毎日叩ける
シェル芸を覚えよう！
～朝のシェル芸勉強会～

Yasuhiro Yamada (@grethlen)

2016-08-27 第6回もう初心者向けでないなんて言わないよ

絶対午前のシェル勉強会/第24回◯◯o◯裏番組シェル芸勉強会 午前の部

$ whoami

なまえ: Yasuhiro Yamada

おしごと:

アプリケーションエンジニア

電子決済まわりが本業

しゅみ:

アニメ鑑賞

シェル芸

Twitter:
@grethlen

Qiita:
greymd

危険じゃないよ

シェル芸歴

芸歴　本格的に端末開き始めたのがおおよそ6年前くらい

シェル芸関連で開発

egzact — シェルの弱点を補うコマンド達

Github: greymd/egzact

cureutils — 次世代IT人材育成ツール

Github: greymd/cureutils

記事(Qiita)

【たのしいな】様々なコマンド達を何も考えずにつないで遊ぶ (1079 stock)

テストの数を減らそう！プリキュアで学ぶPICT (305 stock)

プリキュアで学ぶワンライナーWebスクレイピング (122 stock)

シェル芸の学び方

ボトムアップ

→知識を土台から固める

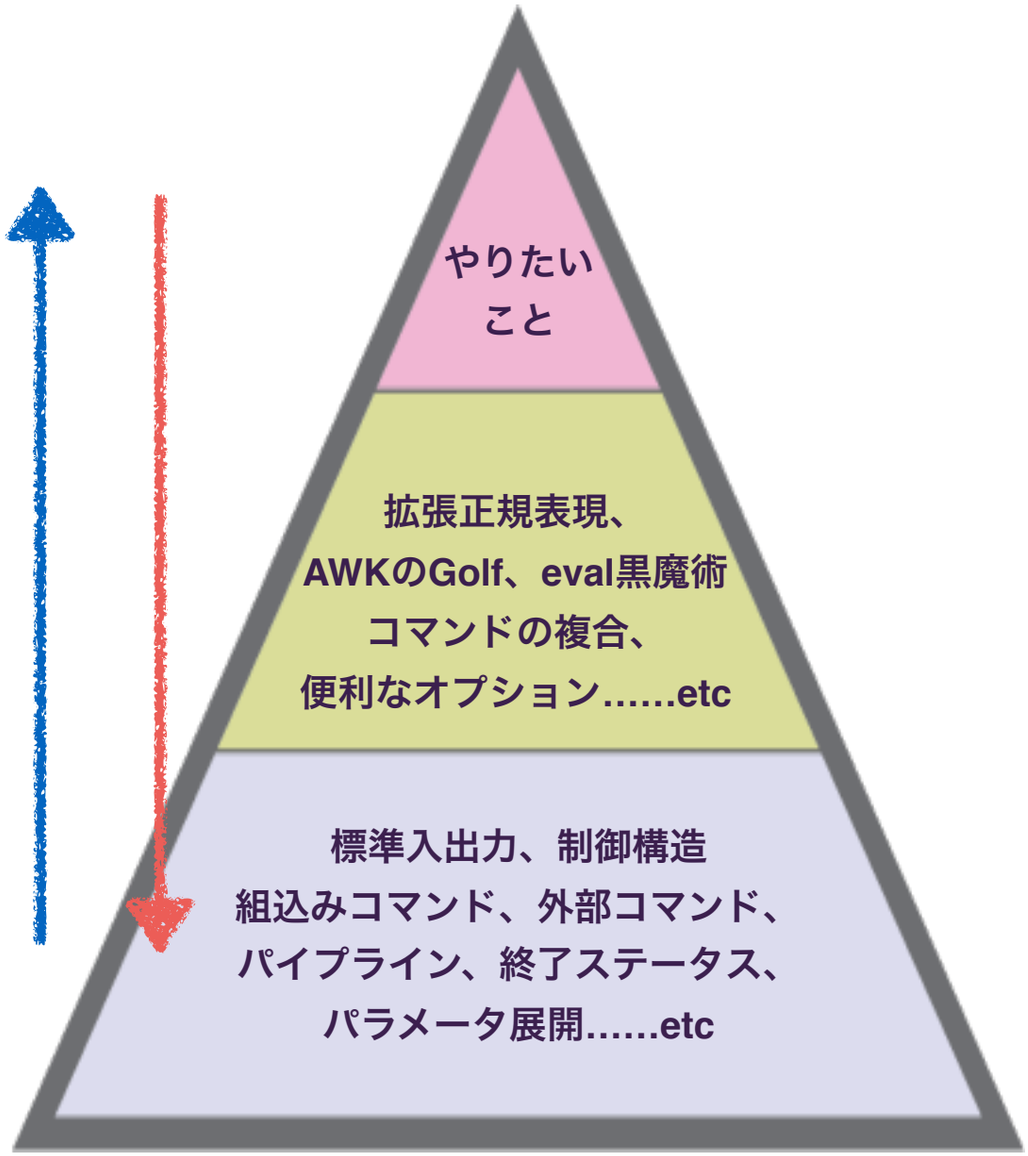
→例: 体系的にまとまった資料を読む

トップダウン

→やりたいことを先に決める

→最低限必要な知識から得る

→例: ググった結果をリバースエンジニアリング



ボトムアップなアプローチ

体系的にまとまった資料を読む

- シェルスクリプト基本リファレンス
  (著: 山森 丈範)
- Software Designでピンときたら買う
  (例: 2016年6月号 bash 再入門)
- シェル芸本
- など…(最後の方のスライドにまとめてます)

トップダウンなアプローチ

やりたいことをググる

→出てきたコマンドをリバースエンジニアリング

勉強会で課題を解く

→ 分からない課題はシェル芸人に一旦やってもらう

