MS-DOS上で使えるUNIX風コマンドの解説

佐藤, 賢二
九州大学情報処理教育センター

https://doi.org/10.15017/6769093

出版情報：情報処理教育広報. 14 (1), pp.3-18, 1991-08. Educational Center For Information Processing, Kyushu University
バージョン：
権利関係：

ＭＳ－ＤＯＳ上で使えるＵＮＩＸ風コマンドの解説

佐藤賢二 *

１．はじめに

ＵＮＩＸは１９６９年に Ken Thompson と Dennis Ritchie によって開発されたＯＳであり、その後多種多様なマシンに移植されてきた。最近大学や企業の研究開発部門において急速に普及しているエンジニアリングワークステーションでは、ＵＮＩＸが標準的なＯＳとなっている。一方、パソコンではＭＳ－ＤＯＳが標準的なＯＳとなっている。別々に開発されたこの２つのＯＳの上では、ファイルシステムの違いやシングルプロセス／マルチプロセスの違いにより、コマンドのバリエーションが相当違う。また、似かよった機能を持つコマンド同士を比較しても、使い勝手が微妙に違う。よって、普段ワークステーションを使っていると、パソコン上でＵＮＩＸのコマンドが使えないことを不便に感じることがしばしばある。

このような理由で、ＵＮＩＸのコマンドをＭＳ－ＤＯＳ上に移植した例がいくつか報告されている。もちろん、ファイルシステムやプロセス数の違いからくる制限があるので、完全な移植ができないコマンドもあるが、それでも主要な機能は実現されている。本解説ではこのようなコマンドのうち、情報処理教育センター（以下、単にセンターという）のパソコン端末であるＦＭＲ－６０ＨＸにインストールしているものについて解説する。本解説の目的は、このような「ＵＮＩＸ風の」コマンドをＭＳ－ＤＯＳ上で利用する方法を解説することにより、現在のセンターでは一般には利用できないＵＮＩＸの持つ魅力を伝えることである。

本解説はＭＳ－ＤＯＳユーザ向けに解説したつもりなので、ＭＳ－ＤＯＳのコマンドやファイルシステムについては特に説明をしない。ＭＳ－ＤＯＳについては文献[3]を参考にして欲しい。また、オリジナルのＵＮＩＸコマンドについては文献[1][2]を参考にして欲しい。

２．ＵＮＩＸ風のコマンド

本節ではＵＮＩＸのコマンドをＭＳ－ＤＯＳ上に移植したものについて解説する。エディタやシェルについては３節～５節で解説する。混乱を避けるために、以後の解説ではＭＳ－ＤＯＳのコマンドを大文字で、ＵＮＩＸから移植されたコマンドを小文字で表記する。また、例ではユーザが入力する部分にアンダーラインを引いている。ほとんどのコマンドに対しては、ｍａｎコマンドでオプションの詳しい説明を見ることができる。例えば、

```
F:¥> man ls
```

と入力すれば、ｌｓコマンドの詳しい説明が得られる。

・ｌｓコマンド

```
ls [-1abcdfgilnopqrstuxACFR] [names...]]
```

ｌｓコマンドは、ファイルの大きさやファイルに対する権限（読み／書き／実行）やディレクトリ内のファイルの一覧などを表示する。ＭＳ－ＤＯＳのＤＩＲコマンドよりも高機能である。names... には複数のファイルおよびディレクトリを指定することができる（ワイルドカードも使える）。指定しないとカレントディレクトリを指定したのと同じことになる。－Ｒオプションを使うとｌｓコマンドを再帰的に実行する。つまり、サブディレクトリの内容も順次表示する。

---

* 情報処理教育センター

例）カレントディレクトリの内容を表示する。

```
F:¥>ls
address1.txt    address2.txt    dir1
```

例）カレントディレクトリの内容を詳しく表示する。

```
F:¥>ls -l
total 3
-rw-rw-rw-  1 root          104 Jul 16  9:40 address1.txt
-rw-rw-rw-  1 root          139 Jul 15 17:38 address2.txt
drwxrwxrwx  2 root            0 Jul 16  9:56 dir1
```

例）カレントディレクトリの内容を表示する。－Ｒオプションがあるので、サブディレクトリに対しても再帰的にｌｓコマンドを実行する。－Ｆオプションがあるので、ファイルタイプも表示する（ディレクトリなら "/" が、実行可能なファイルなら "*" が付く）。

```
F:¥>ls -RF
address1.txt    address2.txt    dir1/

./dir1 :
test.exe*
```

・ｃａｔコマンド

```
cat [-nsuvV] [file...]
```

ｃａｔコマンドは、ファイルの表示を行う。ＭＳ－ＤＯＳのＴＹＰＥコマンドと異なり、複数のファイルを指定して連結表示することができる。これと出力リダイレクトを使うとファイルのマージができる。ファイル名を指定しないと、標準入力（キーボード）から読み込む。

例）カレントディレクトリの address1.txt を表示する。

```
F:¥> cat address1.txt
崎谷    健次郎  815     福岡    092-123-XXXX
角松    敏生    814-01  福岡    092-456-XXXX
鳥山    雄二    812     福岡    092-789-XXXX
```

