解　説

拡張可能な画面エディタ EMACS†

斉　藤　康　己††

0. まえがき

本稿では代表的な画面エディタである EMACS の解説を行う．前半で EMACS の大まかな構造と機能，画面の見え方などを EMACS を知らない人にもわかるように説明する．後半では内部に少し立ち入ってデータ構造や再表示のアルゴリズムなどを解説する．また，あまり知られていない便利な機能，うまい使い方，カスタマイズの方法などにも言及する．

なお，EMACS と一口に言っても Stallman が MIT の ITS 上に作ったもの[1]，それを DEC の TOPS-20 に移したもの，Multics 上の LISP で書いたもの[2]，LISP Machine 上の ZMACS[3]，さらには UNIX 上の C で書かれたもの[4] など色々の計算機上で開発されたものが存在する．ここではオリジナルの EMACS に最も近く，日本で一番良く使われていると思われる TOPS-20 のもとで動く EMACS[5] を解説することにする．

さらに EMACS は 50 種以上の端末をサポートしているが，ここでは DEC の VT-100（または VT-100 コンパチブルな）端末での使用を前提に説明を進めていく．

1. EMACS の動き

端末の前に座って EMACS を起動すると，数秒後に画面が消去され，画面の一番上にバージョンナンバと “type ^- (the help character) for help” というメッセージが現われる．

この時点で EMACS はユーザからの入力を待ちかまえている．“a” のキーを打つと，先程表示されていたヘルプメッセージは消え，画面左上すみに “a” という文字が現われ，カーサがこの “a” の右で点滅している．それと同時に画面の下から 3 行目あたりに，“EMACS (Fundamental) Main:” という行が現われる．この行はモード行と呼ばれ，Major, Minor のモードや，バッファ名，日付や今見ているファイル名などを常時表示しておくことができる．その下の数行はエコー領域と呼ばれ，ファイルを読み込む時のファイル名の入力や，探索する文字列の入力などに使われる．画面の残りの 20 行ぐらいがバッファの表示に使われる．バッファには文字の列をたくわえることができる．したがってユーザが入力した “a” という文字はバッファの先頭に挿入され，かつ画面上に表示されているわけである．引き続き “b”，“c” とキーを打っていくと “a” のうしろに “bc” の文字が表示されるとともにカーサは右へ移動していく．復帰改行を押すと次の行の先頭へカーサが移る（図-1 参照）．ここまでは紙のかわりにスクリーンになっただけでタイプライタと同じような振舞をすると言ってよい．あと削除キーでカーサの左の一文字を消去できることを知っていれば一応タイプライタの代わりとして EMACS を使うことができる．

2. EMACS の基本コマンド

以上の説明だけではすでに打ち込んだテキストの任意の個所への変更や探索，置換などの機能がはたせない．そこでコマンドの登場となる．

EMACS のコマンドはキーに割り当てられたものと名前で呼ぶものの 2 つに大別できる．より正確には，すべてのコマンドに名前がついていて，そのうち頻繁に使われるコマンドは効率よく起動するためにキーに割り当てられている．キーとしてはコントロールキー (CTRL というキーと他のアルファベットのキーを同時に押す．以下 C- で示す．) とメタキー (VT-100 上では ESCAPE のあとに他のキーを押す，したがって 2 打鍵となる．MIT で最初に使われた Knight キーボードや LISP マシンのキーボードでは META と刻印された CTRL と同じようなキーがあって，その META キーと他のキーを同時に打てばよい．以下 M- で示す．) それにコントロールメタ

† The Extensible Screen Editor EMACS by Yasuki SAITOH (Musashino Electrical Communication Laboratory).

†† 日本電信電話公社武蔵野電気通信研究所情報通信基礎研究部

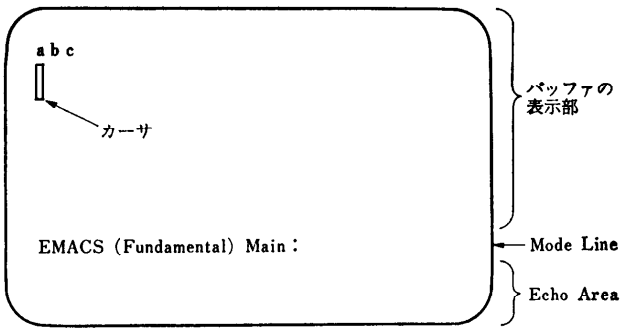

図-1 EMACS の画面

キー（VT 100 上ではコントロール-Z のあとで他のキーを打つ．以下 C-M- で示す．）を使用する．これで，百以上のコマンドを1打鍵または2打鍵のキー列に割り当てることができる．この割り当てはユーザが自由に変えられるようになっている．またユーザの定義したコマンドも同様に任意のキーに割り当てることができる．

これらのコマンドキーは編集の任意の時点で打つことができ，打たれるとすぐにそのキーに割り当てられているコマンドが実行され，その結果を反映するように画面が書きかえられる．

実は EMACS 全体の制御構造は，おおまかに言えばコマンド読み込み－コマンド実行－画面の再表示のループになっていて，まさに LISP の READ-EVAL-PRINT ループと同じような構造になっている．したがって通常の印刷可能文字にも“自分自身を挿入せよ”というコマンドが割り当ててある．

EMACS には豊富なコマンド群が用意されていて，コントロール－X などのコマンドキー拡張文字を使用したものまで含めると，キーに割り当てられたものだけで 200 にも達する（**付録 1** 参照）．

ライブラリと呼ばれるパッケージに含まれているコマンドまで数えると，さらにコマンドの数は多くなるが，ライブラリの解説は後にして，ここでは TEACH-EMACS と呼ばれる EMACS の初心者用の CAI プログラムに含まれているコマンド及び我々が通常よく使用するコマンドを中心に基本コマンドのみを機能別に解説することにする．

2.1 カーサの移動

挿入，削除などの編集操作はすべてカーサの周りで行われる．EMACS ではバッファ内の文字の位置を表わすのにポイントと呼ばれるポインタを使用する．ポイントは常に文字と文字の間を指している．このポイントの位置をユーザに知らせるために EMACS は常にポイントの右の文字の上にカーサを表示している．したがってカーサ移動コマンドで画面の上ではカーサが移動するが，実は内部ではポイントを動かしているわけである．

主なカーサ移動用のコマンドは付録1で Simple Cursor Motion と分類してある．

EMACS にはこの他にも英語の単語単位でカーサを動かしたり，文単位，パラグラフ単位で動かすもの，LISP の S 式単位で動かすものなど多くのカーサ移動のコマンドが用意されている．

