第3回 UNIXゼミ

ゼミ担当者　：　折戸　俊彦，荒久田　博士，鈴木　和徳
指導院生　　：　片浦　哲平，谷口　義樹
開催日　　：　2003 年 6 月 6 日

ゼミ内容: 第 2 回 UNIX ゼミにおいて，UNIX 上で使用されているエディタである vi や簡単なファイル操作を行うための基本的なコマンドについて紹介した．第 3 回 UNIX ゼミでは，vi の他によく用いられる Emacs というエディタやファイル操作を行う際に便利なワイルドカードについて紹介する．

1 Emacs

前回紹介した vi の他に，UNIX 上ではよく Emacs という非常に高機能なエディタが使われる．Emacs は，Emacs Lisp というプログラミング言語を用いることでエディタの機能をさらに拡張することができるが，この節では Emacs の基本的な使い方についてのみ紹介する．

1.1 Emacs でのキー操作

Emacs では，コマンド操作を行うときにはほかのエディタとは違ったキー操作をする．これ以降の説明では，Table 1 のような表記ルールに従って基本となるファイル編集に必要となる操作を説明する．

Table 1　Emacs でのキー操作

C-x	Ctrl キーを押しながら ‘x’ を押す
M-x	Alt キーを押しながら ‘x’ を押す
C-x C-c	Ctrl キーを押しながら ‘x’ を押し，次に Ctrl キーを押しながら ‘c’ を押す

1.2 Emacs でのファイル編集

ここでは，Emacs でのデフォルトのキー操作を説明する．これらのキー操作は設定によって変更することも可能である．

基本的には，$ emacs ファイル名　と入力し，ファイルを編集・作成する．

Emacs でテキストを開くためには，“C-x C-f” と打つ．そうすると，カーソルが画面の下の方に移るので，ここで編集したいファイル名を入力する．

編集したファイルを保存するためには”C-x C-s” と打つ．保存するファイル名を変えたい場合は “C-x C-w” と打ち，その後にカーソルが画面の下の方に移るので，保存したいファイル名を入力する．

コマンド操作をしようとして途中で操作を取り消したい場合は “C-g” と打つ．

文字を消去したい場合は消したい文字の上にカーソルを移動して “C-d” と打つ．DEL キーを押すと，カーソルの前の文字が消去されるので注意する．

文字列を検索したい場合は，“C-s” を入力し，続けて検索したい文字列を入力すると，1 文字ごとに入力文字列に一致する文字列にカーソルが移動する．“C-r” とすると，ファイルの先頭側に検索を実行する．

Emacs を終了するためには “C-x C-c” と打つ．この際，未保存のファイルは保存するか聞かれるが，‘y’ を押すと保存され，‘n’ を押すと破棄することができる．

1.3 vi と Emacs の比較

前項で紹介したように Emacs は非常に高機能なエディタであり，わざわざ変わった操作を行わないと使えない vi を使う必要性は感じられないかもしれない．しかし，UNIX では以下のような理由により vi を使うことがある．

- Emacs は UNIX の標準エディタではないので，環境によっては Emacs が入っていないこともある．vi はほぼすべての UNIX システムに標準で用意されている．
- ちょっとした編集作業の場合は Emacs を起動するよりも，vi の方が処理が速い．

Table 2 に Emacs の編集操作の方法を掲載する．

Table 2　Emacs のコマンド表

Emacs	機能
C-x C-f	ファイルを読み込む
C-x C-s	ファイルを保存する
C-x C-w	ファイルを別名で保存する
C-g	コマンド操作を取り消す．
C-d	文字を消去する
C-s	ファイル末尾方向への検索
C-r	ファイル先頭方向への検索
C-x C-c	エディタを終了する
M-x replace-string	文字列の置き換え
M-g 50	50 行目にカーソル移動

2　ワイルド カード

ワイルド カードを用いると一度に複数のファイルを扱うことが可能であることから，コマンド ライン上で複数のファイルを扱う際には便利である．1 つのコマンドで複数のファイルやディレクトリを同時に操作したい場合はワイルド カードと呼ばれる特殊な記号を用いる．複数のファイルを指定できるコマンドであれば，すべてのワイルド カードを使用可能である．ワイルド カードには Table 3 のようなものがある．

