

LPI-Japan主催 LPICレベル1技術解説無料セミナー

2011/10/01

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

セミナーについて

LPICについて

自己学習環境の整備

各項目の説明



自己紹介1

株式会社
ネットマイスター

- 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(7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



自己紹介2

株式会社
ネットマイスター

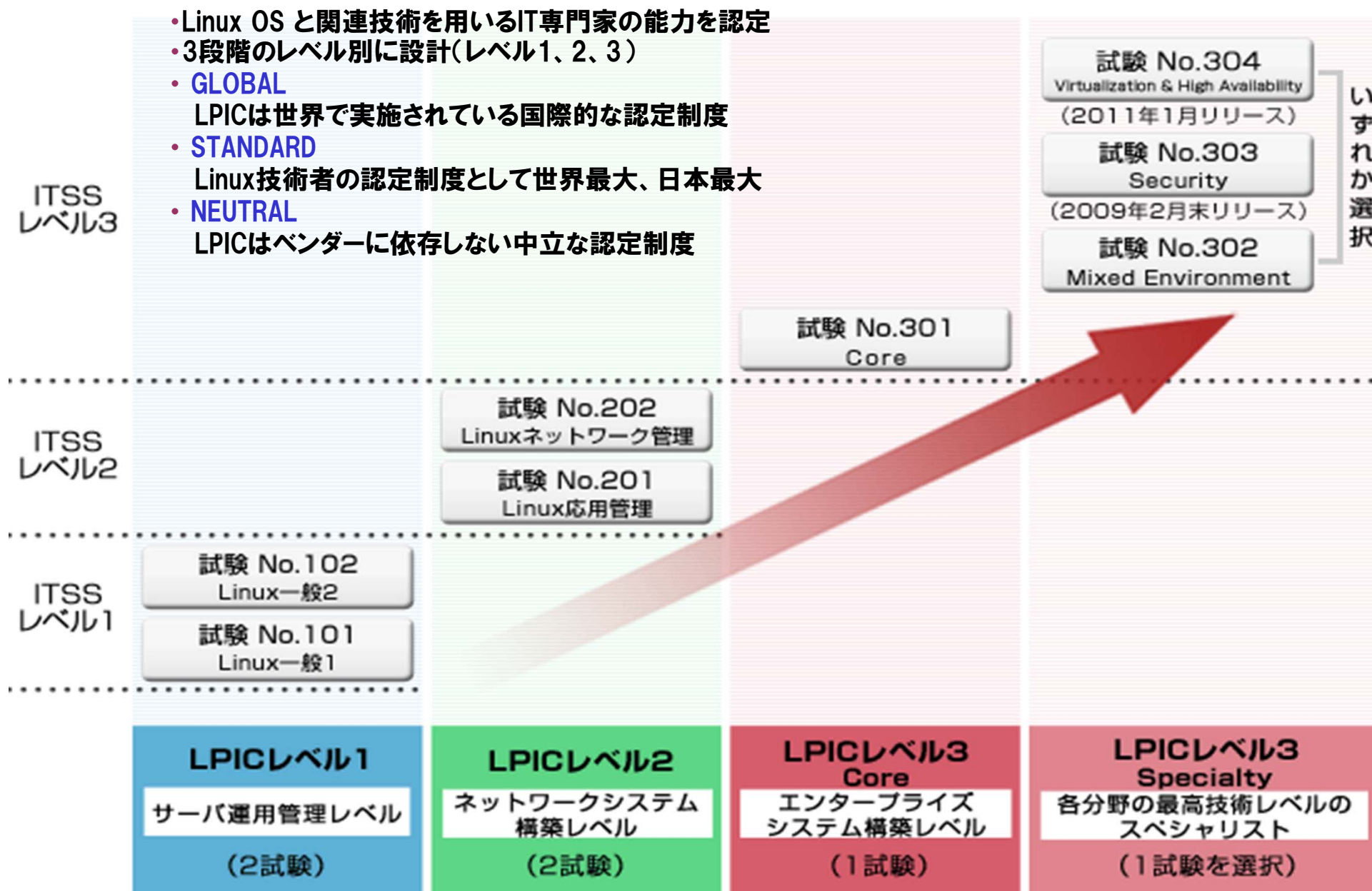
- 普段は、自宅作業でMac。常駐作業でLinuxを使用しています。
- 自宅サーバが存在。1台のマシンで12個のLinuxが同時に動いています。メールサーバとかは、全部そいつでやらせています。
- Linux歴は、10-11年。
- UNIX歴を含めると、18年以上触っています。
- いわゆる「サバ管」上がりです。なので、妙な経験値だけはいっぱいあります。