2.2 削除，Kill と Yank

前に述べたように削除キーでポイントの左の一文字を削除できるが，その他に以下の削除コマンドがある．

C-D ポイントの右の1文字を削除する．

C-K ポイントから行末までの文字列を削除する．ポイントが復帰改行の直前にあるときはその復帰改行を削除する．

C-Y C-K で削除した文字列を復元する．

C-Y は Yank コマンドと呼ばれ，テキストの一部を他の場所へ移動したり，コピーを取って，それをある位置に挿入したりする際に C-K や C-W などのコマンドとペアで使われる．

2.3 ファイルの入出力

“C-X C-V”というコマンドを打つとエコー領域にプロンプトが出る．そこにファイル名を入力すると，そのファイルの中味が現在のバッファの中味になる．以前のバッファに変更が加えられていれば，それを前もって出力するかどうかをユーザに尋ねてくる．ファイル名の入力にはエスケープによるコンプリション* が使える．

“C-X C-S”はバッファの中味に変更があれば，C-X C-V で読み込んだのと同じファイル名でその中味を書き出す．変更がない場合には何もしない．

“C-X C-W”はバッファの中味をユーザが指定した名前のファイルに書き出す．

以上が主に使われる入出力コマンドであるが，他にファイルの中味をポイントの位置に挿入したり，バッファの一部をファイルに書き出したりするコマンドも

* ファイル名を途中まで打ち込んでおいて，ESCAPE キーを打つと，それまで打ち込んだ文字列で名前が始まるファイルが存在するかどうかをチェックし，もし唯一に決まればファイル名を最後まで自動的に完成してくれる機能．

用意されている．また，通常はオペレーティング・システムのコマンドを使って行うファイル名の変更，ファイルの削除，コピーなどの仕事も EMACS のコマンドとして実行できる．

付 録 1 EMACS のコマンドリスト

付録 1(No. 1)

```
This summary contains a list of
all commands, grouped by topic.
Not all M-X commands are included.

Prefix Characters
  Altmode (^R Prefix Meta)
  C-Z (^R Prefix Control-Meta)
  C-Q (^R Quoted Insert)
  C-U (^R Universal Argument)
  C-X
  M-X (^R Extended Command)
  C-M-X (^R Instant Extended Command)
  C-digits, M-digits, C-M-digits
  C-Minus, M-Minus, C-M-Minus
Simple Cursor Motion
  C-A (^R Beginning of Line)
  C-B (^R Backward Character)
  C-E (^R End of Line)
  C-F (^R Forward Character)
  C-H (^R Backward Character)
  C-N (^R Down Real Line)
  C-P (^R Up Real Line)
  C-R (^R Reverse Search)
  C-S (^R Incremental Search)
  M-< (^R Goto Beginning)
  M-> (^R Goto End)
  C-X C-N (^R Set Goal Column)
Lines
  Return (^R CRLF)
  C-O (^R Open Line)
  M-= (^R Count Lines Region)
  C-X C-O (^R Delete Blank Lines)
  C-X C-T (^R Transpose Lines)
Killing and Un-killing
  Rubout (^R Backward Delete Character)
  C-D (^R Delete Character)
  C-K (^R Kill Line)
  C-W (^R Kill Region)
  C-Y (^R Un-kill)
  M-W (^R Copy Region)
  M-Y (^R Un-kill Pop)
  C-M-W (^R Append Next Kill)
  C-X G (^R Get Q-reg)
  C-X T (^R Transpose Regions)
  C-X X (^R Put Q-reg)
  M-X Overwrite Mode
Scrolling and Display Control
  C-L (^R New Window)
  C-V (^R Next Screen)
  M-R (^R Move to Screen Edge)
  M-V (^R Previous Screen)
  C-M-R (^R Reposition Window)
  C-M-V (^R Scroll Other Window)
  M-X View Buffer
  M-X View File
The Mark and the Region
  C-Space (^R Set/Pop Mark)
  C-< (^R Mark Beginning)
  C-> (^R Mark End)
  M-@ (^R Mark Word)
  M-H (^R Mark Paragraph)
  C-M-@ (^R Mark Sexp)
  C-M-H (^R Mark Defun)
  C-X H (^R Mark Whole Buffer)
  C-X C-P (^R Mark Page)
  C-X C-X (^R Exchange Point and Mark)
Whitespace and Indentation
  Tab (^R Indent According to Mode)
  Linefeed (^R Indent New Line)
  M-Tab (^R Tab to Tab Stop)
  M-M (^R Back to Indentation)
  M-\ (^R Delete Horizontal Space)
  M-^ (^R Delete Indentation)
  C-M-O (^R Split Line)
  C-M-\ (^R Indent Region)
  C-X Tab (^R Indent Rigidly)
  M-X Edit Indented Text
  M-X Edit Tab Stops
  M-X Edit Tabular Text
  M-X Indent Tabs Mode
  M-X Tabify
  M-X Untabify
Words, Sentences and Paragraphs
  M-A (^R Backward Sentence)
  M-B (^R Backward Word)
  M-D (^R Kill Word)
  M-E (^R Forward Sentence)
  M-F (^R Forward Word)
  M-H (^R Mark Paragraph)
  M-K (^R Kill Sentence)
  M-T (^R Transpose Words)
  M-[ (^R Backward Paragraph)
  M-] (^R Forward Paragraph)
  M-Rubout (^R Backward Kill Word)
  C-X Rubout (^R Backward Kill Sentence)
  M-X Atom Word Mode
  M-X Edit Syntax Table
Filling Text
  M-G (^R Fill Region)
  M-Q (^R Fill Paragraph)
  M-S (^R Center Line)
  C-X . (^R Set Fill Prefix)
  C-X F (^R Set Fill Column)
  M-X Auto Fill Mode
Exiting, Quitting
  C-G
  C-] (Abort Recursive Edit)
  C-M-Z (^R Exit)
  C-X C-Z (^R Return to Superior)
  M-X Compile
  M-X Rerun CCL
  M-X Top Level
  M-X Undo
Pages
  C-X L (^R Count Lines Page)
  C-X P (^R Narrow Bounds to Page)
  C-X [ (^R Previous Page)
  C-X ] (^R Next Page)
  C-X C-P (^R Mark Page)
  M-X View Page Directory (in PAGE)
  M-X What Page
Lisp
  M-( (^R Make ())
  M-) (^R Move Over ))
```