Table 3　ワイルド カードの種類

記号	置き換え
*	0 文字以上の任意の文字列
?	任意の 1 文字
[ ]	[ ] 内に指定された任意の 1 文字
[^]	[ ] 内に指定されていない任意の 1 文字
[!]	[ ] 内に指定されていない任意の 1 文字
{,}	「,」で区切られた文字列のいずれか

2.1　ワイルド カードの記述例

Table 4　ワイルド カードの記述例

記述例	意味
*tmp	tmp で終わるファイル．tmp も含む．
tmp	tmp という文字列を含むファイル
[a-z]*	a から z までの文字で始まるファイル
[-a-z]*	ハイフン，a〜z から始まるファイル
[a-zA-Z]*	アルファベットで始まるファイル
[0-9]	ファイル名の中に数字が含まれているもの
[!0-9]*	数字で始まらないファイル
??	名前が 2 文字のファイル
??*	名前が 2 文字以上のファイル
tmp/*	tmp というディレクトリ下の全ファイル

Table 4 にワイルド カードの記述例を示す．アルファベット全体を指したい場合は [a-Z] と書かずに，[a-zA-Z] と書く必要がある．文字コードの大小で範囲を判断しているため，[a-Z] とは [0x61-0x5A] のことであり，範囲内に何も含まない．逆に，[A-z] においては間に余計なコードを含んでしまう．その他，注意すべき項目は {test1,test2}という記述である．「,」の後にスペースを入れると「{test1,」と「test2}」の 2 引数と判断されてしまう．

以下に Table 3 または Table 4 の記号の使用例を示す．

Fig. 1 は*の使用例である．ls ttya*のように使うと，ttya で始まるカレントディレクトリの全てのファイルを表示することができる．

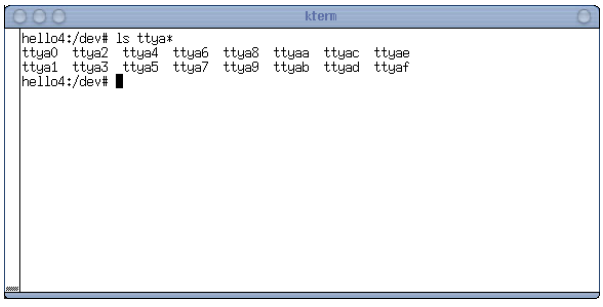

Fig. 1　*の使用例

Fig. 2 は?の使用例である．ls ttya?のように使うと，ttya+「任意の一文字」のカレントディレクトリ内のファイルを全て表示することができる．

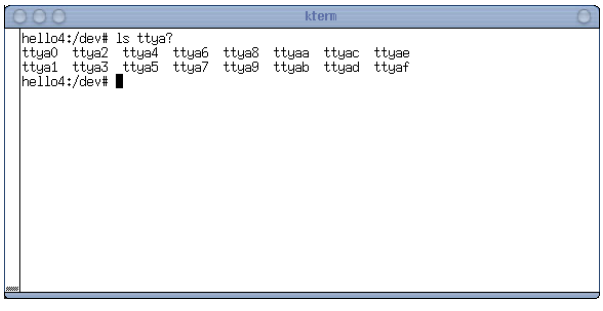

Fig. 2　?の使用例

Fig. 3 は [ ] の使用例である．ls tty[abcde]*のように使うと，tty の後に a,b,c,d,e のうちのどれかがくるカレントディレクトリ内のファイルを全て表示することができる．

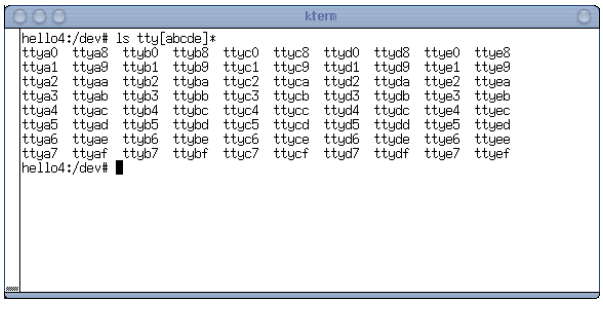

Fig. 3　[ ] の使用例

Fig. 4 は [-] の使用例である．ls tty[a-e]*のように使うと，Fig. 3 と同じ意味を表わし，tty に続いて a,b,c,d,e のうちのどれかがくるカレントディレクトリ内のファイルを全て表示することができる．つまり [abcde] は [a-e] と記述することができるといえる．