LPIC体系

株式会社
ネットマイスター





【LPIC レベル1】 ～ファーストレベルLinux専門家～

Linuxサーバの構築・運用・保守等の基本操作が行える技術者。

サーバ構築・運用・保守技術者、DBエンジニア、カスタマーサポートなど

【LPIC レベル2】 ～アドバンスドレベルLinux専門家～

Linuxのシステムデザイン、システム構築、ネットワーク構築、維持、トラブルシューティングが行える技術者。

レベル1に加え、システム開発・企画・保守のリーダー、組込開発技術者、インストラクターなど

【LPIC レベル3】 ～市場価値の高いLinuxプロフェッショナル～

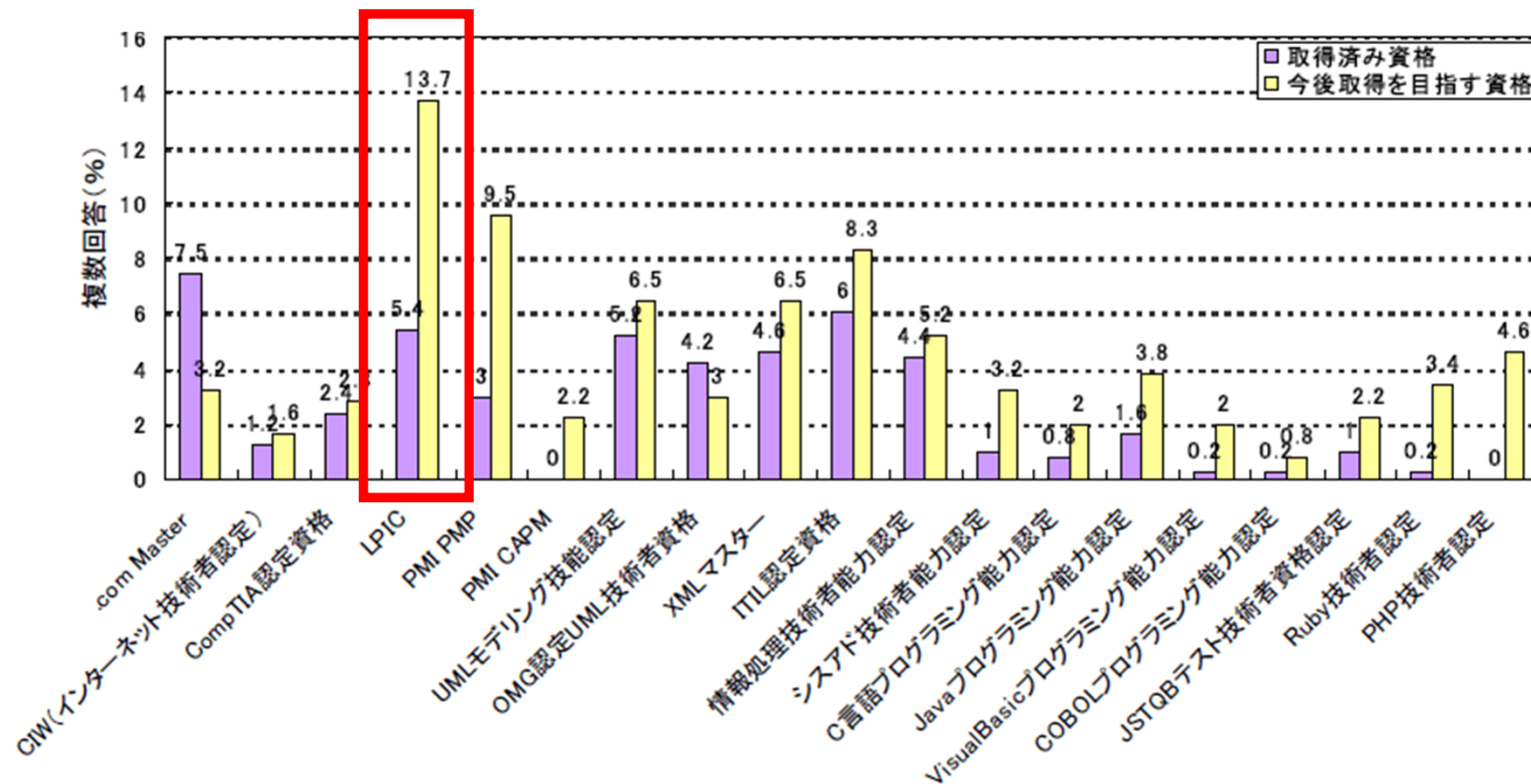
Linux、Windows、UNIXなどの混在環境での大規模システムの構築・ネットワーク構築や高度なセキュリティ技術を有する技術者。

