.SUPED.09/29/98.08:45:22.MAINT APPLIED.02/09/98.03:50:28.MAINT
:PTF.UQ11749 :STAT.APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ11840 :STAT.APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ11880 :STAT.APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ11882 :STAT.APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ12249 :STAT.APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ12400 :STAT.APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ12461 :STAT.APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ13024 :STAT.APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ13234 :STAT.APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ13241 :STAT.APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ13302 :STAT.APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ13394 :STAT.APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ13723 :STAT.APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ13745 :STAT.APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ13818 :STAT.APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ13878 :STAT.SUPED.09/01/98.17:31:22.MAINT APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ12909 :STAT.SUPED.04/14/98.01:39:39.MAINT
:PTF.UQ14176 :STAT.APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ14339 :STAT.SUPED.12/16/98.03:47:11.MAINT APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ12003 :STAT.SUPED.04/14/98.01:39:39.MAINT
:PTF.UQ14493 :STAT.APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ15015 :STAT.SUPED.11/04/98.12:45:42.MAINT APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ12509 :STAT.SUPED.04/14/98.01:39:39.MAINT
:PTF.UQ11648 :STAT.SUPED.04/14/98.01:39:39.MAINT
:PTF.UQ15135 :STAT.APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ13484 :STAT.SUPED.04/14/98.01:39:39.MAINT
:PTF.UQ13483 :STAT.SUPED.04/14/98.01:39:39.MAINT
:PTF.UQ13482 :STAT.APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ15161 :STAT.APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ15160 :STAT.APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ15159 :STAT.APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ15258 :STAT.SUPED.12/16/98.03:47:11.MAINT APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ13163 :STAT.SUPED.04/14/98.01:39:39.MAINT
:PTF.UQ15274 :STAT.APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ13017 :STAT.SUPED.04/14/98.01:39:39.MAINT
:PTF.UQ15268 :STAT.APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ15740 :STAT.SUPED.07/20/98.13:12:44.MAINT APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ15803 :STAT.SUPED.05/22/98.12:34:24.MAINT APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ15850 :STAT.SUPED.06/04/98.12:35:34.MAINT APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ15885 :STAT.SUPED.07/20/98.13:12:44.MAINT APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ15028 :STAT.SUPED.04/14/98.01:39:39.MAINT
:PTF.UQ16336 :STAT.SUPED.02/22/99.09:59:44.P688198H APPLIED.04/14/98.01:39:39.MAINT
:PTF.UQ09386 :STAT.SUPED.05/22/98.12:34:24.MAINT
:PTF.UQ11608 :STAT.APPLIED.05/22/98.12:34:24.MAINT
:PTF.UQ11499 :STAT.APPLIED.05/22/98.12:34:24.MAINT
:PTF.UQ16725 :STAT.APPLIED.05/22/98.12:34:24.MAINT
:PTF.UQ16869 :STAT.APPLIED.05/22/98.12:34:24.MAINT
:PTF.UQ14471 :STAT.SUPED.05/22/98.12:34:24.MAINT
:PTF.UQ15516 :STAT.APPLIED.05/22/98.12:34:24.MAINT
:PTF.UQ16827 :STAT.APPLIED.05/22/98.12:34:24.MAINT
:PTF.UQ17075 :STAT.APPLIED.05/22/98.12:34:24.MAINT
:PTF.UQ17253 :STAT.SUPED.09/01/98.17:31:22.MAINT APPLIED.05/22/98.12:34:24.MAINT
:PTF.UQ17340 :STAT.APPLIED.05/22/98.12:34:24.MAINT
:PTF.UQ17393 :STAT.APPLIED.05/22/98.12:34:24.MAINT
:PTF.UQ11671 :STAT.APPLIED.05/22/98.12:34:24.MAINT
:PTF.UQ13099 :STAT.APPLIED.05/22/98.12:34:24.MAINT
:PTF.UQ15753 :STAT.APPLIED.05/22/98.12:34:24.MAINT
:PTF.UQ16721 :STAT.APPLIED.05/22/98.12:34:24.MAINT
:PTF.UQ15031 :STAT.SUPED.05/22/98.12:34:24.MAINT
:PTF.UQ17053 :STAT.APPLIED.05/22/98.12:34:24.MAINT
:PTF.UQ17059 :STAT.APPLIED.05/22/98.12:34:24.MAINT
:PTF.UQ17054 :STAT.APPLIED.05/22/98.12:34:24.MAINT
:PTF.UQ17479 :STAT.SUPED.09/01/98.17:31:22.MAINT APPLIED.05/22/98.12:34:24.MAINT
:PTF.UQ16332 :STAT.SUPED.05/22/98.12:34:24.MAINT
:PTF.UQ17544 :STAT.APPLIED.05/22/98.12:34:24.MAINT
:PTF.UQ17624 :STAT.SUPED.12/16/98.03:47:11.MAINT APPLIED.05/22/98.12:34:24.MAINT
:PTF.UQ17740 :STAT.APPLIED.05/22/98.12:34:24.MAINT
:PTF.UQ17798 :STAT.SUPED.09/01/98.17:31:22.MAINT APPLIED.05/22/98.12:34:24.MAINT
:PTF.UQ18044 :STAT.APPLIED.06/04/98.12:35:34.MAINT
:PTF.UQ18668 :STAT.SUPED.12/16/98.03:47:11.MAINT APPLIED.06/11/98.16:20:28.MAINT
:PTF.UQ14962 :STAT.APPLIED.07/20/98.13:12:44.MAINT
:PTF.UQ15320 :STAT.APPLIED.07/20/98.13:12:44.MAINT
:PTF.UQ14727 :STAT.APPLIED.07/20/98.13:12:44.MAINT
:PTF.UQ16155 :STAT.APPLIED.07/20/98.13:12:44.MAINT
```