Fig. 5 は [!] の使用例である．ls tty[!a-z]*ように使う

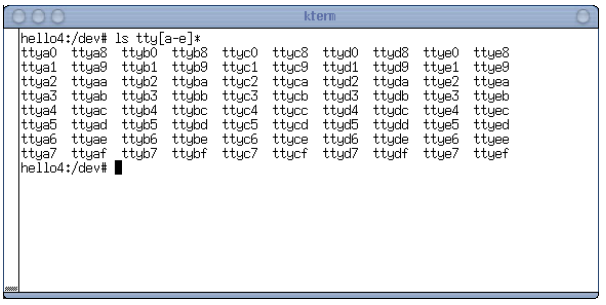

Fig. 4　[-] の使用例

と，tty に続く文字が a〜z でないカレントディレクトリ内のファイルを全て表示することができる．

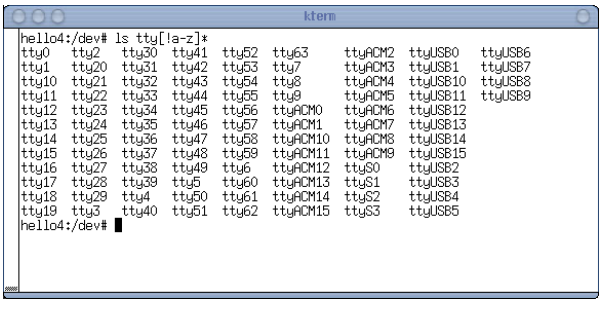

Fig. 5　[!] の使用例

Fig. 6 は [^] の使用例である．ls tty[^]*のように使うと ls tty[!a-z]*と同じことを表わし，tty に続く文字が a〜z でないカレントディレクトリ内のファイルを全て表示することができる．

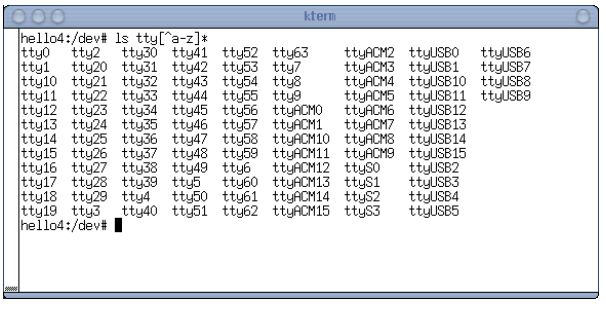

Fig. 6　[^] の使用例

Fig. 7 は, の使用例である．ls `*{USB,tec}*`のように使うと USB，または tec の文字列を含むカレントディレクトリ内のファイルを全て表示することができる．

2.2　ワイルド カード の仕組み

ワイルド カード はシェルが展開する．つまり，コマンドの引数にワイルド カード を含むファイルを指定した場合，コマンドが引数を処理する前に，シェルがワイルド カード を解釈し，複数の引数に展開した上でコマンドに

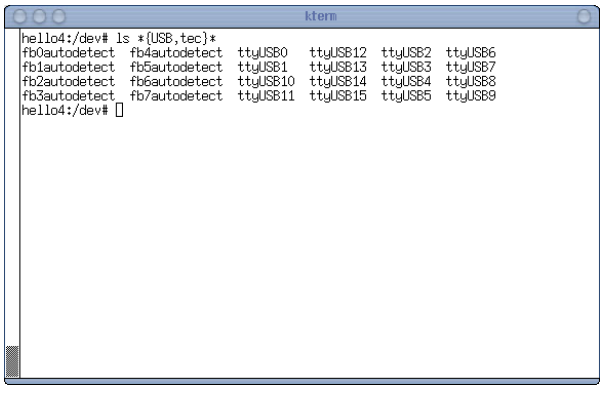

Fig. 7　{,} の使用例

受け渡す．

2.3　ワイルド カード 使用上の注意

cp のような「複数のファイルを引数とし，その並び順に意味があるコマンド」の場合や，find のようなコマンド 自身もワイルド カード が処理できる場合には思わぬ動作をする可能性がある．

