UTSのエディタ

松尾, 文碩
九州大学大型計算機センター研究開発部

https://doi.org/10.15017/1468156

出版情報：九州大学大型計算機センター広報. 20 (5), pp.424-440, 1987-09-25. 九州大学大型計算機センター
バージョン：
権利関係：

UTSのエディタ

松尾文碩*

1. UTSで使用可能なエディタ

UTSのエディタは，日本語関係を除けば，次の7種がある[1, 2]．

- ed　　行エディタ
- vi　　画面エディタ
- ex　　行エディタ(コマンド行が画面の下部にある)
- sed　　フィルタとして使われる非会話型行エディタ
- bfs　　大きなファイルを読むだけのシステム
- ned　　画面エディタ
- scope　　C言語構文向画面エディタ

上記システムのうち，nedとscopeは一部の端末でしか使えない．

ここでは，UNIX固有のedとUNIXの標準画面エディタであるviの使用法を述べる．edよりviの方が便利である．しかし，viは端末の画面制御を行うため，terminfo[3]に登録されている端末でしか使えない．したがって，登録されていない端末でUTSを使う場合，edを使わざるをえない．また，viを使いこなすためには，ある程度edのことを知っている必要がある．viは後述するように，edと同様に，コマンドモードとテキストモードの二つのモードがある．このため，viを嫌う人も多い．使いやすさからいえば，パーソナルコンピュータのWordStarやWordMasterの方が上だ．しかし，UNIXではこれらのエディタは動かない．モードのないエディタで，UNIXで動くものにemacsがある．近いうちにemacsを導入する計画である．

なお，紙面の都合で，edとviのすべての機能を紹介することはできない．しかし，本稿程度のことを知っていれば，まず不自由はないであろう．

2. edエディタ

図1に示すように，edエディタに入るにはedと入力してreturnキーを押せばよい．エディタから抜け出るには，qと入力してreturnキーを押す(quit要求)．図1では，最後のreturnキーを省略している．edエディタでは，コマンドの最後には，returnキーを押さなければならないが，この節では以後returnキーを押す部分は省略する．

edエディタは，コマンドモードのとき，通常，促進記号を出さず，黙っている．相手が黙っていると，生きているか死んでいるかわからず不安だという人は，Pを入力すればよい

昭和62年7月28日受理

*九州大学大型計算機センター研究開発部

(prompt要求). 促進記号として * が出るようになる.

edエディタは, バッファとよぶ一時的ファイル上のテキストを編集の対象とする. edと入力して最初にコマンドモードになったときは, バッファは空っぽである. すでに, 再編集したいテキストがファイルに保存されている場合, これをバッファにもってくるには,

```
e doc
```

のように入力する(edit要求).
ここで, docはそのテキストが入っているファイル名である. いまバッファが空っぽでなく, 何かテキストが入っている場合には, そのテキストは破壊され, バッファはdocのテキストで置き換わる. 逆に, バッファの内容をファイルに書くには,

```
w doc
```

のように入力する(write要求). eとwが実行されると, テキストの文字数が表示される.

```
ed
```

```
e ファイル名
```

の代わりに,

```
ed ファイル名
```

とすることができる. いったん, eまたはedでファイル名を指定すると, wの後のファイル名は省略することができる. 省略すると最後に指定したファイルに書かれる. 記憶しているファイル名を知りたいときには

```
f
```

