



LPI-Japan主催 LPICレベル1技術解説無料セミナー

2012/12/09

株式会社ネットマイスター

CEO

岡田 賢治



■本日のセミナーの流れ

本日のセミナーの流れ

自己紹介

LPICについて

セミナーについて

試験準備の進め方

自己学習環境の整備

各項目の説明



- 自己紹介
 - 株式会社ネットマイスターの代表/CEO
 - リストラが無い会社です。
 - 業務内容:
 - ドキュメント書き
 - 専門学校非常勤講師(8年目)
 - 受託開発(JavaによるWebアプリ・PHPのWebアプリ・JavaのSwingアプリ・Perlのクローラ等)
 - 「IPAにつとめたことは、ただの1度も無い！」
- 絶賛お仕事募集中です。



E-Mail: okada.knj@gmail.com



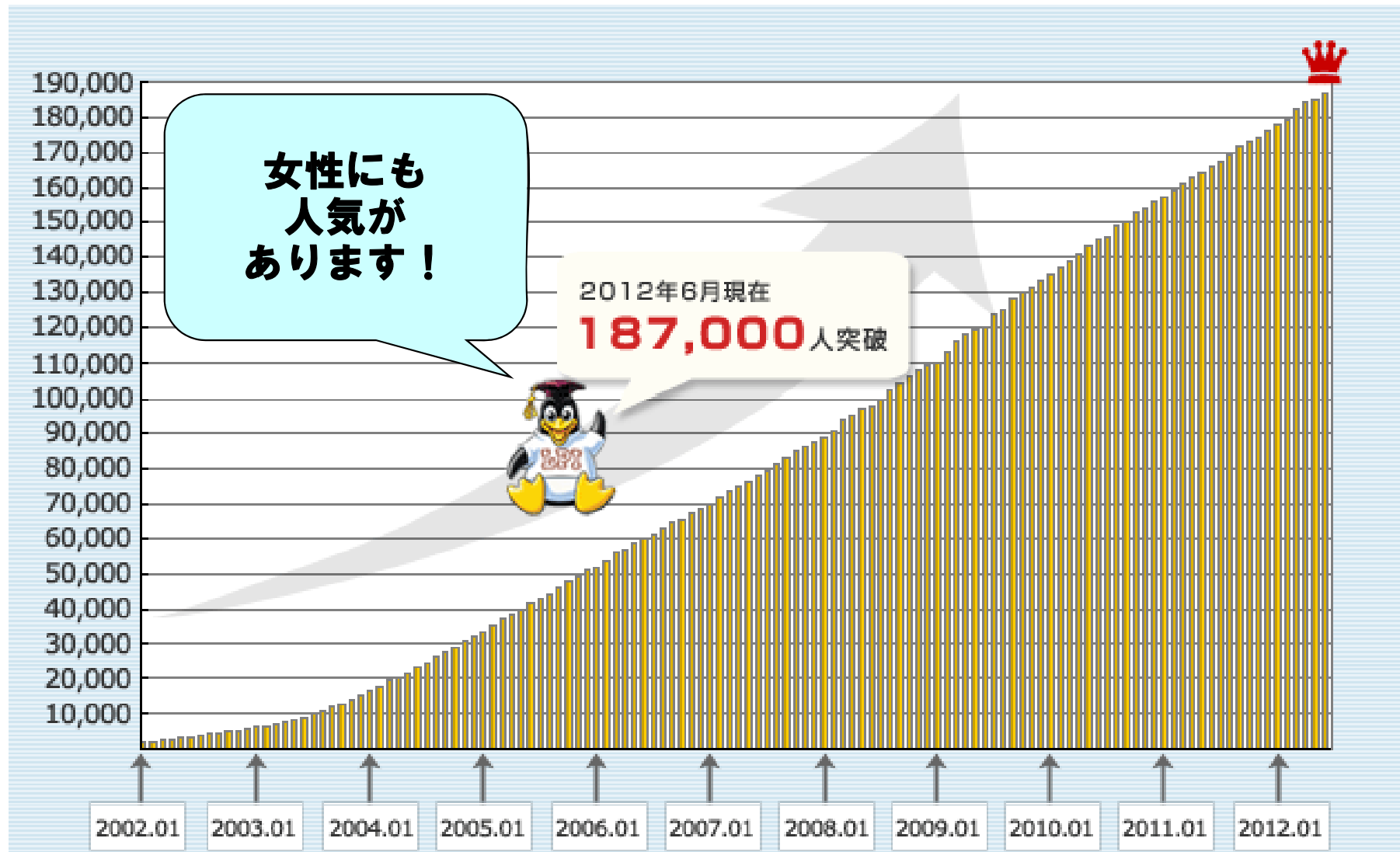
- 普段は、自宅作業でMac。常駐作業でLinuxを使用しています。
- 自宅サーバが存在。1台のマシンで12個のLinuxが同時に動いています。
メールサーバとかは、全部そいつでやらせています。
- Linux歴は、11-12年。
- UNIX歴を含めると、19年以上触っています。
- いわゆる「サバ管」上がりです。
なので、妙な経験値だけは
いっぱいあります。





Linuxの普及により、LPIC受験数が伸び続けている

2012年6月現在





LPIC体系

株式会社
ネットマイスター

- Linux OS と関連技術を用いるIT専門家の能力を認定
- 3段階のレベル別に設計(レベル1、2、3)
- **GLOBAL**
LPICは世界で実施されている国際的な認定制度
- **STANDARD**
Linux技術者の認定制度として世界最大、日本最大
- **NEUTRAL**
LPICはベンダーに依存しない中立な認定制度

ITSS
レベル3

ITSS
レベル2

ITSS
レベル1

試験 No.102
Linux一般2

試験 No.101
Linux一般1

試験 No.202
Linuxネットワーク管理

試験 No.201
Linux応用管理

試験 No.301
Core

試験 No.304
Virtualization & High Availability
(2011年1月リリース)

試験 No.303
Security
(2009年2月末リリース)

試験 No.302
Mixed Environment

い
ず
れ
か
選
択

LPICレベル1

サーバ運用管理レベル

(2試験)

LPICレベル2

ネットワークシステム
構築レベル

(2試験)

LPICレベル3 Core

エンタープライズ
システム構築レベル

(1試験)

