Hatena

Linuxカーネルの読み方

Naoya Ito

担当章

- リバース **proxy**
- キャッシュ
- **Linux**
- **Web**サーバー

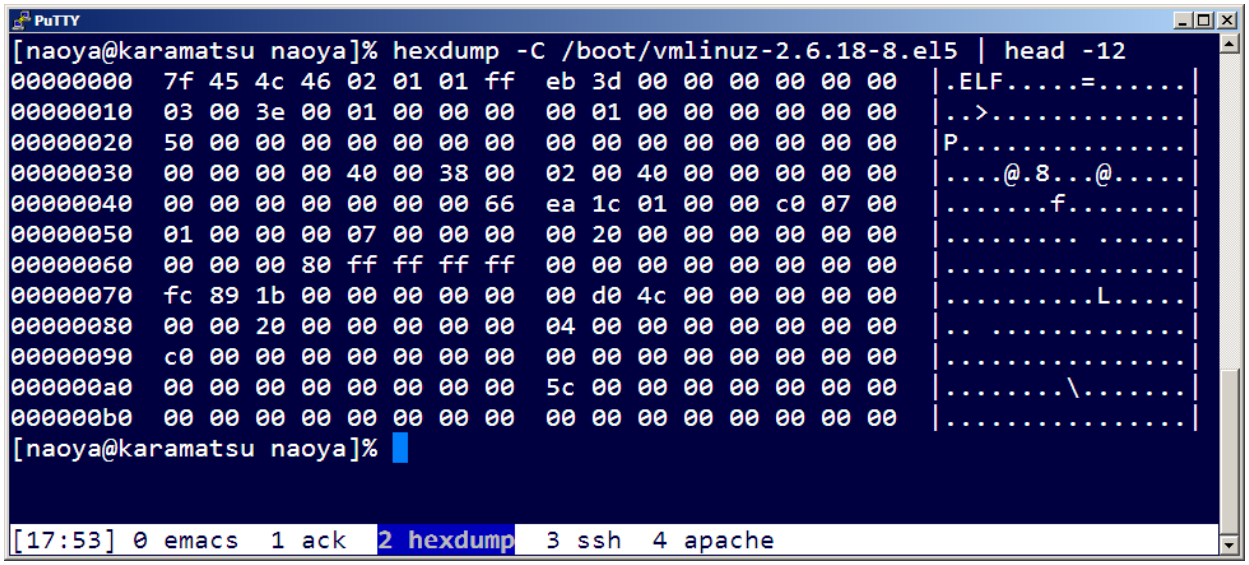
Hatena

Linux カーネルを読む

- 「よーし、**Linux** カーネル読むぞー」

Hatena

早速カーネルを読む!



- 「なんとかわいいバイナリ」
- 「**0x000000c4** は俺の嫁」
- 「いやいや、そこは **0x0000006d** でしょう**?** なおや氏」

Hatena

… という話ではなく

- ~~**Linux** カーネルの読み方~~
- **Linux** カーネルのソースコードの読み方

Hatena

お話したいこと

- **Linux**カーネルを読む
- カーネルのソースを読んで良かったこと
- **Linux** カーネルに触れたきっかけ
- カーネルソースの読み方

Hatena

Hatena

カーネルのソースを読んで良かったこと

以前の私#1

- 負荷が過剰
  - 「このサーバーはディスクが重そう」
  - (実際どこが正確な原因なんだろう...)
  - (そもそも"負荷"って何だろう...)

Hatena

以前の私#2

- **load average が 10.00**
  - 「**Google** で調べると**CPU**数で割って **1.00** 以下なら **OK** らしい。」
  - **(**なんで**CPU**数で割るんだろう**...)**
  - **(**そもそも **load average** って何だ**...)**

Hatena

以前の私#3

- **Apache** がアクセス過多
  - 「**prefork** から **worker** にすればマルチスレッドになって速くなるかも」
  - **(**なんでスレッドだと速いのかしら**...)**
  - **(**とりあえず収まったから良いか**...)**

Hatena

以前の私#4

- **I/O** が高くてディスクがもたない
  - 「サーバー増やしてみる**? SCSI** のディスクに変えてみようか」
  - **(I/O** ってどうやったら減らせるの**...?)**
  - **(SCSI** にしたところで大して速くならない**...)**