と入力する(file要求). また,

```
f ファイル名
```

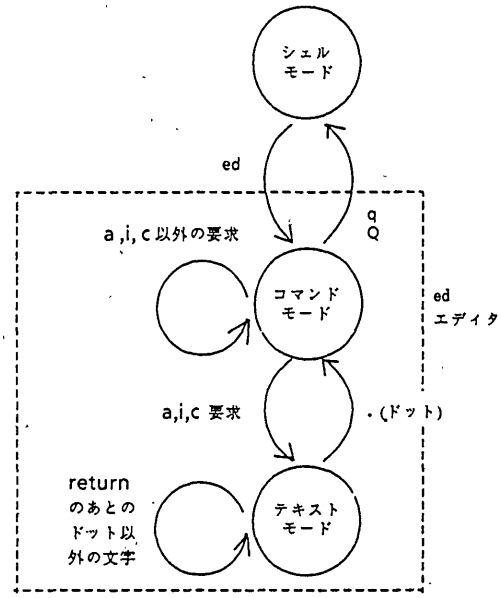

図1. edエディタのモード

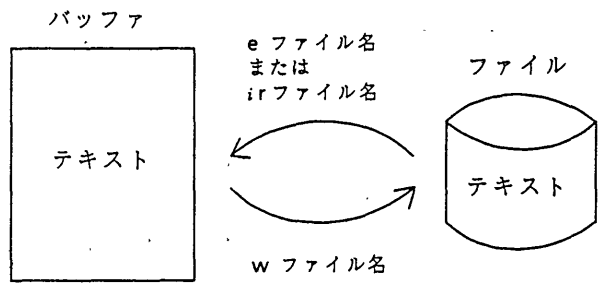

図2. バッファーファイル間のテキスト転送
(*i* は行番号)

とすると，ファイル名が変わる．しかし，バッファは変化しない．

バッファに新しくテキストをつくる場合は，aと入力する．すると，エディタは入力モードになる．そこで，入力した文字はすべてバッファに入る．例外は，returnキーを押したあとに．(ドット)を入力すると，コマンドモードに戻る．図3.のように入力すると行-1から行-nまでがバッファに入る．各行の終わりでreturnキーが押されているが，図3では省略されている．つまり，一行は，returnキーで終わる．テキストもreturnキーに対応してLF(ラインフィード)の制御キャラクタ(16進数0A)が入る．行-1，行-2，---はそれぞれ行番号1, 2, ---によって参照される．

```
a
行-1
行-2
 ⋮
 ⋮
行-n
.
```

図3. a要求

さて，これまでにでてきた，e, w, a などをedエディタでは要求(request)という．要求はすべて1文字である．すべての要求は，コマンドモードでのみ受付けられる．エディタが要求を理解できなかったり，うまく実行できなかったときは，?を返す．?が返ってくるのは，利用者の単純なミスのことが多く，すぐにわかる．エディタに駄目な理由を聞きたいときは，

h

を入力してみるとよい(help要求)．すると，エディタはエラーメッセージを出す．エラーのとき，常にエラーメッセージが欲しいときには，

H

と入力すればよい．

エディタからシェルのコマンドを呼ぶには，コマンドの前に!をつける．たとえば，

!ls

のようにする．コマンドの実行が終わると再びedのコマンドモードに戻る．バッファは変化しない．

q要求では?が返ってくることがある．これは警告である．バッファを保存せずにエディタから抜けようとしたためである．もう一度q要求をすれば，シェルに戻る．2度q要求をするのが嫌いな人は，

Q

と入力すればよい．一度でシェルに戻る．

2.1 行編集

行編集の要求は，一般的に次の形式である．

開始行番号,終了行番号　要求

開始行と終了行が同じであるとき，すなわち，ある特定の一行に対する要求の場合は，

行番号　要求

の形式でよい.

よく使用される要求には,次のものがある.ここで,mとn,iは,行番号であり,$m \leqq n$.

m, np	第m行から第n行までのすべての行を表示せよ(print要求).
m, nn	第m行から第n行まで行番号をつけてすべての行を表示せよ.
m, nd	第m行から第n行までのすべての行を消去せよ(delete要求).
m, nw doc	第m行から第n行までのすべての行をdocというファイルに書け.m,nを省略すると,バッファ中の全行がファイルに書かれる.
ir doc	docというファイルのテキストを,バッファの第i行のあとに挿入せよ(read要求).iを省略すると,最後の行のあとに挿入される.
m,nm i	第m行から第n行までのすべての行を第i行のあとに移せ(move要求).
m,nti	第m行から第n行までのすべての行を第i行のあとにコピーせよ.