- cp での失敗例

  test で始まるファイルを別のディレクトリにコピーするつもりで「cp test*」したとする．このディレクトリに test1 と test2 があった場合，コマンドはシェルによって「cp test1 test2」に展開される．つまり，test1 を test2 にコピーするという処理になってしまう．

- find での失敗例

  test で始まるファイルを指定したい場合,「find . -name test*」としても実行できない．正規表現にマッチするカレントディレクトリ内の全ファイルを探したい場合は,「find . -name ”test*”」する必要がある．

3　プロセスに関するコマンド

第 1 回 UNIX ゼミではプロセスとジョブという内容があったが，今回，プロセスに関する基本的なコマンドを紹介する．Table 5 にコマンドを示す．

Table 5　プロセスに関するコマンド 表

コマンド	機能
ps	実行中プロセスの情報を表示
top	プロセスの CPU 占有率を表示
kill	プロセスを停止させる

ps または top コマンドでプロセス ID を調べ，kill [プロセス ID] とすることでプロセスを殺すことがで

きる．

```
kazunoris@suzukik:~$ ps
 PID TTY          TIME CMD
1797 pts/0    00:00:00 bash
1805 pts/0    00:00:01 emacs
1816 pts/0    00:00:00 ps
kazunoris@suzukik:~$ kill 1805
 PID TTY          TIME CMD
1797 pts/0    00:00:00 bash
1816 pts/0    00:00:00 ps
```

Table 6 に ps のよく使われるオプションを紹介する．

Table 6　コマンド ps のオプション

オプション	機能
-a	全てのプロセスの状態を表示
-u	ユーザ名を含めてプロセスを表示
-x	制御端末を持たないものを含めて表示

これらのオプションを使った例を次ページに一部示す．

また top で CPU の占有率を調べ，占有率の高いプロセスを殺したりもする．また kill コマンドを実行しても終了できない場合がある．そのときは kill -9 のオプションを使うことで強制終了することができる．

top の使用例を次ページに一部示す．CPU 占有率の高い順に表示され，自動的に情報が更新される．

4　マウントとは

Windows にはドライブという概念があり，C ドライブ，D ドライブのように呼ぶが，Linux にはドライブという概念はない．すべてルートディレクトリを出発点としてアクセスできるようになっている．そのため，CD-ROM や複数のハードディスクを扱う場合，マウントというしくみを用いる．

マウントとは，別のディレクトリツリーをルートディレクトリ以下のディレクトリツリーに結合することを言う．例えば，CD-ROM ドライブを/mnt/cdrom というディレクトリにマウントした場合，CD-ROM の中にある「test」というディレクトリは「/mnt/cdrom/test」として扱うことができる．こうすることで，ユーザーはファイルの物理的な場所を意識せず，統一した方法でファイルを扱うことができる．

4.1　マウント

ファイルシステムをマウントするときは mount コマンドを用いる．mount コマンドは「mount オプション デバイス名 マウントポイント」の書式でスーパーユーザのみが使用できる．

```
# mount -t iso9660 /dev/cdrom /mut/cdrom
```

ただし，/etc/fstab で設定するとその設定に従ったマウント方法であれば，一般ユーザでもマウントできるようになる．一般ユーザが行う場合は「mount デバイス名 (またはマウントポイント)」のような書式になる．

```
$ mount /floppy
```

「mount」のみで実行した場合は現在のマウント状況が表示される．一般ユーザでも実行できる．

```
orito@hello4:~$ mount
/dev/hda3 on / type ext2
proc on /proc type proc
devpts on /dev/pts type devpts
```

4.2　アンマウント

FD，CD-ROM などの取はずし可能なデバイスは，マウントを解除 (アンマウント) してから取り出す．アンマウントは umount コマンドを用いる．

```
# umount /floppy
```

/etc/fstab の設定で，一般ユーザでもマウント可能になっている場合は，一般ユーザでアンマウントすることができる．

5　netstat コマンド

netstat コマンドを用いると，ネットワークへの接続状況を表示する，ネットワークに流れたパケットの統計を取る，自分のマシンのルーティングテーブルを表示するといったようにネットワークに関する様々な情報を取得することが可能である．

5.1　接続状況の表示

ネットワークへの接続状況を表示するためには，-a オプションをつけて netstat を実行する．こうすることで，現在有効なインターネット接続，UNIX ドメインソケットなどの接続状況が表示される．インターネット接続だけを表示したい場合は，“-A inet” のように，表示するプロトコルの種類を指定する．