```
:PTF.UQ16154 :STAT.APPLIED.07/20/98.13:12:44.MAINT
:PTF.UQ16153 :STAT.APPLIED.07/20/98.13:12:44.MAINT
:PTF.UQ18307 :STAT.APPLIED.07/20/98.13:12:44.MAINT
:PTF.UQ18407 :STAT.APPLIED.07/20/98.13:12:44.MAINT
:PTF.UQ18553 :STAT.APPLIED.07/20/98.13:12:44.MAINT
:PTF.UQ18703 :STAT.SUPED.07/20/98.13:12:44.MAINT
:PTF.UQ17017 :STAT.SUPED.07/20/98.13:12:44.MAINT
:PTF.UQ19020 :STAT.APPLIED.07/20/98.13:12:44.MAINT
:PTF.UQ19084 :STAT.APPLIED.07/20/98.13:12:44.MAINT
:PTF.UQ18082 :STAT.SUPED.07/20/98.13:12:44.MAINT
:PTF.UQ19230 :STAT.APPLIED.07/20/98.13:12:44.MAINT
:PTF.UQ19184 :STAT.APPLIED.09/01/98.17:31:22.MAINT
:PTF.UQ11654 :STAT.SUPED.09/01/98.17:31:22.MAINT
:PTF.UQ10225 :STAT.SUPED.09/01/98.17:31:22.MAINT
:PTF.UQ10223 :STAT.SUPED.09/01/98.17:31:22.MAINT
:PTF.UQ19629 :STAT.APPLIED.09/01/98.17:31:22.MAINT
:PTF.UQ19697 :STAT.APPLIED.09/01/98.17:31:22.MAINT
:PTF.UQ19724 :STAT.APPLIED.09/01/98.17:31:22.MAINT
:PTF.UQ19822 :STAT.APPLIED.09/01/98.17:31:22.MAINT
:PTF.UQ13456 :STAT.SUPED.09/01/98.17:31:22.MAINT
:PTF.UQ19809 :STAT.APPLIED.09/01/98.17:31:22.MAINT
:PTF.UQ20197 :STAT.APPLIED.09/01/98.17:31:22.MAINT
:PTF.UQ14718 :STAT.APPLIED.09/01/98.17:31:22.MAINT
:PTF.UQ14904 :STAT.SUPED.09/01/98.17:31:22.MAINT
:PTF.UQ20280 :STAT.APPLIED.09/01/98.17:31:22.MAINT
:PTF.UQ16068 :STAT.APPLIED.09/29/98.08:45:22.MAINT
:PTF.UQ16952 :STAT.APPLIED.09/29/98.08:45:22.MAINT
:PTF.UQ20355 :STAT.APPLIED.09/29/98.08:45:22.MAINT
:PTF.UQ20649 :STAT.APPLIED.09/29/98.08:45:22.MAINT
:PTF.UQ20747 :STAT.APPLIED.09/29/98.08:45:22.MAINT
:PTF.UQ21256 :STAT.APPLIED.09/29/98.08:45:22.MAINT
:PTF.UQ21191 :STAT.APPLIED.11/04/98.12:45:42.MAINT
:PTF.UQ21264 :STAT.APPLIED.11/04/98.12:45:42.MAINT
:PTF.UQ21401 :STAT.APPLIED.11/04/98.12:45:42.MAINT
:PTF.UQ21430 :STAT.APPLIED.11/04/98.12:45:42.MAINT
:PTF.UQ21610 :STAT.APPLIED.11/04/98.12:45:42.MAINT
:PTF.UQ17440 :STAT.SUPED.11/04/98.12:45:42.MAINT
:PTF.UQ21733 :STAT.APPLIED.11/04/98.12:45:42.MAINT
:PTF.UQ22127 :STAT.APPLIED.11/04/98.12:45:42.MAINT
:PTF.UQ20754 :STAT.SUPED.11/04/98.12:45:42.MAINT
:PTF.UQ22258 :STAT.APPLIED.11/04/98.12:45:42.MAINT
:PTF.UQ22366 :STAT.APPLIED.11/04/98.12:45:42.MAINT
:PTF.UQ19081 :STAT.SUPED.11/04/98.12:45:42.MAINT
:PTF.UQ22658 :STAT.APPLIED.11/04/98.12:45:42.MAINT
:PTF.UQ22139 :STAT.APPLIED.12/16/98.03:47:11.MAINT
:PTF.UQ15022 :STAT.SUPED.12/16/98.03:47:11.MAINT
:PTF.UQ13201 :STAT.SUPED.12/16/98.03:47:11.MAINT
:PTF.UQ13197 :STAT.SUPED.12/16/98.03:47:11.MAINT
:PTF.UQ19132 :STAT.APPLIED.12/16/98.03:47:11.MAINT
:PTF.UQ22647 :STAT.APPLIED.12/16/98.03:47:11.MAINT
:PTF.UQ21349 :STAT.SUPED.12/16/98.03:47:11.MAINT
:PTF.UQ21292 :STAT.SUPED.12/16/98.03:47:11.MAINT
:PTF.UQ21380 :STAT.APPLIED.12/16/98.03:47:11.MAINT
:PTF.UQ21914 :STAT.APPLIED.12/16/98.03:47:11.MAINT
:PTF.UQ21937 :STAT.APPLIED.12/16/98.03:47:11.MAINT
:PTF.UQ22149 :STAT.SUPED.12/16/98.03:47:11.MAINT
:PTF.UQ22131 :STAT.SUPED.12/16/98.03:47:11.MAINT
:PTF.UQ22941 :STAT.SUPED.12/16/98.03:47:11.MAINT
:PTF.UQ22374 :STAT.SUPED.12/16/98.03:47:11.MAINT
:PTF.UQ22929 :STAT.APPLIED.12/16/98.03:47:11.MAINT
:PTF.UQ17849 :STAT.SUPED.12/16/98.03:47:11.MAINT
:PTF.UQ23162 :STAT.SUPED.02/22/99.09:59:44.P688198H APPLIED.12/16/98.03:47:11.MAINT
:PTF.UQ22567 :STAT.SUPED.12/16/98.03:47:11.MAINT
:PTF.UQ23574 :STAT.APPLIED.12/16/98.03:47:11.MAINT
:PTF.UQ23707 :STAT.APPLIED.12/16/98.03:47:11.MAINT
:PTF.UQ23631 :STAT.SUPED.12/16/98.03:47:11.MAINT
:PTF.UQ23801 :STAT.SUPED.02/23/99.15:30:52.P688198H APPLIED.12/16/98.03:47:11.MAINT
:PTF.UQ24272 :STAT.APPLIED.02/22/99.09:59:44.P688198H
:PTF.UQ22435 :STAT.APPLIED.02/22/99.09:59:44.P688198H
:PTF.UQ22623 :STAT.APPLIED.02/22/99.09:59:44.P688198H
:PTF.UQ17945 :STAT.SUPED.02/22/99.09:59:44.P688198H
:PTF.UQ21785 :STAT.APPLIED.02/22/99.09:59:44.P688198H
:PTF.UQ24360 :STAT.APPLIED.02/23/99.15:30:52.P688198H
```

J.5 RSCS SRVAPPS

```
:PTF.UV59323 :STAT.SUPED.04/25/96.14:59:48.RSCS32GA APPLIED.01/11/96.15:05:45.RSCS32GA
:PTF.UV59325 :STAT.APPLIED.01/11/96.15:05:45.RSCS32GA
:PTF.UV59359 :STAT.SUPED.04/25/96.14:59:48.RSCS32GA APPLIED.01/11/96.15:05:45.RSCS32GA
:PTF.UVRSU01 :STAT.SUPED.04/25/96.14:59:48.RSCS32GA APPLIED.01/11/96.15:05:45.RSCS32GA
:PTF.UV59386 :STAT.APPLIED.04/25/96.14:59:48.RSCS32GA
:PTF.UV59438 :STAT.APPLIED.04/25/96.14:59:48.RSCS32GA
:PTF.UV59450 :STAT.SUPED.04/25/96.14:59:48.RSCS32GA
:PTF.UV59413 :STAT.SUPED.04/25/96.14:59:48.RSCS32GA
:PTF.UV59462 :STAT.APPLIED.04/25/96.14:59:48.RSCS32GA
:PTF.UV59482 :STAT.SUPED.05/27/98.00:58:24.MAINT APPLIED.04/25/96.14:59:48.RSCS32GA
:PTF.UV59483 :STAT.APPLIED.04/25/96.14:59:48.RSCS32GA
:PTF.UV59512 :STAT.APPLIED.04/25/96.14:59:48.RSCS32GA
:PTF.UV59523 :STAT.SUPED.07/24/96.15:11:12.RSCS32GA APPLIED.04/25/96.14:59:48.RSCS32GA
:PTF.UV59533 :STAT.APPLIED.04/25/96.14:59:48.RSCS32GA
:PTF.UV59543 :STAT.APPLIED.04/25/96.14:59:48.RSCS32GA
:PTF.UV59567 :STAT.APPLIED.04/25/96.14:59:48.RSCS32GA
:PTF.UV59548 :STAT.SUPED.04/25/96.14:59:48.RSCS32GA
:PTF.UV59572 :STAT.APPLIED.04/25/96.14:59:48.RSCS32GA
:PTF.UV59570 :STAT.SUPED.04/25/96.14:59:48.RSCS32GA
:PTF.UV59536 :STAT.SUPED.04/25/96.14:59:48.RSCS32GA
:PTF.UV59510 :STAT.SUPED.04/25/96.14:59:48.RSCS32GA
:PTF.UV59577 :STAT.APPLIED.04/25/96.14:59:48.RSCS32GA
:PTF.UV59588 :STAT.SUPED.04/08/97.20:18:44.RSCS32GA APPLIED.04/25/96.14:59:48.RSCS32GA
:PTF.UVRSU03 :STAT.SUPED.07/24/96.15:11:12.RSCS32GA APPLIED.04/25/96.14:59:48.RSCS32GA
:PTF.UV59392 :STAT.SUPED.04/08/97.20:18:44.RSCS32GA APPLIED.07/24/96.15:11:12.RSCS32GA
:PTF.UV59613 :STAT.APPLIED.07/24/96.15:11:12.RSCS32GA
:PTF.UV59618 :STAT.SUPED.04/08/97.20:18:44.RSCS32GA APPLIED.07/24/96.15:11:12.RSCS32GA
:PTF.UV59622 :STAT.SUPED.07/24/96.15:11:12.RSCS32GA
:PTF.UV59635 :STAT.SUPED.12/12/96.11:39:47.RSCS32GA APPLIED.07/24/96.15:11:12.RSCS32GA
:PTF.UV59624 :STAT.SUPED.07/24/96.15:11:12.RSCS32GA
:PTF.UV59671 :STAT.SUPED.04/08/97.20:18:44.RSCS32GA APPLIED.07/24/96.15:11:12.RSCS32GA
:PTF.UV59672 :STAT.APPLIED.07/24/96.15:11:12.RSCS32GA
:PTF.UVRSU05 :STAT.SUPED.12/13/96.13:04:09.RSCS32GA APPLIED.07/24/96.15:11:12.RSCS32GA
:PTF.UV59632 :STAT.APPLIED.12/12/96.11:39:47.RSCS32GA
:PTF.UV59642 :STAT.SUPED.12/12/96.11:39:47.RSCS32GA
:PTF.UV59715 :STAT.SUPED.02/04/98.17:36:14.MAINT APPLIED.12/12/96.11:39:47.RSCS32GA
:PTF.UV59741 :STAT.APPLIED.12/12/96.11:39:47.RSCS32GA
:PTF.UV59762 :STAT.APPLIED.12/12/96.11:39:47.RSCS32GA
:PTF.UV59768 :STAT.APPLIED.12/12/96.11:39:47.RSCS32GA
:PTF.UV59757 :STAT.SUPED.12/12/96.11:39:47.RSCS32GA
:PTF.UV59820 :STAT.SUPED.02/04/98.17:36:14.MAINT APPLIED.12/12/96.11:39:47.RSCS32GA
:PTF.UVRSU07 :STAT.SUPED.04/08/97.20:18:44.RSCS32GA APPLIED.12/13/96.13:04:09.RSCS32GA
:PTF.UV59840 :STAT.SUPED.02/04/98.17:36:14.MAINT APPLIED.04/08/97.20:18:44.RSCS32GA
:PTF.UV59784 :STAT.SUPED.04/08/97.20:18:44.RSCS32GA
:PTF.UV59858 :STAT.APPLIED.04/08/97.20:18:44.RSCS32GA
:PTF.UV59867 :STAT.SUPED.02/04/98.17:36:14.MAINT APPLIED.04/08/97.20:18:44.RSCS32GA
:PTF.UV59868 :STAT.APPLIED.04/08/97.20:18:44.RSCS32GA
:PTF.UV59882 :STAT.APPLIED.04/08/97.20:18:44.RSCS32GA
:PTF.UV59897 :STAT.SUPED.02/04/98.17:36:14.MAINT APPLIED.04/08/97.20:18:44.RSCS32GA
:PTF.UV59937 :STAT.APPLIED.04/08/97.20:18:44.RSCS32GA
:PTF.UV59949 :STAT.APPLIED.04/08/97.20:18:44.RSCS32GA
:PTF.UVRSU09 :STAT.SUPED.02/04/98.17:36:14.MAINT APPLIED.04/08/97.20:18:44.RSCS32GA
:PTF.UV59912 :STAT.APPLIED.02/04/98.17:36:14.MAINT
:PTF.UV59919 :STAT.APPLIED.02/04/98.17:36:14.MAINT
:PTF.UV59921 :STAT.SUPED.02/04/98.17:36:14.MAINT
:PTF.UV60001 :STAT.APPLIED.02/04/98.17:36:14.MAINT
:PTF.UV60069 :STAT.APPLIED.02/04/98.17:36:14.MAINT
:PTF.UV59971 :STAT.APPLIED.02/04/98.17:36:14.MAINT
:PTF.UV60079 :STAT.APPLIED.02/04/98.17:36:14.MAINT
:PTF.UV60095 :STAT.APPLIED.02/04/98.17:36:14.MAINT
:PTF.UV60056 :STAT.SUPED.02/04/98.17:36:14.MAINT
:PTF.UV60040 :STAT.SUPED.02/04/98.17:36:14.MAINT
:PTF.UV60039 :STAT.SUPED.02/04/98.17:36:14.MAINT
:PTF.UV59969 :STAT.SUPED.02/04/98.17:36:14.MAINT
:PTF.UV60104 :STAT.APPLIED.02/04/98.17:36:14.MAINT
:PTF.UV60106 :STAT.APPLIED.02/04/98.17:36:14.MAINT
:PTF.UVRSU11 :STAT.SUPED.05/27/98.00:58:24.MAINT APPLIED.02/04/98.17:36:14.MAINT
:PTF.UV60097 :STAT.APPLIED.05/27/98.00:58:24.MAINT
:PTF.UV60109 :STAT.SUPED.05/27/98.00:58:24.MAINT
:PTF.UV60115 :STAT.APPLIED.05/27/98.00:58:24.MAINT
:PTF.UV59910 :STAT.SUPED.05/27/98.00:58:24.MAINT
:PTF.UV60135 :STAT.APPLIED.05/27/98.00:58:24.MAINT
:PTF.UV60161 :STAT.APPLIED.05/27/98.00:58:24.MAINT
:PTF.UV60166 :STAT.APPLIED.05/27/98.00:58:24.MAINT
:PTF.UV60173 :STAT.APPLIED.05/27/98.00:58:24.MAINT
```