例）同様に、address2.txt を表示する。

```
F:¥> cat address2.txt
中森    明菜    820-07  嘉穂    0948-12-XXXX
森高    千里    818     筑紫野  092-345-XXXX
仙道    敦子    816     大野城  092-678-XXXX
森口    博子    810     福岡    092-901-XXXX
```

例）address1.txt と address2.txt を連結して juusho.txt に出力する。

```
F:¥> cat address?.txt > juusho.txt

F:¥> cat juusho.txt
崎谷    健次郎  815     福岡    092-123-XXXX
角松    敏生    814-01  福岡    092-456-XXXX
鳥山    雄二    812     福岡    092-789-XXXX
中森    明菜    820-07  嘉穂    0948-12-XXXX
森高    千里    818     筑紫野  092-345-XXXX
```

仙道	敦子	816	大野城	092-678-XXXX
森口	博子	810	福岡	092-901-XXXX

・ｃｐコマンド

(1) cp [-fir] file1 file2
(2) cp [-fir] files directory

ｃｐコマンドは、ファイルのコピーを行う。(1)の使い方では、一つのファイル file1 を、別のファイル file2 にコピーする。(2)の使い方では、複数のファイルを directory の下に同じ名前でコピーする。－ｒオプションを使うと、file1 や files にディレクトリを指定した場合、ディレクトリの下を全部コピーする。ＭＳ－ＤＯＳでは、ファイルのコピーはＣＯＰＹコマンド、ディレクトリのコピーはＸＣＯＰＹコマンドという具合いに別々に扱っていたが、ｃｐコマンドでは統一的に扱っている。

例）juusho.txt を juusho2.txt にコピーする。
F:¥> cp juusho.txt juusho2.txt

例）dir1 を dir2 にコピーする。
F:¥> cp -r dir1 dir2

例）address?.txt を dir1 の下にコピーする。
F:¥> cp address?.txt dir1

・ｍｖコマンド

(1) mv [-ir] file1 file2
(2) mv [-ir] files directory

ｍｖコマンドは、ファイル名の変更やファイルの移動を行う。ＭＳ－ＤＯＳでは、同じディレクトリ内でファイル名の変更するコマンド（ＲＥＮコマンド）はあるが、ファイルを別のディレクトリに移動するコマンドは無い。ｍｖコマンドはこの２つの機能を統一的に実現している。コマンドの文法やオプション等はｃｐコマンドとほぼ同じである。

例）juusho2.txt を juusho3.txt に改名する。
F:¥> mv juusho2.txt juusho3.txt

例）dir2 を dir1 の下に移動する。
F:¥> mv -r dir2 dir1

・ｄｕコマンド

du [-as] [names...]

ｄｕコマンドは、ディレクトリの容量をキロバイト単位で表示する。names...を省略すると、カレントディレクトリを指定したのと同じことになる。－ａオプションを指定すると、ディレクトリだけでなくファイルの容量も表示する。－ｓオプションを指定すると、個々のディレクトリの容量は表示せず、合計容量だけを表示する。

例）カレントディレクトリの合計容量を表示する。
F:¥> du -s
40

例）カレントディレクトリの容量を表示する。サブディレクトリやファイルの容量

もすべて表示する。

```
F:¥> du -a
1       ./address1.txt
1       ./address2.txt
1       ./juusho.txt
1       ./juusho3.txt
14      ./dir1/test.exe
1       ./dir1/address1.txt
1       ./dir1/address2.txt
14      ./dir1/dir2/test.exe
16      ./dir1/dir2
35      ./dir1
40
```

・ｆｉｎｄコマンド

find path... expression

ｆｉｎｄコマンドは、指定されたパスから再帰的にディレクトリを降りて行き、条件に合致するファイルを検索する。検索したファイルに対してコマンドを実行することもできる。expression の詳しい書き方については文献[1][2]を参考にして欲しい。expression の記述に使うオペレータのうち、オリジナルのｆｉｎｄコマンドでは使えるが、ＭＳ－ＤＯＳに移植したｆｉｎｄコマンドでは使えないものは次の１２個である：-fstype, -prune, -links, -nouser -nogroup, -inum, -ls, -cpio, -ncpio, -xdev, -user および -group。

例）カレントディレクトリからスタートして、address1.txt というファイルを捜す。見つかったファイルのパスを表示する。

```
F:¥> find . -name address1.txt -print
./address1.txt
./dir1/address1.txt
```

例）カレントディレクトリからスタートして、一週間以内にアクセスされなかったファイルを捜す。見つかったファイルに対しては、まずパスを表示し、そのあとｌｓコマンドを実行する。ｌｓコマンドの代わりに、後述するｒｍコマンドを使うと、古いファイルを掃除することができる。逆に、一週間以内にアクセスされたファイルを捜したい場合は、＋７を－７にする。

```
F:¥> find . -atime +7 -print -exec ls -la {} ";"
./dir1/test.exe
-rwxrwxrwx  1 other        13600 Jun 27 15:51 ./dir1/test.exe
./dir1/dir2/test.exe
-rwxrwxrwx  1 other        13600 Jun 27 15:51 ./dir1/dir2/test.exe
```

