画面エディタマイクロEMACSの改良

本田道夫

I　マイクロEMACSとGNU－EMACS

エディタは，プログラムの開発や原稿の作成などにおいて計算機を利用するときもっとも頻繁に利用するユーティリティプログラムの１つであり，その機能や使いがっての良さが，プログラム開発などの効率に大きく影響する。このようなこともあり，[1] で述べたように，著者らは研究上使用する必要のある計算機のうちできるだけ多くのもので，共通コマンド体系の画面エディタを開発してきた。

その共通のコマンド体系としては，Richard Stallman により開発され，DEC (Digital Equipment Corporation) のシステム 2020 や，各種 Unix ワークステーションの上で稼働している「画面エディタEMACS」を参考にした。Unix ワークステションは，プログラムの開発環境が整備されており，計算機科学や人工知能などの研究のためのソフトウエアが数多く実現され，近年とくに普及してきており，われわれの研究室だけでなく，上記の分野を研究している研究室でもよく利用されている。そのため，出張先でも同じコマンド体系のエディタが利用でき，非常に便利である。ただし，ワークステーション上のEMACSは，GNU－EMACSと呼ばれるものであり，DEC 2020 上のEMACSとは，そのコマンド体系が若干異なる。

一方，著者の身近にあるパソコンや 16 ビットの Unix マシン（ワークステーションという程ではない）の上で従来から開発してきた画面エディタ（以後，

総称してマイクロEMACS)は，DEC 2020上のEMACSのコマンド体系に近いものであり，近年普及がめざましく，著者の研究室でも備えているUnixワークステーションを利用するときには，若干ではあるが戸惑うこともあった。

また最近では，身近なパソコンやUnixワークステーション上でも，数式を含む原稿作成のためのソフトウエアとして有名なTexが利用できるようになった。ただしTexの制御コマンドを含むテキストの作成は，TexではなくEMACSのようなエディタを利用する。その場合，従来のパソコン上のマイクロEMACSではテキストの領域が最大64キロバイトと小さいため，原稿を不自然に分割する必要が生じたこともある。また，16ビットUnixマシン上のマイクロEMACSではサポートしていたマルチウインドウやマルチバッファなどの有用な機能を，パソコン上のマイクロEMACSでは備えていなかったし，GNU－EMACSなどで有用に機能しているマーク位置を複数個(10個程度)保持するためのマークリング，編修中に削除された部分を複数個(10個程度)保存するためのキルバッファリングなども備えていなかった。

これらのことから，次の点を目標として，パソコン上のマイクロEMACSを新たに全面的に作成し直した。

(1) GNU-EMACSとの仕様の統一(ただし，かなり小さいコマンドサブセット)

(2) 扱えるテキストの大量化

(3) マルチバッファ，マルチウィンドウの機能追加

(4) 複数のマークとキルバッファなどの機能の追加

また，従来EMACSの説明書としては，DEC 2020上のEMACS，あるいはGNU－EMACSに付随していた英語版のマニュアルとそれを翻訳したもの[2]しか無かった。ただし，これらのマニュアルはEMACSの機能の多さを反映し，その分量も多く，しかも細部に渡って記述していることもあり，学部学生などの初心者には非常に読みづらいようである。一方，最近では学生の教育あるいは卒業研究でも，パソコンだけでなくUnixのワークステーションを利用することもあり，したがってGNU－EMACSなどを使用する場合

も生じている。幸い，新バージョンのマイクロEMACSはGNU－EMACSのコマンド体系のかなり小さいサブセットのものではあるが，比較的よく用いるコマンドを備えており，しかも，その程度の数のコマンドを知っていれば，さしあたりGNU－EMACSを用いるのにも不自由しないと思われる。そこでマイクロEMACSのマニュアルとしても，GNU－EMACSの簡易版マニュアルとしても利用できるように，マニュアルの整備を図った。