```
:PTF.UVRSU13 :STAT.SUPED.11/04/98.12:40:08.MAINT APPLIED.05/27/98.00:58:24.MAINT
:PTF.UV60250 :STAT.APPLIED.11/04/98.12:40:08.MAINT
:PTF.UV60262 :STAT.APPLIED.11/04/98.12:40:08.MAINT
:PTF.UV60282 :STAT.APPLIED.11/04/98.12:40:08.MAINT
:PTF.UVRSU15 :STAT.APPLIED.11/04/98.12:40:08.MAINT
```

J.6 C/VM 3.1 SRVAPPS

```
:PTF.UN85704 :STAT.SUPED.09/12/97.10:13:55.P654033A APPLIED.02/08/96.17:38:35.P654033A
:PTF.UQ05262 :STAT.SUPED.09/12/97.10:13:55.P654033A
:PTF.UQ05264 :STAT.APPLIED.09/12/97.10:13:55.P654033A
:PTF.UQ05608 :STAT.APPLIED.09/12/97.10:13:55.P654033A
```

J.7 TCP/IP FL310 SRVAPPS

```
:PTF.UQ14371 :STAT.APPLIED.02/08/98.18:07:35.MAINT
:PTF.UQ14304 :STAT.APPLIED.02/08/98.18:07:35.MAINT
:PTF.UQ14326 :STAT.APPLIED.02/08/98.18:07:35.MAINT
:PTF.UQ14296 :STAT.APPLIED.02/08/98.18:07:35.MAINT
:PTF.UQ14340 :STAT.APPLIED.02/08/98.18:07:35.MAINT
:PTF.UQ14297 :STAT.APPLIED.02/08/98.18:07:35.MAINT
:PTF.UQ14328 :STAT.APPLIED.02/08/98.18:07:35.MAINT
:PTF.UQRSU01 :STAT.SUPED.04/13/98.17:16:37.MAINT APPLIED.02/08/98.18:07:35.MAINT
:PTF.UQ14530 :STAT.APPLIED.04/13/98.17:16:37.MAINT
:PTF.UQ15832 :STAT.APPLIED.04/13/98.17:16:37.MAINT
:PTF.UQ15833 :STAT.APPLIED.04/13/98.17:16:37.MAINT
:PTF.UQ16101 :STAT.APPLIED.04/13/98.17:16:37.MAINT
:PTF.UQRSU02 :STAT.SUPED.06/09/98.16:36:23.MAINT APPLIED.04/13/98.17:16:37.MAINT
:PTF.UQ16441 :STAT.APPLIED.06/09/98.16:36:23.MAINT
:PTF.UQ17557 :STAT.APPLIED.06/09/98.16:36:23.MAINT
:PTF.UQ16551 :STAT.APPLIED.06/09/98.16:36:23.MAINT
:PTF.UQ17712 :STAT.APPLIED.06/09/98.16:36:23.MAINT
:PTF.UQ18436 :STAT.APPLIED.06/09/98.16:36:23.MAINT
:PTF.UQ18443 :STAT.APPLIED.06/09/98.16:36:23.MAINT
:PTF.UQRSU03 :STAT.SUPED.09/01/98.17:17:11.MAINT APPLIED.06/09/98.16:36:23.MAINT
:PTF.UQ16409 :STAT.APPLIED.09/01/98.17:17:11.MAINT
:PTF.UQ16241 :STAT.APPLIED.09/01/98.17:17:11.MAINT
:PTF.UQ16430 :STAT.APPLIED.09/01/98.17:17:11.MAINT
:PTF.UQ17221 :STAT.APPLIED.09/01/98.17:17:11.MAINT
:PTF.UQ18012 :STAT.APPLIED.09/01/98.17:17:11.MAINT
:PTF.UQ18488 :STAT.APPLIED.09/01/98.17:17:11.MAINT
:PTF.UQ18799 :STAT.APPLIED.09/01/98.17:17:11.MAINT
:PTF.UQ18801 :STAT.APPLIED.09/01/98.17:17:11.MAINT
:PTF.UQ16766 :STAT.APPLIED.09/01/98.17:17:11.MAINT
:PTF.UQ17066 :STAT.APPLIED.09/01/98.17:17:11.MAINT
:PTF.UQ17202 :STAT.APPLIED.09/01/98.17:17:11.MAINT
:PTF.UQ17430 :STAT.APPLIED.09/01/98.17:17:11.MAINT
:PTF.UQ19569 :STAT.APPLIED.09/01/98.17:17:11.MAINT
:PTF.UQ20449 :STAT.APPLIED.09/01/98.17:17:11.MAINT
:PTF.UQRSU04 :STAT.APPLIED.09/01/98.17:17:11.MAINT
:PTF.UQ16552 :STAT.APPLIED.10/08/98.08:55:11.P735FALQ
:PTF.UQ17932 :STAT.APPLIED.10/08/98.08:55:11.P735FALQ
:PTF.UQ19568 :STAT.APPLIED.10/08/98.08:55:11.P735FALQ
:PTF.UQ20817 :STAT.APPLIED.10/08/98.08:55:11.P735FALQ
:PTF.UQ19055 :STAT.APPLIED.02/26/99.10:07:16.P735FALQ
:PTF.UQ22841 :STAT.APPLIED.02/26/99.10:07:16.P735FALQ
```

Appendix K. General instructions for downloading packages

To obtain the packages, you can download them off the accompanying CD-ROM or from the following sites:

`http://pucc.princeton.edu/~neale/vmoe.html`

Alternatively, look at the following web site for a readme text file. This file directs you to many additional web sites for downloading the tools and programs.

`ftp://www.redbooks.ibm.com/redbooks/SG245458`

The process for getting an application from an archive source into production is illustrated in Figure 201:

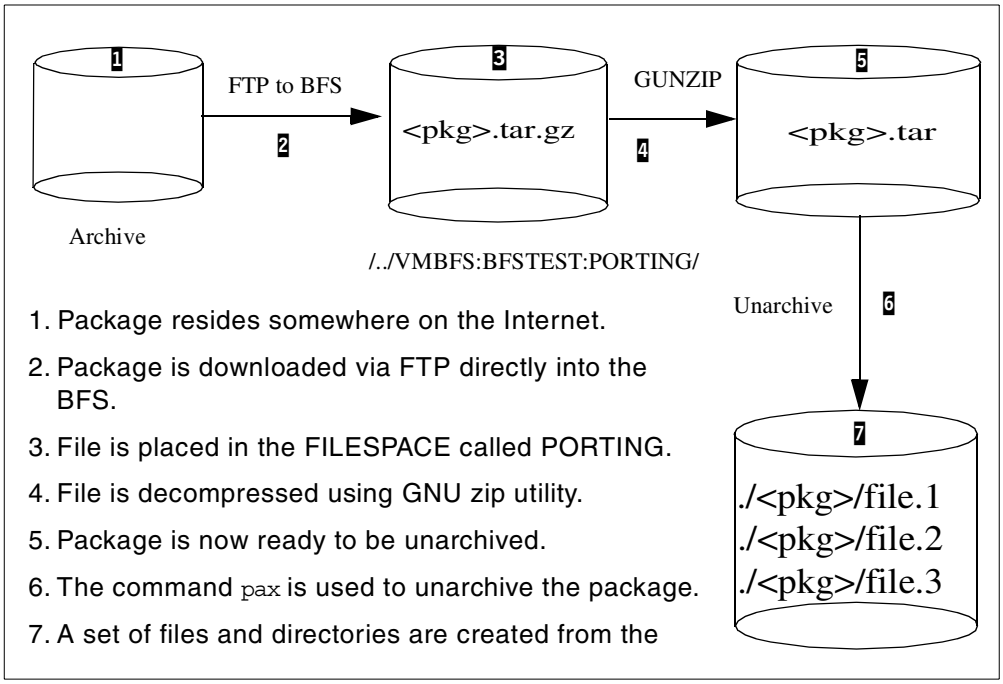


Figure 201. Overview of download process

K.1 Package names

By convention, we use an uppercase name when we speak about a package. During the residency, we worked on the following packages:

- APACHE: A Web server
- BYACC: A parser more powerful than the OpenEdition for VM/ESA `yacc`
- DAYTIME: A daytime server
- DCRON: A server to start commands at the appropriate time
- GNUDIFF: Tools to compare files
- FLEX: A faster LEXical analyzer than the OpenEdition for VM/ESA `lex`
- GDBM: A database manager
- GZIP: A tool to compress file

- INETD: A multithreaded server for Internet connection
- INN: A newsgroup server
- IRC: A chat server
- JACORB: A JAVA implementation of CORBA
- LDAP: A directory service for your Web browser
- LIBASCII: Functions to improve character translation from ASCII to EBCDIC or from EBCDIC to ASCII
- MAKE: The GNU `make` with more functions than the OpenEdition for VM/ESA `make`
- MQM: A JAVA MQseries client
- RCS: A tool to manage multiple revision of files
- REGINA: A REXX running under OpenEdition for VM/ESA
- SAMBA: A Common Internet File System between UNIX, Windows, OS/2, MVS, VMS, Apple Macs and so on
- SYSLOGD: A server to route messages, to consolidate logs from different platforms and to distribute specific messages

K.2 Format of the packages

The packages are available on the joined CD-ROM and they are also available on the Internet. Multiple formats are used to store the packages. The major package formats and extensions used to name package files are:

1. The **VMARC** extension:

   A file named SAMBA VMARC is a file compressed with the VMARC tool. VMARC is a VM tool available on Internet:

   http://www.ibm.com/s390/vm.

   On this site, instructions are given to download VMARC MODULE and to unpack the file in VMARC format.

   To unpack a file in VMARC format:

   1. Download the file to VM in BINARY
   2. Run the file through this pipeline (where `fn` is the name of the file you downloaded): `PIPE < fn VMARC A | fblock 80 00 | > fn VMARC A F 80`
   3. Unpack the resultant file using VMARC: `VMARC UNPK fn VMARC A`

2. The **`.tar`** extension:

   A `.tar` suffix indicates a file archived with `tar` UNIX tool. A `tar` file is not compressed and is downloaded directly to the BFS. To unarchive a `tar` file use the following command in your OpenEdition for VM/ESA session:

   ```
   pax -o from=ISO8859-1,to=IBM-1047 -rf gzip.tar
   ```

3. The **`.z`** extension:

   A `.Z` suffix specifies a file compressed with `compress` UNIX tool.

   - To uncompress file type in your OpenEdition for VM/ESA session:
     `uncompress make.tar.Z`

- To uncompress and unarchive file: `pax -o from=ISO8859-1,to=IBM-1047 -rzf gzip.tar.Z`

4. The **.gz** extension:

   A .gz suffix indicates a file compressed with GNU gzip. More details are available at 4.2.1, “What is gzip” on page 52. To uncompress the file use command:

   ```
   gzip /home/porting/make-3.77.tar.gz
   ```

K.3 The CD-ROM content

We created a CD-ROM where we put all packages used during the residency. The files are stored in four directories:

/Tools	Contains all the basic tools to be installed first.
/NonPortedPackage	Contains packages in original format. These packages are not ported to OpenEdition for VM/ESA.
/PortedPackage	Contains the ported packages written in two formats (VMARC and `.tar.gz`).
/Book	Contains this book in PDF format.

The following table shows all the CD-ROM directories and all the filenames and their extensions as a reference for your convenience.

Table 8. Files available on the CD-ROM

CD-ROM Directory	File name	Package name	Download method	Size
/Tools	vmarc.module	VMARC	note 1	13KB
	vmarc.helpcms	VMARC	note 1	31KB
	gzip-1.2.4.tar.Z	GZIP	note 3	482KB
	gzip.vmarc	GZIP	note 2	499KB
	XPG4.tar.gz	LIBXPG4	note 3	289KB
	libxpg4.vmarc	LIBXPG4	note 2	297KB
/NonPortedPackage	apache_1.3.6.tar.gz	APACHE	note 3	1349KB
	byacc.1.9.tar.Z	BYACC	note 3	85KB
	dcron.tar.gz	DCRON	note 3	77KB
	diffutils-2.7.tar.gz	GNUDIFF	note 3	305KB
	flex-2.5.4a.tar.gz	FLEX	note 3	373KB
	gdbm-1.7.3.tar.gz	GDBM	note 3	81KB
	gzip-1.2.4.tar.gz	GZIP	note 3	216KB
	inn-2.2.tar.gz	INN	note 3	1114KB
	irc2.10.2p1+Cr21+Sc8-a2-nostats2.tar.gz	IRC	note 3	573KB
	JacORB_0.9f1.tar.gz	JACORB	note 3	1246KB
	ldap-3.3.tar.Z	LDAP	note 3	2061KB
	make-3.77.tar.gz	MAKE	note 3	653KB
	mqm.tar.gz	MQM	note 3	416KB
	rcs-5.7.tar.gz	RCS	note 3	276KB
	Regina-0.08f.tar.gz	REGINA	note 3	460KB
	samba-2.0.3.tar.gz	SAMBA	note 3	2033KB
/PortedPackage	apache_1.3.6.tar.gz	APACHE	note 3	2798KB
	byacc.tar.gz	BYACC	note 3	177KB
	DAYTIME.tar.gz	DAYTIME	note 3	208KB
	dcron.tar.gz	DCRON	note 3	50KB
	gnudiff-2.7.tar.gz	GNUDIFF	note 3	538KB
	flex-2.5.4.tar.gz	FLEX	note 3	921KB
	gdbm-1.7.3.tar.gz	GDBM	note 3	188KB
	INETD.tar.gz	INETD	note 3	101KB
	inn-1.5.1.tar.gz	INN	note 3	4863KB

CD-ROM Directory	File name	Package name	Download method	Size
/PortedPackage	irc2.8.21.tar.gz	IRC	note 3	898KB
	JacORB_0.9e.tar.gz	JACORB	note 3	1228KB
	ldap-3.3.tar.gz	LDAP	note 3	5317KB
	libascii.tar.gz	LIBASCII	note 3	111KB
	make-3.74.tar.gz	MAKE	note 3	873KB
	mqm.tar.gz	MQM	note 3	416KB
	patch-2.0.tar.gz	PATCH	note 3	152KB
	RCS.tar.gz	RCS	note 3	1338KB
	Regina-0.08c.tar.gz	REGINA	note 3	1565KB
	samba-1.9.18p10.tar.gz	SAMBA	note 3	4478KB
	SYSLOGD.tar.gz	SYSLOGD	note 3	374KB
	apache.vmarc	APACHE	note 2	2870KB
	byacc.vmarc	BYACC	note 2	183KB
	daytime.vmarc	DAYTIME	note 2	216KB
	dcron.vmarc	DCRON	note 2	55KB
	gnudiff.vmarc	GNUDIFF	note 2	553KB
	flex.vmarc	FLEX	note 2	947KB
	gdbm.vmarc	GDBM	note 2	194KB
	inetd.vmarc	INETD	note 2	105KB
	innd.vmarc	INN	note 2	4995KB
	ircd.vmarc	IRC	note 2	928KB
	jacorb.vmarc	JACORB	note 2	1260KB
	ldap.vmarc	LDAP	note 2	5454KB
	libascii.vmarc	LIBASCII	note 2	121KB
	gnumake.vmarc	MAKE	note 2	897KB
	mqm.vmarc	MQM	note 2	427KB
	patch.vmarc	PATCH	note 2	158KB
	rcs.vmarc	RCS	note 2	1373KB
	regina.vmarc	REGINA	note 2	1605KB
	samba.vmarc	SAMBA	note 2	4595KB
	syslogd.vmarc	SYSLOGD	note 2	384KB

K.4 The CD-ROM download methods

Methods to download a package from the CD-ROM are described in the following notes.

K.4.1 Using TCP/IP FTP

Note 1:VMARC MODULE install

To install VMARC in VM, do the following:

1. Download the files vmarc.module and vmarc.helpcms to VM in BINARY,