・ｒｍコマンド

rm [-rfiv] file1 ... filen

ｒｍコマンドは、ファイルの削除を行う。－ｒオプションを指定すると、サブディレクトリに対しても再帰的にｒｍコマンドを実行する。

例）juusho3.txt を削除する。－ｉオプションを指定すると、削除してよいかどうかユーザに問い合わせる。また、－ｖオプションを指定すると、verbose mode（コマンド実行時に各種のメッセージを表示するモード）で実行する。

```
F:¥> rm -iv juusho3.txt
```

```
remove juusho3.txt? y
File juusho3.txt removed.
```

例）dir1 を削除する。

```
F:¥> rm -vr dir1

File dir1¥test.exe removed.
File dir1¥address1.txt removed.
File dir1¥address2.txt removed.
File dir1¥dir2¥test.exe removed.
Directory dir1¥dir2 removed.
Directory dir1 removed.
```

・ｗｈｉｃｈコマンド

```
which [-a] commandname...
```

ｗｈｉｃｈコマンドは、commandname... で指定した名前を持つバッチファイル（拡張子が .bat）または実行可能ファイル（拡張子が .exe または .com）がどこのディレクトリにあるかを表示する。これを使うと、コマンドがどこのディレクトリに置いてあるかを調べることができる。また、環境変数 PATH で定義されるコマンドサーチパスに従って検索し、最初に見つかったコマンドのパスを表示するので、同じファイル名を持つコマンドが複数あるような状況で実際にどのコマンドが実行されるかを調べることもできる。commandname... には複数のコマンドを書くことができる。－ａオプションを使うと最初に見つかったコマンドのパスを表示した後も検索を続け、見つかったコマンドのパスを全て表示する。

例）ＦＬコマンドを捜す。

```
F:¥> which FL
e:/localbin/fl.bat
```

例）ＦＬコマンドを全部捜す。

```
F:¥> which -a FL
e:/localbin/fl.bat
e:/bin/fl.exe
FL not found in
./
.
e:/localbin
e:/bin
c:/
e:/etc
```

・ｔａｉｌコマンド

```
tail [-|+n[lbc]] [file...]
```

ｔａｉｌコマンドは、ファイルの最後の方だけを表示する。

例）juusho.txt の最後から２行目から表示を初め、最後まで表示する。

```
F:¥> tail -2l juusho.txt
仙道　　敦子　　816　　大野城　092-678-XXXX
森口　　博子　　810　　福岡　　092-901-XXXX
```

・ｗｃコマンド

```
wc [-lwc] [file...]
```

ｗｃコマンドはファイル中に含まれる行数、単語数、文字数を表示する。－ｌ、－ｗ、－ｃオプションはそれぞれ行、単語、文字に対応する。オプションを指定しない場合は全てのオプションを指定したのと同じことになる。

例）address?.txt にマッチする各ファイルに対して行数、単語数、文字数を表示する。最後に合計も表示する。

```
F:¥> wc address?.txt
       3       15       97 address1.txt
       4       20      137 address2.txt
       7       35      234 total
```

・ｄｉｆｆコマンド

```
diff [-aBCFHINpT] file1 file2
```

ｄｉｆｆコマンドは２つのファイルの内容を比較し、異なっている部分を表示する。２つのファイルの内容が全く同じなら何も表示しない。最も簡単な使い方としては、一方がもう一方のバックアップコピーかどうかを調べるという使い方がある。

例）コピーしたファイルを比較する。ファイルにはＡＳＣＩＩキャラクタでない文字すなわち日本語が含まれるので、－ａオプションが必要である。２つのファイルは等しいので、結果は何も表示されない。

```
F:¥> cp juusho.txt juusho.bak
F:¥> diff -a juusho.txt juusho.bak
```

例）同様に、２つの異なるファイルを比べる。file1 は file2 を包含しているので、file1 にのみ出現する行が "< " に続いて表示される。最初の行に表示されるのは、file1 が file2 に等しくなるための操作を、ＵＮＩＸのラインエディタの編集コマンド風に表示したものである。この場合は「file1 の１行目から３行目までを削除すれば file2 と等しくなる」という意味である。

```
F:¥> diff -a juusho.txt address2.txt
1,3d0
< 崎谷　健次郎　815　　福岡　　092-123-XXXX
< 角松　敏生　　814-01　福岡　　092-456-XXXX
< 鳥山　雄二　　812　　福岡　　092-789-XXXX
```

・ｃｕｔコマンド

```
(1) cut -c<list> [file...]
(2) cut -f<list> [-d<char>] [-s] [file...]
```

ｃｕｔコマンドは、ファイルの各行の特定の列（カラム）を表示する。－ｃオプションを使うと、<list>で指定した列を表示する。－ｆオプションを使うと、<list>で指定したフィールドを表示する。フィールドは、句切り記号で区切られた行の一部分のことである。－ｄオプションで句切り記号を指定することができる。指定しない場合はタブ記号が句切り記号になる。行によっては句切り記号が１つもない場合が考えられるが、－ｓオプションを指定するとこのような行を表示しない。

例）address1.txt の第１フィールドから第２フィールドまでと第４フィールドを表示する。

```
F:¥> cut -f1-2,4 juusho.txt
```

```
崎谷　　健次郎　福岡
角松　　敏生　　福岡
鳥山　　雄二　　福岡
中森　　明菜　　嘉穂
森高　　千里　　筑紫野
仙道　　敦子　　大野城
森口　　博子　　福岡
```