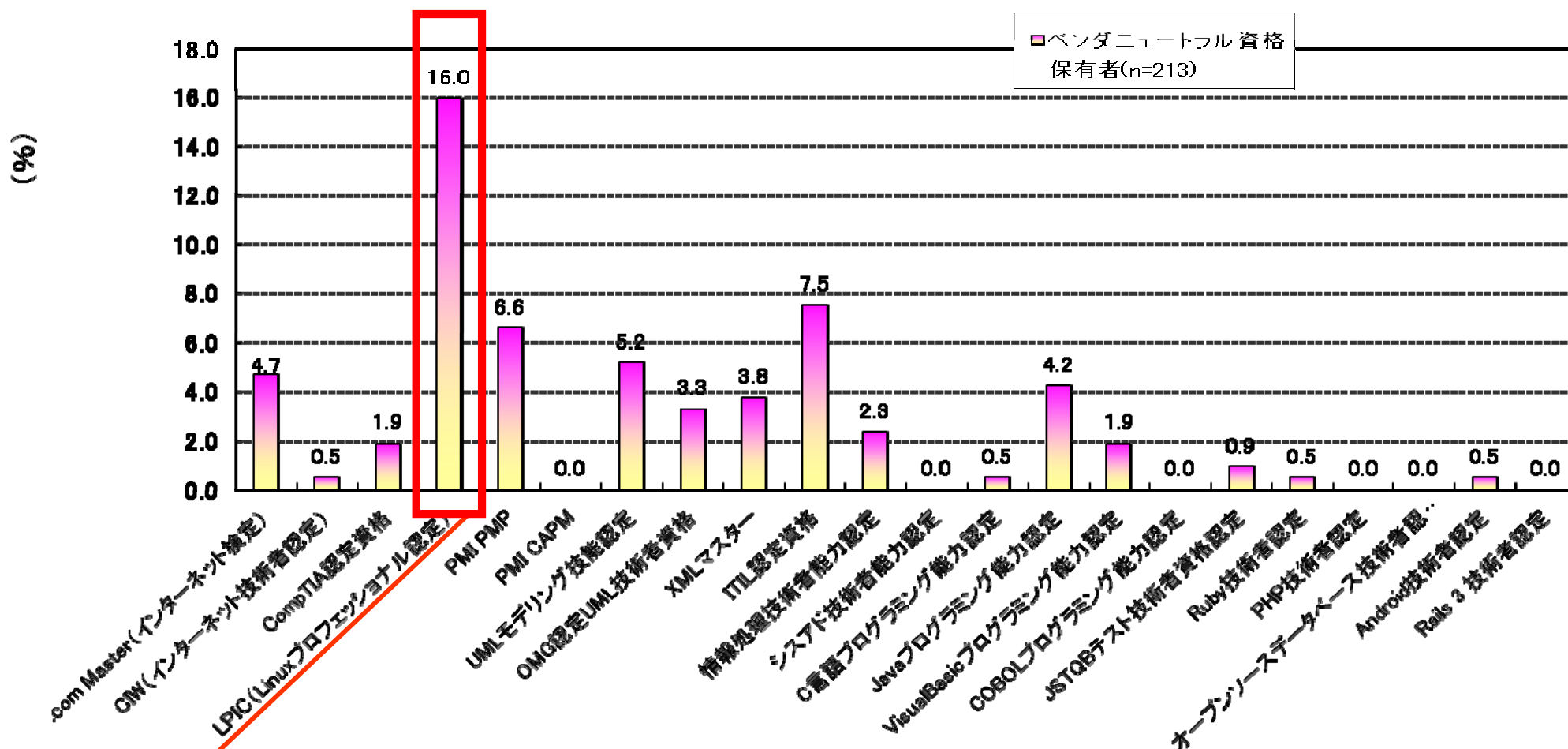
LPICレベル3 Specialty

各分野の最高技術レベルの
スペシャリスト

(1試験を選択)



アイティメディアによるアンケート
「最もキャリアアップに繋がったベンダニュートラル資格」において、
LPICが1位！



出典:ITメディア@IT自分戦略研究所 読者調査結果(2011年11月、回答213)



- これからLPIC Lv1受験の準備を始める人、ある程度始めている人が対象
- 個々の項目については、よく理解できると思います。
- でも、知識が「バラバラ」になりませんか？
- それらが有機的につながると理解度も向上（すると考えています）
- その「つながり構築」の手助けになるセミナーになれば、と考えています。
- Linuxが好きになっていただけたら・・・



- 岡田の保有資格 (LPI-Japan関連)
LPIC Lv1 (2011年12月取得 101, 102)
OSS-DB Silver (2007年当時 PostgreSQL CEとして取得)
304 (仮想化・高可用運用 2012年6月取得 Lv2, 301を持っていないので、Lv3 Speciality認定は無し)
- Lv1は、職業・経歴より「とれて当たり前」・・・逆に怖い
- 304は、「急いで取ってください」と言われ、6月になんとか
- みなさんのLv1は、私の304の経験に近いのではないかと
- その経験談を1つ



■すべきこと

- コマンドを徹底的に覚える。特にオプション周り。
ls の -a オプション(. で始まる隠しファイルも表示する)に、--all というのがあるのをご存知ですか？両方とも覚える。
- 体系的な理解。「一行文章」では無く、全体を理解するように。
304は範囲も広く、何よりも資料が少ない(対策本は1冊だけ)
このセミナーも「体系的な理解」を目指している。
個人的には、体系的に理解していた分野が多く出題されたので、304が合格できたと判断
- 試験を予約してください
基本情報処理技術者試験は、日程が確定している
LPICやCCNA等は、日程が自由に設定できる
＝いつでも受けられるし「いつまでも延ばせる」
1ヶ月先でも2ヶ月先でもいいので、必ず先に試験を予約すること。それを目指して勉強しましょう。



■すべきこと(続き)

- ムラを無くす

得意分野と不得意分野の差をできるだけ無くす
不得意分野があると、仮に合格しても試練の道が...

特に、すでにLinuxのオペレーションをしている方

- スケジュールには余裕を持って

「* *までにLPIC Lv1を取りなさい」

リテークポリシーの関係で、不合格後すぐに再受験できない

万が一不合格のことも考え、リテークポリシーから逆算して日程を決める

- 受験の最後は「深呼吸をして」

■すべきではないこと

- 周りに受験することを伝える

変なプレッシャー

逆にそれを利用？



■「必ず手を動かしてください」

Linuxを「体で覚えること」をお勧め

体を動かして覚える >> 本・ペーパーベースの勉強

実際にLinuxを触る手法

1. 実機を使う

(余っている)PCにLinuxをインストール

利点: 実際の運用環境に近い環境で学習可能

欠点: 余っているPCが必要(ノートPCはNG)

基本コマンド操作習得の前に、インストール方法を覚えることが
要求される

作業環境の保持が不可

(一度上書き保存してしまうと前の状態に戻れない)





2. 仮想PC環境を使う

■PCの上で、仮想マシンを実現するソフトウェアを利用し、「あたかも実機があるかのように」操作が可能

Win: VMWare Workstation, VMWare Player(*), VirtualBox(*)

Mac: Parallels, VMWare Fusion, VirtualBox(*)

利点: 既存のコンピュータ上で作業可能

スナップショットが利用できる(ことが多い)ので、作業の経過が保存可能

欠点: (有償のものは)購入する必要がある
基本コマンド操作習得の前に、インストール方法を覚える必要・・・1.と同じ

vmware®

|| Parallels®





- お勧めはRedHat系のディストリビューション
- RedHatというディストリビューションは有償
- CentOS・・・RedHatの互換ディストリビューションとして有名
- Fedora・・・RedHatが配布している、次期RedHatのベータ版
- パッケージ回りの手順(rpmコマンド等)が、そのまま利用可能
- 参考書等が充実している
- Linux標準教科書もCentOSがベース
- インストール時に、ファイヤーウォール機能・SELinuxの機能はOFFに・・・非常に重要な機能
- それらの機能により、学習時に教科書と「ずれ」が発生する可能性