実際に，表示するプロトコルをインターネット接続のみにして接続状況を表示したものを Fig. 9 に示する．この表示の意味は，左から，

- プロトコルの種類
- 受信されなかったバイト数
- 送信されなかったバイト数

```
suzukik@mikilab:~$ ps aux
USER       PID %CPU %MEM   VSZ  RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0  1272  408 ?        S    May19   0:04 init [2]
root         2  0.0  0.0     0    0 ?        SW   May19   0:00 [keventd]
root         3  0.0  0.0     0    0 ?        SWN  May19   0:00 [ksoftirqd_CPU0]
root         4  0.0  0.0     0    0 ?        Z    May19   0:15 [kswapd <defunct>
root         5  0.0  0.0     0    0 ?        SW   May19   0:00 [bdflush]
```

Fig. 8　オプションつきの ps コマンド（一部）

```
 17:46:51 up 9 days,  8:08, 25 users,  load average: 0.07, 0.08, 0.06
222 processes: 220 sleeping, 1 running, 1 zombie, 0 stopped
CPU states:   0.6% user,   1.0% system,   0.0% nice,  98.4% idle
Mem:    513280K total,   499184K used,    14096K free,    60520K buffers
Swap:  1595980K total,    30000K used,  1565980K free,   259048K cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT %CPU %MEM   TIME COMMAND
11624 suzukik   17   0  1064 1064   748 R     0.7  0.2   0:00 top
25794 egami      9   0  1836 1808  1464 S     0.1  0.3   0:00 sshd
    1 root       8   0   448  408   388 S     0.0  0.0   0:04 init
    2 root       9   0     0    0     0 SW    0.0  0.0   0:00 keventd
    3 root      19  19     0    0     0 SWN   0.0  0.0   0:00 ksoftirqd_CPU0
    4 root       9   0     0    0     0 Z     0.0  0.0   0:15 kswapd <defunct>
```

Fig. 9　top コマンドの使用例（一部）

- 接続元 (自分自身) の IP アドレスとポート番号

  IP アドレスからホスト名がわかる場合はホスト名が，ポート番号からサービス名がわかる場合はサービス名が表示される．-n オプションをつけると，ホスト名とサービス名の解決は行われない．
- 接続先の IP アドレスとポート番号
- 接続状態

  ESTABLISHED は接続されている状態，LISTEN は接続要求待ちの状態である．

となっている．このなかで，*（ホスト名を解決しない場合は，0.0.0.0）となっているものは，アドレスは自分が持つインターフェースの任意のもの，ポート番号は任意の値を用いることができるということを意味している．

5.2　NIC ごとの統計量表示

NIC(Network Interface Card) ごとに，どれだけのパケットがやりとりされているかを調べるためには，-i オプションをつけて実行する．こうすることで，データリンク層レベルでどれだけのパケットがやりとりされたかが把握できる．

NIC ごとの統計情報を表示したものを Fig. 10 に示す．この表示は左から順に，

- インターフェース名
- 最大転送単位
- メトリック数（ネットワーク間の距離）
- 受信関連の統計値

  順に、正常パケット数，エラーパケット数，破棄パケット数，オーバーロードパケット数を示す．
- 送信または転送関連の統計値
- 各種フラグ

を意味している．

5.3　プロトコルレベルでの統計量表示

プロトコルレベルでどれだけのパケットがやりとりされているかを調べるためには，-s オプションをつけて実行する．そうすると，IP や ICMP，TCP や UDP といったプロトコルレベルでどれだけのパケットが受信されたかやどれだけのパケットが受信エラーになったか，どれだけのパケットが送信されたかといったことが把握できる．