・ｕｎｉｑコマンド

```
uniq [-cdu -n +n] [input [outout]]
```

同一内容の行が連続する場合、それを１つにまとめて表示する。ファイル中に含まれる同一内容の行を削除したいときに使う。連続していなければ同一内容の行とは見なさないので、sort コマンドと一緒に使うことが多い。－ｃオプションを付けると各行の先頭に重複回数を付けて表示する。－ｄオプションは、重複している行すなわち重複回数が２以上の行だけを表示する。逆に、－ｕオプションは重複していない行だけを表示する。

例）juusho.txt の第４フィールドをｃｕｔで切り出し、ＭＳ－ＤＯＳのＳＯＲＴコマンドでソートした後、ｕｎｉｑで重複行を省く。同じ住所に何人住んでいるかを見るために、－ｃオプションを付ける。

```
F:¥> cut -f4 juusho.txt |SORT |uniq -c
    1 嘉穂
    1 大野城
    1 筑紫野
    4 福岡
```

・ｃｏｍｐｒｅｓｓコマンド

```
compress [-cCdf?hkKvV][-b maxbits][-Iinpath][-Ooutpath][filenames...]
```

ファイルの圧縮と復元を行う。

例）juusho.txt を圧縮して小さくする。－ｖオプションで verbose mode を指定する。拡張子の最後の文字は自動的に "z" に置き換えられる。

```
F:¥> compress -v juusho.txt
juusho.txt: Compression: 20.16% -- replaced with juusho.txZ
```

例）圧縮した juusho.txz を元に戻す。

```
F:¥> compress -vd juusho.txt
juusho.txZ:  -- replaced with juusho.txt
```

・ｐｓコマンド

```
ps [-abdfkpx] [-i pid[,size]]
```

ＵＮＩＸはマルチプロセスのＯＳである。つまり、同時に複数のプログラムを走らせることができる。現在動作しているプロセスの数や、それぞれの識別番号（PID）や、そのプロセスを起動したプロセス（親プロセス）の識別番号や、プロセスが占有するメモリの大きさなどの情報を表示するのがＵＮＩＸのｐｓコマンドである。ＭＳ－ＤＯＳはシングルプロセスのＯＳなので、ｐｓコマンドは意味が無いようにも思えるが、メモリに常駐しているプログラムの情報を知りたいときに役に立つ。

例）４節で説明するヒストリ機能プログラムを常駐させた後、ｐｓコマンドを実行する。３行目に表示されているのがそのプログラムである（ｈｉｓ＿ｋｎｊコマン

ドはバッチコマンドなので、バッチファイルの中で実行しているコマンドが表示される）。２行目は名前が無いが、実際にはＣＯＭＭＡＮＤコマンドである。ヒストリ機能プログラムとｐｓコマンドはこれから起動されているので、親プロセスＩＤ（PPID）が 415b になっている。

```
F:¥> ps
UID      PID  PPID SIZE
root     3fad bf2a   6.70kb (no name process) C:¥ /E:20 /P
root     415b 415b   3.37kb (no name process)
root     425b 415b  22.17kb h:/history/history.com  -IR -P -DH:¥HISTORY
root     47f3 415b  17.79kb f:/ps.exe
```

3．シェルの利用

［シェルとは］

ＵＮＩＸを使っているときに入力したコマンドは、そのまま実行されるのではなく、コマンドインタプリタによって解釈されてから実行される。このコマンドインタプリタがシェルである。ＭＳ－ＤＯＳで言えばＣＯＭＭＡＮＤコマンドがシェルに近い働きをする。シェルにはｓｈ、ｂｓｈ、ｃｓｈ、ｔｃｓｈなどいくつかの種類があり、それぞれ機能が異なる。ここで解説するのは、ＵＮＩＸのｃｓｈに近い機能を持つＭＳ－ＤＯＳのｃｓｈである。

［ｃｓｈの起動と終了］

ＭＳ－ＤＯＳのプロンプトが出ている状態で

```
F:¥> csh
```

と入力すると、プロンプトがパーセント記号に変わり、

```
%
```

となる。ｃｓｈを終了してＭＳ－ＤＯＳに戻るときは

```
% exit
```

と入力する。ｃｓｈを起動したディレクトリに _cshrc というファイルがあれば、ｃｓｈはプロンプトを出す前に、_cshrc に書いてあるコマンドを実行する。

例）以下のような _cshrc を用意してｃｓｈを起動する。

```
F:¥> cat _cshrc
echo    "Welcome to DOS-C-Shell¥!¥!"

F:¥> csh
Welcome to DOS-C-Shell!!
%
```

－Ｌオプションを使ってｃｓｈを起動すると、_cshrc に書いてあるコマンドを実行した後に、カレントディレクトリにある _login というファイルに書いてあるコマンドを実行する。また、－Ｌオプションを使ってｃｓｈを起動した場合は、ｃｓｈを終了するときに、ｃｓｈを起動したディレクトリにある _logout というファイルに書いてあるコマンドを実行する。

例）以下のような _login と _logout を用意して、－Ｌオプションを付けてｃｓｈを起動する。１つ前の例で出てきた _cshrc も、同じディレクトリにあるとする。

```
F:¥> cat _login
echo    "You logged in."
```

```
F:¥> cat _logout
echo    "You logged out."

F:¥> csh -L
Welcome to DOS-C-Shell!!
You logged in.
% exit
You logged out.

F:¥>
```