$は,バッファの最後の行を示すのに用いることができる.したがって,

```
1,$ p
```

はバッファの全テキストを表示する.1,$ は,で置き換えることができる.つまり,

```
,p
```

は,上と同じ要求である.また,

```
,d
```

とすると,バッファにある全テキストが消去される.

edエディタは,バッファのテキストのある行を指差している.これを現在行(current line)という.現在行は,ドット(dot)とも呼ばれ,.で表す.現在行は要求によって変化する.どう変化するかは,要求によって決まる.通常は,変化の最後の行である.現在行は,利用者によっても変えることができる.第5行を現在行にしたければ,5と入力すればよい.このとき,第5行が表示される.また,現在行を1行進めるには,+と入力する.一行戻すのは,−である.+++と入力すれば,3行進むし,−−−−と入力すれば4行戻る.行編集要求の行番号として,.を使うことができる.

```
.,.p
```

は,

```
p
```

でよい.p,d要求において行番号を省略すると,現在行を指定したことになる.

```
.,$d
```

は,現在行から最後の行までを消去する.現在行がどこにあるかを知りたいときには,

```
. =
```

と要求する.すると,現在行の行番号が表示される.

行編集要求の行番号には,＋と－を用いて.(現在行)と$(最終行)との相対番号をあたえることができる.つぎに代表例を二つ示す.

```
$－9,$p          最後の10行を表示せよ.
.－5,.＋5p       現在行を中心に前後11行を表示せよ.
```

上の表現において,.または＋のどちらかは省略することができる.つまり,.3または＋3は,

```
. + 3
```

の省略形である.また,－のあとの数字が省略されたときは,その数字は1である.たとえば,$.－は$－1の省略形である.

バッファのテキストを追加挿入するには,a要求かi要求を行う.

```
7a
追加テキスト
.
```

とすると,第7行のあとに追加テキストが挿入される.

```
8i
追加テキスト
.
```

は第8行の前に追加テキストを挿入する.8iは7aと同じ効果をもたらす.c要求は消去要求と挿入要求を組み合わせたものである.

```
15,20c
追加テキスト
.
```

は,第15行から第20行までを消去し,第14行のあとに追加テキストを挿入する.
同様に,

```
10c
追加テキスト
.
```

は,第10行を消去し,第9行のあとに追加テキストを挿入する.p,d要求同様,a,i,c要求において,行番号を省略すると現在行を指定したことになる.

行番号の代わりに英小文字を使うことができる.行番号を英小文字に割り当てる形式は

```
行番号k 英小文字
```

である.たとえば,

```
10ka
```

とすれば,第10行は,

```
'a
```

で参照することができる.

行の指定を常に行番号だけで行わなければならないとすると,編集作業は楽ではない.行に含まれる文字列で行を指定するには,

```
/if/,/}/d
```

のようにする.この場合,ifを含む行から}を含む行までが消去される.この場合,}を含む行はifを含む行と同じかあとにあることを仮定している.ifと}を含む行のサーチは,現在行から始め,最後の行まで行き,その間になければ,先頭行に行き,見つからなければ現在行に戻ってくる.その間に最初にifを含んでいる行が/if/である./}/の意味も同様である./if/と/}/のサーチは同時である./}/のサーチを/if/が見つかったあとから始めるようにするためには,,の代わりに;を使い,

```
/if/;/}/d
```

のようにする.

/---/に対しては,$と.と同様,+と−による相対行を指示することができる.たとえば,

```
/main/ +1,$p
```

と入力すると,mainを含む行のつぎの行から最後の行までが表示される.また,省略形も同様である.上の要求は,

```
/main/1,$p
```

と書くことができる.

行番号を入れて現在行を変えたように,

```
/continue/
```

と入力すると/continue/を含む行が見つかれば,/continue/が現在行となる.行番号を入れたときと同様,サーチに成功すればこの行が表示される.

文字列のサーチで,/---/の代わりに?---?を使うと,サーチの方向が逆になる.つまり,現在行から始めて行1の方向へ探していく.この間に見つからなければ,バッファの最後に行き,現在行の方向へサーチする.見つからなければ現在行まで戻る.