■そもそもファイルとは？

コンピュータは計算機

元データ → 計算 → 新データ

メモリ → CPU → メモリ

- メモリにデータを置いておく必要がない
- 一時保存しておく
- 他のマシンへ持っていく

⇒補助記憶装置に出力・ファイルという形式

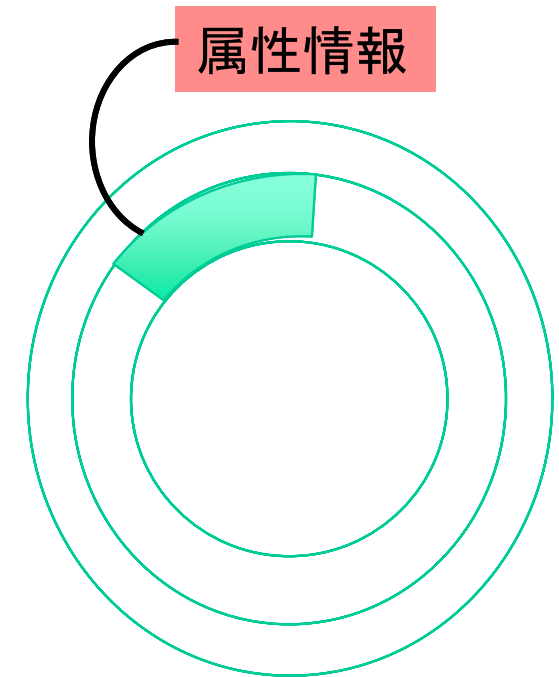
⇒UIはファイル操作を前提に設計

⇔メモリは「主記憶装置」



■ファイルの正体

- ファイルは、「属性情報」と「実体」からできている
ファイルの属性情報・・・ファイル名・ファイルサイズ・アクセス権・・・
ファイルの実体・・・ディスク上に置いてあるデータそのもの
- 「実体」と「属性情報」の関係が切れる＝「名無し」
⇒データではない！
⇒時が来たら上書きされる



■UNIX/Linuxにおける「情報」

UNIX/Linuxには、**i-node**という「属性情報」がある。



■i-nodeとは？

UNIX/Linuxには、i-nodeという「属性情報」がある。

i-nodeが保持しているデータ

ファイルかディレクトリか特殊 ファイルか	ハードリンクの数
アクセス権	ファイルサイズ
ファイルの所有者	ファイルの作成日時
ファイルの所属グループ	ファイル名

どこかで見たことはありませんか？

実は、lsの-lオプション出力です。

```
# ls -l AAA
```

```
-rw-r--r-- 1 root root 3490 2010-04-07 13:53 AAA
```



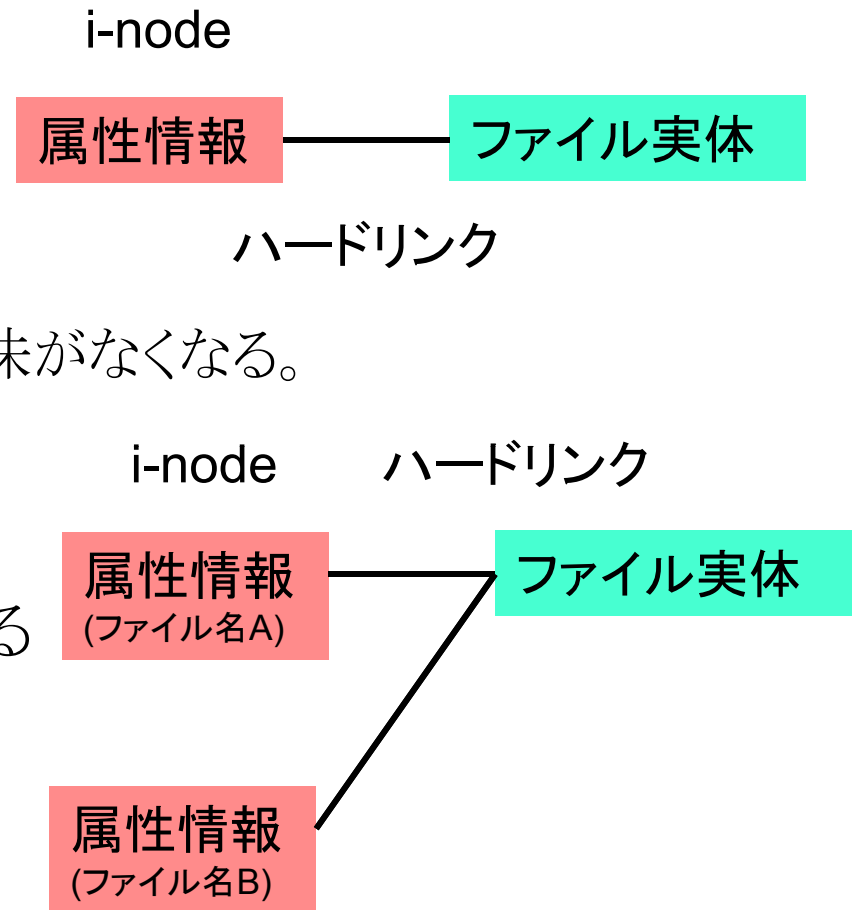
■リンク

- ハードリンクとシンボリックリンクがある
- ハードリンクは、いくつ作ってもディスクの使用量は変わらない
- シンボリックリンクは、リンク先を消すと意味がなくなる。

ファイル名が**A**というファイルがある
実体には、複数の**i-node**情報を対応付ける
これを実現するのがハードリンク

In A B

ファイル名**A**の指す実体に対して、ファイル名**B**という**i-node**”も”作ってくれ。
「実体の数が増えない」「ディスクの使用量が変わらない。」