これ以外にも、ＭＳ－ＤＯＳで言えばバッチファイルに相当するファイルを用意しておき、ｃｓｈから呼び出すことで、ファイル内のコマンドをまとめて実行することができる。このようなファイルをシェルスクリプトと言う。バッチファイルと同様に、シェルスクリプトにも条件分岐や繰り返し実行などの制御命令を書くことができる。シェルスクリプトの文法は文献[1][2]を、ＭＳ－ＤＯＳ版のｃｓｈコマンド特有の制限事項については man csh で表示されるファイルを参考にして欲しい。

例）以下のようなシェルスクリプト a.csh を用意して実行する。

```
% cat a.csh
if ( -e core ) then
        echo "You have a core file."
endif

switch ("$term")
        case pc9801:
                echo "Terminal type is pc9801."
                breaksw
        case fmr60:
                echo "Terminal type is fmr60."
                breaksw
        default:
                breaksw
endsw

% set term=fmr60
% echo "This is a core file." > core
% source a.csh
You have a core file.
Terminal type is fmr60.
```

［ヒストリ機能］

ｃｓｈの機能の一つに、入力したコマンドの履歴を記憶して後から参照するヒストリ機能がある。これを有効に使うと、コマンド入力の手間が軽減される。似たような長いコマンドを入力することが多い場合に有効である。ＭＳ－ＤＯＳのテンプレート機能は直前に入力したコマンドしか参照できないが、ｃｓｈでは設定した長さの履歴を参照できる。

例）まず、保存する履歴の長さを、ｃｓｈのコマンドであるｓｅｔコマンドを使って設定する。その後、適当なコマンドを１０回投入する。最後に、履歴を見る。

```
% set history=20
% pwd
～中略～
% ls -laR | wc -l
% history
1       set history=20
2       pwd
3       commpress juusho.txt
4       compress juusho.txt
5       compress -d juusho.txt
6       ls
7       du -s
8       cat juusho.txt
9       cat juusho.txt | sort
10      cut -f4 juusho.txt |SORT |uniq -c
11      ls -laR | wc -l
12      history
```

例）履歴を参照する。最初は番号を使って参照する。次に名前を使って参照する。最後に、直前のコマンドを参照する。

```
% !6
ls
address1.txt    address2.txt    dir1
% !p
pwd
f:/
% !!
pwd
f:/
```

［別名定義機能（ａｌｉａｓ）］

ｃｓｈでは、コマンドに別の名前を付けて参照する別名定義機能（ａｌｉａｓ）もある。あるコマンドを実行するときによく使うオプションを設定しておいたり、長い名前のコマンドに短い別名を付けることでコマンド入力の手間を軽減したい場合に役に立つ。ＭＳ－ＤＯＳでもバッチファイルを使えばこれに近いことが出来るが、ａｌｉａｓではコマンドごとにファイルを用意しなくてもよい。普通は _cshrc にａｌｉａｓコマンドを書いておき、ｃｓｈの起動と同時に自分の好みの別名を使えるように設定する。

例）_cshrc にａｌｉａｓコマンドを書いておき、ｃｓｈを起動する。

```
F:¥> cat _cshrc
#       copy this file and modify
#       .cshrc shell script
set     history=40
set     noclobber
set     prompt=`pwd`'%

# alias list
alias   dir     'ls -CF'
alias   H       'history 20'
```

```
alias   ls      'ls -CFa'
alias   rm      'rm -i'
alias   cd      'cd ¥!* ;set prompt = `pwd`"% "'

echo    "Welcome to DOS-C-Shell¥!¥!"

F:¥> csh
Welcome to DOS-C-Shell!!
f:/% cd dir1
f:/dir1% ls
./              ../             test.exe*
```

4. エディタとページャ

・ｋｅｍａｃｓ

高機能エディタｅｍａｃｓの一種であるｍｅｍａｃｓ（ＭｉｃｒｏＥｍａｃｓ）を日本語化したもの。日本語（２バイトコード）が扱え、メッセージが日本語化されている。less e:¥etc¥kemacs.tut でチュートリアルファイルを見ることができる。このチュートリアルファイルは、読みながらｋｅｍａｃｓのコマンドを実行して理解するように書いてあるので、例えば

```
F:¥> cp e:¥etc¥kemacs.tut .
F:¥> kemacs kemacs.tut
```

と入力してｋｅｍａｃｓを起動し、実際に編集操作を行いながら読むのもよい。

例）juusho.txt を編集する。

```
F:¥> kemacs juusho.txt
```

と入力してｋｅｍａｃｓを起動する。(ESC-X)を押した後、ｈｅｌｐと入力してリターンキーを押すと、画面が２つに分割され、下の画面にヘルプファイルが表示される（図１）。終了するときは(CTRL-X)(CTRL-C)と入力する。ここで、(ESC-X)は「エスケープキーを押した後、Ｘキーを押す」ことを表し、(CTRL-X)(CTRL-C)は「コントロールキーを押したまま、ＸキーとＣキーを続けて押す」ことを表している。