付録 1(No. 2)

```
  C-M-( (^R Backward Up List)
  C-M-) (^R Up List)
  C-M-@ (^R Mark Sexp)
  C-M-A (^R Beginning of Defun)
  C-M-B (^R Backward Sexp)
  C-M-D (^R Down List)
  C-M-E (^R End of Defun)
  C-M-F (^R Forward Sexp)
  C-M-G (^R Format Code)
  C-M-H (^R Mark Defun)
  C-M-K (^R Kill Sexp)
  C-M-N (^R Next List)
  C-M-P (^R Previous List)
  C-M-Q (^R Indent Sexp)
  C-M-T (^R Transpose Sexps)
  C-M-U (^R Backward Up List)
Files
  M-. (^R Find Tag)
  M-~ (^R Buffer Not Modified)
  C-X C-F (Find File)
  C-X C-Q (^R Set File Read Only)
  C-X C-S (^R Save File)
  C-X C-V (^R Visit File)
  C-X C-W (Write File)
  M-X Append to File
  M-X Auto Save Mode
  M-X Copy File
  M-X Delete File
  M-X Insert File
  M-X Prepend to File
  M-X Rename File
  M-X Revert File
  M-X Save All Files
  M-X Set Visited Filename
  M-X Write Region
File Directories
  C-X D (^R DIRED)
  C-X C-D (^R Directory Display)
  M-X Clean Directory
  M-X List Files
  M-X Reap File
  M-X View Directory
Buffers
  C-X C-B (List Buffers)
  C-X A (^R Append to Buffer)
  C-X B (Select Buffer)
  C-X K (Kill Buffer)
  M-X Insert Buffer
  M-X Kill Some Buffers
  M-X Make Space
  M-X Rename Buffer
  M-X What Available Space
Comments
  M-LF (^R Indent New Comment Line)
  M-; (^R Indent for Comment)
  M-N (^R Down Comment Line)
  M-P (^R Up Comment Line)
  C-M-; (^R Kill Comment)
  C-X ; (^R Set Comment Column)
Case Conversion
  M-C (^R Uppercase Initial)
  M-L (^R Lowercase Word)
  M-U (^R Uppercase Word)
  C-X C-L (^R Lowercase Region)
  C-X C-U (^R Uppercase Region)
Minor Corrections
  M-$ (^R Correct Word Spelling)
  M-' (^R Upcase Digit)
  M-_ (^R Underline Word)
  C-X _ (^R Underline Region)
Windows
  C-M-V (^R Scroll Other Window)
  C-X 1 (^R One Window)
  C-X 2 (^R Two Windows)
  C-X 3 (^R View Two Windows)
  C-X 4 (^R Visit in Other Window)
  C-X O (^R Other Window)
  C-X ^ (^R Grow Window)
  M-X Compare Windows
Narrowing
  C-X N (^R Narrow Bounds to Region)
  C-X P (^R Narrow Bounds to Page)
  C-X W (^R Widen Bounds)
Status Information
  C-X = (What Cursor Position)
  C-X L (^R Count Lines Page)
  M-X List Loaded Libraries
  M-X List Variables
  M-X List Redefinitions
  M-X What Page
Keyboard Macros
  C-X ( (^R Start Kbd Macro)
  C-X ) (^R End Kbd Macro)
  C-X E (^R Call Last Kbd Macro)
  C-X Q (^R Kbd Macro Query)
  M-X Name Kbd Macro
  M-X View Kbd Macro
Libraries
  M-X Kill Libraries
  M-X List Library
  M-X Load Library
  M-X Run Library
Variables
  M-X Edit Options
  M-X Kill Local Q-register
  M-X Kill Local Variable
  M-X Kill Variable
  M-X Make Local Q-register
  M-X Make Local Variable
  M-X Set Key
  M-X Set Variable
  M-X View Variable
Mail
  C-X M (Send Mail)
  C-X R (Read Mail)
  M-X Check Mail
Minibuffer
  C-% (^R Replace String)
  M-Altmode (^R Execute Minibuffer)
  M-% (^R Query Replace)
  C-X Altmode (^R Re-execute Minibuffer)
```

2.4 アーギュメント

EMACS のほとんどのコマンドには数のアーギュメントを与えることができる．通常は次のコマンドが与えられたアーギュメントの回数分繰り返される．

アーギュメントを与えるコマンドは C-U で，C-U のあとに数字を入力する．数字なしで C-U のみのときは 4 を，C-U C-U と 2 回打てば 4×4=16 を与えたことになる．

2.5 マークとリージョン

編集作業をしているとテキストの一部に対してある処理を行いたいという状況がしばしば生じる．それにはまずテキストの一部を何らかの方法で指定しなければならない．EMACS ではマークされた位置と現在のポイントのある位置で囲まれた領域をリージョンと呼び，これに各種の操作をほどこすことができる．

マークを付けるにはカーサを望みの位置まで持っていって C-Space を打てばよい．マークされた位置は長さ16のスタックに保持されているので，ポップしていくと次々に以前マークした位置へカーサを戻すことができる．リージョンに関連したコマンドは付録1 The Mark and the Region を参照のこと．

2.6 インクリメンタル・サーチ

探索は実はカーサ移動のためのコマンドの一種である．EMACS にはごく普通の与えられた文字列の探索もあるが，あまり使われず，もっぱらこのコマンドが使われている．なぜなら，以下に述べるように，このコマンドを使うと探索文字列の変更が容易でかつ会話的にいろいろなことができるからである．

C-S でインクリメンタル・サーチは起動される．エコー領域に “I-Search:” というプロンプトが出てユーザからの入力待ちになる．ユーザが探したい文字列，たとえば “abc” を入力していくと，カーサはバッファ内で最初にみつけた “abc” の右へ移動する．ユーザの入力がゆっくりだと “a” を打った所で，カーサはポイント以降で最初に現われる “a” の右へ移り，“b” を入力した所で最初の “ab” の右へというふうに順々に動いていく．入力をまちがえて望んでいない文字を打ってしまったときは，削除キーを押せば，最後に打ち込んだ文字はエコー領域から消え，バッファの方でも以前に探索ずみの位置へカーサを戻してくれる．また2番目，3番目の同一文字列の出現位置を探したいときは C-S を2度，3度と打てばよい．

さらに C-R は逆向きのインクリメンタル・サーチを起動するためのキーであるが，すでにインクリメンタル・サーチに入った状態で C-R を押すと現在の探索文字列を逆向きに探索してくれる．

2.7 ヘルプ及びセルフ・ドキュメント

Stallman が EMACS を紹介した論文[1]の表題には “Self-Documenting” という形容詞が入っていた．充実したヘルプ機能と，拡張可能であるにもかかわらず，拡張された部分も含めてすべての機能に関する説明を EMACS の中から直接読めるというのは，EMACS の大きな特徴である．

