本稿は Linux Japan 誌 2000 年 1 月号に掲載された記事に補筆修正したものです.

貧相コマンドライン生活 その 2

今回は枕話は無しにして，すぐ本題に入ります. Plamo と Vine と Redhat の/bin に含まれる共通のコマンドを調べたところ,

```
arch ash bash cat chgrp chmod chown cp cpio
csh date dd df dmesg dnsdomainname domainname
echo ed false gawk grep gunzip gzip hostname
kill ln login ls mail mkdir mknod more mount
mt mv netstat nisdomainname ping ps pwd red
rm rmdir sh stty su sync tar tcsh touch true
umount uname ypdomainname zcat
```

となっていて，この総勢 55 個のコマンドの使い方を覚えようということなのですが，さっそく謝らなければいけません．今回改めて調べてみると，ash gawk grep は共通でなかったのです．ですから，総勢 52 個ということで話を進めます.

ネットワーク関連

hostname

*domainname はネットワークのドメインを表示あるいは設定するコマンドですが，man を実行してみるとなぜか hostname のマニュアルが表示されます．これは間違いではありません．なぜなら，実体は hostname だけで，他はそのリンクになっているのです．ドメインの管理は DNS，NIS，YP のどれによってなされている筈なので，それに対応して先頭に dns，nis，yp がついたコマンドで操作することになります．引数を与えずに実行すると，現在の設定状況が表示されます．例えば,

```
aya:~$ hostname
aya
aya:~$ domainname
(none)
aya:~$ dnsdomainname
film.s.dendai.ac.jp
```

という実行例からは，ホスト名が aya，ドメイン名が film.s.dendai.ac.jp と設定されていることが判ります．この設定はもちろんブート時になされる筈です．grep を用いて

```
grep hostname /etc/rc.d/*
```

と場所を探してみると，Slackware 系（Plamo）なら /etc/rc.d/rc.M に

```
/bin/hostname `cat /etc/HOSTNAME | cut -f1 -d .`
```

とあり，/etc/HOSTNAME ファイルに記述された文字列，“ホスト名. ドメイン名”（FQDN）からホスト名だけを切り出した結果を設定する手順になってます．Red Hat 系ではちょっと違うようです．/etc/rc.d/rc.sysinit に

```
... hostname ${HOSTNAME} ...
```

という個所があり，HOSTNAME というシェル変数が使われます．この変数はどこで定義されるかというと，rc.sysinit を読まないといけません．最初の方に

```
# Read in config data.
if [ -f /etc/sysconfig/network ]; then
    . /etc/sysconfig/network
else
    NETWORKING=no
    HOSTNAME=localhost
fi
```

とあって，/etc/sysconfig/network を読んでいるみたいです．さっそくその中身を確かめると,

```
bash# cat /etc/sysconfig/network
NETWORKING=yes
FORWARD_IPV4="yes"
HOSTNAME="aya"
GATEWAYDEV="eth0"
GATEWAY="192.168.0.1"
```

確かに，HOSTNAME が設定されています．この調子でドメイン名がどのように設定されるか調べてみるのは読者の皆さんにお任せしましょう.

netstat

ネットワークに関するの種々の情報を表示するコマンドです．筆者は，ネットワーク経路情報を表示するオプション -r しか使ったことがないのですが，今回 man で調べてみるとマスカレード関連（-M）や netlink の情報まで表示するようになっていますね.

ping

あるホストとの物理的な接続状況を応答時間で調べるためのコマンドです．このコマンドに関しては，送るパケットサイズを指定する-s packetsize しかオプションを使ったことがありません．応答に 1000 ミリ秒以上かかってしまうのは，かなりシンドイ状況です．さらに状況が悪いと，送った順番が入れ替わって応答されることがあります．そういう場合には，telnet 接続によるリモート作業は絶望的な状況になりますから，黙っ

て諦めましょう.(残り 45)

マシンの識別

arch

arch はマシンのハードウエア識別を表示します．結局は CPU 種別に基づくのですが，Linux では現在 i386，i486，i586，alpha，sparc，arm，m68k，ppc，mips が区別されるようです.

uname

OS，ハードウェア，ホスト名等のシステムの情報を出力するコマンドです．ハードウェアを示すオプション -m を付けて実行すると arch と同じ結果を得ます．全てを表示するオプション-a，OS のバージョンを表示するオプション -r を付けて実行させると，

```
aya:$ uname -a
Linux aya 2.0.36 #7 Fri Sep 24 09:17:31 JST
1999 i586 unknown
aya:$ uname -r
2.0.36
```

のように表示されます．連載第 8 回でも取り上げましたが，login 直後のシステムからのメッセージ /etc/motd，/etc/issue を作成する時にこのシステム情報が用いられています．具体的には，Plamo の場合は /etc/rc.d/rc.S にその部分が含まれます．Vine や LASER-Linux では，/etc/rc.d/rc.local に当該部分があります.(残り 43)