→ 気になったフレーズを覚えて帰って使ってみる

毎日端末を開いて覚えたフレーズを叩くのオススメ

カレンダーの確認とか、

生活に必要な作業がGOOD（後述）

どっちのアプローチがいいの？

今の上達レベルに応じてバランスを取るのが大切

ボトムアップだけだと手が動かない。

トップダウンだけだと知識が蜂の巣に。

プチシェル芸勉強会を
これから開きます

進め方(1/3)

入門者（黒い画面がまだこわいおともだち）

画面に出るコマンド写経あるいはググって下さい。

飽きたら変態を観察して下さい。こわくないです。

初心者（grepでログを確認するのがせいぜいのおともだち）

あたらしいコマンドとオプションを覚えましょう。

後で資料を見返してみましょう。

中級者（xargsで改行を除くのが習慣化してるおともだち）

知らないTipsがあると思うので、是非持ち帰って下さい。

変態（AWKでGolfするおともだち）

偉そうなこと言っててすみません。変態回答期待してます。

進め方(2/3)

動作環境

問題はUbuntu14.14のbashで動作確認済み。

資料自体は既にWebで見れます。

できるだけ答えを見ないで進めて下さい。

実施中はトップダウンな学びを。

資料自体は後から読み返してボトムアップな学びのための索引としてご活用下さい。

進め方(3/3)

端末の説明

```
$ コマンド -オプションやその他 '文字列リテラル' ファイル
    # コメント
```

- 先頭の $ はプロンプトを表す記号で入力は不要。
- 文末の逆スラッシュ \ は改行（あってもなくてもOK）

[持ち帰りポイント]

説明だらだら

下準備 (1/2)

1. 端末を開いて下さい

2. 下記のコマンドを実行して、ファイルをダウンロードしてください。

```
$ wget goo.gl/PGDjKb -O holidays
```

※ wgetがない人はこちら

```
$ curl -L goo.gl/PGDjKb > holidays
```

下準備 (2/2)