Hatena

根拠がない

- 重そう
- **Google**で調べると
- 速くなるかも
- 変えてみよう

根拠がないので対症療法しか取りようがない

Hatena

現在の私#1

- 負荷が過剰
  - **OS** の出力する統計情報から原因を特定
- **load average** が **10.00**
  - **load average** を上昇させるプロセスを特定
  - プロセスの挙動からボトルネックを特定

Hatena

現在の私#2

- **Apache** がアクセス過剰
  - なぜ応答を返せないか原因を統計から特定
  - プロセス **or** スレッドは両者の差異点がどこかを考慮して選択
- **I/O** が高くてディスクが持たない
  - **I/O** 分散計画をキャッシュを前提に立てる

Hatena

カーネルのソースに触れたことで

- **OS** の動作原理や統計情報の意味をソースコードレベルで理解することができた
- 根拠を持って負荷分散戦略を立てられるようになった
- スケーラブルなアプリケーション設計に自信が持てるようになった

抽象レイヤが透けて見える

Hatena

結果

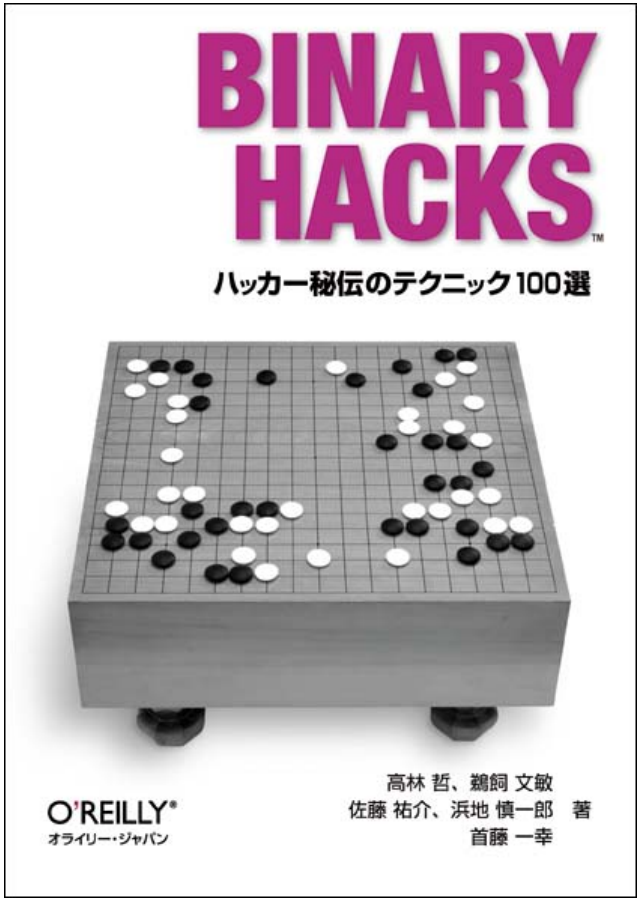
- 『サーバー/インフラを支える技術』執筆

Hatena

Hatena

Linux カーネルに触れたきっかけ

バイナリアンになりたくて



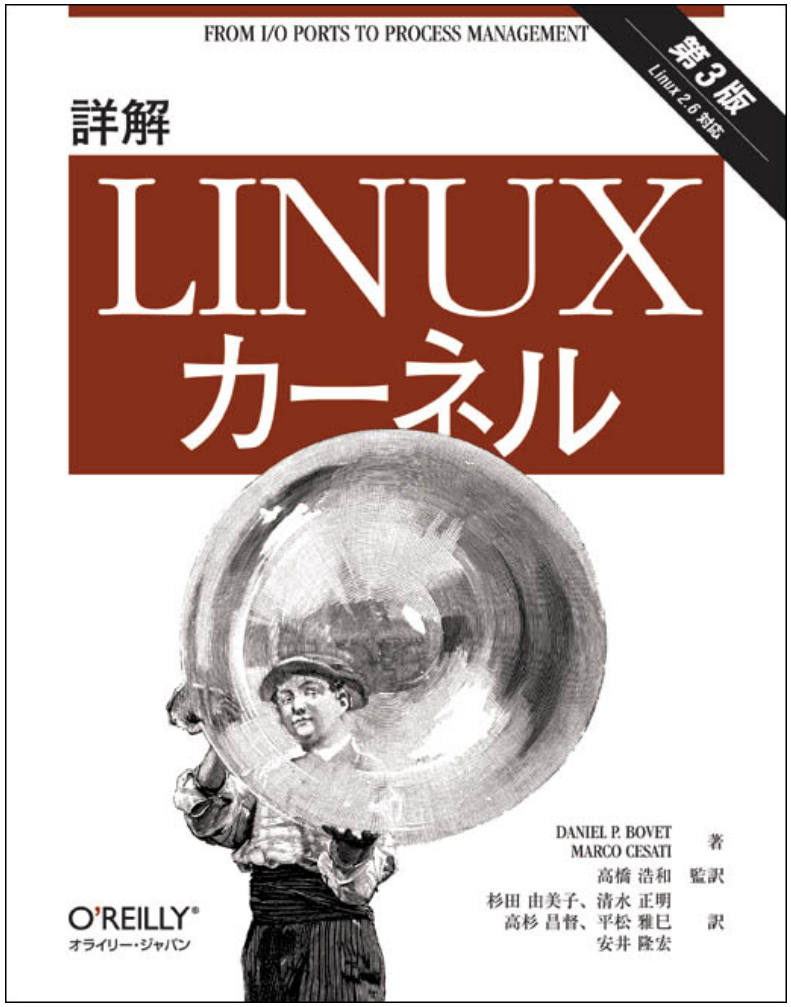
Hatena

苛立ち

- ローレイヤのことが分からない
  - 分かりたい
- 何に手をつけて良いかが分からない

Hatena

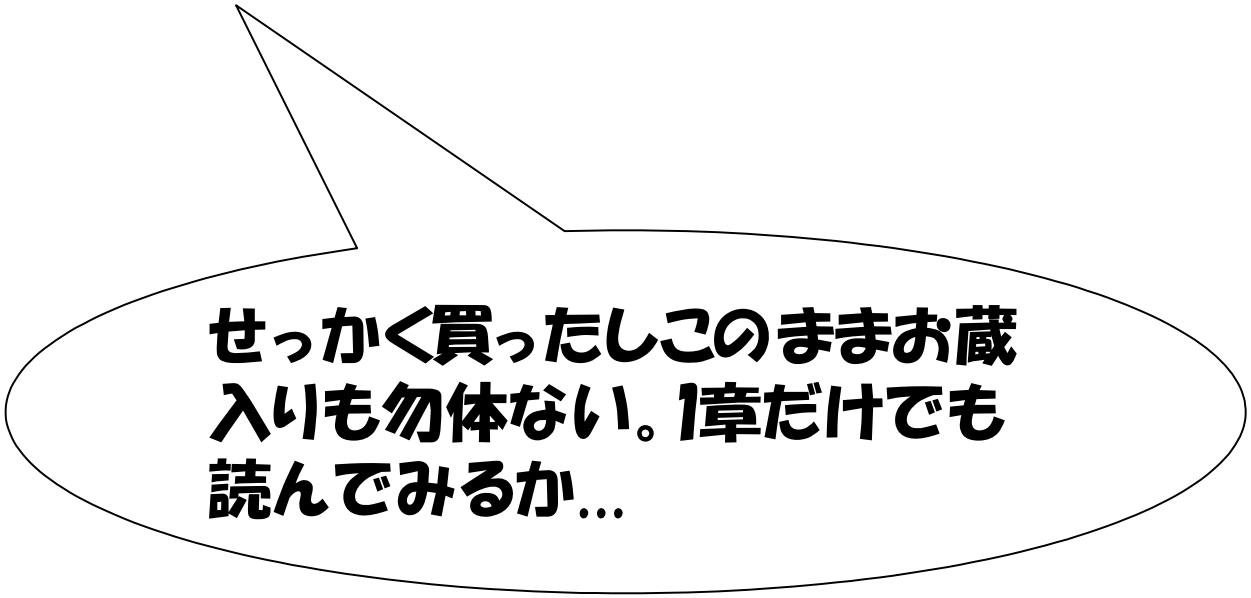
きっかけ


FROM I/O PORTS TO PROCESS MANAGEMENT
第3版
Linux 2.6 対応
詳解
LINUX
カーネル
DANIEL P. BOVET
MARCO CESATI
著
高橋 浩和 監訳
杉田 由美子、清水 正明
高杉 昌督、平松 雅巳
安井 隆宏
訳
O'REILLY
オライリー・ジャパン