■シンボリックリンク

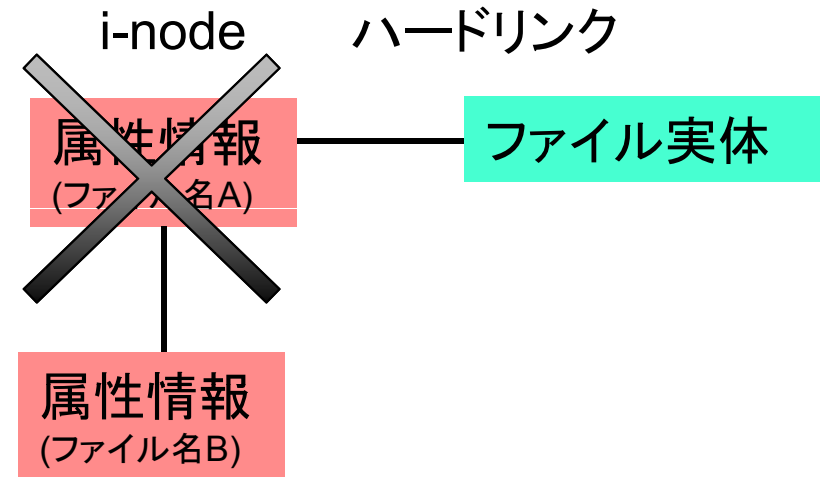
i-nodeに対して対応付けをする

`ln -s A B`

Aを削除すると、

- 実体は「名無し」
- Bは指している相手がいなくなる。

⇒「シンボリックリンクは、指している相手を消してはいけない」



ハードリンクの参照値

```
-rw-r--r-- 1 root root 3490    2010-04-07  13:53    AAA
```



■リンク

リンクを消す

実体は「名無し」・・・ファイルではない

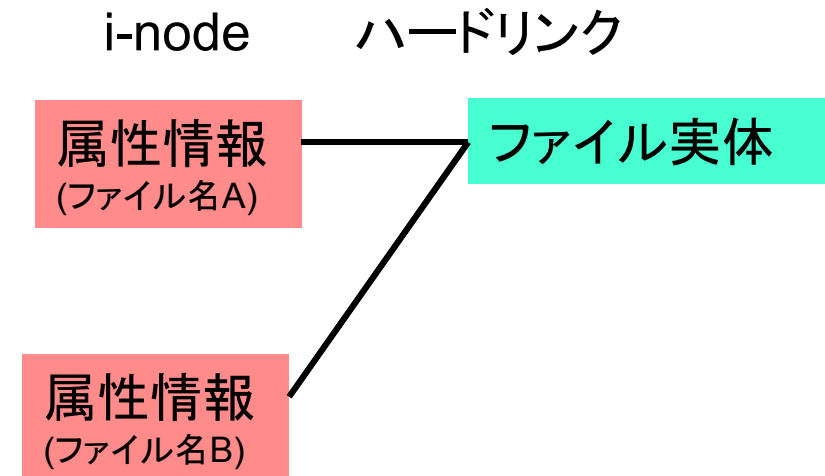
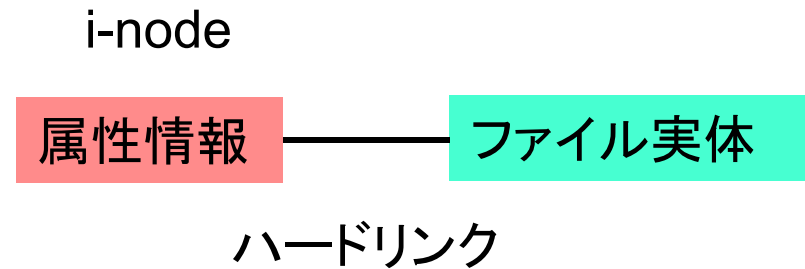
これがrmの正体(主題103.3)

リンクを、作って消す

ファイル名**A**のファイルへ、
ファイル名**B**のリンクを作る

ファイル名**A**の属性情報を消す

これがmvの正体(主題103.3)





■ハードディスクの分割

ハードディスクは、Ultra-ATAとSCSIが存在

ハードディスクの領域にデバイスのIDが割り当てられる

Ultra-ATAが/dev/hdXX、SCSIが/dev/sdXXになる・・・最近ではsdaで統一

1台目の装置・・・a、2台目の装置・・・b

1つめのパーティション・・・1、2つめのパーティション・・・2

■例:

Ultra-ATAの1台目のディスクで、3つめのパーティション・・・/dev/sda3

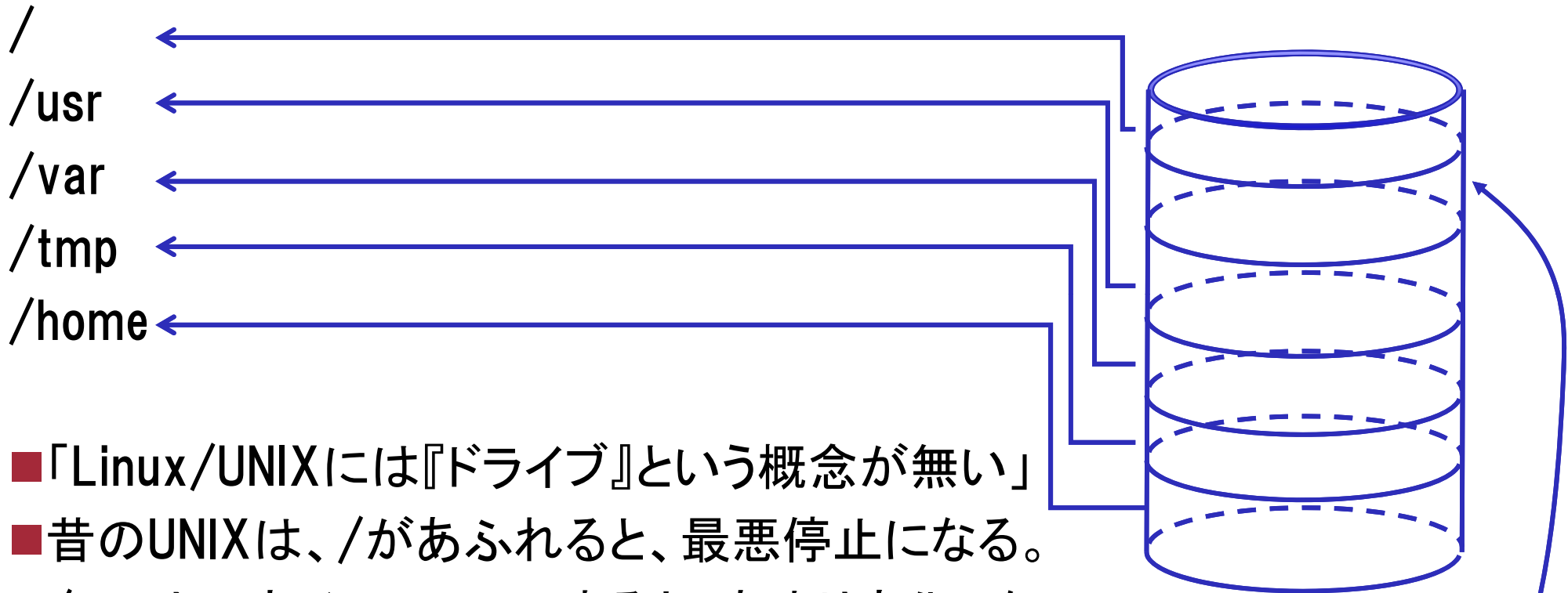
SCSIの3台目のディスクで、4つめのパーティション・・・/dev/sdc4

Serial-ATAは・・・/dev/sdXXで認識。SCSIとしての扱いになります。

USBの外付けHDD, フラッシュメモリ等・・・/dev/sdXXで認識。



■ ディスクの領域をディレクトリに割り当てる



- 「Linux/UNIXには『ドライブ』という概念が無い」
- 昔のUNIXは、/があふれると、最悪停止になる。
- /usrは一度インストールすると、あまり変化しない
- /var, /tmp, /homeは、作業の内容によっては「あふれる」
- /のファイルシステムは、前3つのファイルシステムと分ける必要
- スワップ領域は、実メモリの2倍のサイズ



- ディスクを利用する場合は、初期化＝フォーマットが必要
ファイルシステムには、ext2, ext3, reiserfs等が存在
mkfs.{ext2, ext3, reiserfs}でフォーマットしてから利用

■ 「ファイルシステム」という単語

「ファイルシステム」の定義が数種類ある

1. ディスクをディレクトリに割り当てている構造
2. ディスクの1つの領域(/dev/sda1)
3. その領域がフォーマットされている、その種類(ext2, ext3…)