3. ls を打ってファイルがあることを確認。

```
$ ls
holidays
```

4. ファイルの内容を確認

```
$ head holidays
2015-01-01 木 元日 1

2015-01-02 金 三が日 1

2015-01-03 土 三が日 1

…
```

このファイルは？

2015～2021年までの日本の日付一覧

フォーマット（[ ] はスペース）

```
YYYY-MM-DD[ ]曜日[ ]イベント名[ ]休日フラグ
```

休日フラグ（休日=1, 平日=0)

12/30～01/03までは休日扱い

（仕事納め、大晦日、三が日）

お盆は8/13～8/16に設定

適宜個人の状況に応じて書き換えて下さい。

Q1

2016年の日付だけを標準出力に表示して下さい。

※中級者以上の方は”g”のつくコマンド禁止

```
$ ? ? ?
2016-01-01 金 元日 1
2016-01-02 土 三が日 1
2016-01-03 日 三が日 1
2016-01-04 月 平日 0

…
```

パターンマッチをしよう(1/2)

解答例

```
$ grep '2016' holidays
2016-01-01 金 元日 1

2016-01-02 土 三が日 1

2016-01-03 日 三が日 1

…
```

[grepコマンドの使い方の基本]

```
1. grep <抽出したいパターン> <ファイル名>
2. cat <ファイル名> | grep <抽出したいパターン>
```

パターンマッチをしよう(2/2)

別解

```
$ cat holidays | grep 2016
```

grepの使い方がわからない人にはこんなやり方も。

```
$ cat holidays | sed -n '/2016/p'
$ cat holidays | awk '/2016/'
$ cat holidays | perl -ne '/2016/ and print'
```

注: grepは速いので、大抵の場合grepを使ったほうがいいです。

Q2

2016年2月は何日あるか

数えてみましょう。

```
$ ? ? ?
 29
```

行数をカウントしよう(1/2)

解答例

```
$ cat holidays | grep '2016-02' | wc -l
      29
```

行数をカウントしよう(2/2)

下記は wc -l (行数カウント)と
ほぼ同じ挙動をする。

- awk 'END{print NR}'
- grep -c .
- nl -nln | cut -f 1 | tail -n 1
- perl -pE '}{ say $.'

Q3

2016年9月19日は何の日？

イベント名のみを表示してみましょう。

中級者以上の人は”a”のつくコマンド禁止

```
$ ？ ？ ？
敬老の日
```

特定のフィールドを抽出(1/2)

解答例

```
$ cat holidays | grep '2016-09-19' | \
awk '{print $3}'
```

[フィールド]

文字列をスペースで区切ったときの文字のまとまりのこと。上記例は左から３番目のフィールドを表示。

2015-01-01[スペース]木[スペース]元日[スペース]1

$1　　$2　　$3　　$4

特定のフィールドを抽出(2/2)

下記の例は全て、３フィールド目を抽出する例

```
• awk '$0=$3'

• cut -d ' ' -f 3

• perl -anle 'print $F[2]' # Perlは0インデックス

• perl -aple '$_=$F[2]'

• self 3  # Open usp Tukubai 使用
```

Q4

今日の情報を表示してみましょう。

ただし、コマンドに一切数字を使わずに。

```
$ ? ? ?
2016-08-27 土 休日 1
```

コマンド置換をつかおう(1/2)

解答例

```
$ cat holidays | grep $(date +%F)
```

[コマンド置換]

$(～) あるいは バッククオート（`）を使った

　`～`　という表現は「コマンド置換」。

　～ に書いたコマンドの結果を文字列に

したもののようにふるまう。

コマンド置換をつかおう(2/2)

コマンド置換の例

```
$ date +%F
2016-08-27

$ echo $( date +%F )  #(1)
2016-08-27

$ echo '2016-08-27'   #(2)
2016-08-27
```