Hatena

Linuxカーネル本の概論を読んだ

- 3版発売決定
  - 2版をずっと「積ん読」にしていた



Hatena

知りたかったことが書かれていた

- プロセスとは何か
- スレッドとは何か
- 仮想メモリとは何か
- ファイルがどうキャッシュされるか
- システムコールの裏側

Hatena

ソースを開く

- **load average** とは何かを調べる
  - 半端無く難しいと思い込んでいた
  - 書籍片手に **grep** したら意外と分かった

Hatena

Linux カーネルの設計

- モノリシックカーネル
- 複雑な設計パターンはあまりない
  - ややオブジェクト指向 **(VFS** など**)**
  - ただし量は多い **... 500M**ステップ

Hatena

カーネル開発者に感謝

- ソースコードの裏には幾多の試行錯誤
  - 開発者 **>>>>** ソース読者
- 素晴らしい学習教材に感謝

Hatena

Hatena

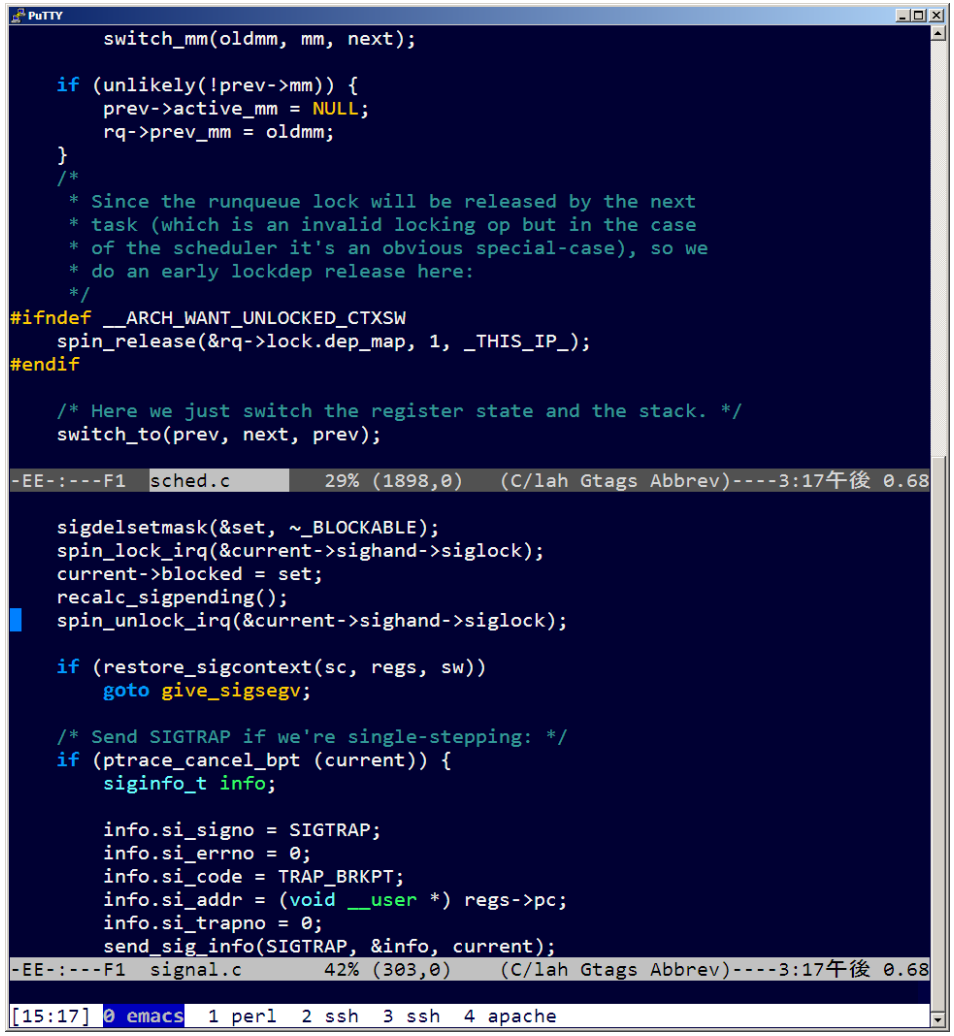
カーネルソースの読み方

使っている道具

- **GNU Emacs**
- **GNU GLOBAL (gtags)**
- **App::Ack (ack)**
- 書籍数冊
- 紙とペン

Hatena

Emacs



Hatena

GNU GLOBAL

- ソースコードタグシステム
  - 関数の定義場所、呼び出し場所にジャンプ
  - **Emacs** と連携

```
if (likely(prev != next)) {
    rq->nr_switches++;
    rq->curr = next;
    ++*switch_count;

    context_switch(rq, prev, next);
} else {
    ...
```