大規模システム開発・企画・設計、システムコンサルタントなど



LPICの認知度

@IT自分戦略研究所「身に付けたいスキル、何ですか？」の中で
「今後取得を目指す資格(ベンダーニュートラル)」で
“5年連続” LPICが1位！



出典: @IT自分戦略研究所 読者調査結果(2010年11月、回答504)



- 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・・・



■「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

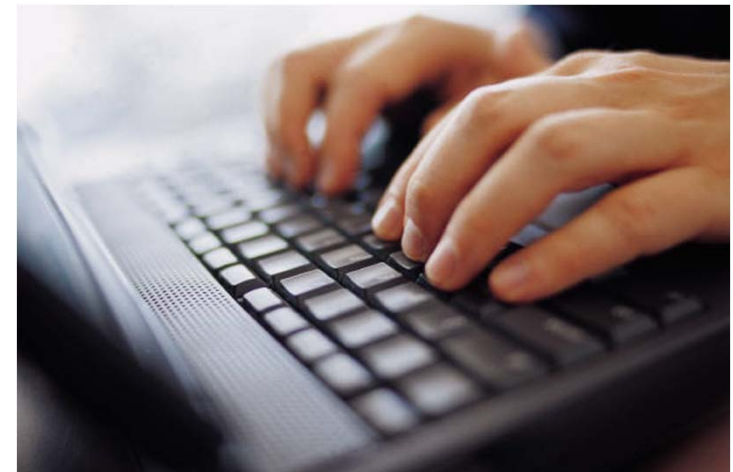
1. プリインストール環境

職場・学校にあるLinuxマシンを使う

利点：設定はいらない

欠点：環境がある人しか利用できない

管理者権限の操作は、管理の都合でNGになることが多い





2. 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はお勧めしません)

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)





3. 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・・・2.と同じ

vmware®
|| Parallels®





4. KNOPPIXを利用する

- CD/DVDから起動するLinux
- HDDに書き込まない

利点: 今利用しているPCでそのまま利用可能

欠点: データが保存できないので、
作業の継続が困難
一部制限があり、
教材内容と差異

- 2., 3.をお勧めします。
- 特に3.でのVirtualBox環境はお勧めです。





■ ディストリビューション

Linuxの大元は、基本部分(カーネル)

- 動作するプログラム＝パッケージやインストーラを追加し、多くの人が利用できるようにまとめたもの
- 大きく分けてRedHat, Debian, Slackwareの系統が存在
- RedHat系統がNo.1のシェアを持つ
- 前述のKNOPPIXもディストリビューションの1つ(Debian系統)





自習環境の整備6

株式会社
ネットマイスター

- お勧めはRedHat系統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





UNIXというOSの存在

- 1969年・・・UNIXの誕生
- 1974年・・・UNIX公開
- 1970年代後半・・・バークレーでのBSD開発。TCP/IP実装。
以降、様々な開発が行われる。

UNIXは商標

- UNIXはThe Open Groupが審査を行う
- 認められたOS・・・UNIX OS(と名乗ってよい)
- 認められてないOS・・・「UNIX系OS」「UNIX互換OS」「UNIX-like OS」
という