ただし，本稿では，新マイクロEMACSの仕様と処理方式について簡単に述べ，マニュアルは研究ノートとする。

II　新マイクロEMACSの仕様

新マイクロEMACSの概念やコマンド体系の詳細な仕様は，研究ノートとするマニュアルに記しているので，ここでは，主としてGNU-EMACSとの違いについて述べることにする。マイクロEMACSは基本的にはGNU－EMACSコマンド体系のかなり小さいサブセットのものとして設計している。しかし，GNU－EMACSではキー入力の簡単なコマンドとしては（著者の知る限りでは）備わっていないけれども，使用経験から有用と思われるいくつかの機能があることにも気がついた。また，ほぼ同様な機能ではあるが，多少仕様を変更した方が操作性がよくなるのではないかという機能にも気がついた。そこでマイクロEMACSでは，新たなコマンドをGNU－EMACSでは使用していないキーに割り当て，また，仕様を変更してもさして混乱することはないと思われる変更を取り入れることにした。ただし，GNU－EMACSでは，これらの機能をマクロ定義機能などを用いて既存のコマンドを組合せて新たなコマンドとして定義し，さらにキーバインド機能により簡単なキー入力に対応させることも可能であることはことわっておく。

そのような追加機能や仕様変更は次の７点であるが，以下ではその理由を述べることにしたい。

- 検索中の文字入力の間違いの訂正
- カーソルの上下の行への移動コマンド

- 画面上の行の先頭と最後へのカーソルの移動
- 画面の最上行と最下行へのカーソルの移動
- 数指定による目的行へのカーソルの移動
- 領域のファイルへの書き出し
- 定義時には実行しないようなマクロの定義方法

(1) 検索中の文字入力の間違いの訂正

他のエディタと大きく異なるものとして，ＥＭＡＣＳにはインクリメンタル探索の機能がある。一般に他のエディタの検索機能では，文字列探索のコマンドにつづいて目的とする文字列を入力したあと，探索開始を指示する特殊な文字（たとえばエスケープなど）を入力すると探索を始める。しかし，ＥＭＡＣＳでは文字列探索のコマンドにつづいて入力される１文字入力ごとに，それまで入力された文字列を探索し，カーソルを検索された位置まで移動する。これは文字の入力を間違えて検索が失敗すればただちに入力ミスが発見できるなど，慣れると非常に有用な機能である。

しかし，間違えた入力の文字列がたまたまテキスト中に存在する場合には，その間違えた文字列の探索が成功することになる。したがって，目的とする文字列を探索するためには，その間違えた（おそらく最後に入力された）文字を取り消す必要がある。しかし，筆者が知る限りでは，ＧＮＵ－ＥＭＡＣＳには，その間違えた文字だけを簡単に取り消す方法はないようである。取り消そうと思ってバックスペース（ＢＳ）を入力すると，それは探索を終了してテキストのカーソル位置からの逆方向への１文字削除のコマンドとなり，探索文字列の最後の文字の削除とはならない。

そこで，マイクロＥＭＡＣＳではＧＮＵ－ＥＭＡＣＳでは使用していないＤＥＬキーを，探索文字列の最後の文字列の削除に使用することにした。

この場合，間違えた文字列の探索は成功しているので，カーソルはその文字列の直後に位置しているが，ＤＥＬキーで最後の文字が取り消されると，その取り消された文字が入力される直前の位置にカーソルが戻る。さらに文字を入力して，その位置から探索を続けることができる。

(2) カーソルの上下の行への移動コマンド

GNU－EMACS，およびマイクロEMACSともテキスト中で改行と改行の間がCRT画面の横方向の文字数を越えるような長い行を扱うことが可能であり，画面上では折り返して2行以上に渡って表示される。ただし，それらの行が連続した行であることを示すために折り返す行の最後に ！ が表示される。なお，以後「行」という言葉を，「テキスト上での改行と改行の間の文字の並びとしての行」と，「画面上で表示されている行」の2通りの意味で用いる。ただし，以後のほとんどの場合，前後関係からどちらの意味での行であるかは明らかであろうと思われる。とくに，両者を区別する必要のある場合には，前者を「テキスト上の行」，後者を「画面上の行」という呼び方をする。