・ｎｇ

ｋｅｍａｃｓと同様にｅｍａｃｓの一種であるが、ｍｇ（ＭｉｃｒｏＧｎｕＥｍａｃｓ）をベースにしている。日本語を扱うことはできるが、メッセージは日本語化されていない。コマンド名やファイル名の補完機能などの特徴がある。ｖｉと並んで最近のワークステーションにおける標準的なエディタとなりつつあるｎｅｍａｃｓに似た操作感を実現している。

例）juusho.txt を編集する。

```
F:¥> ng juusho.txt
```

と入力してｎｇを起動する。(ESC-X)を押した後、スペースキーを押すと、以下のように画面が２つに分割され、(ESC-X)に続くコマンドの候補が表示される（図２）。この状態から、例えばｂｓと入力してスペースキーを２回押すと、最下行の表示が

```
M-x bsmap-mode
```

となり、リターンキーを押すとこのコマンドが実行される。終了するときはｋｅｍａｃｓと同様に(CTRL-X)(CTRL-C)と入力する。

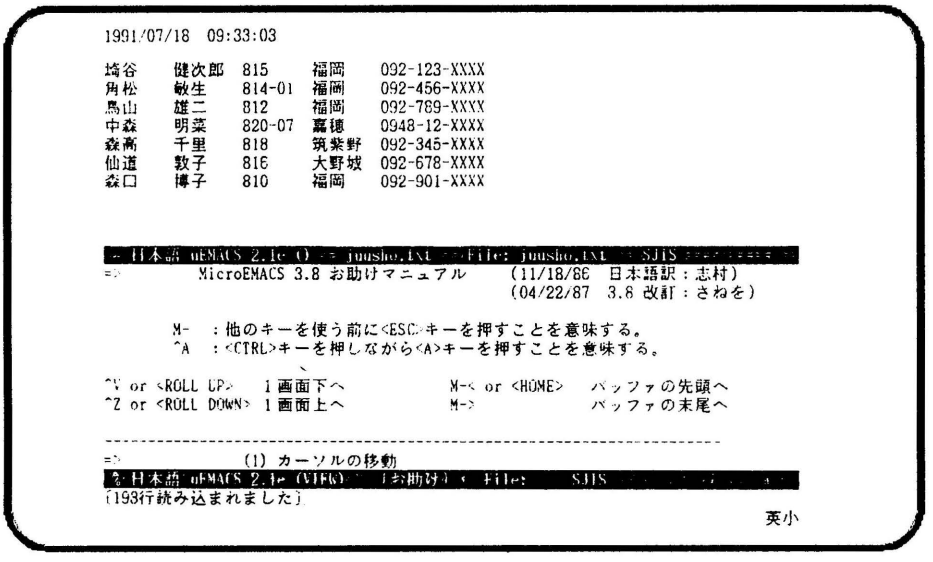

図1. kemacsの画面

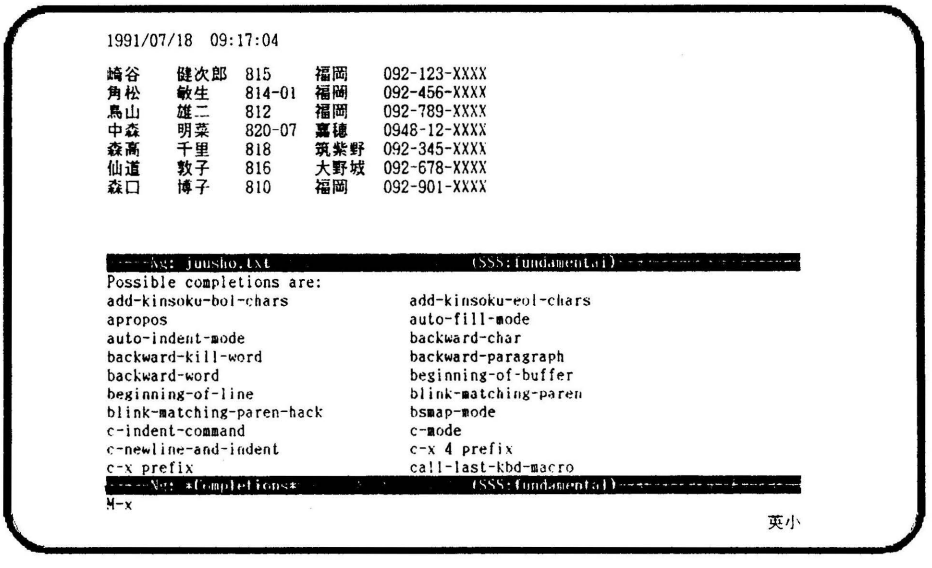

図2. ngの画面

・ｊｓｔｅｖｉｅ

ｖｉが持つ多くの機能を実現したエディタで、操作感もｖｉに近い。：キーを押した後ｈｅｌｐと入力すると、簡単なヘルプ画面が表示される。

例）juusho.txt を編集する。

F:¥> jstevie juusho.txt

と入力すると、以下の画面が表示される（図３）。終了するときはＺＺ（シフトキーを押したままｚを２回押す）と入力する。

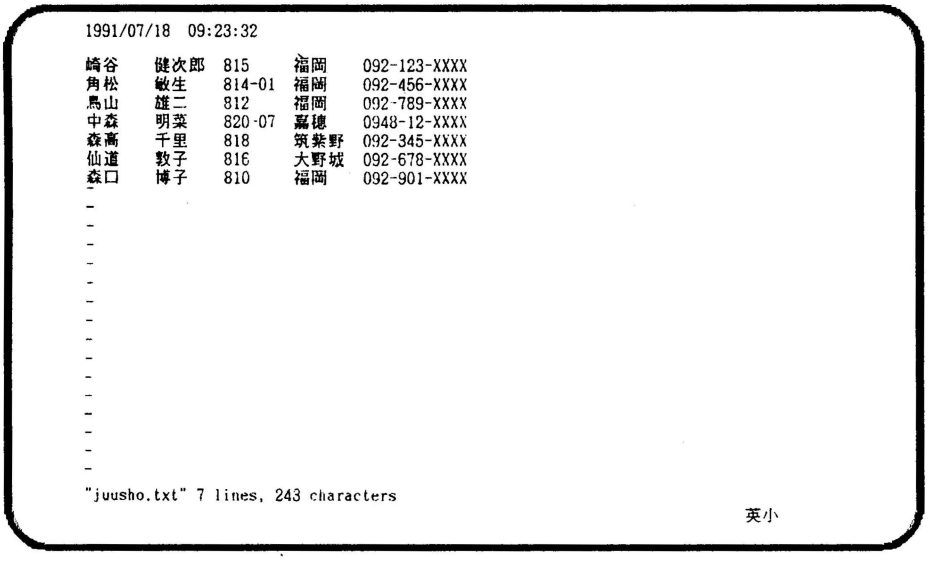

図３．　ｊｓｔｅｖｉｅの画面

・ｌｅｓｓ