すべて「ファイルシステム」です。

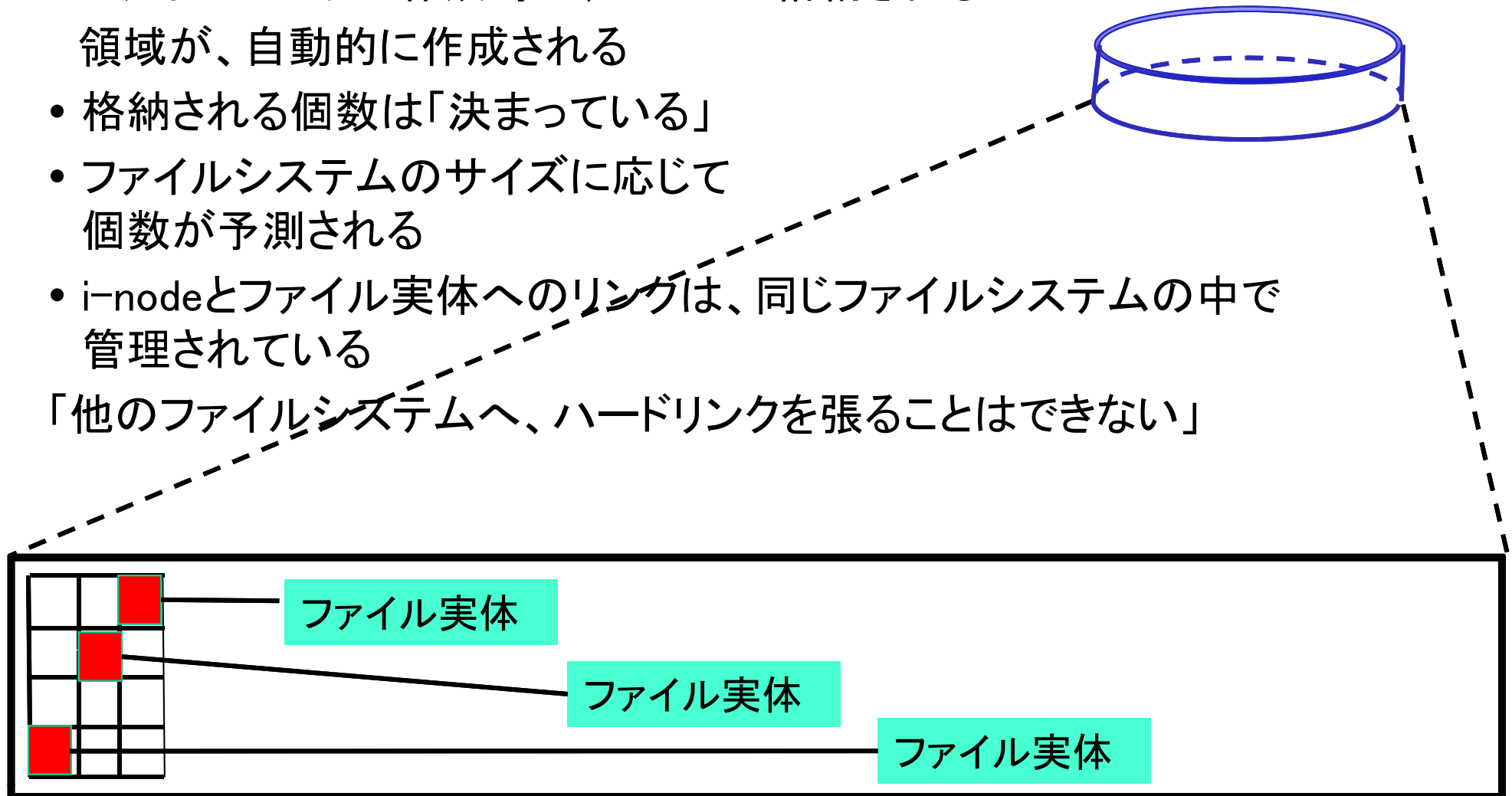
用法を間違えないように。

reiserには、i-nodeの仕組みがありません。



■ファイルシステムとi-node

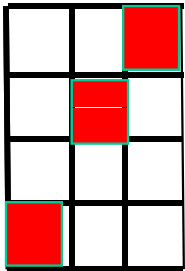
- ファイルシステム作成時に、i-nodeが格納される領域が、自動的に作成される
 - 格納される個数は「決まっている」
 - ファイルシステムのサイズに応じて個数が予測される
 - i-nodeとファイル実体へのリンクは、同じファイルシステムの中で管理されている
- 「他のファイルシステムへ、ハードリンクを張ることはできない」





■問題

ファイルシステムでi-nodeの個数が足りないというエラーが出た。



・・・がいっぱいになってしまった、ということ。

確認する方法: `df -i`

そもそも、どうして発生したのか？

ファイルの実体の消費よりも、i-nodeの消費が多かった
小さなファイルが、たくさん集まった・・・1つあたりのファイルサイズが、予想より小さかった。

mke2fsの-iオプションでファイルシステム作成時に設定する。



■最新ファイルシステム

- 機能の上限が上昇

	Ext3	Ext4	備考
ファイルシステムの最大サイズ	16TiB	4EiB	
1ファイルあたりの最大サイズ	2TiB	16TiB	
サブディレクトリの数	32000	無制限	

- ext3との互換性

本体側	Ext3	Ext4	Ext3	Ext4
ディスク側	Ext3	Ext3	Ext4	Ext4
機能	Ext3として動作	Ext3として動作	条件付き(*)でExt3として動作	Ext4として動作

「前方互換性」と「後方互換性」



■LVMとは

ディスクの構成で、「論理層」という仕組みを用いることで、より効率的に利用できる仕組み

登場人物は4人

PE(Physical Extent):

PVの構成要素・後程紹介

PV(Physical Volume):

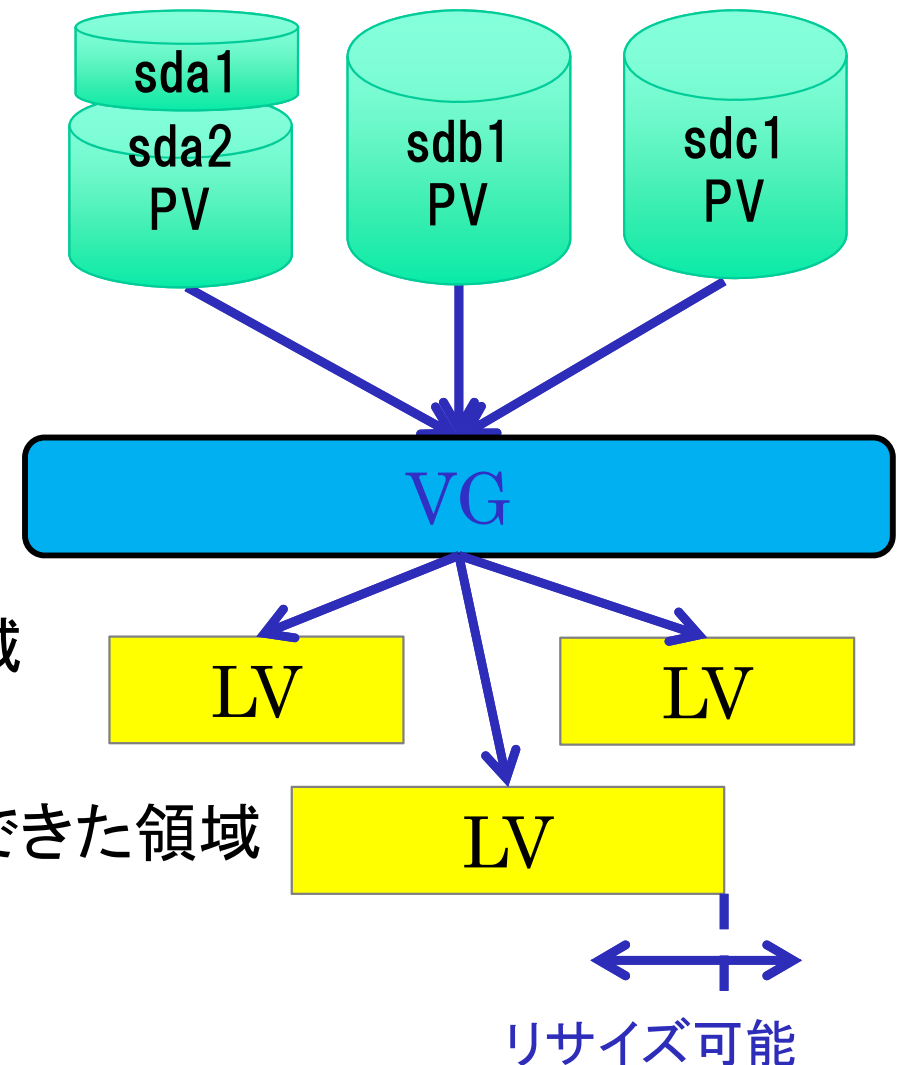
ディスクの一部やディスク全体を
構成する、文字通り物理的な領域

VG(Volume Group):