画面上で数行に渡る長い行の最初の行にカーソルがあるとき，次の行に移動するコマンド（コントロールN）を入力した場合，GNU－EMACSでは，テキスト上での次の行にカーソルが移動する。したがって，長い行の画面上の途中の行にカーソルを移動するときには1文字あるいは1単語単位の移動などのコマンドを繰り返して用いる必要がある。しかし，使用した経験からは，このような移動のコマンドでは操作性はあまりよくない。したがって，マイクロEMACSでは，次行への移動コマンドは，画面上での1行の移動とした。

もっとも，マイクロEMACSでの使用では，たとえば，「テキスト上の次行へカーソルを移動して，ある操作をする」ようなマクロの定義では，画面上で2行以上に表示される長い行がある場合には多少こみいった定義をする必要がでてくる。しかし，使用経験からは，マクロ定義よりも画面上の1行の移動を重視したほうが操作性がよいと判断した。

(3) 画面上の行の先頭と最後へのカーソルの移動

EMACSには行の先頭と最後へのカーソルの移動のためのコマンド（コントロールAとコントロールE）がある。しかし，これらのコマンドはそれぞれテキスト上の行の先頭および最後にカーソルを移動させるコマンドであるので，画面上で2行以上に表示されている長い行の場合には，画面上では何行か上の行の先頭，あるいは何行か下の行の最後にカーソルが移動することになる。

しかし，実際に使用した経験からであるが，原稿などのテキストを作成しているときには，画面上で2行以上に渡って表示されている長い行の場合には，テキスト上の行の途中である画面上の行の先頭あるいは最後にカーソルを移動させることも多い。ただし，テキスト上の行の先頭および最後への移動のコマンドも必要であるので，マイクロEMACSでは，GNU－EMACSで通常は使用されていないキー（コントロール＾）を用いて，コントロール＾Aとコントロール＾Eで画面上の先頭および最後への移動のコマンドとした。

(4) 画面の最上行と最下行へのカーソルの移動

これも使用した経験からであるが，現在表示されている画面の最初あるいは最後の行にカーソルを移動することもあれば有用な機能であるので，コントロール＾＜と，コントロール＾＞のコマンドで備えた。

(5) 数指定による目的行へのカーソルの移動

プログラム開発中には，コンパイラにより出力されるエラー行にカーソルを移動して修正することを非常に頻繁に行う。しかし，頻繁に使用するには，GNU－EMACSに用意されている，先頭からの行数を指定してその行へカーソルを移動させる「ESC X goto-line 行数」のコマンドは，キー入力が多すぎる。そこで，マイクロEMACSでは，「コントロール＾G行数」で行えるようにした。

(6) 領域のファイルへの書き出し

GNU－EMACSには，マークとカーソルの間の「領域」を別のバッファ（マニュアルのバッファの説明参照）に書き出すコマンドはあるが，ファイルに書き出す機能はないようである。その領域を編修する場合にはそれでもよいが，使用経験からは，単にファイルに取り出すことが目的の場合も多い。もちろん，そのバッファを選択し内容をファイルに書き出せばよいが，多少手間がかかる。また，Unix ワークステーションのように大きなメモリを利用できる場合には多くのバッファを開くこともさして抵抗はないが，パソコンの場合には，ファイルに書き出すだけでバッファを開くことには多少の抵抗がある。そのために，ファイルへ直接に書き出すコマンド，コントロール＾コントロールWを