less [-dstwcCepmMqQuU] [-hN] [-bN] [-xN] [+command] [file...]

ページャの一種で、長いテキストファイルを読むのに使う。ＭＳ－ＤＯＳのＭＯＲＥよりも高機能になっている。ページ単位や行単位で上下にスクロールでき、特定の文字列を含む行やファイルの先頭あるいは最後にジャンプすることもできる。エディタと違ってファイルの編集はできないが、ファイルの一部分しか主記憶に読み込まないので、非常に大きなファイル、例えば主記憶の容量（１メガバイト）よりも大きいファイルを読むのに便利である。

例）拡張子 .txt を持つファイルを読む。

F:¥> less *.txt

と入力すると、以下の画面が表示される（図４）。この例では拡張子 .txt を持つファイルは３つあり、１つめのファイル juusho.txt を現在読んでいる、ということが最下行に表示されている。

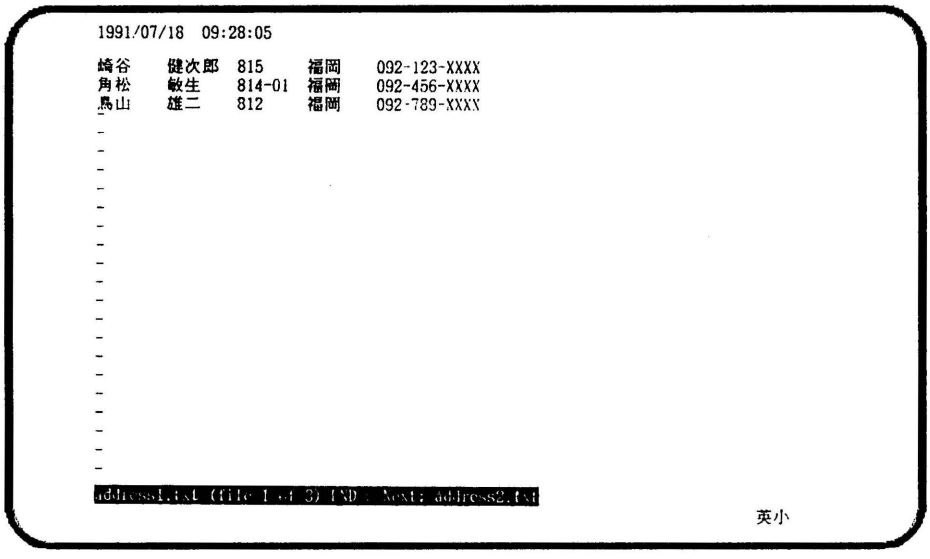

図４．　ｌｅｓｓの画面

５．メモリ常駐型のヒストリ機能プログラム

ここまではＵＮＩＸ風のコマンドに限定して説明してきたが、本節では特別にその制限を離れ、ＵＮＩＸ風でないコマンドについて解説する。最近のＵＮＩＸのシェルはファイル名・コマンド名の補完機能や行内編集機能を持っていて、ここで解説するヒストリ機能プログラムを使うことでその便利さを実感して貰いたいのが理由である。

・ｈｉｓ

センターで利用できるバージョン（Ver 4.48x）では以下の機能が実現されている。

(a) 入力文字列履歴の検索・再入力
(b) 行内での強力な編集機能
(c) 別名定義(alias)機能
(d) ディレクトリ変更履歴の検索・再入力
(e) 複文記述機能（複行連続入力代行）
(f) 履歴に対するバッチ実行機能
(g) 環境変数の展開
(h) ファイル名・コマンド名の補完