1つ以上のPVが組み合わさってできた領域

LV(Logical Volume):

VGから「切り出して」使う領域





■実際の流れ:

1. PVを作る

```
pvcreate /dev/sda2
```

2. VGを作成する

```
vgcreate lpic.vg /dev/sdb1
```

VGに参加させることもできる

```
vgextend lpic.vg /dev/sdc1
```

3. VGからLVを切り出す

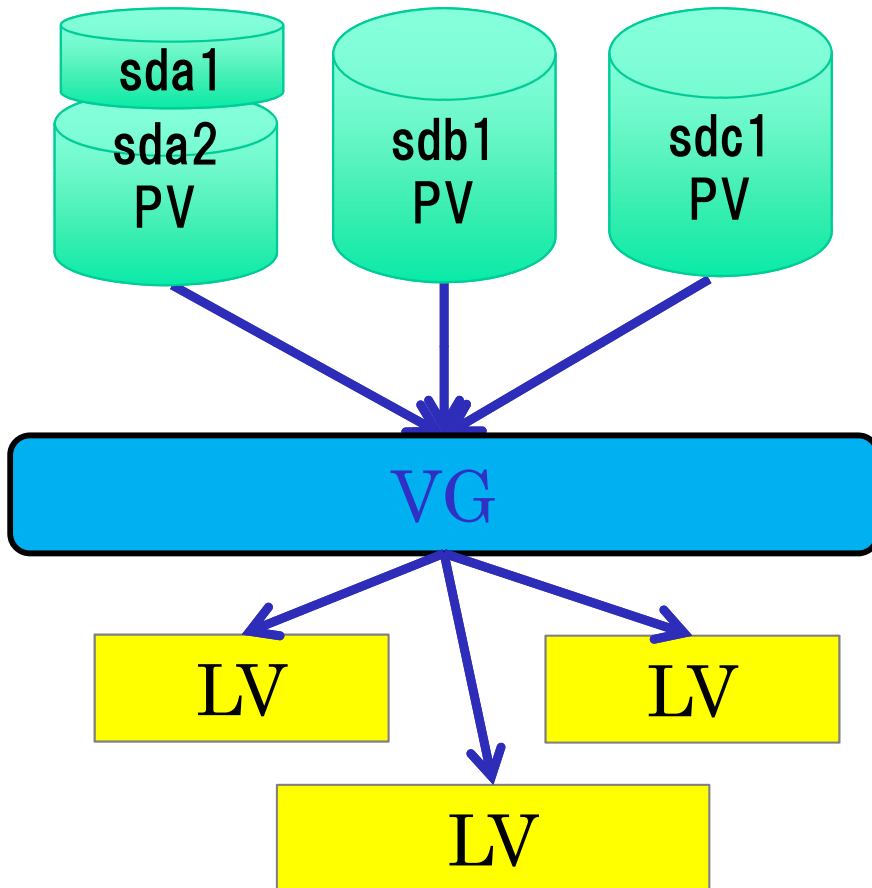
```
lvcreate -L 1G -n /dev/lpic.vg/lv.data
```

4. lvをフォーマットして、マウント

```
mkfs.ext4 /dev/lpic.vg/lv.data
```

```
mkdir /data
```

```
mount /dev/lpic.vg/lv.data /data
```





■LVMのコマンド群

PV

VG

LV

作成: create

削除: remove

検索: scan 全部あります

確認: display

上記区割りに入らないコマンド

pvmove: PVのディスクを未使用の状態にする

vgextend: PVをVGに追加する

vgreduce: VGからPVを抜けさせる

vgsplit: 1つのVGを2つのVGへ分割する



■LVMを用いる利点

シナリオ1: ディスク容量の不足

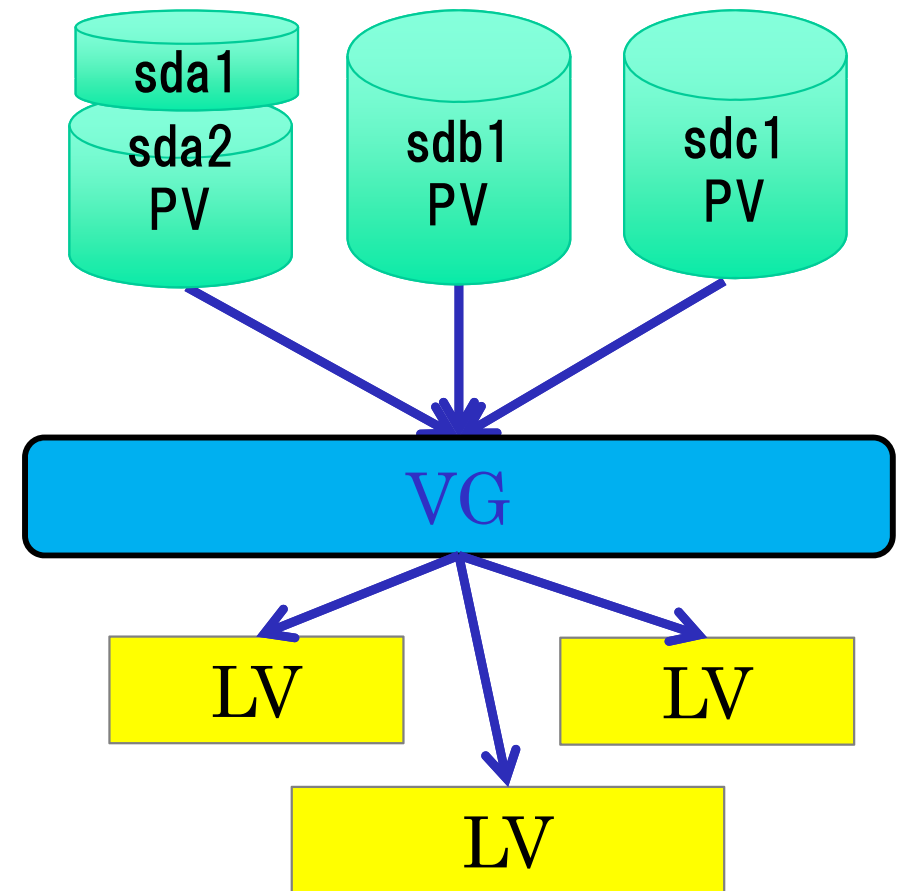
既存のファイルシステムの容量が足りなくなった。

従来の方法: ディスクを増設。

fdisk, mkfs.ext3, mount, rsyncでミラーし利用開始

LVM:

ディスクを増設, fdisk, pvcreate, vgextendで新しいPVをVGに追加, lvextendでLVの容量を増やし、resize2fsで容量変更





■シナリオ2: ディスクが異音を...