Linuxの誕生

- 1991年にヘルシンキ大学の学生、ライナス・トルヴァーズが開発開始
- MINIXという、教育機関向けUNIXが存在
 - 荒削りながら、先進的機能が多数存在
- 「MINIXをもっと洗練させ、一般利用のUNIX OSにしよう」
 - ・・・MINIXの開発者が「NO」
 - ・・・じゃあ、独自に作ろう
- 「今、僕はMINIXみたいなOSを作っている。もう少しで公開できるよ」
 - MINIXのメーリングリスト(ネットニュース)に投稿
 - MINIXに不満を持っていた連中が、開発に参画
 - 以降、開発コミュニティにより開発が継続。企業も参加。
 - ・・・UNIX-likeのOSです。



- 当初は、インストールも複雑で、マニアしか使えない
 - ・・・インストールを楽にするインストーラを作ろう
- インストール後のソフトウェア追加の作業も複雑
 - ・・・誰でも簡単にソフトウェア追加ができる、パッケージの仕組みを考えよう
 - ・・・ディストリビューションの誕生

大きな転機

- 前述の通り「マニアによるマニアのためのOS」
- 一般利用とはほど遠い、企業利用もあまり浸透していなかった
- DBで有名なOracleが、DBの動作環境OSにLinuxを含めると宣言(1990年代後半)
- Linuxでの企業利用が注目を浴びる