```
ftp totvm1
Connected to totvm1.itso.ibm.com.
220-FTPSERVE IBM VM Level at TOTVM1.ITSO.IBM.COM, 15:03:07 EDT MONDAY
04/12/99
220 Connection will close if idle for more than 5 minutes.
User (totvm1.itso.ibm.com:(none)): porting
331 Send password please.
Password: ********
230 PORTING logged in; no working directory defined
ftp> cd sfstest:.
250 SFS working directory is SFSTEST:PORTING.
ftp> bin
200 Representation type is IMAGE.
ftp> put vmarc.module
200 Port request OK.
150 Storing file 'vmarc.module'
250 Transfer completed successfully.
112800 bytes sent in 0.16 seconds (705.00 Kbytes/sec)
ftp> 221 Quit command received. Goodbye.
```

2. Run the files through this pipeline:

```
PIPE < VMARC MODULE A | deblock cms | > VMARC MODULE A
PIPE < VMARC HELPCMS A | deblock cms | > VMARC HELPCMS A
```

Note: if you are running VM/ESA 2.2.0, you will need to apply APAR VM61031 in order for VMARC MODULE to work correctly on your system.

Note 2: Package in VMARC Format

The package you want to get has an extension `.vmarc`, so do the following:

1. On your PC workstation, use your local FTP to send the file from the CD-ROM download the VMARC file to VM :

```
ftp totvm1
Connected to totvm1.itso.ibm.com.
220-FTPSERVE IBM VM Level at TOTVM1.ITSO.IBM.COM, 15:03:07 EDT MONDAY
04/12/99
220 Connection will close if idle for more than 5 minutes.
User (totvm1.itso.ibm.com:(none)): porting
331 Send password please.
Password: ********
230 PORTING logged in; no working directory defined
ftp> cd sfstest:.
```

```
250 SFS working directory is SFSTEST:PORTING.
ftp> bin
200 Representation type is IMAGE.
ftp> put inetd.vmarc
200 Port request OK.
150 Storing file 'inetd.vmarc'
250 Transfer completed successfully.
112800 bytes sent in 0.16 seconds (705.00 Kbytes/sec)
ftp> 221 Quit command received. Goodbye.
```

2. Fix the VMARC file after download:

```
PIPE < INETD VMARC | fblock 80 00 | > INETD VMARC A F
```

3. Unpack the VMARC file:

```
VMARC UNPK INETD VMARC A
```

4. Place tar file in BFS:

```
OPENVM PUTBFS INETD TAR A /home/porting/inetd.tar.gz (BFSLINE NONE
```

When the file transfer is completed, skip to K.6, “Unarchiving the package files” on page 319.

Note 3: Package in (.tar.gz) or (.tar.Z) Format

The package that you want to load has an extension `.tar or .gz` ,so do the following:

1. On your PC workstation, use your local FTP to send the file from the CD-ROM download the tar file to VM:

```
ftp totvm1
Connected to totvm1.itso.ibm.com.
220-FTPSERVE IBM VM Level at TOTVM1.ITSO.IBM.COM, 11:26:45 EDT MONDAY
04/12/99
220 Connection will close if idle for more than 5 minutes.
User (totvm1.itso.ibm.com:(none)): porting
331 Send password please.
Password: ********
230 PORTING logged in; no working directory defined
ftp> cd /../VMBFS:BFSTEST:PORTING/
250 BFS working directory is /../VMBFS:BFSTEST:PORTING/
ftp> binary
200 Representation type is IMAGE.
ftp> put inetd.tar.gz
200 Port request OK.
150 Storing file 'inetd.tar.gz'
250 Transfer completed successfully.
212880 bytes sent in 0.88 seconds (241.63 Kbytes/sec)
ftp> quit
221 Quit command received. Goodbye.
```

2. When the file transfer is completed, skip to K.6, “Unarchiving the package files” on page 319.

Note 4: Book in pdf Format

The book sg245458.pdf is in PDF format. To read it, you need Adobe Acrobat Reader. This is available at:

```
http://www.adobe.com/prodindex/acrobat
```

K.4.2 Using IND$FILE

If you have no TCP/IP connectivity between your PC and VM, then you can use IND$FILE (also known as SEND and RECEIVE). We recommend you download packages in vmarc format.

1. On the PC where the CD-ROM is mounted (in this example as a 'D:' drive), send the file to the 3270 (in this example, the A session is used):

   ```
   send d:inetd.vmarc a:inetd vmarc a (crlf recfm f lrecl 80
   ```

2. If the file is a VMARC file, do this step to unpack it:

   ```
   VMARC UNPK INETD VMARC A
   ```

3. Place the file in the BFS:

   ```
   OPENVM PUTBFS INETD TAR A /home/neale/inetd.tar.gz (BFSLINE NONE
   ```

K.5 Downloading from Internet

Use your preferred browser to download a package from a World Wide Web site: Download the package file from the Web site to your workstation disk space; see Figure 202 on page 319.

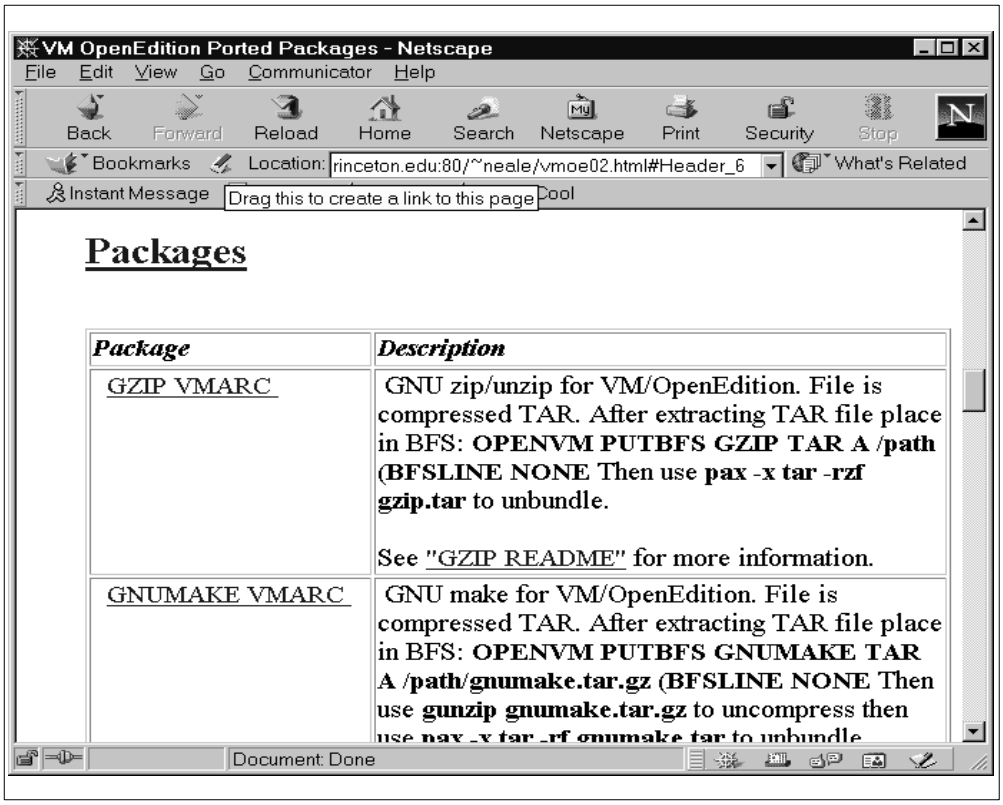


Figure 202. Downloading file from Web site

Depending on the format of the file, follow the procedure in either “Note 3: Package in (.tar.gz) or (.tar.Z) Format” on page 317 or “Note 2: Package in VMARC Format” on page 316.

K.6 Unarchiving the package files

1. The file selected has been placed into the BFS. To proceed, enter the shell:

   ```
   shell
   ```

2. Uncompress the archive:
   1. If its name ends with .Z: `uncompress inetd.tar.Z`
   2. If its name ends with .gz: `gunzip inetd.tar.gz`

Note: If it happens that you receive the following message please ignore it.