プロトコルレベルでの統計量表示したものを Fig. 11 に示す．

5.4　ルーティングテーブルの表示

ある IP パケットがある場合に，送信先アドレスの情報から次にどの IP アドレスにパケットを送信すればいいかを記述したものがルーティングテーブルである．

ルーティングテーブルを表示するには，-r オプションをつけて実行する．これによって，どの IP アドレスが書かれているとどの IP アドレスにパケットが送られるか，ということが確認できる．ルーティングテーブルを

```
suzukik@mikilab:~$ netstat -aA inet
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 *:time                  *:*                     LISTEN
tcp        0      0 *:pop3                  *:*                     LISTEN
tcp        0      0 *:www                   *:*                     LISTEN
tcp        0      0 *:8470                  *:*                     LISTEN
tcp        0      0 *:ssh                   *:*                     LISTEN
tcp        0      0 *:smtp                  *:*                     LISTEN
tcp        0      0 mikilab.doshisha.ac:ssh 202.23.143.73:28521     ESTABLISHED
tcp        1      0 mikilab.doshisha.:36431 mikilab.doshisha.:36428 CLOSE_WAIT
tcp        0      0 mikilab.doshisha.a:pop3 mikilab.doshisha.:46403 TIME_WAIT
tcp        0      0 mikilab.doshisha.ac:ssh 202.23.147.71:3454      ESTABLISHED
tcp        0      0 mikilab.doshisha.ac:ssh 202.23.143.73:1068      ESTABLISHED
tcp        0      0 mikilab.doshisha.ac:www ip68-5-197-194.oc:47360 TIME_WAIT
tcp        0      0 mikilab.doshisha.a:pop3 mikilab.doshisha.:46405 TIME_WAIT
tcp        0      0 mikilab.doshisha.ac:ssh 202.23.143.73:29230     ESTABLISHED
tcp        0      0 mikilab.doshisha.ac:ssh 202.23.147.71:4156      ESTABLISHED
tcp        1      0 mikilab.doshisha.:40408 mikilab.doshisha.:40405 CLOSE_WAIT
tcp        0      0 localhost:pop3          localhost:46408         TIME_WAIT
tcp        0      0 mikilab.doshisha.ac:ssh FLA1Aae158.kng.mes:1047 ESTABLISHED
tcp        0      0 localhost:pop3          localhost:46411         TIME_WAIT
tcp        0      0 mikilab.doshisha.ac:ssh 202.23.147.71:54262     ESTABLISHED
tcp        0      0 mikilab.doshisha.ac:ssh p3030-ipad02daian:64835 ESTABLISHED
tcp        0      0 mikilab.doshisha.ac:ssh 202.23.147.71:54263     ESTABLISHED
tcp        0      0 mikilab.doshisha.a:pop3 mikilab.doshisha.:46414 TIME_WAIT
tcp        0      0 mikilab.doshisha.ac:ssh 202.23.143.73:28390     ESTABLISHED
tcp        0      0 mikilab.doshisha.a:pop3 mikilab.doshisha.:46416 TIME_WAIT
tcp        1      0 mikilab.doshisha.:55372 mikilab.doshisha.:55369 CLOSE_WAIT
tcp        1      0 mikilab.doshisha.:39566 mikilab.doshisha.:39563 CLOSE_WAIT
tcp        0      0 mikilab.doshisha.ac:ssh 202.23.143.73:eklogin   ESTABLISHED
tcp        0      0 mikilab.doshisha.ac:ssh 202.23.143.73:32825     ESTABLISHED
tcp        1      0 mikilab.doshisha.:42552 mikilab.doshisha.:42549 CLOSE_WAIT
tcp        0      0 mikilab.doshisha.ac:ssh 202.23.143.73:1596      ESTABLISHED
tcp        1      0 mikilab.doshisha.:39570 mikilab.doshisha.:39567 CLOSE_WAIT
tcp        1      0 mikilab.doshisha.:50198 mikilab.doshisha.:50195 CLOSE_WAIT
tcp        0      0 mikilab.doshisha.ac:ssh 202.23.147.71:4589      ESTABLISHED
tcp        1      0 mikilab.doshisha.:55364 mikilab.doshisha.:55361 CLOSE_WAIT
tcp        1      0 mikilab.doshisha.:39555 mikilab.doshisha.:39552 CLOSE_WAIT
tcp        0      0 mikilab.doshisha.ac:ssh kutf21910260203.ge:3020 ESTABLISHED
tcp        0      0 mikilab.doshisha.ac:ssh 202.23.143.73:1396      ESTABLISHED
tcp        1      0 mikilab.doshisha.:55368 mikilab.doshisha.:55365 CLOSE_WAIT
tcp        0      0 mikilab.doshisha.ac:www gate.toyama-u.jp:3558   TIME_WAIT
tcp        0      0 mikilab.doshisha.ac:ssh 202.23.143.73:1032      ESTABLISHED
tcp        1      0 mikilab.doshisha.:40123 mikilab.doshisha.:40120 CLOSE_WAIT
tcp        0  19824 mikilab.doshisha.ac:www p6e41f6.freedc01.a:1851 ESTABLISHED
tcp        1      0 mikilab.doshisha.:44686 mikilab.doshisha.:44683 CLOSE_WAIT
tcp        0      0 mikilab.doshisha.ac:ssh 202.23.143.73:1037      ESTABLISHED
tcp        1      0 mikilab.doshisha.:45784 mikilab.doshisha.:45781 CLOSE_WAIT
tcp        0      0 mikilab.doshisha.ac:ssh 202.23.143.73:1167      ESTABLISHED
tcp        0      0 mikilab.doshisha.ac:ssh 202.23.143.73:28123     ESTABLISHED
tcp        1      0 mikilab.doshisha.:45769 mikilab.doshisha.:45766 CLOSE_WAIT
tcp        0      0 mikilab.doshisha.ac:ssh 202.23.143.73:2524      ESTABLISHED
tcp        0      0 mikilab.doshisha.ac:ssh 202.23.147.71:2370      ESTABLISHED
tcp        1      0 mikilab.doshisha.:45774 mikilab.doshisha.:45771 CLOSE_WAIT
tcp        0      0 mikilab.doshisha.ac:ssh 202.23.143.73:2065      ESTABLISHED
tcp        0      0 mikilab.doshisha.ac:ssh 202.23.147.71:1030      ESTABLISHED
tcp        1      0 mikilab.doshisha.:45778 mikilab.doshisha.:45775 CLOSE_WAIT
tcp        1      0 mikilab.doshisha.:37554 mikilab.doshisha.:37551 CLOSE_WAIT
tcp        0      0 mikilab.doshisha.ac:ssh 202.23.147.71:1029      ESTABLISHED
tcp        0      0 mikilab.doshisha.ac:ssh 202.23.143.73:1047      ESTABLISHED
udp        0      0 mikilab.doshisha.ac:ntp *:*
udp        0      0 localhost:ntp           *:*
udp        0      0 *:ntp                   *:*
```