プログラムの本体は H:¥HISTORY¥HISTORY.COM だが、ｃｓｈのｈｉｓｔｏｒｙコマンドと区別するためにセンターでは H:¥LOCALBIN¥HIS.BAT というバッチファイルを用意して使っている（ｃｓｈと一緒に使うことはできない）。以下では単にｈｉｓコマンドと呼ぶ。

例）ｈｉｓコマンドを実行し、プログラムをメモリに常駐させる。解放するときは－ｑオプションを使う。常駐したときに表示されるように、(CTRL-])を押すと、メニュー形式のヘルプ画面が表示される。

```
F:¥> his

F:¥>H:¥HISTORY¥HISTORY -ir -p -dH:¥HISTORY    &4
HISTORY.COM (Ver 4.48x)  (C) 1986,87,88  by T.Oka
          Hit  CTRL-]  to show help menu.

F:¥> his -q

F:¥>H:¥HISTORY¥HISTORY -ir -p -dH:¥HISTORY -q   &4
Remove HISTORY.COM
```

例）以下の例ではプログラムが常駐した状態とする。(CTRL-L)でコマンドの履歴を新しいものから２０個表示する。

```
HISTORY LIST...
[1] adddev sysstd
    ～中略～
[19] cat address?.txt > juusho.txt
[20] less *.txt
```

例）(CTRL-W)で１つ前コマンドをコマンド入力行に表示する（ＭＳ－ＤＯＳのテンプレート機能 PF3 と同じ）。実行したい場合はそのままリターンキーを押す。

```
F:> less *.txt
```

例）リターンキーを押さずに、もう一度(CTRL-W)を押すと、２つ前のコマンドを表示する。

```
F:¥> cat address?.txt > juusho.txt
```

例）コマンドを編集する。(CTRL-S)と(CTRL-D)で左右、(CTRL-G)で一文字消去、(CTRL-B)で行頭／行末へ移動する。カーソルの位置をアンダーラインで示す。

```
F:¥> cat address?.txt > juusho.txt_
        ↓ (CTRL-B)を１回押して、行頭に移動
F:¥> cat address?.txt > juusho.txt
        ↓ (CTRL-D)を１１回押す
F:¥> cat address?.txt > juusho.txt
        ↓ (CTRL-G)を１回押して、１文字削除する
F:¥> cat address.txt > juusho.txt
        ↓ 2 を入力する
F:¥> cat address2.txt > juusho.txt
```

例）別名定義機能を使う。

```
F:¥> ls ls -1Fa(ESC-I)
alias   ls      ls -1Fa

F:¥> ls
address1.txt
address2.txt
dir1/
dir2/
juusho.txt
```

例）コマンド名の補完機能を使う。

```
F:¥> o
      ↓ (CTRL-U)を押す
F:¥> oaksetup
      ↓ もう一度(CTRL-U)を押す
F:¥> oakrep
```

6．終わりに

本解説ではＭＳ－ＤＯＳ上で使えるＵＮＩＸ風コマンドについて解説した。行いたい作業によってＭＳ－ＤＯＳのコマンドと使い分けて頂ければ幸いである。また、解説したコマンドはすべて、作者が著作権を放棄したパブリック・ドメイン・ソフトウェア（ＰＤＳ）か、あるいは著作権の放棄はしないが一定の条件の下でコピーフリーであることを宣言したフリー・ソフトウェア（ＦＳ）である。現在センターのパソコン端末で利用できるＰＤＳやＦＳはほとんどがＵＮＩＸからの移植だが、他にも有用なＰＤＳやＦＳがネットワークを介して流通している。これらのソフトウェアを順次整備していくことで、少しでも使いやすい環境にしていきたいと考えている。特に情報処理教育センターのように多数の端末を有して集合教育を行う所では、その操作環境を少しでも新しいものにしていく必要がある反面、契約などの問題により新しいソフトウェアを導入することが難しい。ＰＤＳやＦＳを整備することで、このジレンマを少しは解消できないだろうか。

謝辞

九州大学教養部数学教室の宮脇伊佐夫先生および情報科学の広川佐千男先生と宮原哲浩先生にはｌｓ、ｃａｔ、ｃｐ、ｍｖ、ｄｕ、ｆｉｎｄ、ｔａｉｌ、ｗｃ、ｃｕｔ、ｕｎｉｑ、ｃｏｍｐｒｅｓｓ、ｐｓ、ｃｓｈ、ｋｅｍａｃｓおよびｌｅｓｓの整備をして頂いた。また、ｒｍ、ｄｉｆｆ、ｗｈｉｃｈ、ｎｇ、ｊｓｔｅｖｉｅおよびｈｉｓに関しては、九州大学工学部情報工学科修士２年の下川俊彦氏にソフトウェアのリストやアーカイブファイルを提供して頂いた。ここに記して感謝する。

参考文献

[1] Brian W. Kernighan and Rob Pike 著，石田晴久　監訳：
ＵＮＩＸプログラミング環境，アスキー出版局。

[2] Mark G.Sobell 著，安居院猛　監訳，大垣一隆・新井幸宏　訳：
実用ＵＮＩＸシステム，工学社。

[3] 日本語ＭＳ－ＤＯＳ™Ｖ３．１ユーザーズリファレンス／ＦＭ　Ｒシリーズ，
富士通株式会社。