ディスクが異音を出し始めた＝壊れ始めた

従来の方法: ディスクを増設。

fdisk, mkfs.ext3, mount, rsyncでミラーし利用開始

新しいディスクを増設する。

従来であれば、

従来の方法: ディスクを増設。fdisk, mkfs.ext3, mount, rsyncでミラーし、壊れかけのディスクを外す。



■ LVM:

ディスクを増設

```
pvcrate /dev/sdc1
```

```
vgextend lpic.vg /dev/sdc1
```

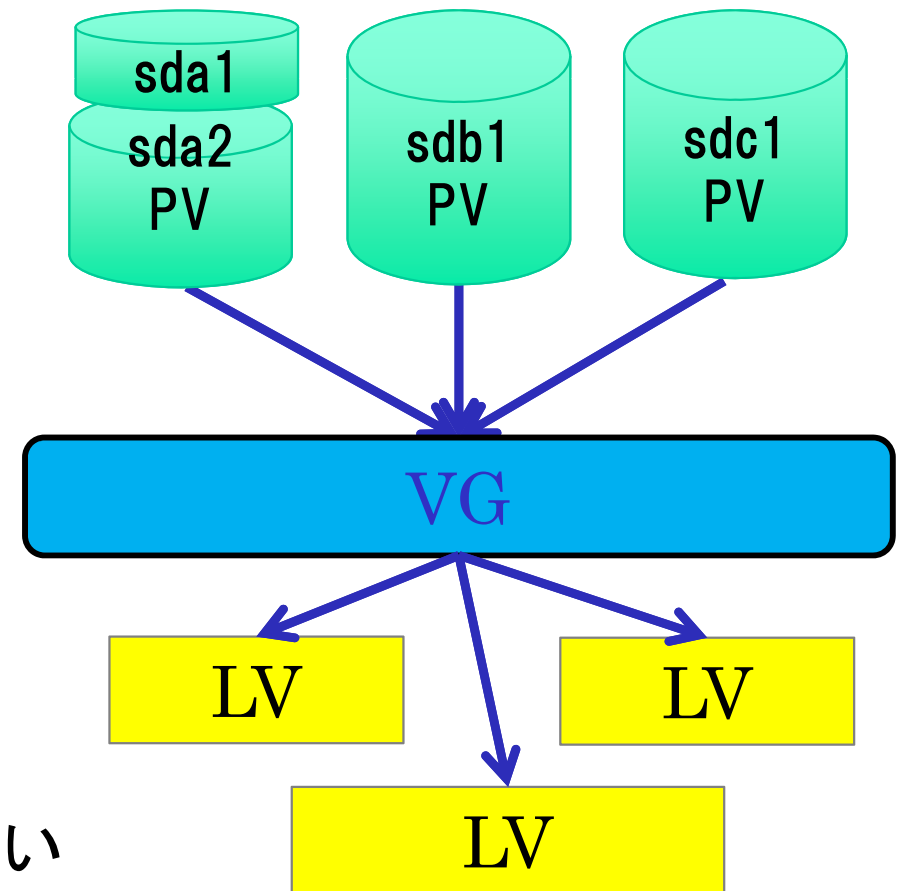
ディスクを未使用に。

```
pvmove lpic.vg /dev/sdb1
```

```
vgreduce lpic.vg /dev/sdd1・・・「不参加」
```

使用領域は、同じlpic.vgに参加している、
別のPV移動

- ・・・使用領域合計が、残りディスク
容量以上だったとき、pvmoveはできない
- ・・・移動する最低単位＝PE





- 人間が手で行う操作を、スクリプトに記述し、自動実行できるようにする
・・・シェルスクリプト
- 深夜定時の作業や、内容が同じオペレーション等を、繰り返し実行する場合に、作成しておくと便利
- プログラミングを行う
- プログラミングを行うため、条件分岐やループ等の制御構文が含まれている
- `vi hoge.sh`でシェルスクリプトを作成後
`chmod a+x hoge.sh`で実行権限を与え、
`./hoge.sh`で実行する



■if文による条件分岐

```
if [ 条件 ]; then ... ; elif [ 条件 ] then ...; else ... fi
```

条件式の書き方

```
if [ "$INPUT" = "OKADA" ]; then      $INPUTの内容がOKADAであったら真
```

```
if [ $COUNT -eq 2 ]; then          $COUNTの値が2であったら真  
他に-ne(≠)、-gt(>)、-lt(<)、-ge(≥)、-le(≤)等が利用可能
```

```
if [ -f /home/OKADA ]; then  /home/OKADAというファイルがあったら真  
-d...ディレクトリが存在したら真    -L...シンボリックリンクであったなら真  
-x...そのファイルが実行可能であったなら真、  
-w...そのファイルが書き込み可能であったなら真
```



■ ループ whileとforが存在

ここでは、forの便利な使い方を紹介

```
#!/bin/bash
```

```
for i in A B C D
```

```
do
```

```
    echo ${i};
```

```
done
```

ループが1回まわるたびに、echo \${i}が実行。そのとき\${i}には、A,B,C,Dが順に代入されている

```
for i in `ls /hoge`
```

/hogeディレクトリに存在している、ファイル・ディレクトリ名の一覧を\${i}に代入

```
for i in `cat a.txt`
```

a.txtの1行1行を\${i}に代入する



■シェルスクリプトの開始と終了

開始

利用するシェルの種類を記述

`#!/bin/bash`

終了

必ず`exit`と`exit`ステータスを記述

正常終了のときは、`exit 0`

異常終了のときは、0以外の値を利用する

この`exit`ステータスを正しく記述する。

`a.sh && b.sh`(`a.sh`が正常終了したとき、`b.sh`を実行する)のとき、`a.sh`から`exit 0`が返ってきているかどうか、を判定している



正規表現

■ただ「なんとなく」の人、多くありませんか？

「いまいち、つかみどころがない」

「なんで勉強するのか、いつ使うのかわからない」

少しの勉強で、身に付きます。得点源にしましょう。

例: ABCという文字列が含まれる行を探したい(Windowsの検索機能でも実現可能)

■「080で”始まる”行を探したい」「XYZで”終わる”行を探したい」

ふつうに検索をかけてしまうと、「080を含む行」「XYZを含む行」も対象

「凝った検索条件」を表記する”言語”・・・正規表現

grepコマンドほか、様々なコマンドの中で使われる。

Perl, Java等でも標準で利用可能・・・利用範囲は無限



■ grepコマンド

正規表現を利用し、ファイルの中から対象の文字列/パターンを検索する

grep (オプション) パターン 対象ファイル

オプション: よく使われるオプション `-r` 対象ファイルのディレクトリを、再帰的に中まで検索を行う。

パターン: 検索対象のパターンです。通常の文字列の場合もあれば、正規表現で記述されることもある。

対象ファイル: ファイル以外にディレクトリを指定することも可能。
ディレクトリを指定すると、そのディレクトリに含まれているファイル全件が対象。



- 正規表現
- 文字列を指定すれば、それが文字列そのまま
- 例: Okada...Okadaを検索する
- 特殊な意味を持つ記号と組み合わせ、パターンを作成する