```
gunzip: inetd.tar.gz: decompression OK, trailing garbage ignored
```

3. Unarchive the files if its name ends with .tar:

   ```
   pax -rf inetd.tar
   ```

4. See what the unarchive created:

   ```
   ls -ld INETD
   drwxrwxrwx   1 neale    system         0 Mar 11 05:16 INETD
   ```

5. Go to the directory just created:

   ```
   cd INETD
   ```

6. See what is in there:

```
ls -l
total 624
-rw-r--r--   1 neale    system       379 Oct 29  1997 Makefile
-rwxrwxrwx   1 neale    system    224442 Nov  6 08:11 inetd
-rw-r--r--   1 neale    system     43980 Nov  6 08:11 inetd.c
-rw-rw-rw-   1 neale    system       993 Feb  1 02:27 inetd.conf
-rw-rw-rw-   1 neale    system     36960 Nov  6 08:11 inetd.o
```

The next steps are either the compilation and linkedit, or the installation. These steps are package-dependent.

Appendix L. Important license information

Some of the packages described in this book are covered under one of the following licenses.

L.1 GNU license Version 1 information

GNU GENERAL PUBLIC LICENSE
Version 1, February 1989

Preamble

The license agreements of most software companies try to keep users at the mercy of those companies. By contrast, our General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. The General Public License applies to the Free Software Foundation's software and to any other program whose authors commit to using it. You can use it for your programs, too.

When we speak of free software, we are referring to freedom, not price. Specifically, the General Public License is designed to make sure that you have the freedom to give away or sell copies of free software, that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of a such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must tell them their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any work containing the Program or a portion of it, either verbatim or with modifications. Each licensee is addressed as "you".

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this General Public License and to the absence of any warranty; and give any other recipients of the Program a copy of this General Public License along with the Program. You may charge a fee for the physical act of transferring a copy.

2. You may modify your copy or copies of the Program or any portion of it, and copy and distribute such modifications under the terms of Paragraph 1 above, provided that you also do the following:

a) cause the modified files to carry prominent notices stating that you changed the files and the date of any change; and

b) cause the whole of any work that you distribute or publish, that in whole or in part contains the Program or any part thereof, either with or without modifications, to be licensed at no charge to all third parties under the terms of this General Public License (except that you may choose to grant warranty protection to some or all third parties, at your option).

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the simplest and most usual way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this General Public License.

d) You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

Mere aggregation of another independent work with the Program (or its derivative) on a volume of a storage or distribution medium does not bring the other work under the scope of these terms.

3. You may copy and distribute the Program (or a portion or derivative of it, under Paragraph 2) in object code or executable form under the terms of Paragraphs 1 and 2 above provided that you also do one of the following:

a) accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Paragraphs 1 and 2 above; or,

b) accompany it with a written offer, valid for at least three years, to give any third party free (except for a nominal charge for the cost of distribution) a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Paragraphs 1 and 2 above; or,

c) accompany it with the information you received as to where the corresponding source code may be obtained. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form alone.)

Source code for a work means the preferred form of the work for making modifications to it. For an executable file, complete source code means all the source code for all modules it contains; but, as a special exception, it need not include source code for modules which are standard libraries that accompany the operating system on which the executable file runs, or for standard header files or definitions files that accompany that operating system.

4. You may not copy, modify, sublicense, distribute or transfer the Program except as expressly provided under this General Public License. Any attempt otherwise to copy, modify, sublicense, distribute or transfer the Program is void, and will automatically terminate your rights to use the Program under this License. However, parties who have received copies, or rights to use copies, from you under this General Public License will not have their licenses terminated so long as such parties remain in full compliance.

5. By copying, distributing or modifying the Program (or any work based on the Program) you indicate your acceptance of this license to do so, and all its terms and conditions.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein.

7. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of the license which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the license, you may choose any version ever published by the Free Software Foundation.

8. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author

to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

9. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

10. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to humanity, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.>
> Copyright (C) 19yy <name of author>
>
> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 1, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software

Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) 19xx name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
This is free software, and you are welcome to redistribute it
under certain conditions; type ‘show c’ for details.

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than ‘show w’ and ‘show c’; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the
program ‘Gnomovision’ (a program to direct compilers to make passes
at assemblers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice

That’s all there is to it!

L.2 GNU license Version 2 information

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not

compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to

be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.>
> Copyright (C) 19yy <name of author>
>
> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

> Gnomovision version 69, Copyright (C) 19yy name of author
> Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
> This is free software, and you are welcome to redistribute it
> under certain conditions; type 'show c' for details.

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than 'show w' and 'show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

> Yoyodyne, Inc., hereby disclaims all copyright interest in the program 'Gnomovision' (which makes passes at compilers) written by James Hacker.
>
> <signature of Ty Coon>, 1 April 1989
> Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

L.3 LDAP license information

```
Copyright (c) 1992-1996 Regents of the University of Michigan.
All rights reserved.

Redistribution and use in source and binary forms are permitted
provided that this notice is preserved and that due credit is given
to the University of Michigan at Ann Arbor. The name of the University
may not be used to endorse or promote products derived from this
software without specific prior written permission. This software
is provided ``as is'' without express or implied warranty.
```

L.4 Apache license information

```
/* ====================================================================
 * Copyright (c) 1995-1999 The Apache Group.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the Apache Group
 *    for use in the Apache HTTP server project (http://www.apache.org/)."
 *
 * 4. The names "Apache Server" and "Apache Group" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    apache@apache.org.
 *
 * 5. Products derived from this software may not be called "Apache"
 *    nor may "Apache" appear in their names without prior written
 *    permission of the Apache Group.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the Apache Group
 *    for use in the Apache HTTP server project (http://www.apache.org/)."
 *
 * THIS SOFTWARE IS PROVIDED BY THE APACHE GROUP ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE APACHE GROUP OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
```

```
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This software consists of voluntary contributions made by many
 * individuals on behalf of the Apache Group and was originally based
 * on public domain software written at the National Center for
 * Supercomputing Applications, University of Illinois, Urbana-Champaign.
 * For more information on the Apache Group and the Apache HTTP server
 * project, please see <http://www.apache.org/>.
 *
 */
```

L.4.1 Supplementary license for src/regex component

```
Copyright 1992, 1993, 1994 Henry Spencer.  All rights reserved.
This software is not subject to any license of the American Telephone
and Telegraph Company or of the Regents of the University of California.

Permission is granted to anyone to use this software for any purpose on
any computer system, and to alter it and redistribute it, subject
to the following restrictions:

1. The author is not responsible for the consequences of use of this
   software, no matter how awful, even if they arise from flaws in it.

2. The origin of this software must not be misrepresented, either by
   explicit claim or by omission.  Since few users ever read sources,
   credits must appear in the documentation.

3. Altered versions must be plainly marked as such, and must not be
   misrepresented as being the original software.  Since few users
   ever read sources, credits must appear in the documentation.