ヘルプを要求するには色々な方法があるが，一番一般的なのはヘルプ文字 (C-_, コントロールアンダバー) を押すことである．EMACS が入力待ちの状態にいるときに C-_ を押すとエコー領域に “Doc (? for help):” というプロンプトが現われる．ここで疑問符を入力すると**図-2**のようなヘルプメッセージが画面に表示される．図-2 からわかるようにヘルプには色々なオプションがある．個々のオプションの説明は省略するが，最も良く使われるオプションは A (Apropos) で，ユーザの与えた文字列をコマンド名に含むようなコマンドをすべて列挙してくれる．これをうまく利用すると望みのコマンドにたどりつくことができる．

また List で始まるコマンドで，各種のコマンド，ライブラリ，変数などを網羅的に列挙して表示してくれるものもある．

以上説明したヘルプの機能を十分に活用すると，コ

```
You are at top level.
Type a Help option to say which kind of help you want:
C   says what a certain Command (character) does.  You type the character.
D   Describes a function.  You type the name.
A   lists all functions Apropos a keyword.  You type the keyword.
L   tells you the Last 60 characters you typed.
R   describes current Recursive editing level.
N   prints a file of EMACS news.
I   runs the INFO program for detailed documentation.
Q or Rubout  Quits -- you don't really want help.
More advanced options:
  T - run TECDOC; V - run List Variables; W - run Where Is;
  M - print the contents of the mode line (for printing ttys).
  SPACE repeats previous A, D, T, V or W request.
For a basic introduction to EMACS, run the program TEACH-EMACS.
```

図-2 C-_? で表示されるヘルプメッセージ

マンドの使い方，動作に関する疑問はほとんどマニュアルを参照することなく解決することができる．

2.8 その他の基本コマンド

名前で呼ぶコマンドがすでにいくつか出てきたが，それを呼ぶには M-X というコマンドを使用する．M-X を打つとエコー領域に“M-X”というプロンプトが出てくるので，ここへ起動したいコマンドの名前を打ち込めばよい．この入力にはファイル名のコンプリションと同じ入力法が使えるのでコマンド名を完全に覚えておく必要はない．

他のユーザからメッセージがきたり，何らかの理由で画面が乱れた時に，画面を消去し同じ内容を再表示するためのコマンドが C-L である．

ファイルをバッファに読み込んで，ながめている時には一画面分ずつ上下にスクロールするコマンドが必要になる．これを行うのが C-V と M-V で，C-V は次の一画面分を表示し，M-V は前の一画面分へ戻るコマンドである．

EMACS を終了し，EMACS から飛び出るのには C-M-Z を使う．またサブプロセスとして EMACS が動いている時 (LISP の LEDIT など) に，親プロセスに戻るには，C-X C-Z を利用する．

また一般に何か変なことがおこって様子がおかしくなった時，進行中の動作を途中でやめたい時などは C-G を打つ．これは LISP (Maclisp) と同じである．

3. EMACS の構成—中核 TECO と豊富なライブラリ群

以上説明してきたコマンドを知っていれば，一応 EMACS を各種テキストやプログラムの編集に使うことができる．しかし EMACS の拡張可能という大きな特徴の真価はもっと複雑な作業をしようと思った時に現われる．その点はこの節の最後に解説する．

EMACS が拡張可能であるのは EMACS の母体である，TECO がエディタであると同時にプログラミング可能な言語でもあることによる．この点を明らかにするためにまず EMACS の成り立ちから説明する．

3.1 EMACS の成り立ち

図-3 に EMACS 全体の成り立ちを図示した．EMACS は TECO を核にして，その上に何段かの拡張をほどこした形になっている．これは EMACS の発展の経緯をそのまま反映している．

まず初めに TECO と言う文字エディタがあった．

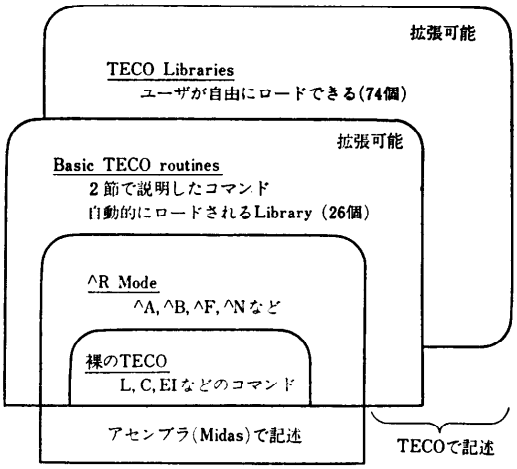

図-3 EMACS のなりたち

Stallman が TECO への拡張としてバッファの中味を常に画面に表示し，かつコマンドを簡単なキー入力で起動できるモードを開発した．これが ^R モードである．最初のうちは編集の一部だけを ^R モードに入って行い，通常の編集は ^R モードから抜け出て行うという利用にとどまっていたらしい．そのうちにユーザが ^R モードのままで使える色々なコマンドを自分で開発し出して序々に現在のような使われ方に近づいてきた．それでもまだその頃の TECO には長い名前の関数や変数を扱う機能がなかったのでユーザの書いたプログラムは現在の TECO プログラム以上に判じ物のようだったらしい．そこで TECO を長い名前が使えるように改良し，かつそのように改良した TECO で今まで多くのユーザが開発していたコマンドを整理しつつ書きなおしていくことによって EMACS が誕生した．

ひとたび EMACS が動くようになると，その魅力に魅せられた人たちが，役に立つコマンドをどんどん書くようになった．それをまた取り込むという形で EMACS は発展し，ユーザの書いたルーチンでも大変便利でほとんどのユーザが使うようなもの (word abbrev や DIRED など) はシステムに組み込まれ，その他のものもライブラリという形で，EMACS の重要な構成要素となったのである．

3.2 ライブラリ

EMACS で利用できるコマンドは，ごく一部 (C-F, C-K, self-insert などアセンブラで書いてあり，組み込みコマンドと呼ばれるもの) を除いては，すべて TECO でプログラムしてあり，ライブラリという形

付 録 2 EMACS のライブラリリスト

付録 2(No. 1)