記号	意味
.	任意の1文字
^	行頭
\$	行末
*	直前の文字の0回以上の繰り返し
?	直前の文字の0回か1回の繰り返し
\	エスケープシーケンス

記号	意味
[]	[abc]で、aもしくはbもしくはc
	[a-z]で、小文字アルファベット全部
	[^a]で、aではない(否定)
+	直前の文字の、1回以上の繰り返し(拡張)
{n}	直前のn回繰り返し(拡張)
{n, m}	n回以上m回以下の繰り返し(拡張)



■ 正規表現の例

<code>^Okada</code>	Okadaで始まっている行
<code>Kenji\$</code>	Kenjiで終わっている行
<code>^Okada\$</code>	Okadaの直前が行頭で、Kenjiの直後が行末 ⇒Okadaとしか書かれていない行
<code>^\$</code>	行頭の直後、行末⇒空行
<code>[lL][pP][iI]</code>	LPI, LPi, IPI, IPI, lpi何でもマッチする
<code>[a-zA-Z]</code>	アルファベット全部にマッチする
<code>[0-9]</code>	数値全部にマッチする



■ grepと正規表現5

これは何のパターンでしょうか1(日本限定)?

```
^[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]$
```

数字3桁-(ハイフン)数字4桁・・・郵便番号ですね

```
grep -E ^[0-9]{3}-[0-9]{4}$ ファイル
```

で指定すると、繰り返しの意味の{}が利用できます。

これは何のパターンでしょうか2(日本限定)?

```
^0[7-9]0-[0-9][0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]$
```

070, 080, 090の後に-(ハイフン)数字4桁-(ハイフン)数字4桁・・・携帯電話の番号ですね。

```
同じくgrep -E ^0[7-9]0-[0-9]{4}-[0-9]{4}$ ファイル
```

でもOKです。



- Linuxでプログラムが動くときの実行単位が、プロセス
- mv, mkdir等、動作がすぐ終了するものも、「瞬間」プロセスとして動作
- 動作がすぐ終了するもの…コマンド
動き続けるもの…デーモン、サーバプログラム

ps…プロセスを表示するコマンド

- a…全ユーザのプロセスを表示
- u…プロセスを実行しているユーザを表示
- x…(制御端末が無い)デーモンプログラムも表示

実行例:

```
root    2527  0.0  0.9 25916 10348 ?        SN   Jun15   0:00 /usr/bin/python
```

top…動作しているプロセスを、リアルタイムで表示するコマンド。
条件を与え、並び替えることが可能。



■killコマンド

コマンド名がコマンド名だけに、「プロセスを“殺す”＝停止させる」と思い込んでいる人いませんか？

「プロセスに対して、シグナルを送る」のが、正しい役割です。

入力例

kill (シグナルの種類) プロセスID kill -TERM 555 あるいは kill -15 555

シグナルの種類

有名なシグナルの種類は、以下の通り

シグナル番号	シグナル名	働き
1	HUP	再起動する
2	INT	割り込みのシグナルを送る
9	KILL	プロセスを強制終了する
15	TERM	プロセス終了(シグナル指定しないkillはこれ)



■ KILL(9)とTERM(15)の違い

killコマンドを、シグナルの種類を指定しないで利用すると、TERM(15)が送られる。

同じくKILL(9)シグナルもある。KILL(9)は、強制停止。

■ 両者の違いは？

ソフトウェアによっては、シグナルによって処理を行ったり、「後片付け」を行う後片付け・・・ファイル書き込み処理や、ネットワーク通信等の、正常終了。

「TERMシグナルを受け取ったら、正常終了の処理をしろ」と設定

しかし、TERMシグナルすらも受け付けないトラブルも起きうる・・・KILL(9)

KILL(9)は、いっさいの設定が行えず、ただ強制停止
従って、「後片付け」しない場合のトラブルも発生しうる
TERM(15)が利用できない場合の、最終手段。



■Upstart

従来の起動方式・・・SysVinit

起動に時間がかかる: SysVinitは、起動が「シリアル」である
順番に起動する

upstartは様々なサービスを並列に起動⇒起動時間が短縮される

■systemd

従来はサービスとランレベルで制御

systemd ...「ターゲット」という名称のグループ化 「ユニット」と呼ぶ

systemctlというコマンドで操作



■ grub

起動時の処理をつかさどる機能…「ブートローダ」

grubでは機能が足りなくなってきた⇒grub2の登場

例: LVM, RAIDからできる

GPTが利用できる

GPT: 今までのMBR(Master Boot Record)方式だと、ディスク最大容量が2TBまで

2TBを越えるディスクはGPTではないと管理ができない。

MacOSX (IntelMac初期から)・Windows(Vistaから)使われ始めている

設定ファイル: /etc/grub.dにある

インストールコマンド: grub-mkconfigを利用



- Linux/UNIXはインターネット/ネットワークに強い

- インターネットとは？

IPアドレスを持った機器同士が、IP(インターネットプロトコル)を利用し、通信しているネットワーク

TCP/IPの挙動を簡単に説明したいと思います。

- LinuxマシンがTCP/IPの通信をするのに、必要な設定

最低、2つ(3つ)の設定項目が必要です。

IPアドレス: インターネット上の機器に割り当てられた、固有のアドレス

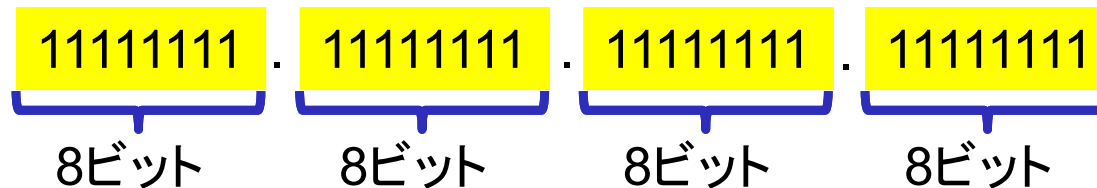
サブネットマスク: サブネットに分けるときに、ネットワークアドレスの指定

ブロードキャストアドレス: ネットワーク全体に通信を行うときのアドレス

```
ifconfig eth0 inet 192.168.1.10 netmask 255.255.255.0 broadcast 192.168.1.255 up
```

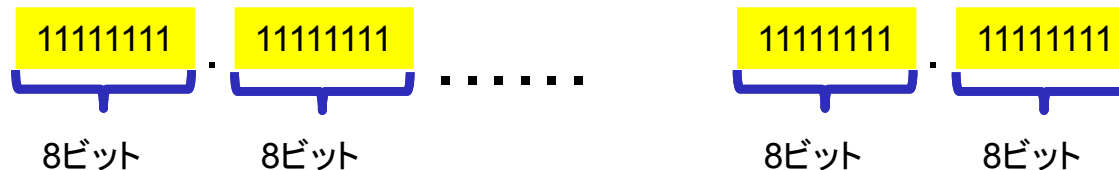


現行IP(v4)



=32ビットアドレス空間
(約43億個のIPアドレス・
理論値)

IPv6



=128ビットアドレス空間
(約43億の4乗個のIPアド
レス・理論値)

表記方法

例: 2012:00c8:0001:0000:0000:1234:2345:dcba

省略形を実現

1. 先頭の0は削除可能
2012:c8:1:0:0:1234:2345:dcba
2. 連続した0は省略可能
2012:c8:1::1234:2345:dcba



- 皆様の更なるご発展をお祈り申し上げます。
- ご清聴ありがとうございました。
- 参考書籍



LPI-Japan 発行



翔泳社 発行



インプレス 発行



秀和システム 発行