2.2 代入と行分割

代入要求は,文字列を他の文字列で置き換える要求である.この一般形は,

```
開始行,終了行 s/旧文字列/新文字列/
```

である.s要求は,開始行から終了行までのすべての行において,最初に出現する旧文字列を新文字列で置き換える(substitute要求).開始行と終了行の指定の方法や省略時の意味は,p,d要求と同じである.以下に二つの使用例を示す.

```
s/stract /struct/      現在行の最初に出現するstractをstructで置き換える.
.,.+2s/char//          現在行からその二行下までの間で最初に現れるcharを消去する.
```

s要求の旧文字列は,別の旧文字列を指定するまでは記憶されていて,同じ旧文字列のs要求において旧文字列を省略することができる.たとえば,

```
s/int/char/
+1s//long/
```

としたとき,二番目の旧文字列はintである.また,直前のサーチの文字列も同様な省略で旧文字列として使うことができる.たとえば,つぎのような使い方ができる.

```
/int/
s//long/
```

intを含む行を探し,そのintをlongで置き換える.
(二つの要求の間にintを含む行が表示される)

この代わりに

```
/int/s//long/
```

としても同じである.

文字列の指定には,必ずしも/で区切らなくてもよい.ほかの文字でよい.たとえば,

```
s|long|int|
```

のようにすることができる.指定文字列が/を含む場合に便利だ.ただし,指定文字列に含まれる文字を区切り記号に使ってはならない.

```
smminimummmaximumm
```

とすると,minimumをmaximumに代えるのか,あるいはminiをummmaximumに代えるのかなどのあいまい性があり,エディタは要求を理解できない.

行の分割にもs要求を用いる.現在行

```
---}{---
```

であるとき,

```
s/}{/}\
{/
```

とすると,

```
---}
{---
```

と二行になる.第m行から第n行までを一行にまとめるには,

```
m,nj
```

とする.

```
j
```

は,

```
.,.+1j
```

の意味である.

誤って,代入要求をしてしまった場合,

```
u
```

とすると,もとの形に戻る(undo要求).この要求は,最後に実行された代入要求だけに有効である.もとに戻ったかどうかの確認をしたければ,u要求のあとにp要求を行えば

よい.たとえば,

```
up
```

と入力する.

s要求には,gとpの修飾子をつけることができる. g修飾子は,出現するすべての旧文字列を新文字列で置き換える指示である.たとえば,

```
,s/int/long/g
```

とするとバッファにおけるintのすべての生起がlongで置き換わる.p修飾子は,代入された結果を表示する.二つの例を示す.

```
s/x/y/p          現在行の最初のxをyに書き換え,表示せよ.
,s/x/y/g p       バッファのすべてのxをyに書き換え,変化のあった最後の行を表示せよ.
```

2.3 大域編集

gは要求としても使うことができる(global要求). g要求の一般形は,

```
g/文字列/1個以上の要求
```

である.バッファの全行をサーチして文字列が見つかるたびに要求を実行する.たとえば

```
g/if/p
```

は, ifを含む全行を表示する.また,

```
g/int/s//long/g p
```

は,すべてのintの生起をlongに変え,変更のあった行はすべて表示する.

```
,s/int/long/g p
```

では,変更のあった最後の行しか表示しない.また, g要求はs要求と異なり,intを含む行がみつからなくても,?を返さない. g要求では,その後に複数の要求が伴ってもよい.ただし,g要求は許されない.g要求が複数行になるときは,最後の行を除いて\で終わらなければならない.例を示す.

```
g/temp/s//t/\
a\                                  } tempがあるとtに代えそのあとに
/* temp is replaced by t */\          /*---*/を挿入する.
.
```

gの代わりにvとすると,指定した文字列を含まない行に対して要求が実行される.

g要求あるいはv要求を実行中に,この要求を中止したければ,

```
del
```

と打つ.ある文字列を含む全行の表示を要求しておいて,途中で表示を止めたいときなどに有効である.しかし,delはgv要求だけでなく,すべての要求の中止に使うことができる.

2.4 文字列の指定

文字列の指定において,行の先頭位置と最終位置は,それぞれ ^ と $ によって指示できる.いくつかの例を示す.