■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

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/hda3

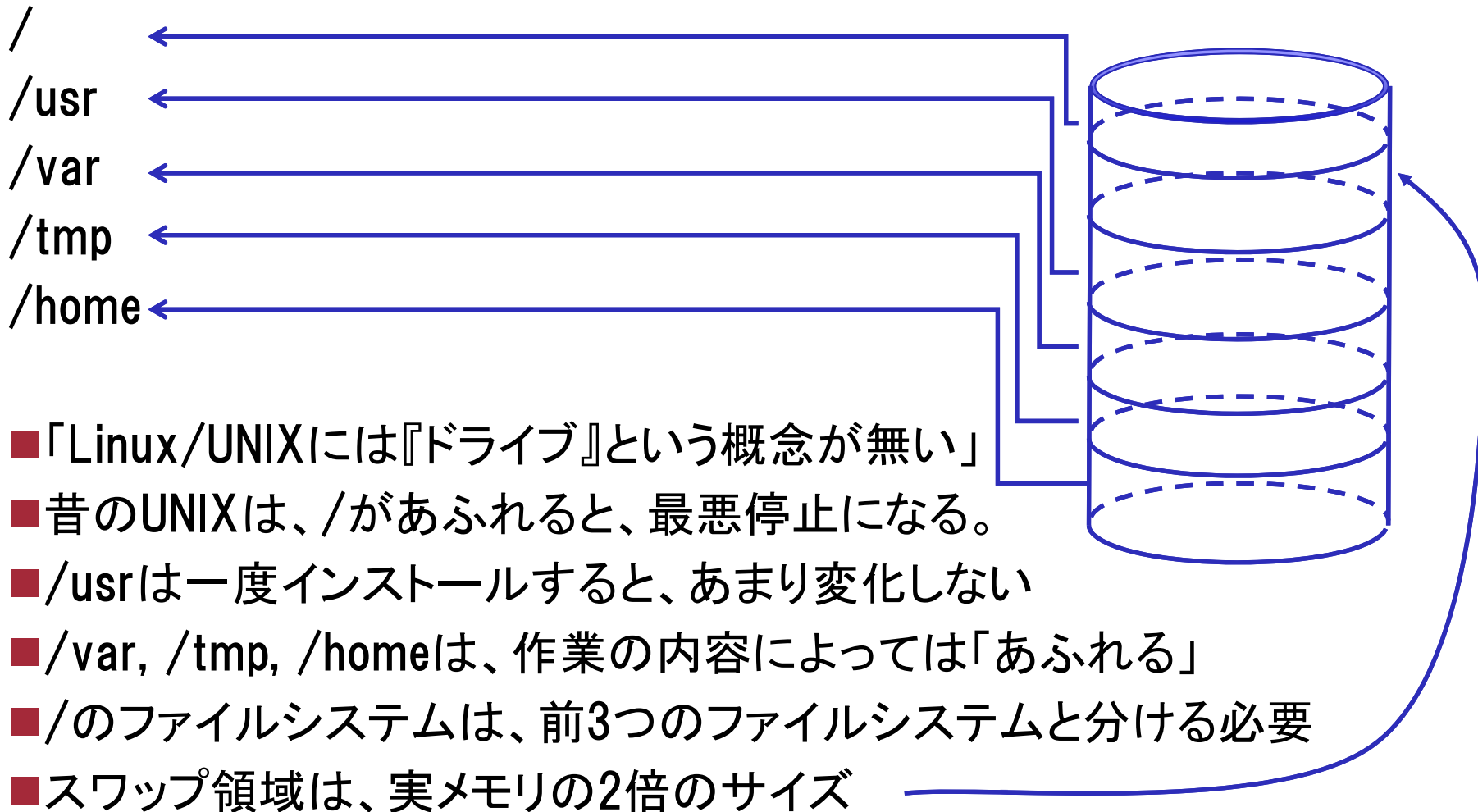
SCSIの3台目のディスクで、4つめのパーティション・・・/dev/sdc4

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





- ディレクトリにディスクの領域を割り当てる・・・マウントする、という
・・・mountコマンド(解除はunmount)
- 起動時に、自動的にマウントするときの設定情報・・・/etc/fstabに記述

```
/dev/sda1  /          ext3  defaults    1 1
```

- ファイルシステムのマウント状況確認・・・dfコマンド

通常のdfコマンド出力・・・わかりづらい

```
/dev/sda1          101086   19135   76732  20%   /
```

-hオプションを利用することで

```
/dev/sda1          99M   19M   75M  20%   /
```

-Hオプションを利用すると

```
/dev/sda1          104M   20M   79M  20%   /
```

-Hは、1KB=1000B, 1MB=1000KBで計算する

-hは、1KiB=1024B, 1MiB=1024KiB・・・これ1KiB(キビバイト)=1024B,
1MiB(メビバイト)=1024KBという



■ 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。



- 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。



- 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
```



■サブネットの計算

ネットワークアドレス				ホストアドレス		
192	.	168	.	1	.	10
11111111	.	11111111	.	11111111	.	00000000
255	.	255	.	255	.	0 = /24

この場合、192.168.1.0～192.168.1.255までが1つのサブネット
192.168.1.0と192.168.1.255は特殊なアドレスなので、使わない。

ネットワークアドレス				ホストアドレス		
192	.	168	.	1	.	10
11111111	.	11111111	.	11111100	.	00000000
255	.	255	.	252	.	0 = /22

この場合、192.168.0.0～192.168.3.255までが1つのサブネット
192.168.1.10/24と192.168.2.20/24は、別サブネット
192.168.1.10/22と192.168.2.20/22は、同一サブネット



- 同一サブネットは、直接通信ができる
- 別サブネットは、「デフォルトルータ」といわれる機械を経由しないと、通信できない。

「近所(=同一サブネット)は直接お話し、遠方(=別サブネット)はルータに任せる」と覚える。

デフォルトルータの設定(デフォルトルータが192.168.1.1だったとき)

```
route add default gw 192.168.1.1
```

デフォルトゲートウェイ、ということもある

netstat -rコマンドで確認することができる

- 機械は、数値を覚えるのが得意。人間は名前を覚えるのが得意。

IPアドレス ⇔ ホスト名・ドメイン名

名前解決の機構



■名前解決

/etc/hostsに、ホスト名とIPアドレスを列挙する
DNSサーバに問い合わせる

設定例:

IPアドレスの設定 /etc/sysconfig/network-scripts/ifcfg-eth0 (Redhat系統)

DEVICE=eth0

BOOTPROTO=static

BROADCAST=192.168.1.255

HWADDR=XX:XX:XX:XX:XX:XX

IPADDR=192.168.1.10

NETMASK=255.255.255.0

NETWORK=192.168.1.0

ONBOOT=yes



設定例:

名前解決の設定 /etc/resolv.conf

(hogehoge.jpというドメイン下のマシンで、DNSのネームサーバが
192.168.1.2であったとき)