シェル

シェルの説明は詳しくはできませんが，今や単独の sh や csh は存在せず，sh は bash， csh は tcsh のリンクとなっているようです．対話シェルとして bash を sh の名前で呼び出した場合には ~/.bashrc を読み込まなくなることは覚えておくべきでしょう．同様に，ログインシェルとして sh の名前で呼び出した場合は ~/.bash_profile や ~/.bash_login を読み込まなくなります．また，起動時のシェルスクリプトは全て “#!/bin/sh” で始まっており bash になってからの拡張機能を前提としていないのが素晴らしい考え方ですね.(残り 38)

シェルの組み込みコマンド

echo のバージョンを調べようと思って，man にあるように --version オプションを付けて起動すると

```
aya:~$ echo --version
--version
```

となってしまいまいます．これはなぜでしょうか？ 答えは type echo とすれば判ります.

```
aya:~$ type echo
echo is a shell builtin
```

そうです，bash の組み込みコマンドなのです．もし，シェルとは独立た外部コマンドの echo を使うならば，絶対パスを指定して

```
aya:~$ /bin/echo --version
echo (GNU sh-utils) 1.16
```

となって一安心．このように，機能が拡張された bash では，従来独立して存在していた外部コマンドが組み込み（builtin）コマンドとなって取り込まれています．echo の他には kill，pwd，test（[） などです．組み込みコマンドにすると，外部コマンドを子プロセスとして起動する必要がなくなるので，動作が速くなるという利点があります．しかし，一方，シェル自身が肥大化するという弊害もあります.

bash の組み込みコマンド type でそのコマンドがシェルにどう解釈されるかを確かめることが出来ますので，調べてみましょう.

```
aya:~$ type echo test [ kill
echo is a shell builtin
test is a shell builtin
[ is a shell builtin
kill is a shell builtin
```

となり，確かに組み込みコマンドと解釈されています.

```
type -all コマンド名
```

とすると，解釈し得る全てのコマンドが表示されます.(残り 35)

true と false

あまりみかけないコマンドです．man true によると，何もせずに正常終了する（ 返り値が 0 ）という機能を持っているとのこと．反対に，false は異常終了をする（返り値が 1）コマンドです．シェルスクリプトで while ルーチンを用いて無限ループを形成する場合などに使われます．例えば，次のような数当てクイズでは，正解されるまでずーっと入力を促すようなルーチンが必要ですが，while true で実現しています.

```
#!/bin/sh

echo "13 までの数を当てましょう！"
VAL=$[RANDOM % 13 + 1]

while true
do
  echo -n "いくつ？ "
  read ANS
  DIFF=$[$ANS-$VAL]
  if   [ $DIFF -lt 0 ]; then echo "小さいです"
  elif [ $DIFF -gt 0 ]; then echo "大きいです"
  elif [ $DIFF -eq 0 ]; then
         echo "正解です !! "; break
  fi
done
```

もっとも，bash の内部コマンドを使って `while :` とした方が実行速度があがるので，この例は無理矢理の感がします (^^;（残り 33）

標準エディタ ed

この 1970 年代からある，UNIX の標準ラインエディタ ed をあまり使ったことがないので，説明するのは心苦しいのですが.... 管理者必須スクリーンエディタ vi は ed のコマンドを踏襲しているので，温古知新ということでちょっと触ってみるのも良い経験でしょう．そうはいっても，検索や置換も備わっている本格的なエディタです．ed は最初コマンドモードで起動されます．入力モードと区別するために，プロンプト文字列を設定するオプション `-p strings` 付きで起動してみましょう．

```
aya:~$ ed -p ">>" ed.txt
ed.txt: No such file or directory --いきなり警告
>>a  ---------------------- append：行入力の開始
#!/bin/sh
echo "Use ed!"
. ----------------- 行頭のピリオド入力で入力終了
>>1p ------------------------ print：1 行目の表示
#!/bin/sh
>>1,$p ---------------- 1 行目から最終行までの表示
#!/bin/sh
echo "Use ed!"
>>w ---------------- write：ファイルへの書き込み
25 -------------------------- 書き込まれた文字数
>>q --------------------------------- quit：終了
aya:~$
```

うーむ，深い．ところで ed のコマンド体系は `vi` にも継承されています．`vi` に留まらず，シェルの起動

```
!cmdline
```

や正規表現による置換の書式

```
s/re/replacement/
```

などは，ほとんどのコマンドに共通です．やはり ed は（使い難いかもしれないが）標準エディタであり，今でも覚える価値がありそうですね．

`red` は機能の制限された ed で，例えば，シェルの起動などができません．ed のリンクとなっています．（残り 31）

ファイルの属性変更

chgrp，chown，chmod

`ch*` はファイルの属性を変更（change）するコマンドです．chgrp は所属グループを，chown は所有（owner）ユーザーを，chmod は許可モード（permission）を設定します．これらは良く知られたコマンドと思いますから，一つだけ便利なオプションを紹介します．それは，`-R` で，ディレクトリ以下全てのファイルを再帰的（recursive）に変更してくれます．

touch

ファイルのタイムスタンプ（アクセス時刻，修正時刻）を変更します．アクセス（access）時刻はファイルの中身を見る操作がなされた場合に変更され，修正（modified）時刻は実際に内容が修正された場合に変更されます．`ls -l` で表示される時刻は修正時刻です．アクセス時刻を知りたければ `ls -lu` とします．

