北海道大学工学部研究報告
第121号（昭和59年）

Bulletin of the Faculty of Engineering.
Hokkaido Uuiversity. No. 121 (1984)

UNIXシステムの特徴についての考察

伊　達　　惇*

（昭和58年12月27日受理）

On Some Properties of UNIX Operating System

Tsutomu DATE

(Received December 27, 1983)

Abstract

UNIX is a computer operating system developed first at Bell Laboratories and now used widely for research in operating systems, languages, computer networks, and other topics in computer science, and also for document preparation as utilities for the editing, transformation, analysis, and publication of text of all types.

We discuss in this paper how useful the UNIX system is for English speaking people, or how available or not it is for us Japanese speaking people. By dealing with those which are observed in the first use by the author, we consider non-quantitative measures such as comfortability or satisfactoriness in regard to managing the system, which will differ widely in different levels of knowledges or experiences of users.

1. はじめに

UNIXは，1969年にBell Telephone Laboratoriesにおいて，単一ユーザーのためのミニコン（DEC PDP7およびPDP9）用のシステムとして開発され，その後，多数ユーザーのためのタイム・シェアリング・システムとして，膨大なソフトウエア・サポートを付与されるに至っている[1)2)3)]。発展の主役は，上記のBell LaboratoriesおよびCalifornia大学Berkeley校である。1971年には，約600の研究所，事務所において，コンピュータ教育，文書処理，サービス・データ管理等のために使われ，1977年には，PDP11/70のほかInterdata 8/32においても使用可能となっている。その間，オペレーティング・システムもアセンブリ言語からCコンパイラにおきかえられ，その後もさまざまな改良が加えられて今日に至っている。

日本でも，UNIX-VAXシステムは，東京大学を中心として関東地方のいくつかの大学間ネットワークとして機能している。しかし，これはあくまでも英語を日常語とする人達のためのシステムであり，日本語を日常語とする人達の間で，このシステムが全く同じレベルで利用，普及が進められるとは考えにくい。その相違を考えるための資料を提供することは，本稿の目的の一つである。

UNIXを構成するソフトウェアは，Cコンパイラ，テキスト・エディタ，タイプセッティング・プログラム，シンボリック・デバッガ，アセンブラ，ローダのほかに，FORTRAN，BASIC，

*情報工学専攻　情報科学

SNOBOL, APL, ALGOL, PASCAL, TMG, 等の言語を多数含んでいる。従来の言語処理系にはない特徴として，ファイルシステムの階層構造化，互換性，非同期処理，多数の言語を含む100以上のサブシステム保持，等を設計フィロソフィーに掲げ[3]，それらを実現している。

この種の言語処理系の評価は，さまざまな角度から与えることが可能であるが，〝初めての者にとって入りやすい〟，〝多少使った者にとって使いやすい〟，〝熟練した者にとって使用法の拡張が容易で便利である〟，という判断基準を考えることは必要であり，かつ妥当であろう。それぞれの判断基準に対する要請は，相互に併立しえないものがあり，主観的要素の多い判断であるから，それぞれの判断基準に対して，別個の人間がそれぞれの充足度をはかったのでは，公平な比較は得られない。一方，ある一人の人間が，これらのいずれの立場からの要請に対しても，その判断を行うための最適な状態にある，ということを期待することも無理である。そこで，まず第一の判断基準を考える場合は，使い慣れてからでは遅過ぎるので，筆者がIllinois大学に滞在して利用に接した最初の時機に感じたことをまとめる必要がある，と考えたことが本稿作成の主要な動機である。

第三の判断基準に対しては，Illinois大学における利用状況を観察することによる推量という方法をとることとした。日本では，研究者は日常の研究活動にあたって，ペンと紙，タイプライタと用紙，研究室関係者が共有する文献，パソコン，研究者仲間への掲示板，等をそれぞれ使い分けているが，Illinois大学では，それらはすべて，このUNIX-VAXシステムに集中しつつあるといっても過言ではない。ほとんどすべての研究者が一日に一回は端末機に接している可能性がきわめて高い現状では，それは可能となっている。これは，特定の大学に限ることではない。日本では，漢字を利用することが大きな制約となって，ワードプロセッサと呼ばれるものの普及発展が遅れているが，漢字入力の問題以外にも，利用者およびその環境に大きな相違が感じられるので，比較検討を行った。

2. UNIX（Berkeley）システム利用の概要と特徴

Illinois大学のDepartment of Computer ScienceでUNIXシステムは，California大学 Berkeley校で発展させられたものに依拠している。本節では，Illinois大学のシステムについて概観する。

2.1 ハードウェアシステムの構成

ここで使用する計算機システムは，Department of Computer Scienceの学科の中にあるVAXシステムと，Computer Service Office管理下のCYBERを主要機としているが，UNIXは学科内のVAXを中心に利用している。ハードウェア構成は次の通りである[2]。

DEC VAX 11/780 2台（主記憶4Mb, テープドライブ各800/1600bpi, disk各1200Mb, 144 communication port）が主要機であり，DEC VAX 11/750 7台がwork stationの役割りを果す。