4. This notice may not be removed or altered.
```

L.5 LIBASCII license information

```
*********************************************************************
*                                                                   *
* libascii - ascii-ebcidic interface layer - README file            *
*                                                                   *
* Version 1.1.9                                                     *
*                                                                   *
* To report problems or ask questions send e-mail to:               *
*                                                                   *
*             libascii@nvet.ibm.com                                 *
*                                                                   *
* Copyright:   Licensed Materials - Property of IBM.                *
*              (C) Copyright IBM Corp. 1997, 1998.                  *
*              All rights reserved.                                 *
*                                                                   *
* License information:                                              *
*                                                                   *
*   The libascii source code is provided free of charge and         *
*   may be distributed freely.  No fee may be charged if you        *
*   distribute the libascii source code (except for such things     *
*   as the price of a disk or tape, postage ).  The libascii        *
*   makefile will compile and produce a libascii.a archive file.    *
*   The libascii.a archive may be link edited  with any software    *
*   vendor  product.  Any software vendor product that is link      *
*   edit with libascii.a archive is free to distribute and charge   *
*   for that product.                                               *
*                                                                   *
*   THIS PROGRAM IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY       *
*   KIND, EXPRESS  OR IMPLIED, INCLUDING THE IMPLIED  WARRANTIES    *
*   OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.        *
*   IBM does not warrant uninterrupted or error free operation of   *
*   the  Program, or that the Program is free from claims by a      *
*   third party of copyright, patent, trademark, trade secret,      *
*   or any other intellectual property infringement. IBM  has       *
*   no obligation to provide service, defect correction, or any     *
*   maintenance for the Program. IBM  has  no  obligation to        *
*   supply any Program updates or enhancements to you even if       *
*   such are or later become available.                             *
*                                                                   *
*    Under no circumstances is IBM liable for any of the            *
*    following:                                                     *
*                                                                   *
*      1.  third-party claims against you for losses or damages;    *
*      2.  loss of, or damage to, your records or data; or          *
*      3.  direct  damages,  lost  profits, lost savings,           *
*          incidental, special, or indirect damages or other        *
*          consequential damages,  even if IBM or its authorized    *
*          supplier, has been advised of the possibility of         *
*          such damages.                                            *
*                                                                   *
*    Some jurisdictions do not allow these limitations or           *
*    exclusions, so they may not apply to you.                      *
*                                                                   *
*                                                                   *
*********************************************************************
```

L.6 Patch license information