CATALOG OF LIBRARIES

Libraries Used Explicitly

```
ABSTR  contains commands for making documentation files.
BABYL  is a subsystem for reading, sending and editing mail.
BBNLIB contains a few commands that people at BBN like.
BSHACK has functions for operating on lines containing overprinting.
BUGHUNT contains commands for putting your name into each comment you edit.
CACHE  implements a cache for speeding up EMACS subroutine calls.
CHESS  implements commands for editing pictures of chess boards.
COBOL  implements COBOL mode.
COLUMNS implements commands for converting single-column into double-column.
COMPLT provides completion for buffer names and variable names.
DELIM  implements commands for moving over balanced groupings of parentheses.
DM     redefines commands to be convenient on Datamedia 2500 terminals.
DM3025 redefines commands to be convenient on Datamedia 3025 terminals.
DOCLSP prints documentation from the MacLisp manual on a specified function.
DOCOND is a macro processor and conditionalizer for text files.
DOCTOR contains DOCTOR mode, a psychiatrist.
DRAW   offers functions for editing pictures made of characters.
EAKMACS EAK's personal library, useful as an example.
EFORK  implements commands for running programs in separate inferior forks.
FDB    has functions for examine file descriptor blocks.
FIXLIB has functions for examining and patching EMACS functions.
FTPLIB is used for EMACS maintenance on Arpanet sites.
HAZ1510 redefines commands to be convenient on Hazeltine 1510 terminals.
IBM370 implements IBM370 mode, for editing 370 assembler code.
INFO   peruses tree-structured documentation files.
INTER  is the EMACS side of the EMACS-to-Interlisp interface.
INTERLISP-MODE is an alternate Interlisp Mode, different from the default one.
IVORY  is EAK and ECC's alternate generator for EMACS.
JOURNAL implements journal files.
JUSTIFY implements an auto-justfy mode similar to auto fill mode.
LABELS is for arranging a file of addresses for printing mailing labels.
LEDIT  is the EMACS side of the EMACS-to-MacLisp interface.
LONG-FILENAMES provides help in handling files which have long names.
LSPUTL contains useful functions for searching and manipulating Lisp code.
LUNAR  is Moon's personal library, which contains some useful commands.
MACCNV does part of the work of converting MACRO-10 code to MIDAS code.
MAICHK checks for arrival of mail freguently and automatically.
MAZLIB is a game for solving mazes.  It's fun to play.
MKDUMP aids in dumping your own customized environment.
MODE2  implements a second additional mode line with additional information.
MODLIN implements a fancier mode line display.
MOVE   provides commands specially for copying and moving many pieces of text.
MQREPL works with TAGS to do Query Replaces on files in a tag table.
NCOLUMNS has functions for turning single-column text into many columns.
NOVICE implements restricted subsets of EMACS, for beginners.
NVT100 defines the arrow keys and numeric keypad of VT-100 to perform editing.
NVT132 is like NVT100 but for the VT-132.
NVT52  defines the arrow keys and numeric keypad of VT-52 to perform editing.
OUTLINE implements Outline Mode, for editing outlines.
OUTLINE-MODE implements a different flavor of Outline Mode.
PAGE   defines commands for viewing only one page of the file at a time.
PERSONAL has functions for keeping notes on your current projects.
PHRASE has commands for moving over and killing phrases of text.
PICTURE contains Edit Picture, the command for editing text pictures.
PRINT  contains "Print File" which formats a file and sends it to a printer.
PURIFY generates libraries from EMACS source files.
RENUM  renumbers figures, equations, theorems or chapters.
```

付録 2(No. 2)

```
SCRLIN contains alternatives for C-N and C-P which move by screen lines.
SEND-MAIL sends mail to another user.
SLOWLY redefines commands and options to suit slow terminals.
SORT   implements the sorting commands.
SPLIT  contains the commands for breaking up large files into small subfiles.
SYSTEM implements various commands useful for communicating with the O/S.
TAGGEN updates tag tables, the same function as the TAGS program.
TALK   initiates and accepts links with other users.
TDEBUG is a debugger for TECO programs.
TEACH-C100 has commands to define the programmable function keys of the C-100.
TIME   causes the current time of day to be displayed in the mode line.
TMACS  contains miscellaneous useful functions
TVLIB  customizes EMACS to resemble TVEDIT.
VT100  defines the arrow keys and numeric keypad of the VT-100 appropriately.
VT52   defines the numeric keypad of the VT-52 to supply numeric arguments.
XLISP  contains functions for global stylistic transformations of Lisp code.


Libraries automatically Loaded when needed
------------------------------------------
AUX    implements several commands which are part of the standard EMACS.
BABYLM contains the part of Babyl that implements mail sending.
BARE   contains the definitions of all built-in functions for documentation.
BASIC20 implements BASIC20 mode.
BCPL   implements BCPL mode.
BLISS  implements BLISS mode.
CLU    implements CLU mode.
DIRED  implements the commands for editing and listing directories.
EINIT  is used in building and dumping EMACS.
EMACS  is the main body of standard EMACS.  Always loaded.
FORTRAN implements FORTRAN mode.
GRIND  implements C-M-G.
HERMES interfaces between EMACS and a superior HERMES fork.
KBDMAC implements keyboard macros.
MMAIL  interfaces between EMACS and a superior MM fork.
MUDDLE implements Muddle mode.
PASCAL implements PASCAL mode.
PCL    implements PCL mode, for editing command files.
PL1    implements PL1 mode.
SAIL   implements SAIL mode.
SCRIBE implements SCRIBE mode.
TAGS   implements the TAGS package.
TEX    implements TEX mode.
TRMTYP implements the Set Terminal Type command.
TWENEX holds commands for the Twenex version of EMACS only.  Always loaded.
WORDAB implements Word Abbrev mode.
```

式でファイルにソースコードが収めてある.

各ライブラリの中には, TECO の関数の定義が複数個並んでいて, 各々が "関数名, ドキュメント, ボディ" という形式を取っている. このドキュメントの部分がヘルプに使われるわけである. 一応 TECO にもコンパイラがあってコンパイル (不要な空白を取り除いたり, インデックスを作ったりする) をするとEMACS に直接ロードできるファイルができる.

現在100近いライブラリが存在し, そのうち26個は EMACS を起動すると自動的にロードされる (付録 2参照). 26個のうち約半数は色々な言語のソースプログラムを編集するためのコマンド類を集めたもので, 下で説明する Major モードをセットアップするのに使われる. 残りのライブラリの中で, インクリメンタルサーチとか, two window モード, Query Replace などが定義されている. (これらの説明は 4 章で行う.)