(1)と(2)のコマンドは結果的に同じ。

Q5

2016年に、各曜日がいくつあるか表示してみましょう。

```
$ ? ? ?
  53 土
  52 日
  52 月
  52 木
  52 水
  52 火
  53 金
```

数え上げ(1/2)

解答例

```
$ cat holidays | grep '2016' | awk '{print $2}' | \
  sort | uniq -c
```

[よく使う数え方のフレーズ]

- `sort | uniq -c`
- `sort | uniq -c | sort -n`

数え上げ(2/2)

sortとuniqの使い方が分からない人向け回答

```
$ cat holidays | \
  grep '2016' | awk '{print $2}' | \
  awk '{a[$1]++}END{for (k in a){print a[k],k}}'

$ cat holidays | \
  grep '2016' | awk '{print $2}' | \
  perl -0777 -anlE 'grep{!$h{$_}++} @F; say
"$h{$_} $_" for keys(%h)'
```

Q6

2016年08月03日 から

2016年09月30日 まで表示してみましょう。

```
$ ? ? ?
2016-08-03 木 平日 0
～
2016-09-30 金 平日 0
```

あるパターンから
あるパターンまで抽出(1/2)

解答例

```
$ cat holidays | \
awk '/2016-08-03/,/2016-09-30/'
```

[特定の2パターン間の出力]

```
awk '/開始パターン/,/終了パターン/'
```

あるパターンから
あるパターンまで抽出(2/2)

別解

```
# sedを使う
$ cat holidays | \
sed -n '/2016-08-03/,/2016-09-30/p'
# perlを使う
$ cat holidays | \
perl -nle 'print if /2016-08-03/../2016-09-30/'
```

半ば強引に

```
$ cat holidays | \
  grep '2016-08-03' -A 50 | grep '2016-09-30' -B 50
```

Q7

2015年と2016年を比較して、
片方にしか存在しない祝日を表示して下さい。
２つあります。

```
$ ？ ？ ？
#  実際解いて確認してみてね
```

プロセス置換をつかおう(1/3)

解答例

```
$ diff \
<(grep 2015 holidays | awk '{print $3}' | sort | uniq)
<(grep 2016 holidays | awk '{print $3}' | sort | uniq)
```

[プロセス置換]

<(〜) という表現は「プロセス置換」。

<(〜) の箇所は、 〜 に書いたコマンドの結果を記述したファイルのように振る舞う。

プロセス置換をつかおう(2/3)

プロセス置換例

```
$ echo A B C > abc.txt

$ cat abc.txt | sed 's/A/B/'          # (1)
B B C

$ cat <( echo A B C ) | sed 's/A/B/' # (2)
B B C
```

(1)と(2)のコマンドは結果的に同じ。

プロセス置換をつかおう(3/3)

プロセス置換を使わない別解

```
$ cat holidays | grep -e '2015' -e '2016' | \
  awk '{print $3}' | sort | uniq -u
```

Q8

※ 上級者向け問題

有給を使って計画的に連休を作るために

2015 〜 2021年の間で

休日→平日→休日

となっている3日間を全て抽出してみましょう。

```
$ ? ? ?
2015-11-01 2015-11-02 2015-11-03
2016-02-11 2016-02-12 2016-02-13
2016-05-01 2016-05-02 2016-05-03
…
```

年月日がわかればよいので、曜日や休日フラグを出力してもOK。

出力方法はおまかせします。

注意

複数行にまたがって
重複する日があるかも。

{08-09, 08-10, 08-11} のトリオ

{08-11, 08-12, 08-13} のトリオ

日付	曜日	内容
2020-08-09	日	休日
2020-08-10	月	平日
2020-08-11	火	山の日
2020-08-12	水	平日
2020-08-13	木	お盆

一行内でループを回そう

解答例

```
$ cat holidays | tr '\n' '\t'| awk -F'\t' \
'{for(i=3;i<=NF;i++){print $(i-2),$(i-1),$i}}' \
| grep ' 1.* 0.* 1$'
```