このほかに，数値計算，解析やデーターベース研究のために PRIME 650 1台（主記憶2 2 Mb, テープ・ドライブ800/1600bpi, disk600Mb）がおかれている。これらの計算機システムは，孤立して置かれているわけではなく，3 Mbpsのローカル・ネットワークEthernetを介して，後述するような大型機と接続して利用することができるしくみになっている。

これらの周辺機器として学科内で使用可能のものに，PRINTRONIXラインプリンタ2台，multi-penページプリンタ 1台，QUMEページプリンタ 1台，IMAGENレーザプリンタ1台，がある。

2.2 UNIXシステム利用の現状と特徴

前項で述べたDepartment of Computer ScienceがするUNIX-VAXシステムは，孤立して存在するわけではなく，大学全体の計算機サービスを担うComputer Services officeの管理下にある大型機Cyber175/174あるいはIBM4341, VAX780等と連動することが可能である。その手続きも簡単であり，UXIXのコマンドの引数としてシステムコールを行うだけでよい。現在の利用者は，学科内について言えば，56名のsenior staffと約250名のstaffおよび大学院生である。アクセスに要する時間は，午前中ならば数秒，午後の混むときでは1分間ほどである。ターンアラウンドタイムで問題となりそうなのは，レーザプリンタ（IMAGEN）である。ソフトウェア処理時間は数分間であるが，100ページ相当の大きなジョブが入ると，それだけで15分以上もかかる上に，その間に用紙あるいは液の補充などが必要となって，30分以上になることもしばしばである。現段階での入出力の弱点はここにある。

2.2.1 User Communication

UNIXの特徴であって日本の計算機利用の立場と最も異なる点の一つはuser communicationである。これは，それを可能にするコマンドがいくつかある，ということだけではなく，次節で述べるようなファイル・ディレクトリシステムの工夫，あるいは，研究上の討論を仲間とかわす習慣などと密接に関連して，システムの特徴となっている。

mailは，登録者に対して，私的もしくは多勢に文章を送るコマンドである。主題と本文をキーボードから入れるのがたてまえであるが，これだけであると実際の討論にはあまり役立たない。なぜなら，同じ式や同じ表を何回もキー・インすることに繁雑さを感じるからである。ファイルに格納されているものの一部が自由に引用記入できるようになって，初めて討論の武器となりうる。このためには，mailコマンドの実行中に，エディタを呼んだり，元に戻ったりすることができるしくみが必要であり，UNIXではそれが可能となっている。受けとったmailの保存あるいは返信発送，未登録者へ宛てたmailの処理方法が決められている等，こまかいところに神経が配られている。

mailの交換は，単一の大学の計算機システムの中にとどまらず，USENETを介して近隣の大学との交換も自由にできるようにしてある。Illinois大学に最も近い他大学システムは，Purdue大学のものである。日本でも関東地方のいくつかの大学で使用中であることは，すでに述べた通りである。

notesfilesは，討論，告知のためのbulletin boardのようなものである。それぞれのファイルには，トピックニュースや連絡事項，それに対する各人からの回答などが入っている。ファイルへ書き込みをする場合には，mailと同様に，エディタの使用が可能である。書き込みを終了したあと，発送をする前に，一回はディスプレイで全文を確認する手順をふまない限り，発送できないしくみとなっている点からは，現場の使用経験が生かされていることが感じられる。

2.2.2 UNIX shellおよびファイル・ディレクトリシステム

UNIX shellは，利用者とVAXのOSとの間のインターフェイスの役割を果すコマンド・インタープリタである。システムコマンドをinputとして他のプログラムを読む媒体でもある。UNIX生誕および成長の歴史から考えて当然のことではあるが，このUNIX shellにはBell Laboratoriesが開発した〝Bourne shell〟，およびCalifornia大学Berkeley校が開発した〝C shell〟がある[3]。ここではC shellを扱っている。

多様なコマンドを有することはいうまでもないが，そのほかに，通常のコマンドの機能をかえて使うためのコマンドラインcharacterがあること，また，I/O処理，ファイル処理の途中で他の

操作を可能とするための Background 操作を有していることは UNIX の特徴である。実際に，前項で述べたレーザプリンタ出力を必要とする場合，それが終るまで端末機が何らの操作も行えなくなる，というのは大変不便なことである。

UNIX のディレクトリは完全な樹枝構造となっており，個々のファイルの identification は，ディレクトリ名の連鎖を伴ってなされる。ファイルにパス名が与えられるということは，前項で述べた研究・討論に際して，他人のファイルにアクセスすることを容易にする。本人が，ファイルの読みとり，書き込み，実行，を拒絶する手続きをしていない限り，そのファイルは，他人にとって，自分のファイルと同じように処理することができるわけである。自分のファイルの内容が自分の知らない間に変えられることが困る場合は，他人の書き込みを禁止しておけばよい。通常，全く指定をしないファイルを作った場合は，「他人が読みとることは可能だが，書き込みや実行はできない」というモードが割りあてられている。このことは，日本におけるファイル利用の習慣と著しく異なっている。