用意した。

(7) 定義時には実行しないようなマクロ定義方法

この方法は，有用というよりも，マイクロEMACSの処理方式によるものである。

いくつかのコマンドを組み合わせたものを定義しておき，簡単なキー操作(コントロールXE）で実行できるキーボードマクロの機能は，使い慣れると非常に便利である。GNU－EMACSでは，このようなマクロを実行しながら定義することができる。つまり実行を確認しながら定義できるという利点があるし，しかもGNU－EMACSには，既に実行したコマンドを逆順に取り消すアンドゥの機能が用意されているので，もし間違えた場合にも訂正は簡単である。しかし，現在のところマイクロEMACSでは，アンドゥの機能は実現されていないので，実行しながら定義することもできるが，実行しないで定義することもできるモードを設けている。

III　新マイクロEMACSの処理方式

(1) 扱えるテキストの大量化

旧バージョンのマイクロEMACSでは，扱えるテキストの大きさは，英字であれば約64000文字，日本語文字であれば約32000文字であった。16ビットパソコンに用いられていたCPU（インテル社8086）ではデータの位置を指す16ビットのレジスタ（以後インデックスレジスタと呼ぶ）だけでは1セグメント64キロバイト(64KB)のメモリ空間しかアクセスできず，セグメントレジスタでセグメントベースを切り替えながら計1メガバイト（1MB）のメモリ空間を利用できるというものである。したがって，64KB以上のテキストを扱うために，[1] で説明した種々のテキスト管理情報，すなわちテキスト中の位置を示す情報などを，インデックスレジスタに持たせる値と，セグメントレジスタに持たせる値の対として管理することにした。

基本的には，このような管理方法の変更でテキスト用のメモリ空間を拡大できる。ただし，旧バージョンでも同じであるが，これらの管理情報は通常はメ

モリに保持されており，使用されるときや値を更新するときにはレジスタに移される。このレジスタとメモリ間のデータ転送は旧バージョンが1命令であったのに対して，新バージョンでは2命令となる。また，新バージョンではこれらの管理情報のうちテキスト中の位置を指す値を更新する場合には，常にセグメントの境界を越えたか否かのチェックの必要があるので，そのためにも命令数が多くなっている。これら以外にもプログラム中では，管理情報同士の値の比較なども行われており，これらの変更に伴う部分ではプログラムのサイズは約2.5〜3倍程度になっている。