3.3 Major モード

EMACS の編集の対象は文書に限らない. たとえば, LISP のようなプログラミング言語で書かれたプ

ログラムの編集にもよく使われる．LISP コードを編集している時には LISP 用の各種コマンドが使えるとありがたい．また別のバッファに移って文書を編集するということも必要になる．こんな時にバッファを変えるたびに編集対象向けのコマンドの再割り当てをしていたのでは面倒である．

そこで EMACS ではバッファごとにモードが設定できるようになっている．こうして設定したモードは別のバッファへ移って戻ってきても保存されているので，バッファを渡り歩いて色々な対象を編集しても一度モードを設定しておけば，どのバッファでも編集対象に合ったコマンドセットが使えることになる．

こうしたモードを EMACS では Major モードと呼び，通常 Tab, Rubout, Line feed キーなどの意味が編集対象の言語に合わせて再定義される．

どんな言語用のモードがあるかは付録2を参照していただきたい．

3.4 Minor モード

EMACS には Major モードとは独立に，Minor モードというものが存在する．こちらは Major モードのようにまとめていくつもの変更を加えるのではなく，ある1つの機能を使えるようにするかしないかのスイッチのようなものである．

Minor モードには Auto Fill, Auto Save, Atom Word, Overwrite, Word Abbrev, Indent Tabs などがある．各モードの解説はここでは省略する．

上で述べた各種の言語用のルーチンのパッケージは M-X で Major モードを設定するだけで利用可能になる．M-X List Redefinitions とやれば，そのモードではどのようなコマンドが使えるかを見ることができる．他のライブラリも M-X Load Library というコマンドにライブラリ名を与えることによってロードできる．ロードしたら，M-X List Library で，ライブラリ内のコマンドにどのようなものがあるかを知ることができる．

何かちょっと複雑そうなことを EMACS でやりたいと思った時には，ユーザはこうやってライブラリの中を探しまわる．なぜなら，かなりの確率で，ユーザのはたしたい機能を実現するコマンドがみつかるからである！

ぴったりのコマンドがみつからない場合にも，簡単なカスタマイズや改良で望みの効果が得られることが多い．筆者は TAGS のライブラリ (4.8 節参照) を利用して，ロングマンの英英辞書を EMACS から検索するユーティリティを作ったし，Journal の機能を利用して各種統計情報を収集している人もいる[8]．とにかく，ヘルプやセルフ・ドキュメントの機能を十分に活用して探しまわると，EMACS ではできないことはないのではないかと思うぐらい豊富なコマンドがそろっている．

それでもやはり望みのコマンドがみつからない時には自分で拡張コマンドを書くことになる．Multics EMACS のように LISP で書けばよいのなら話は別だが，やはり TECO でのプログラミングは難しい．しかし各種ライブラリのソースを見てまねればある程度動くものができ上がる．

4. EMACS の応用コマンド

この章では2章で説明できなかったいくつかのコマンドと，ユーザがライブラリをロードして使うユーティリティのいくつかを解説する．ライブラリに関しては，とてもその全体は説明できないので，どのようなものがあるかは付録2を参照されたい．

4.1 Query Replace

M-X Query Replace または M-% で起動される．後者で起動したときには画面左上にミニバッファ(4.6 節参照) が現われ，そこへ引数を入力していくので復帰改行やエスケープを含んだ文字列でも置き換えの対象にできる．

これが単なる置き換えと違うのは全体を一度に置き換えてしまうのではなく，置き換えたい文字列を見つけるたびにその部分を表示してユーザの判断をあおぐ点である．ユーザは置き換えを実行したければスペースを，置き換えせずに次へ行きたければ削除キーを，置き換えを実行しそれを表示して欲しければコンマを打てばよい．最後までユーザにきかずに置き換えてしまいたいときには“！”，まちがえたので1つ前に戻りたいときには“^”を打てばよい．

4.2 複数バッファと Two Window モード

前にも述べたように EMACS ではいくつでもバッファを持つことができる．C-X B とやってバッファ名を打ち込めば，その名前のバッファが存在すれば，そのバッファを選び，なければ新しく作ることになる．

最近のワークステーションなどではマルチウインドウを売物にしているものも多いが，EMACS では画面を水平に2つに分けた Two Window モードというのがある．上と下のウインドウで，2つのバッファ

の中味を同時に見たり，あるいは1つのバッファの違う部分を見たりできる．

4.3 DIRED

Directory Editor の略で，あるファイルディレクトリの中を掃除したり，どんなファイルがあるかを眺めてみたりする時に大変便利なサブシステムである．

4.4 キーボード・マクロ

編集作業をしているとあるコマンドの列を何回も繰り返したいということがよくある．それを非常に簡単に実現するのが，このキーボード・マクロである．まず "C-X (" で，キーボード・マクロの定義モードに入る．あとは通常の編集とまったく同じようにキーを打ち込んでいけばよい．定義を終了したいと思ったら "C-X)" で定義終了になり，それまでに打ち込んだコマンド列がマクロとして登録される．このマクロを起動するのには C-X E と打てばよい．繰り返し数を指定するアーギュメントを与えることもできる．

4.5 Word Abbrev モード

これは Minor モードの一種で，省略形で入力した文字列を EMACS が自動的にユーザが前もって与えた文字列に変換してくれるというものである．これは省略形ばかりでなくミススペルをなおしたり (thier を their に)，小文字を自動的に大文字にしたり (emacs を EMACS に) というふうにも使うことができる．

4.6 ミニバッファ

ESCAPE を2度打つと画面の左上に3行ほどの空行が現われる．これがミニバッファである．このバッファには TECO のコマンドを直接打ち込むことができる．コマンドを打ち込んでおいて ESCAPE を2度たたけば，そのコマンド列が，現在のバッファに対して実行される．実際はミニバッファは TECO の ^R モードになっているので若干の編集も可能である．

すでに説明した M-% による Query Replace はミニバッファを利用していたし，あとで述べる FS フラッグなどに値をセットしたり，その値を見たりするのにも使われる．また TECO のソートコマンド (^P) を利用したり，簡単な編集はその場で TECO のプログラムを書いて実行することもある．前述のキーボード・マクロなどに比べて TECO で書くと圧倒的に速い．またちょっと変わった使い方として電卓代わりに四則演算をするのに使うこともある．

4.7 LEDIT

LISP (Maclisp) と EMACS の間のインタフェースの EMACS 側の機能を実現しているのがこのライブラリである．LISP の中で (LEDIT) という関数を呼ぶと LISP の子プロセスとして EMACS が起動され，このライブラリが自動的にロードされる．

EMACS のバッファは LISP モードになっていて，括弧の対応表示や改行による自動字下げなどの機能を使いながら LISP プログラムの編集ができる．そのあと簡単なコマンドで編集した結果を持って LISP へ戻ることができる．

この LEDIT が実現している機能は大変重要なもので，LISP によるプログラミングを真に会話的でインクリメンタルなものにしてくれる．

4.8 TAGS

大きなシステムを書くと，どうしてもたくさんの関数やルーチンの定義が複数のファイルに分かれて存在することになる．"あの関数はどのファイルの中で定義したのか" と探しまわると案外時間を取られてしまう．