/^main/	main で始まる行を探せ.
/};$/	}; で終わる行を探せ.
/^$/	空行を探せ
/^main()$/	main() だけを含む行を探せ.

もちろん,これらの記号はs要求の文字列においても使うことができる.二つの使用例を示す.

s/^/§5./	現在行の先頭に §5. を挿入せよ
s/};$/}/g	バッファの全行において}; で終わる箇所を}で置き換える.

文字列のサーチにおいて . は任意の一文字を意味する.また * は,ある文字が0個以上繰り返された文字列を意味する.たとえば,x* はxの0個以上からなる文字列である.いま現在行が

```
main(xxx)
```

であるとし,xxxを消去したい.

```
s/x*//
```

としても何も変化しない.行の先頭で0個のxを探し,見つかるからである.この場合,たとえば,

```
s/(x*/(/
```

とすると,xxxが消去される.0個以上の任意の文字列を指定するには,.* とする.

```
/if(x.*y)/
```

は,if(x < y),if(x < =y) などを探し出す. * は,最長文字列とマッチする.たとえば,

```
s/{.*;//
```

は,現在行の最初の { から行の最後の ; までの文字列を消去する.

また,つぎのように一文字の文字クラスを指定することができる(シェルと同じ).

[,.;:]	,,.,;,: のどれかの文字
[0-9]	0から9までのどれか数字
[a-z]	aからzまでのどれか英小文字
[I-N]	IからNまでのどれか英大文字
[^0-9]	数字以外の任意文字.[の直後の ^ は,このクラスが[0-9]の補集合であることを意味する.
[]^\]	],^,\のいずれかの文字.] を含むためには,第1文字の位置に置かなければならない.^を含むためには第1文字の位置に置いてはならない.置くと補集合の意味をもつ.

＜使用例＞

```
g/[^;}]$/p        行が;,}で終わっていない行を表示せよ.
v/[;}]$/p         上と同じ.
```

文字列の指定において,つぎの文字は特別な意味をもつ.

```
^  .  $  [  *  \  /
```

それらの記号を,文字列で指定するには,前に\をつける.すると,特別な意味からエスケープし,その文字自身を指定したことになる.

＜使用例＞

```
g/[0-9]*\.[0-9]*/p        たとえば,326.07のような文字列を含む
                          行をすべて表示せよ.
```

代入において,旧文字列は新文字列のなかで&で参照することができる.

＜使用例＞

```
s/x + y/(&)/              x + yを(x + y)で置き換えよ.
```

3. viエディタ

viエディタへ入るには,

```
vi ファイル名
```

とする.ファイル名は,既存でも,新規でもよい.既存の場合,テキストがファイルからバッファに読み込まれ,画面上には,先頭部分が表示される.viもedと同様,バッファと呼ばれる一時的ファイルのテキストを編集の対象とする.ファイルが新規のときは,バッファは空っぽである.画面には,左端に~がある行が表示される.~は,行がないことを表わしている.コマンドモードでは,:のあとにedの大部分の要求を入力することができる(全部ではない).:を押すと,カーソルは左下に移動する.そのあとにed要求を入力する.たとえば,ファイルにバッファの中味を移す(図2参照)には:wと入力して,returnキーを押す.edでは,すべての要求の最後にreturnキーを押さねばならないので,前節ではこの部分を明示しなかった.viでは,常にreturnキーを押さなければならないというわけではないので,この節ではreturnキー押下を↙で表す.viからシェルに戻るには,:q↙とする.しかし,バッファが変化を受けているときには,w要求を行わなければ,戻れない.edのようにもう一度q要求を行っても駄目だ.また,Q要求は受け付けない.通常の終り方は,:wq↙(ファイルに書いて終わる)か:q!↙(強制的に終る,edのQと同じ)である.よく使われるed要求には,wとqのほか,次のものがある.

```
:r doc            docというファイルを読め.
:s ---            代入せよ.
:g ---            全バッファを探せ.
:!シェル コマンド  コマンドを実行してviに戻れ.戻ったあと,↙を入
                  力する.
```

: sh ↙　　シェルへエスケープせよ. シェルの処理が終ったら ^d を入れる. ^d は, control キーを押しながら d を入力することを意味し, control-d という.

vi では, ed と異なり, エラーメッセージが画面の最下部に常に出る.