少しわかりやすく

```
cat holidays | tr '\n' '\t'| awk -F'\t' \ # タブ文字を区切り文字に指定
'{
  for(i=3;i<=NF;i++){       # NF(Number of Field)…フィールド数
    print $(i-2),$(i-1),$i # 例: iが3の時$(i-1)は$2として評価される
  }
 }' \
| grep ' 1.* 0.* 1$' # 1(休日) → 0(平日) → 1(休日) (スペース注意)
```

便利な道具を活用しよう

egzact（最終スライド参照）の

convコマンドがあればシンプルに組める！

（手前味噌失礼）

```
$ cat holidays | tr '\n' '\t' | \
  conv fs='\t' 3 | \
  grep ' 1.* 0.* 1$'
```

convコマンド

フィールド区切りの入力を螺旋状に繰り返す。

```
$ echo 'A B C D E' | conv 3
A B C
B C D
C D E
```

このファイルを使った便利技(1/3)

「2ヶ月でできるよね？」と言われた時

→ 実際の営業日数は？

→ 大型連休を挟むと意外に少ない

```
$ cat holidays | \
  awk '/2016-08-01/,/2016-09-30/' |\
  grep '0$' | nl
```

営業日だけ抽出

→ エクセルに貼り付けて進捗管理（白目）

このファイルを使った便利技(2/3)

今年の有給いくつ温存しようかな……。

休日→平日→休日

```
$ cat holidays | grep '2016' | \
  tr '\n' '\t' | conv fs='\t' 3 | \
  grep ' 1.* 0.* 1$'
```

休日→平日→平日→休日

```
$ cat holidays | grep '2016' | \
  tr '\n' '\t' | conv fs='\t' 4 | \
  grep ' 1.* 0.* 0.* 1$'
```

このファイルを使った便利技(3/3)

次の素数の日はいつだろう？気になるなあ

```
$ cat holidays | grep '2016' | \
  tr -d '-' | awk '$0=$1' | \
  factor | awk '$0*=!$3' | \
  sed -r 's/(..)(..)$/-\1-\2/' | \
  grep -f - holidays
```

まとめ

入門者はボトムアップ学習重視が吉だと思います

慣れてきたらやりたいことを作ってトップダウン

やり方がわからない時は……

→ ググってリバースエンジニアリング

→ つよいシェル芸人に聞く

毎日叩くとマイナーな操作・オプションも覚えてくる

なんとか習慣化させるとつよくなれる。

今日の資料の気になった部分を一旦丸暗記するのも手。

参考文献・URL

石井さんの「シェル芸入門」

http://slideck.io/github.com/hisaharu/shellgei/slides/shellgei23AM2.md#/

斎藤さんのAWK一行野郎百烈拳

http://gauc.no-ip.org/awk-users-jp/material/100_one_liners_20131223.pdf

シェルプログラミング実用テクニック

http://gihyo.jp/book/2015/978-4-7741-7344-3

シェルスクリプト基本リファレンス

http://gihyo.jp/book/2011/978-4-7741-4643-0

Software Design 2016年6月号 bash 再入門

http://gihyo.jp/magazine/SD/archive/2016/201606

Perl One-Liners (※洋書)

http://shop.oreilly.com/product/9781593275204.do

支援ツールURL

egzact — 組み合わせ列挙が軽々できるようになる

https://github.com/greymd/egzact

Open usp Tukubai — DBいらずで集計作業がラクチンになる

https://github.com/usp-engineers-community/Open-usp-Tukubai

nixar — 地味にやりにくい作業が楽になるかも

https://github.com/askucher/nixar

MCMD — 億単位のデータに複雑な統計処理ができる

http://www.nysol.jp/

Cureutils — プリティでキュアな便利コマンド

https://github.com/greymd/cureutils