Fig. 10　ネットワーク接続状況の表示

```
suzukik@mikilab:~$ netstat -i
Kernel Interface table
Iface   MTU Met   RX-OK RX-ERR RX-DRP RX-OVR   TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0   1500   0 9483877      0      0      0 11376806      0      0      0 BMRU
lo    16436   0 1778212      0      0      0 1778212      0      0      0 LRU
```

Fig. 11　各 NIC でのパケット送受信統計

```
suzukik@mikilab:~$ netstat -s
Ip:
    11261932 total packets received
    0 forwarded
    0 incoming packets discarded
    11104119 incoming packets delivered
    13155266 requests sent out
Icmp:
    7472 ICMP messages received
    6860 input ICMP message failed.
    ICMP input histogram:
        destination unreachable: 433
        echo requests: 22
        echo replies: 159
    8251 ICMP messages sent
    0 ICMP messages failed
    ICMP output histogram:
        destination unreachable: 8229
        echo replies: 22
Tcp:
    146942 active connections openings
    0 passive connection openings
    2 failed connection attempts
    0 connection resets received
    55 connections established
    10401384 segments received
    12460838 segments send out
    65641 segments retransmited
    91 bad segments received.
    69070 resets sent
Udp:
    660784 packets received
    8229 packets to unknown port received.
    0 packet receive errors
    686004 packets sent
```

Fig. 12　各プロトコルでのパケット送受信統計

```
suzukik@mikilab:~$ netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
localnet        *               255.255.255.0   U        40 0          0 eth0
default         202.23.156.1    0.0.0.0         UG       40 0          0 eth0
```

Fig. 13　ルーティングテーブルの表示

表示したものを Fig. 12 に示す．この表示は左から，

- 送り先の IP アドレス
- 転送先の IP アドレス
- ネットマスク
- 各種フラグ
- 最大セグメントサイズ
- ウインドウサイズ
- 初期ラウンドトリップ時間
- インターフェイス名

を意味している．

参考文献

1) Linux のすべて
   西村めぐみ，株式会社 日本実業出版社
2) UNIX シェルプログラミング
   Bruce Blinn，SOFT BANK