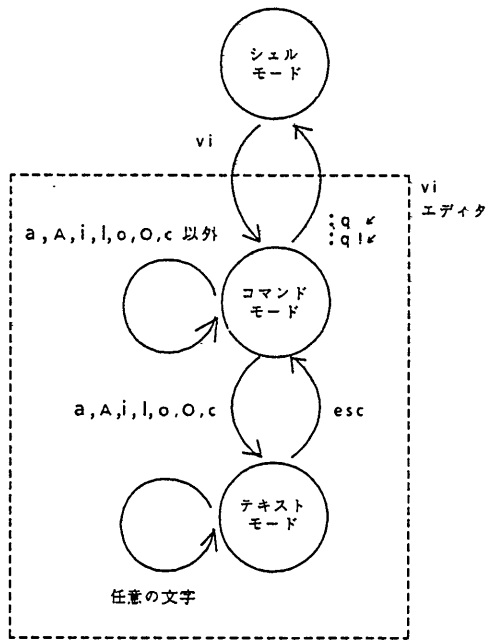

図4. vi エディタのモード

3.1 画面移動

他の画面エディタ同様, 画面にはバッファの一部分しか出ない. 画面は, バッファへの窓(window)の役目をする. 窓を下げるには,

^d

とする. ^d は, 前にも述べたようにcontrol キーを押しながら d キーを押すことを意味する. 窓を上げるには,

^u

とする. ^d と ^u は, スクロールして窓の中の約半分の行を入れ替える. ほぼ全行を入れ替えたいのであれば, ^d の代わりに,

^f

^u の代わりに

^b

とする. ^f と ^b は, いったん画面をクリアしてほぼ全行を入れ替える. 一行下にスクロールするには,

^e

を使う. また, 一行上にスクロールするには,

^y

を使う.

3.2 カーソル移動

カーソル移動には, h, j, k, l キーを使う. 図5に示すように, これらのキーは, キーボード上では, 一列に並んでいる, カーソル移動の方向は, 図5においてキーの上の矢印で示す. カーソル移動のときは, 右手の人差し指から小指までの4本の指をこれらのキーの上に置く. 両側の指で横方向の移動を行う. 向きは自然だ. 内側の二つの指が縦方向である.

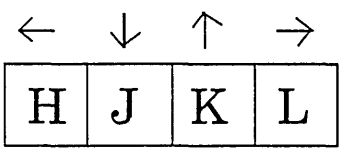

図5. カーソル移動のキーと移動方向

次のようなキーによる移動もできる.

space	右に一文字分カーソルを移動せよ(l と同じ).
back space	左に一文字分カーソルを移動せよ(h と同じ).
^h	back space と同じ.
return	次の行の先頭にカーソルを移動せよ.
+	return と同じ.
−	前の行の先頭にカーソルを移動せよ.
$	行の最後にカーソルを移動せよ.
^	行の先頭にカーソルを移動せよ.
m G	第*m*行にカーソルを移動せよ. *m* がないと, $ と同じ.

単語単位の移動もできる.

w	次の単語の最初にカーソルを移動せよ.
b	前の単語の最初にカーソルを移動せよ.
e	現在カーソルがある単語の最後にカーソルを移動せよ.

w と b, e は, 句読点を一つの単語とみなす. 句読点でカーソルを停止したくないときは, w と b, e の代りに大文字の W と B, E を使えばよい. これらの要求の前には, 数を置くこと

ができる.たとえば,

10 w

とすると10単語先へ進む.この数は,要求の繰り返し回数を指示する.

ある特定の文字列を探して,そこにカーソルを移動させることができる.それには,

/文字列↙

と打つ./を入れたとたん,カーソルは左下に移動し,文字列の入力が可能になる.サーチの方向は,edの/---/と同じである.すなわち,/の入力時にカーソルがあった位置から,バッファの後部へと進み,バッファの最後まで行くと,バッファの先頭に回り,カーソル位置まで戻ってくる.この間,最初に見つかった文字列の先頭位置にカーソルが移動する.同じ文字列の次の生起位置が欲しければ,

n

と打つ.nを入力するたびに,その次の位置に移動する.サーチの方向を逆にしたければ,ed同様,/の代わりに?を使い,

?文字列↙

とする.