```
search hogehoge.jp
```

```
nameserver 192.168.1.2
```




- サーバプログラムの動作
- サーバプログラムは、自サービスのポートをオープンしながら待機
- 自サービスのポート・・・Webサーバ(HTTP, TCP/80番)、メールサーバ(SMTP, TCP/25番)等

利点：常に準備しているので、リクエストがあった時の応答が早い

欠点：プログラムが常駐するので、メモリの利用量が増える

これは、比較的新しい方式



- インターネットスーパーサーバ:inetd(昔の方式)
- inetdに、そのホストで利用するサービスとポート番号を登録
- 「このポートにリクエストが来たら、* * というプログラムが対応する」
 - inetdが、登録してあるサービスの全ポートをオープンし待機
 - リクエストがあると、対応プログラムを起動
 - 利点:inetdが常駐しているだけなので、メモリの節約になる
 - 欠点:リクエストのたびにプログラム起動・タイムラグの発生

Linuxにおいてプログラム起動は、すごい負荷がかかる・・・アクセスが多い環境では不適切



inetdの終わりとxinetdの登場

- inetdは、セキュリティ的に甘い設計
- tcpwrapper/tcpd等を利用し延命処置をしたがNG
アクセス制限等が実現できた
- セキュリティのことを考え、一から作り直したxinetdが登場

「メモリが安いね」

「1つのサーバに1つのサービス」

inetd/xinetdの方式が使われず、常駐型に移行



■ xinetdについて

/etc/xinetd.dに設定ファイルを保存

リクエストに応じて、サーバプログラムを起動

設定ファイル例:(このままでは動きません)

```
service rsync
```

```
{
```

```
    disable = yes
```

```
    socket_type = stream
```

```
    wait = no
```

```
    user = root
```

```
    server = /usr/bin/rsync
```

```
    server_args = --daemon
```

```
    log_on_failure += USERID
```

```
}
```

標準では動作しない。disable=noで動作する

stream=TCP
dgram=UDP

シングルスレッドか
マルチスレッドか

サーバプログラムの実行権限

サーバプログラム本体

サーバプログラム起動時
のオプション

接続失敗時のログ出力



■ xinetdのアクセス制限

/etc/hosts.allow アクセス制限で、アクセス許可の条件を記述
記述例:

ALL: lpi.or.jp ... lpi.or.jpからのアクセスをすべて許可する

/etc/hosts.deny アクセス制限で、アクセス不可の条件を記述
記述例:

rsync: 192.168.10.0/24 ... rsyncのサービスは
192.168.10.0/24のネットワークからのアクセスを拒絶する

■ なぜ/etc/xinetd.dに設定ファイルをおくのか？

... パッケージにとって、設定ファイルを「置く」「消す」は管理しやすい



■実際にポートをオープンし待機している状態

netstatコマンドで確認できる

netstat -a

応答例:

```
tcp      0      0 localhost.localdomain:smtp  *:*    LISTEN
```

netstat -an

応答例:

```
tcp      0      0 127.0.0.1:25              0.0.0.0:*    LISTEN
```

■-nオプション

-nオプションが無いとき、名前があるものは、それが表示(ホスト名・ポート名)

-nオプションがあると、すべて数値で表示される。

ポートの名称(今回はsmtp)はどこで? ... /etc/servicesで変換



- サービスアクセス時の(xinetd管理下の)プログラムは、hosts.allow, hosts.denyで管理できる

- 常駐型でのアクセス制限は？

iptablesというツールが存在します

カーネルレベルでのパケット制御が可能であるため、hosts.allow, hosts.denyよりもよりセキュアな環境が構築可能

```
iptables -A INPUT -i eth0 -s 192.168.10.0/24 -p tcp --dport 80 -j DROP
```

インターフェースeth0から入ってくる、192.168.10.0/24からのパケットで、宛先がTCP/80であるものを、DROP(=拒絶)する



まとめ

株式会社
ネットマイスター

■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