そんな時に TAGS の機能を使うと，ファイルのことはまったく忘れて，関数名からその定義に直接アクセスすることができる．

TAGS は TECO や LISP の他，BASIC, BLISS, FORTRAN, PASCAL アセンブラなど20種以上の言語のソースファイルに適用することができる．

4.9 その他のコマンド

他にもまだまだ役立つコマンドがたくさんあるが，とてもすべては書けないので，筆者が実際に使ってみて役立ったことのあるコマンドの名前だけを以下に列挙しておく (付録1参照)．

M-X Occur $, M-X How many $
M-X Keep Lines $ M-X Flush Lines $
M-X Correct Spelling $
M-X Push $
M-X Tabify $ M-X Untabify $
M-X Strip SOS Line Numbers $
M-X What available Spaces $
M-X ^R Buffer Graph $ in TMACS ライブラリ
M-X Uncontrolify $ in TMACS ライブラリ

また COMPLT というライブラリをロードしておくとバッファ名や変数名の入力にコンプリションがきくようになる．他に便利なライブラリとして PICTURE, DRAW, SORT, JOURNAL, SPLIT などがある (付録2参照)．

5. カスタマイズ

TECO で拡張のためのコマンドを書かずとも，EMACS では簡単なカスタマイズでユーザの思い通りになることが多い．以下にどのような方法があるかを解説する．

5.1 オプションの編集

EMACS の振舞は各種変数の値でコントロールされている．したがって，これらの変数の値を変えればEMACS の動き方を変えてカスタマイズができる．

変数の値を変える一番簡便な方法がオプションの編集である．M-X Edit Options でオプションとして登録されている変数の名前とその値が画面に表示される．さらに EMACS は再帰的に編集モードに入っているので，名前とコメントを見ながらそこに表示されている値を通常の編集コマンドで変更し，C-Z C-Z で飛び出れば自動的に値の変更が完了する．

5.2 シンタックス・テーブル

EMACS の中で単語の切れ目を判断したり，LISP のアトムの判定，または括弧の対応を取ったりするときに使われるのが，シンタックス・テーブルである．このテーブルには各文字ごとに単語の構成要素になるかならないかの情報と，LISP のシンタックスとしてその文字がどういう意味を持つか，及び LISP の閉じ括弧としての機能を持っている文字に関してはそれに対応する開き括弧が示されている．

このテーブルを編集するには，Edit Syntax Table というコマンドを起動すればよい．オプションの編集と同じように変更を加えることができる．

5.3 FS フラッグ

EMACS のレベルではなく TECO のレベルの変数が FS フラッグである．大部分のものは TECO のプログラムの動きや，画面の再表示などを制御するために使われているので，一般のユーザにはあまり関係はないが，中には知っていると便利なものもある．

EMACS から FS フラッグの値を見たり，セットしたりするにはミニバッファを使う．

すべての FS フラッグのリストを得るには M-X List TECO FS Flags $ を実行すればよい．名前のわかっているフラッグの説明は，ヘルプの“T”オプションで Tecdoc に入りその名前を打ち込むと得られる．

5.4 INIT ファイルと EMACS. VARS ファイル

EMACS は起動されると必ず EMACS. INIT というファイルを現在コネクトしているディレクトリから探してきて実行する．もしそこになければ EMACS のディレクトリから標準の INIT ファイルを読み込んでくる．INIT ファイルは TECO のコマンドが並んだもの（あるいはライブラリと同じフォーマットで書いてコンパイルしたもの）なので，何でも書くことができる．

しかし TECO で INIT ファイルを書くのはご免だという人のためにもう1つ INIT ファイルと同じような効果を実現できるものとして EMACS. VARS ファイルがある．こちらのファイルは，TECO ではなく単純なフォーマットで変数への値のセットや，コマンドの文字への再割りつけができるようになっている．

6. インプリメンテーション

EMACS はいわゆるギャップエディタ[7]である．Multics EMACS[2]や LISP マシンの ZMACS[3]などは行を文字列で表現し，行を相方向リンクでつないだものとしてバッファを表現しているのでこれとは違っている．

ギャップエディタでは，バッファを単純な文字の列で表現し，挿入や削除を容易にするためにギャップと呼ばれる大きな空き領域をポイントのある位置に設ける．その位置で挿入や削除が行われる分には他の部分にはまったく触れる必要がないので大変効率的である．ただし大きなファイルでポイントを先頭からファイルの最後へ移動したりすると，ギャップをはるばるバッファの最後へ移すためにほぼバッファ全体をメモリ空間の別の位置へ移動することになり時間がかかる．

一長一短であるが，通常の編集ではかなり局所性があるので十分に実用に耐える効率が得られる．

EMACS（より正確には TECO）をプログラミング言語として見た時には LISP に非常に似ているということが言える．データとしてはリストではなく文字を扱うが，プログラムとデータは同一（文字の列）であるし，再帰的な関数が書けるし，文字列やバッファ等のデータはガーベッジ・コレクション付きでメモリ空間に割りあてられている．また制御構造が LISP の READ-EVAL-PRINT ループと対応していることもすでに述べた．同じ文化の中から生まれた言語なので，これは当然のことかもしれない．

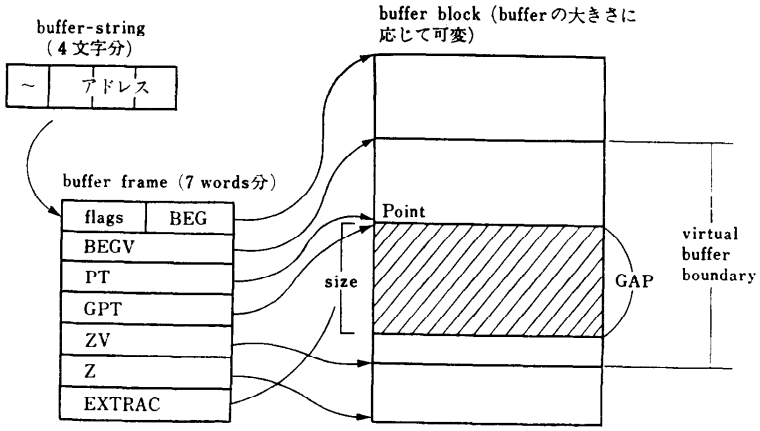

図-4 バッファのデータ構造

6.1 データ構造

EMACS の取り扱うデータは文字の列であるが，EMACS では文字の列に2種類ある．ストリングとバッファである．ストリングは比較的固定的な文字の列を表現するのに使われ，バッファはそのまま EMACS のバッファの表現となっている．

ストリングは本体の文字の列の先頭に4文字分のヘッダがついている．ヘッダの最初の文字は削除文字で以下がストリングであることを示す．残り3文字分(21ビット)でストリングの長さを表わしている．ストリングの長さの最大は18ビット分 $2^{18}-1$ である．