```
Patch Kit, Version 2.0

Copyright (c) 1988, Larry Wall

You may copy the patch kit in whole or in part as long as you don't try
make money off it, or pretend that you wrote it.
```

Appendix M. Special Notices

This publication is intended to help those engaged in porting UNIX applications and solutions to OpenEdition for VM/ESA. The information in this publication is not intended as the specification of any programming interfaces that are provided by VM/ESA. See the PUBLICATIONS section of the IBM Programming Announcement for VM/ESA for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (“vendor”) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AIX	APPN
C/VM	DB2
Home Director	IBM ®
IMS	Language Environment
MQ	MQSeries
NetRexx	NetView
OpenEdition	OS/2
OS/390	SP
System/390	VM/ESA
VTAM	XT

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries. (For a complete list of Intel trademarks see www.intel.com/dradmarx.htm)

UNIX is a registered trademark in the United States and/or other countries licensed exclusively through The Open Group.

SET and the SET logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, and service names may be trademarks or service marks of others.

Appendix N. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

N.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How to Get ITSO Redbooks” on page 341.

- *OpenEdition for VM/ESA Implementation and Administration Guide*, SG24-4747
- *VM/ESA OpenEdition DCE Introduction and Implementation Notebook*, SG24-4554
- *LDAP Implementation Cookbook*, SG24-5110
- *Understanding LDAP*, SG24-4986

N.2 Redbooks on CD-ROMs

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at `http://www.redbooks.ibm.com/` for information about all the CD-ROMs offered, updates and formats**:**

CD-ROM Title	Collection Kit Number
System/390 Redbooks Collection	SK2T-2177
Networking and Systems Management Redbooks Collection	SK2T-6022
Transaction Processing and Data Management Redbook	SK2T-8038
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
Application Development Redbooks Collection	SK2T-8037

N.3 Other Publications

These publications are also relevant as further information sources:

- *OpenEdition for VM/ESA Advanced Application Programming Tools Version 2 Release 1.0*, SC24-5729
- *OpenEdition for VM/ESA Command Reference Version 2 Release 3.0*, SC24-5728
- *OpenEdition for VM/ESA Callable Services Reference*, SC24-5726
- *OpenEdition for VM/ESA Implementation and Administration Guide*, SG24-4747
- *OpenEdition for VM/ESA POSIX Conformance Document*, GC24-5842
- *OpenEdition for VM/ESA Sockets Reference Version 2 Release 1*, SC24-5741
- *OpenEdition for VM/ESA User’s Guide Version 2 Release 1.0*, SC24-5727
- *VM/ESA CMS File Pool Planning, Administration, and Operation*, SC24-5751
- *Language Environment for OS/390 & VM Programming Guide*, SC28-1939

- *Language Environment for OS/390 & VM Programming Reference*, SC28-1940
- *IBM C for VM/ESA Library Reference, Volume 1 Version 3 Release 1*, SC23-3908
- *Programming PERL 5*, ISBN 1-5659-2149-6
- *Apache Server Bible* Mohammed J. Kabir, ISBN 0-7645-3218-9
- *CMS Application Multitasking*, SC24-5766-01
- *Pthreads Programming*, ISBN 1-56592-115-1

N.4 Resources on the Internet

The following Web pages are updated periodically with new documentation, demos, code samples, and utilities to assist with porting UNIX applications.

- http://www.s390.ibm.com/products/oe/libascii.html
- http://www.redbooks.ibm.com/
- http://www.ibm.com/s390/vm/openedition
- http://www.gzip.org/index.html
- http://www.s390.ibm.com/oe/bpxa1por.html
- http://pucc.princeton.edu/~neale/vmoe.html
- http://samba.org/cifs.
- http://www.netcraft.com/survey
- http://www.fastcgi.com
- http://www.apache.org/docs/
- http://www.apache.org/docs/vhosts/
- http://www.inf.fu-berlin.de/~brose/jacorb/
- http://www.inf.fu-berlin.de/~brose/NS_Ref
- http://www.inf.fu-berlin.de/~brose/jacorb/jacorb.ps.
- http://www.ibm.com/vm/openedition
- http://pucc.princeton.edu/~neale/vmoe.html
- http://www.adobe.com/prodindex/acrobat
- http://www.mirc.co.uk/help/jarkko.txt
- http://www.mirc.co.uk/help/jarkko2.txt
- http://www.alphaWorks.ibm.com/tech/irc

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** `http://www.redbooks.ibm.com/`

  Search for, view, download, or order hardcopy/CD-ROM redbooks from the redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this redbooks site.

  Redpieces are redbooks in progress; not all redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the redbooks fax order form to:

	e-mail address
In United States	usib6fpl@ibmmail.com
Outside North America	Contact information is in the “How to Order” section at this site: `http://www.elink.ibmlink.ibm.com/pbl/pbl/`

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the “How to Order” section at this site: `http://www.elink.ibmlink.ibm.com/pbl/pbl/`

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the “How to Order” section at this site: `http://www.elink.ibmlink.ibm.com/pbl/pbl/`

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and redbooks by accessing the IBM Intranet Web site at `http://w3.itso.ibm.com/` and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access `MyNews` at `http://w3.ibm.com/` for redbook, residency, and workshop announcements.

IBM Redbook Fax Order Form

Please send me the following:

Title	Order Number	Quantity

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

☐ Invoice to customer number _____

☐ Credit card number _____

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Glossary

DCE Distributed Computing Environment

A

access permission. A group of designations that determine who can access a particular file and how the user can access the file.

AIX. Advanced Interactive eXecutive.

alias. An alternate name for a shell command. An alias for a command can be defined with options different from those of the command itself. An alias can be a convenient shorthand for a complicated command line, such as a pipeline.

American Standard Code For Information Interchange (ASCII). It is the coded character set used by UNIX. It defines 128 characters and each is represented by 7-bit binary values.

Application Program Interface (API). A formal interface that is intended to be used in the building of applications. It provides a path into system functions.

APPN. Advanced Peer-to Peer Network.

archive. Synonymous with backup, and backup copy.

ASCII. American National Standard Code for Information Interchange.

awk. A file processing language that is well-suited to data manipulation and retrieval of information from text files. *awk* is named for its creators, Alfred V. Aho, Peter J. Weinberger, and Brian W. Kernighan.

B

background. In multiprogramming, the conditions under which low-priority programs are processed.

backslash. The character \ is also known as a reverse solidus. The backslash enables a user to escape the special meaning of a character. That is, typing a backslash before a character tells the system to ignore any special meaning the character might have.

basename. The final, or only, file name in a pathname.

BFS. Byte File System. The (BFS) server of VM/ESA.

Byte File System. A file system where a file consists of an ordered sequence of bytes, rather than records as in the CMS file system. BFS files may be organized into hierarchical directories.

C

canonical input. If a terminal is in canonical mode, input is collected and processed one line at a time. A read request is not returned until the line is terminated or a signal is received. Erase and kill processing is performed on input that was not terminated by one of the line termination characters. Erase processing removes the last character in the line, and kill processing removes the entire line.

character special file. A special BFS file that provides access to an input or output device. The character interface is used for devices that do not use block I/O.

child process. A new process created by an existing process. The new process is thereafter known to the preexisting process as the child process. The preexisting process is called the parent process.

CISC. Complex Instruction Set Computer.

CL. Control Language.

CLP. Control Language Program.

code page. A table showing codes assigned to character sets.

CMS short name. The CMS short name is a numerical identifier unique to every file in the BFS. See also **inode**.

command alias. An abbreviation of a long commandline or a new name for a command.

command interpreter. A program that reads the commands that you type and then processes them. When you are typing commands into the computer, you are actually typing input to the command interpreter. The interpreter then decides how to perform the commands that you have typed. The shell is an example of a command interpreter. Synonymous with command language interpreter.

current directory. The directory that is active and can be displayed with the *pwd* command. Synonymous with current working directory.

current working directory. The BFS directory a user is working in.

D

daemon. Daemon processes on the UNIX system do system-wide functions, such as administration and control of networks. They are not associated with any user.

DCE. Distributed Computing Environment.

default directory. The directory name supplied by the operating system if a name is not specified.

directory. A binary file consisting of a list of other files.

dump. To write at a particular instant the contents of storage, or a part of storage, onto another data medium for the purpose of safeguarding or debugging the data.

E

EBCDIC. Extended binary-coded decimal interchange code.

effective user ID. The current user ID, but not necessarily the user′s login ID. For example, a user logged in under a login ID may change to another user′s ID. This ID becomes the effective user ID until the user switches back to the original login ID.

external link. A file system entry that can be used to: reference data outside of BFS (data residing on a CMS minidisk or in a directory); create an implicit mount point; contain data in an application-defined format.

F

FIFO special file. A type of file with the property that data written to the file is read on a first-in-first-out basis.

file descriptor. In the UNIX environment, the integer that identifies a file.