sched.c 3517行目

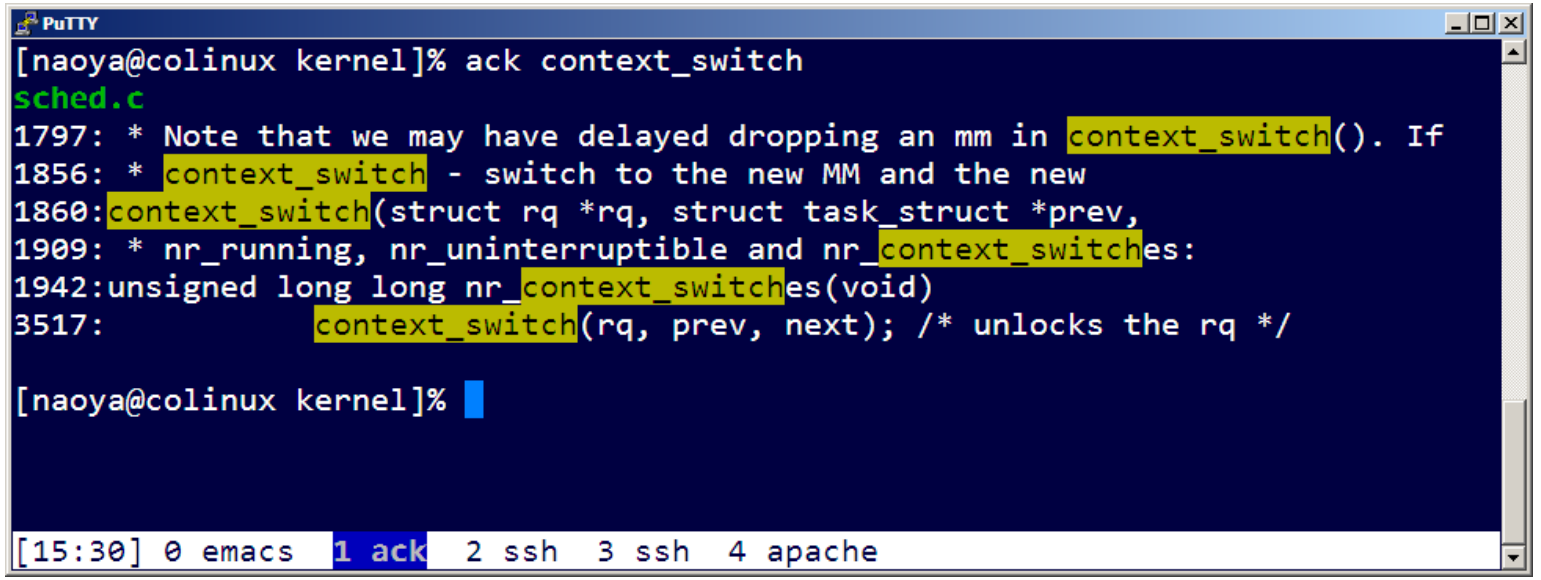
```
static inline void
context_switch(...)
{
    struct mm_struct *mm, *oldmm;

    prepare_task_switch(rq, prev, next);
    mm = next->mm;
    ..
```

sched.c 1860行目

Hatena

App::Ack

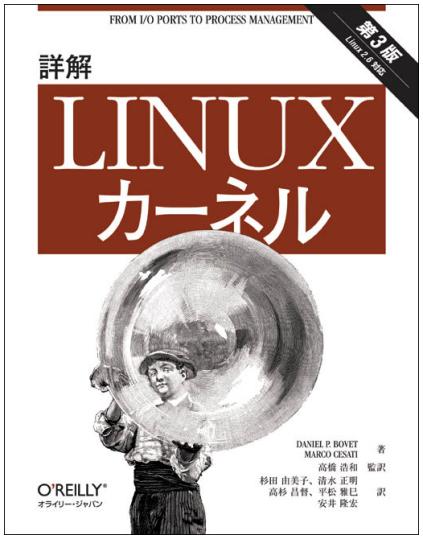
- grep よい良いソースコード検索ツール
  - % sudo cpan App::Ack