なお，8086系統のＣＰＵのレジスタの性質から，原則として管理情報はＤＳレジスタで指されるメモリ空間で保持し，テキストの領域はＥＳレジスタで管理している。

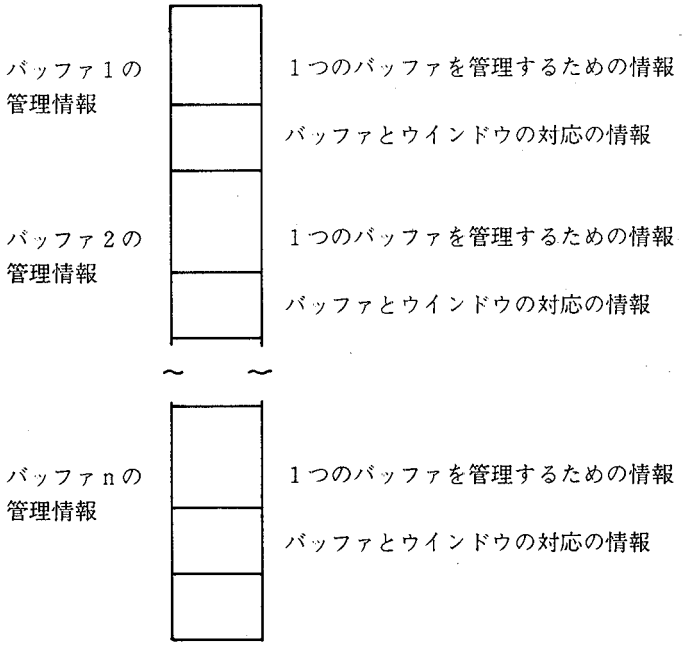

図1　バッファ管理のための情報管理

(2)　マルチバッファ，マルチウィンドウの機能追加

マルチバッファ，マルチウインドウのシステムを実現するには，基本的には1バッファ，1ウインドウであった旧バージョンで管理していた，テキストの最初と最後のメモリ上の位置などの情報の組（[1] 参照）を，複数個管理することが必要になる。ただし，それら以外にも，現在選択されているバッファや，画面に表示されているウインドウ，およびそれぞれのバッファとウインドウの対応などを示す情報も必要である。このような情報管理を実現する方法はいくつか考えられるが，マイクロEMACSでは次の図のような管理方法を採用した。

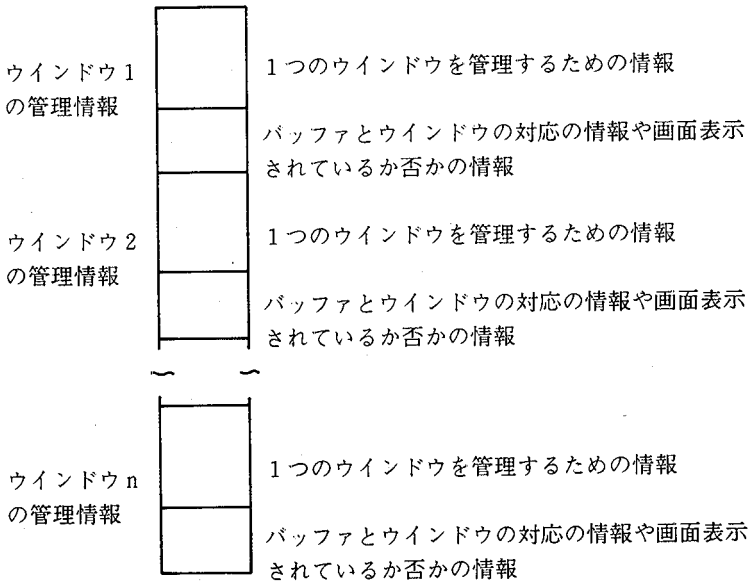

図2　ウインドウ管理のための情報管理

1つのバッファあるいはウインドウを管理するための情報は，すべて同じ構造のものである。バッファとウインドウの対応の情報なども同じ構造のものである。そして，現在選択されているバッファあるいはウインドウに対応する構

造の先頭を8086 CPUのBPとBXのインデックスレジスタにより示し，たとえば，その構造の先頭から4番目の値を参照するには，[BP＋4]あるいは[BX＋4]のようにアクセスする。このように同じ構造のもので情報管理をし，インデックスレジスタを用いることによって，バッファあるいはウインドウの切り替えは，BPやBXで指す管理情報の構造の始まりの位置を切り換えることにより容易に行える。

IV 今後の課題

パソコンやUnixワークステーション上のEMACSの開発・使用は，8ビットパソコンの時代のスクリーンエディタがまだ市販されていないころからであり，ほぼ10年近くなっている。最近ではパソコン上ではかなり高機能のスクリーンエディタが市販されているが，ワークステーション上では現在のところGNU－EMACSが主流であるようである。したがって，パソコンやUnixワークステーションを併用する著者にとっては，両方で共通して利用できる意味で，パソコンでのマイクロEMACSは利用価値のあるものである。ただし，実行されたコマンドを取り消すアンドウの機能は実現する必要を感じている。

なお，これも数年まえからであるがパソコンのグラフィック機能を利用して，英語と日本語とさらに第3の言語を扱える多カ国語が扱えるスクリーンエディタの開発も行っているが，これについても別の機会に報告したい。

参 考 文 献

[1] 本田道夫，中村邦彦「複数の計算機システムにおける共通コマンド体系の画面エディタの開発」，香川大学経済論叢 第60巻 第3号，昭和62年9月

[2] Richard Stallman 著，竹内郁雄・天海良治監訳「GNU Emacs マニュアル」共立出版 1988年