file mode. File mode on a UNIX system; refers to file protection access of a file. There are three types of file access: read, write, and execute, designated by the letters r, w, and x respectively.

file pool. A file pool is a collection of minidisks owned by a particular file pool virtual machine known as a file pool server. The minidisks are used for storing file pool repository data and control data.

file space. Storage allocated to an enrolled user within the Shared File System (SFS) or Byte File System (BFS). In BFS terms, a file space can also be mounted as a file system.

file type. On UNIX systems, it is safe to say that everything is a file. The operating system even represents I/O devices as files.

There are several different kinds of files, each with a different purpose. Regular files contain data. These may be text files, data files, executable files, and so on. Special files are used for device I/O under UNIX. They reside in directory /dev and its subdirectories. Link files allow several file names to refer to a single file on disk. There are two types of links: hard links and soft links. A hard link uses the inode address of a file, while a soft link will use a file′s directory.

foreground process. A process that is a member of a foreground process group. Typically, this is the application the user interacts with.

FTP. File Transfer Protocol.

function key. A key that causes a specified sequence of operations to be performed when it is pressed. Generally used to refer to keys labeled <F n>, f o r example <F1>.

G

GID. An acronym for group identifier.

group ID. A unique number assigned to a group of related users. The GID can often be substituted in commands that take a group name as an operand.

group name. A name that uniquely identifies a group of POSIX users to the system.

gnumake. gnumake is used to maintain groups of programs. It executes commands in a makefile to update one or more target names, where name is typically a program.

gzip. A popular tool for compressing data in files, gzip is similar to the compress utility but more effective. In addition, gzip is smart enough to handle *.Z and *.z files.

H

history file. A file that lists the shell commands that have been processed.

home directory. The current directory associated with the user at the time of login.

I

IBM. International Business Machines Corporation.

IETF. Internet Engineering Task Force.

IFS. Integrated File System.

IMPI. Internal Micro Programmed Interface.

inode. A data structure on disk that describes and stores a file′s attributes, including:

² UID and GID of the file

² File type (regular, directory, and so on)

² Access modes (permissions)

² File creation, modification, and access times

² Inode modification time

² Number of links to the file

² Size of the file

It is possible for two files to have the same inode number when hard links are used for files.

IPX. Internetwork PacketeXchange.

ITSO. International Technical Support Organization.

K

kernel. The part of the OpenEdition for VM/ESA component containing programs for such tasks as I/O, management, and communication.

kill. An operating system command that stops a process.

L

LAN. Local Area Network.

LIC. Licensed Internal Code.

Lightweight Directory Access Protocol (LDAP). LDAP is a fast-growing technology for accessing common directory information.

line mode. Synonym for canonical mode.

locale. The definition of the subset of a user's environment, which depends on language and cultural conventions.

login. In UNIX systems, the act of gaining access to a computer system by entering identification and authentication information at the workstation.

M

MI. Machine Interface**.**

mode. A collection of attributes that specifies a file's type and its access permissions.

mount. To make a file system accessible.

mount point. The path name of the directory on which the file system is mounted.

N

NetBios. Network Basic Input Output System.

NNTP. Network News Transfer Protocol.

noncanonical mode. An input processing mode where input character erase and killing are eliminated, making input characters available to the user program as they are typed.

O

ORB. Object Request Broker.

OSF. Open Software Foundation.

OSI. Open Systems Interconnection.

output file. A file that a program opens so that it can write to that file.

output redirection. The specification of an output destination other than the standard one.

owner. The user who has the highest level of access authority to a data object or action, as defined by the object or action.

P

parent directory. The directory that both contains a directory entry for the given director and is represented by the path name in the given directory.

path name. A BFS file name specifying all BFS directories leading to the file.

Perl. A language for easily manipulating text, files, and processes.

permission mode. A three-digit octal code or a nine-letter alphabetic code that determines how a file can be used. It indicates the read, write, and run permissions for the owner, for the group, and for all others.

ping. A socket program used by network administrators to diagnose network problems. It provides feedback on the reliability of the connection and the time required to reach the target host.

pipe. An interprocess communication mechanism that connects an output file descriptor to an input file descriptor. Usually the standard output of one process is connected to the standard input of another, forming a pipeline.

portable character set. The set of characters described in POSIX.2 2.4 that is supported on all conforming systems.

POSIX. Portable Operating System Interface for Computer Environments; an IEEE operating system standard, closely related to the UNIX system (software writing).

process. A function being performed or waiting to be performed.

prompt. A displayed symbol or message that requests information or operator action.

R

RACF. Resource Access Control Facility, an IBM security product.

relative path name. The name of a BFS directory or file expressed as a sequence of directories followed by a file name, beginning from the current directory.

RISC. Reduced Instruction Set Computer.

root. The starting point of the Byte File System. It can also be the user name for the system user with the most authority.

root directory. The BFS directory that contains all other directories in the system.

RPC. Remote Procedure Call.

S

Samba. A suite of programs which work together to allow a client to access a server's filespace and printers via the SMB (Server Message Block) protocol.

session. The period of time during which a user of a terminal can communicate with an interactive system. It is usually the elapsed time between logon and logoff.

set-group-ID mode bit. In setting file access permissions, the bit that sets the effective group ID of the process to the file's group upon processing.

set-user-ID mode bit. In setting file access permissions, the bit that sets the effective user ID of the process to the file's owner upon processing.

shell. (1) A program that interprets and processes interactive commands from a terminal or file. (2) An interface between the user and the operating system. (3) The working environment for an interactive user.

shell script. A file of shell commands very similar to a CMS EXEC.

Server Message Block (SMB). Protocol for sharing files, printers, serial ports, and communications abstractions such as named pipes and mail slots between computers.

SMTP. Simple Mall Transfer Protocol.

SNA. Systems Network Architecture.

SNMT. Simple Network Management Protocol.

socket. A method of communication between two processes. Sockets allow communication in two directions, in contrast to pipes, which allow communication in only one direction. The processes using a socket can be on the same system or on systems in the same network.

SPX. Sequenced Packet eXchange.

sticky bit. A permission bit that causes an executable program to remain in storage after execution. Sticky bit is the common name of save text mode. The sticky bit can be set, but VM/ESA will take no action based on the setting.

storage group. A collection of minidisks within a file pool. Each storage group is identified by a number. Storage group 1 is for catalog information only. Storage groups 2 through 32767 are for user data.

superuser. A privileged account with unlimited access to all files and commands. Many administrative tasks require superuser status.

symbolic link. A file which points to another path name in the file system. Symbolic links may span VM/ESA BFS file pools.

T

TCP/IP. Transmission Control Protocol/Internet Protocol refers to a family of non-proprietary network protocols, of which TCP, providing host-to-host transmission, and IP, providing data routing from source to destination, are the two important parts.

thread. A technique for concurrent programming by allowing multiple flows of processing within a process. Each thread in a process is a separate processing flow.

TIMI. Technology Independent Machine Interface.

U

UID. An acronym for user identification. See **user ID**.

UUCP. UNIX-to-UNIX Copy Program.

user ID. A unique string of characters that identifies a user to the system. This string of characters limits the functions and information the user can use.

W

white space. Space characters, tab characters, new line characters, and comments. White space separates commands on the command line.

working directory. The active directory used to resolve path names that do not begin with a slash. In similar systems, a working directory may be called the current directory or the current working directory.

****

Index

Numerics

A

B

C

D

E

F

G

I

J

L

M

P

R

S

T

U

V

X

ITSO Redbook Evaluation

Porting UNIX Applications to OpenEdition for VM/ESA
SG24-5458-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at `http://www.redbooks.ibm.com/`
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to `redbook@us.ibm.com`

Which of the following best describes you?
_ Customer _ Business Partner _ Solution Developer _ IBM employee
_ None of the above

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes___ No___

If no, please explain:

What other redbooks would you like to see published?

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

SG24-5458-00

Printed in the U.S.A.

Porting UNIX Applications to OpenEdition for VM/ESA

SG24-5458-00