3.3 追加,消去,変更

カーソルのあとにテキストを追加するには,

a　テキスト　esc

とする(append要求). escは,escキーを押すことを意味する.カーソルのまえに,テキストを追加するには,

i　テキスト　esc

とする(insert要求).カーソルのある行のあとにテキストを追加するには,

A　テキスト　esc

とする.カーソルのある行の前にテキストを追加するには,

I　テキスト　esc

とする.行単位の追加には,次のようにo要求を使う(open要求).

o
テキスト
esc

o要求は,カーソルのある行の次の行からテキストをつくる.上の行につくるには,oの代りに,Oとすればよい.テキストを入力している途中で,修正するには,つぎの機能を使う.

^h　　1文字消去
^w　　1単語消去
^u　　1行消去

escで,編集モードに戻ったあとの消去と置換えの要求は,次のとおりである.カーソルのあるところが対象になる.

x	1文字消去
dw	1単語消去
dd	1行消去
D	行の残りを消去
rc	文字cで置き換え
~	大文字を小文字に小文字を大文字に入れ換え

このうち,上の三つの要求は,その前に繰り返し回数を置くことができる.

5x

とすると,カーソル位置から始まる5文字を消去する.

10dd

は,カーソルのある行から始まる10行を消去する.

d要求の次の文字であるrやwをオブジェクトという.オブジェクトは,要求の作用範囲を指定するために使う.一般的なオブジェクトには,次のものがある.

l	カーソルのある1文字.3lは,カーソルのある文字とそのあとの3文字.(方向は図5と同じ)
h	カーソルの前の1文字.5hは,カーソルの前の5文字.(方向は図5と同じ)
space	lと同じ.
back space	hと同じ.
fc	カーソルから文字cまでの後方文字列(cを含む).
Fc	カーソルから文字cまでの前方文字列(cを含む).
tc	カーソルから文字cの前までの後方文字列(cを含まない).
Tc	カーソルから文字cの後までの前方文字列(cを含まない).
w	カーソルの後方1単語.句読点を一つの単語とみる.2wは後方2単語.
W	カーソルの後方1単語.句読点は単語の切れ目とみない.5Wは後方5単語.
b	カーソルの前方1単語.句読点を一つの単語とみる.6bは前方6単語.
B	カーソルの前方1単語.句読点は単語の区切りとみない.2Bは前方2単語.
e	カーソルの後方1単語.句読点を単語の区切りとみる.範囲は,単語の最後の文字まで.w,W,b,Bは区切り記号も含む.3eは後方の3単語.
$	カーソルからその行の最後までの文字列.
j	カーソルのある行とそのあとの1行.4jは,カーソルのある行とそのあとの4行.(方向は図5と同じ)

k	カーソルのある行とその前の1行. 3kは, カーソルのある行とその前の3行. (方向は図5と同じ)
↙	jと同じ.
+	jと同じ.
−	kと同じ.
G	カーソルのある行から最終行まで. 3Gは, カーソルのある行から第3行までの範囲.
H	画面の最上位行. 7Hは, 最上位行からの上7行.
L	画面の最下位行. 4Lは, 最下位行からの下4行.
/文字列↙	カーソルより後方で文字列を含む最初の行.
?文字列↙	カーソルより前方で文字列を含む最初の行.
)	文の最後. 文は . , ! , ? の文字で終わるか空白行で終る.
(	現在の文の先頭.
}	現在のパラグラフの最後.
{	現在のパラグラフの先頭.
]]	現在のセクションの最後.
[[	現在のセクションの先頭.

パラグラフやセクションの指定に関しては, text formatter[4]と関連しているので, ここでは詳しい説明を省略する.