ファイルのタイムスタンプは `sys/stat.h` を見ると stat 構造体の中で

```
struct stat{
        ....
        timestruc_t     st_atim;
        timestruc_t     st_mtim;
        timestruc_t     st_ctim;
        ...
```

のように 3 種類定義されています．`st_atime`，`st_mtime` は上記の時刻ですが，さらに `st_ctime` があって，ファイルの状態を変更した時刻を保持しています．`ls -lc` で表示させることが可能です．ファイルを検索するコマンド find でも，オプション `-atime`，`-mtime`，`-ctime` などにより，この 3 種類の時刻は区別されています．普段は気にしないのですが，細かい管理が可能なのですね．

この時刻の違いが表に現れる場合いがあります．連載第 7 回でメールの既読・未読をアクセス時刻で判断するルーチンを紹介しましたが，そこではアクセス時刻と（ファイルの大きさ）を判断材料にしました．3 種の時刻を同時に表示するコマンドは標準ではないような

ので，おさらいを兼ねてちょっと作ってみましょう．そうですね，show3time とでもしまショウか（くー，おやじギャグ）．リスト 1 にソースを示します．このコマンドを使って，less や chmod により 3 種類の時刻が変更されていく様子を以下に示します．

```
aya:~$ show3time 12th.tex
atime: Tue Oct 26 10:33:48 1999
mtime: Tue Oct 26 10:35:36 1999
ctime: Tue Oct 26 10:35:36 1999
aya:~$ less 12th.tex
aya:~$ show3time 12th.tex
atime: Tue Oct 26 10:36:09 1999 <--- 更新
mtime: Tue Oct 26 10:35:36 1999
ctime: Tue Oct 26 10:35:36 1999
aya:~$ chmod 700 12th.tex
aya:~$ show3time 12th.tex
atime: Tue Oct 26 10:36:09 1999
mtime: Tue Oct 26 10:35:36 1999
ctime: Tue Oct 26 10:36:35 1999 <--- 更新
```

（残り 27）

リスト 1

```
/*********************************************
   ファイルのアクセス・修正・更新時刻を表示する
   gcc -o show3time show3time.c
   Usage: show3time ファイル名
*********************************************/

#include <stdio.h>
#include <sys/stat.h>
#include <unistd.h>

main(int argc, char** argv)
{
  char filename[80];
  struct stat buf;

  strcpy(filename,argv[1]);

  stat(filename,&buf);

  printf("atime: %s",ctime(&buf.st_atime));
  printf("mtime: %s",ctime(&buf.st_mtime));
  printf("ctime: %s",ctime(&buf.st_ctime));
}
```

ファイルの内容表示

more

前回 more は逆スクロールできないと書いてしまいましたが，正確には行単位で戻ることができないのです．スクリーン単位では f,b で順方向逆方向にスクロールします．機能的には全ての点で less が勝っていると思いますが，サイズは表のように more はあまり変化なくかつ小さいですが，less は バラバラで一定でないことが判ります．

表 1　各配付系における more と less のサイズの比較

	Plamo1.4	Vine1.1	Laser6.0
more	29,296	25,508	27,540
less	106,624	88,820	306,724

zcat

zcat は圧縮ファイルの中身を伸長しながら表示します．で，これは gzip のリンクです．gunzip も同様です．が，zless はシェルスクリプトで，zcat と less を使っています．なんかその場凌ぎにも見えますね (^^;

（残り 24）

突然ですが

残り 24 というところで，残り誌面の関係もあってギブアップします．後は，mknod，cpio などという滅多に使わないコマンドを除いては，まあお馴染みのコマンドですから省略ということで，お開きにしましょう．ちょっと本の紹介しておきます．

『UNIX & コマンド辞典』（1995 年，丸善）

Alan Southerton，Edwin C.Perkins,Jr 著

加藤大典 訳

です．コマンドの詳細を網羅するのではなく，役立ちそうな実例を列挙した小さい辞典で，コマンドの使い方に興味を持たれた方には，読んで面白いものと思います，たぶん．

次回は…未定

うーーん，本稿を書く直前には 『次は /sbin 以下だな，ちょろいぜ』と思っていたのですが，数も多く 100 個近いし，一つ一つのコマンドも慎重に検討（なにしろ /sbin ですからね）しなければならないということで，今回の経験から判断してまず無理とあっさり諦めました．TEX の話でもしようかななんて思っています（かなり未定）．