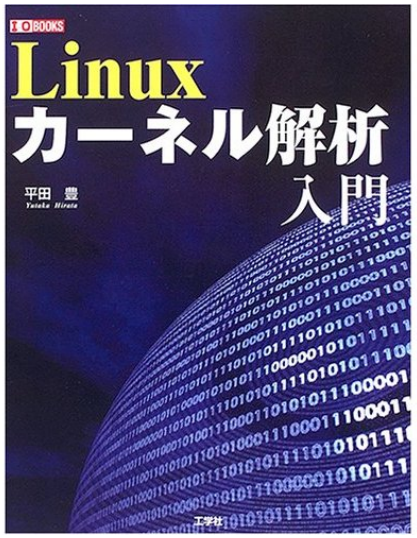


Hatena

書籍数冊

- 別の書籍の同テーマの解説を読む





Hatena

紙とペン

- 処理の流れをメモしながら進める
  - コールスタックが深いと迷子になりやすい

Hatena

Linux Kernel Hack Japan

- http://hira.main.jp/wiki/index.php
  - "ひら"さんによる膨大な覚書き
  - ひらメソッド
  - 分からない時のヒントに

Hatena

ソースを読む**5**つのコツ

- 読む対象を絞る
- いきなり読まない
- 概観を知った上で読む
- 重要なデータ構造を把握する
- テストプログラムのトレースを活用する

Hatena

1. 読む対象を絞る

- 隅々まで読もうとしない
  - **500M**ステップも読めない
  - **(**自分もまだ数**%**です**)**
- テーマを決めて読む
  - 「**load average** の計測方法を調べる」
  - 「コンテキストスイッチの仕組みを調べる」
  - 「**sel ect (2)** の実装を調べる」

Hatena

2. いきなり読まない

- まず全体を把握する
  - ソースツリーの構造
  - 本の目次を読むのと同じ

Hatena

3. 概観を知った上で読む

- 書籍で知りたい箇所の概観を掴む
  - スケジューラの概要 … ふむふむ、なるほど
    → ソース読む
  - 大枠が分かっていれば迷子にならない

Hatena

4. 重要なデータ構造を把握しておく

- 重要なデータ構造
  - **`struct task_struct`**
  - **`struct mm_struct`**
  - **VFS の `struct file_operations`**
  - **`...`**

Hatena

5. テストプログラムを活用

- 例: スレッド生成周りの処理が知りたい
  - **pthread_create()** するプログラムを作る
  - プログラムを **strace** する
  - システムコールの呼ばれ方 → 実装箇所の当たりが付く

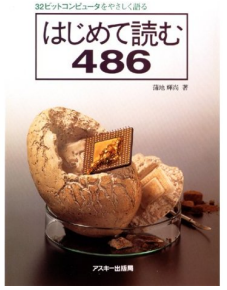
Hatena

例: **usleep(3)** の精度が知りたい

- **http://d.hatena.ne.jp/naoya/20080122**
  - まずテストプログラムで実験
  - **strace** で **nanosleep(2)** の呼び出しを確認
  - 書籍の索引で **nanosleep** を引く
    - タイマ割り込み周りの章を読む
    - タイマ割り込み周りのコードの当たりがつく
  - **ack** や **GNU GLOBAL** を駆使して **sys_nanosleep( )** の実装/周辺を芋づる式に調べていく **(**道筋の記録を忘れずに**)**
  - **nanospeel(2)** の母体になっている動的タイマが割り込み周期に依存していることが分かる → **4ms**

Hatena

読んで分かった必要な知識



- C言語
- gcc 拡張
- GNU開発ツール
- CPU の機能 (x86)
- オブジェクト指向
  - 簡単なパターン (Template Method など)
  - Writing a Simple File System
    - http://www.geocities.com/ravikiran_uvs/article

Hatena

番外編

- 手を動かしたい方は...
  - カーネルモジュール開発が面白い
  - **http://d.hatena.ne.jp/naoya/20071201**
    - フィボナッチ数列を計算するデバイス (アホ)

```
% cat /proc/fib
10
% cat /dev/fib
55
```

Hatena

まとめ

- 巨大なソースに当たるにはコツがあります
  - 堀から埋めていきましょう
- カーネルのソースを読むと負荷分散に強くなります。自信が持てます
- カーネルのソースを読むことで**OS**の動作原理を知ることができます

Hatena

最後に

- カーネル? 何の役に立つの?
  - 「ただ知りたかった」
  - カーネルへの興味が負荷分散に繋がるとは思っていなかった

役に立つかどうかは知らなければ分からない。
知りたいと思う動機を大切に。

Hatena

Hatena

Thank you!