オブジェクトを伴う要求の一つに, c要求がある(change要求). これは, 文字列を新しい文字列で置き換える. *x* をオブジェクトとすると,

c*x* 新文字列 esc

＜使用例＞

c2w in the esc　　2単語を“in the”で置き換えよ.

c要求では, オブジェクトで範囲を指定すると, 範囲の最後の位置に $ が出る. これは, esc で消える.

3.5 移動とコピー

テキストの一部をテキストの別の位置に移動させるには, まず移動する部分を消去し, 続いて移動位置をカーソルで示す. そのあと

p

を入力すると, カーソルのあとに移行する(put要求). 行単位に消去されたならば, カーソルのある行の次の行に移動する. 文字単位の消去ならば, カーソルのある文字の次に移動する. カーソルの前に移動するときは, p の代わりに P を使う.

たとえばカーソルのある行を含めて続く4行を移動するには,

d3j

カーソル移動

p

とする.

コピーをとるときには,y要求を用いる(yank要求). yを入力するカーソル位置と,yのあとのオブジェクトでコピーする範囲を決め,コピーを置く場所にカーソルを移動させ,pまたはPと入力する.現在行をコピーするには,yとオブジェクトの代りにYと打てばよい.

たとえば,カーソル位置から始まる5文字を別の場所にコピーするには

y 5 l

カーソル移動

p

とする.

p(P)要求は,直前のd要求とy要求で指定された範囲の文字列または連続行をカーソルの直後(直前)に置く.その文字列または連続行は,別のd要求またはy要求がなされるまで変化しない.したがって,同じ箇所を複数の場所へコピーするには,y要求を繰り返す必要はない.また,同じ箇所を別のd要求やy要求を挟んで,複数の場所へコピーするにも,y要求を繰り返す必要はない.文字列または連続行を格納するレジスタが用意されている.たとえば,

" a y 5 l

とすると,レジスタaに5文字が格納される.レジスタaの内容をある場所に置くには,カーソルをそこに移動し,

" a p

または,

" a P

とすればよい.もちろん,レジスタ名はaだけでなく,ほかの英字でもよい.

3.6 繰り返しと取り消し

要求を取り消すには,delキーを押す.

.(ドット)を打つと,カーソルの位置に対して,直前の要求をもう一度繰り返す.ドットで繰り返される要求は,

a,i,o,d,c,p

である.

3.7 復元

最後に受けたバッファの変化は,復元することができる.

u

と入力すれば,元に戻る(undo要求).もう一度uを打つと,undoのundoを要求したことになり,変化を受けた状態に戻る.Uは,現在行を変化前の状態に戻す.

u要求以外でも,復元の方法がある.viは,最近消去されたテキストブロックを9個のレジスタに記憶している.ただし,数語からなるテキストブロックは記憶されない.これらのレ

ジスタには1から9までの番号がついている.5番目のレジスタの内容をどこかにいれたかったらそこにカーソルを移動し,

"5p

とする.

何番目かを忘れているときには,レジスタ1から見ていけばよい."1pとして駄目なら,uを打ち,もとに戻す.次に.を入れると要求が繰り返されるが,この場合,レジスタ番号が一つ増える.このプロセスを繰り返すと,レジスタ1から9までの内容を順次出すことができる.

3.8 画面清掃

エラーや割込みなどで,画面が乱れることがある.画面表示とバッファの内容が異なったときは,

^l

とすればよい.画面は,いったん消去されたあと,バッファの内容が画面に再表示される.viエディタでは,行消去を行うと,消去された行は,完全に消えるのではなく,@からなる行に代わる.^lで再表示すると,この@の行は消えてしまう.ただし,この@行を消去するためだけなら^lの代りに,

^r

とした方がよい.この方が再表示が速い.

参考文献

1. マニュアル　UTS エディタ使用手引書 V10L30系用, 24SP-1091-1, 富士通(株), 1987.
2. マニュアル　UTS Editing Guide V10L30, 24SP1091E-1, 富士通(株), 1987. (英語版).
3. 松延栄治　UTSの使用法, 九州大学大型計算機センター広報, Vol.20, No.5, 1987.
4. 松尾文碩　FACOM M780上のUTS ―世界最高速のUNIX―, 同上.