バッファの方は少し複雑で**図-4**のような構成になっている．まず buffer-string と呼ばれるヘッダのみの部分があって，これが buffer frame を指している．buffer-string はストリングとまったく同じ領域に取られる．buffer frame にはバッファの実体である buffer block への各種ポインタが納められている．

6.2 基本ループ

すでに述べたように EMACS の制御構造は概念的には，コマンド読み込み，コマンド実行，画面の再表示のループになっている．ただし再表示は時間のかかるプロセスなので，ユーザからの入力の受付け及びそれの実行の方を優先するようにできている．実際には再表示ルーチンの中で1行の再表示を終了するたびに入力バッファをチェックして，未処理の入力文字が到着している時には再表示を中断して入力の処理へ回るように書かれている．

以上のようにコマンドの実行を画面の再表示プロセスが追いかけるような機構になっているので，時々，コマンドはすでに実行されているのに画面がそれを忠実に反映していないという状況がおこる．

6.3 画面の再表示アルゴリズム

EMACS では再表示を必要最小限に食い止め，かつそのための計算を効率化するためにいくつかの工夫がこらされている．

まず EMACS は画面上のすべての行に対して次の3つの情報を常に保持している．

i) 画面上に表示された各行に対するハッシュ・コード．

ii) 各行の先頭の文字がバッファ内のどの位置の文字に対応しているか（何文字目か）．

iii) 画面上でその行の最後の文字の次の位置．

これらのうち ii) の情報はカーサの位置がバッファ内のどの文字位置にあたるかの計算を効率よく行うのに使われる．iii) の情報は行末へのカーサの移動や，行末への文字の挿入などの再表示を高速化するのに役立つ．i) はある行に加える変更（どの位置にどの文字を出力するか）から容易に計算できるもので，各行への変更を実際に出力する前に，出力しようとしている行に対して計算されたハッシュコードと比較され，もし値が同じなら画面上の行と変更後の行が同じであるとみなして出力を見合わせるようになっている．これで，画面上の行がそのまま生かせる時にはそれをそのまま使うようにしているわけである．さらにこのハッシュコードの情報は画面を何行かスクロールしただけで望みの画面になるかどうかの判定にも使われている．

また EMACS の各コマンドは，そのコマンドによって変更を受けたバッファ内の範囲をでき得る限り報告するようにできているので，その情報も利用して，

再表示は決定される.

さて，本題の再表示アルゴリズムであるが，アセンブラのコードをある程度解読したところ大体次のような構造になっている（細部までは明らかでない）.

i） 入力バッファをチェックし未処理の入力があればコマンド読み込みルーチンへ飛ぶ.

ii） 画面全体の再表示が必要かどうかのチェックをし，必要ならば全画面再表示ルーチンへ．そうでなければ次へ行く.

iii） 画面の一部で再表示が必要なので，まず，行の挿入・削除コマンドのみで変更ができるかどうかをチェックする．できる場合はそうする．そうでなければ以下へ.

iv） 行の中の一部の再表示が必要なわけだから，まず，文字の挿入・削除コマンドを利用して再表示できるかどうかを判定する．できる場合でも画面から消さずに表示したままにできる文字数が少ない場合は行末まで行を削除してしまい新しい内容を出力した方が速いので，その判定をし，それにしたがって再表示する.

v） 文字の挿入・削除でだめな場合は行末まで行を削除し，新しい内容を出力する方式で再表示する.

以上がアルゴリズムの概要だが，行や文字の挿入・削除機能を有しない端末でも EMACS は動く．EMACS を動かすのに最低限必要な端末の機能は，カーサの移動，カーサから行末までの行の削除，全画面の消去の3つで，それらがあれば他は EMACS がシミュレートしてくれるようになっている.

7. あとがき

タイプライタとしての EMACS の使用から説きおこしカスタマイズ，拡張まで含めて EMACS の全貌をある程度説明できたのではないかと思う．ただし筆者自身，本稿を書くためにマニュアル[5]を読み返していて，こんなこともできたのかと思うことが何度もあったので，筆者の個人的な使い方に片寄った解説になってはいないか心配である.

本稿を書くにあたっては，日本語による EMACS の解説がほとんどない点を考慮して[9),10)]，マニュアル的な要素も附加したつもりである．したがって身近に EMACS にアクセスできる人は，本稿などを参考にしながら，是非 EMACS の世界を探検していただきたい.

TECO でのカスタマイズはかなり困難であること，ヘルプの記述の水準がまちまちである程度 EMACS を知っていないと理解がむずかしいこと，コマンドが多すぎて専門家向きだが，素人にはかなり負担になることなど EMACS にもいくつかの欠点がある.

しかし永い年月をかけて，かつたくさんの人の努力を集積して作り上げられた EMACS はやはりソフトウェアの傑作の1つであり，画面エディタのスタンダードである．このことがいくらかでもわかっていただければ幸いである.

参考文献

1) Stallman, R. M.: EMACS The Extensible, Customizable Self-Documenting Display Editor, SIGPLAN Notices, Vol. 16, No. 6, pp. 147-156 (June 1981). これは，MIT AI Memo No. 519 (June 1979) としても入手可.

2) Greenberg, B.: Prose and CONS, (Multics Emacs: a Commercial Text-processing System in Lisp), Conference Record of the 1980 Lisp Conference, pp. 6-12.

3) Wechsler, A. C.: Zmacs Manual, Symbolics 3600 Software Documentation (Sep. 1981).

4) Gosling, James: Unix Emacs, Documentation with the Software.

5) Stallman, R. M.: EMACS Manual for TWENEX Users — A Reference Manual for the Extensible, Customizable, Self-documenting Realtime Display Editor, This manual corresponds to EMACS version 162. MIT AI Memo No. 555 (Oct. 1981).

6) Ciccarelli, Eugene: An Introduction to the EMACS Editor. MIT AI Memo. No. 447 (Jan. 1978).

7) Finseth, C. A.: Theory and Practice of Text Editors or A Cookbook for an EMACS. MIT LCS TM-165 (May 1980).

8) 奥乃 博：画面エディタ EMACS のユーザ特性について，第25回プログラミング・シンポジウム予稿集 (Jan. 1984).

9) 池内克史：MIT AI ラボの EMACS エディタの使い心地，bit, Vol. 12, No. 4, pp. 26-34 (1980).

10) 和田英一：連載 エディタとテキスト処理 8 ディスプレイ・エディタ（1） bit, Vol. 14, No. 13, pp. 76-82 (1982).

（昭和59年 4 月24日受付）