2.2.3 オンラインドキュメンテーション等

UNIX に関係があるマニュアルは存在する[1]が，それを書物として机上に置く人はあまり見かけない。日本では膨大な数のマニュアルを各人が身近かなところに置いて使う習慣があるが，UNIX システムではそういう習慣が形成されない。それは，マニュアルを参照する必要がなくなっているからではなく，ドキュメンテーションが進んで端末機を通じて行えるようになっているからである。マニュアルの内容は，いつでも端末機を通じて読みとることができる。とくに，UNIX 処理の途中でコマンドの利用法が分らなくなったという瞬間に，マニュアルの該当部分を読んで操作を継続する，ということが可能になっている。コマンド名がわかっていて，使用法の細目が不明であるときは，**man コマンド名**を送れば良い。コマンド名の記憶もあやしく，対応の仕方が分らない，というときは，かなり丁寧な HELP 機能が使えるほか，後述する Emacs エディターを使用中のときは，可能な対応の仕方を列挙してもらって，それを見ながら行動をきめる，ということも可能である。

ただし，日本人にとっては，日本語のマニュアルの場合でも該当部分を読んだだけでは良く分らず，関連する他の部分を読んだり，トライ・アンド・エラーをいくつかやって見てはじめて分る，ということがある。英語マニュアルの方が明快である場合が多いが，それでも見かけない単語に出会って，別の辞書を必要としたり，マニュアルの他の部分を往来する，といった手間がかかることもある。

man-k（キーワード）は内容によるドキュメンテーションを可能とし，**man-f（ファイル名）**はそのファイルに関する情報を提供する。

後に書式制御の項でも触れるが，利用者の側の誤操作の影響をなるべく少なくすることが望ましいことは論を待たない。フォーマット上のミスは比較的容易に検出され，修正が可能だが，タイプミスのようなものは容易でない。その検出には，プログラムが辞書を内蔵することが必要であり，UNIX では，すでに公称25000語の辞書（reference file）を有している[2]。これを利用するコマンドは **spell** である。固有名詞については，日本の地名，人名のほとんどがミスタイプリストの中に登場する。TOKYO, YOKOHAMA, NAGOYA, KYOTO, OSAKA はミスタイプではないが，SAPPORO, HOKKAIDO, KYUSHU 等はミスタイプリストに登場する。人名についても，SUZUKI, TANAKA はミスタイプではないが，それ以外のほとんどはミスタイプとされる。SATO, FUKUDA, YUKAWA, TOMONAGA 等はいずれもミスタイプとされる。

文法ミスの発見については，今のところ UNIX は何らの機能を有していない。この問題は，全

く形式的な部分があるので，さらに工夫の余地がある。

以上のことは，UNIXが英語の文書処理を行うものであるがゆえに可能である。そこで可能であるからといって，日本語の誤字，脱字が自動的に検出されることが容易にできる，ということにはただちにはならない。

3. Emacsエディタ

UNIX-VAXシステムは，それだけで非常に便利なワードプロセッサであるというわけには行かず，エディタや書式制御のためのフォーマッタ等，関連するソフトウェアの有効な利用が一体となって，初めてその力を発揮することとなる。そこで使用されるエディタにはいくつかあるが，その中で最も有力なEmacsを対象としてその特徴を検討する。

EmacsはMIT（Massachusetts Institute of Technology）で開発されたものであるが，それには二通りのものがあって，一つは，Richard StallmanがMITで使用していたTECOの拡張として作成したITS-Emacsであり，もう一つは，Bernie GreenbergがMITのMULTICSのために作成したものでありMacLISPで書かれている。ここで扱うUNIXのEmacsはITS-Emacsを元祖とするもので，厳密にはEmacs-likeエディタと呼ぶべきものである[6]。

Emacsが他のスクリーンエディタと異なる点は，拡張性にある。拡張性の主な内容としては，書式を制御したり数式を表現したりする等の目的で作られたソフトウェア・ツールを特殊な方法で記憶しておいて操作を簡単にすることや，コマンドを組合せて利用者がコマンドを作って使用することができること，あるいはMLispと呼ぶLISPに似たインタープリンタを内蔵していて，その利用を可能にしていること，などが挙げられる。

3.1 コマンドの構成

制御キー（コントロール・キーヌはESCキーなど）と一文字の英文字又は記号とから成るコマンドを，以後一字コマンドと呼ぶことにする。Emacsにおける基本的なコマンドは一字コマンドであり，ときとしてこれを二つ組み合せたコマンドが用いられる。しかし，これだけでは，コマンドの種類が十分に得られないか，さもなくば利用者にとってそれらの機能を記憶するための労力が大きくなるという問題がある。

Emacsでは，カーソル移動，消去，探索，置換などの，ごく基本的な操作は一字コマンドで処理しており，より複雑な機能にに対しては，機能な連想しやすい英単語の列で表現するコマンドを使用することとしている。基本的な機能に限定して一字コマンドを使用することによって，「簡便である」，「初心者に理解しやすい」という要請を満すとともに，mnemonicなコマンドを併用することによって，「多くのコマンドを覚えやすい」，「高度な利用が可能である」という要請をも満す，という工夫が見られる。

表1-A～表1-Dに，主たる基本コマンドを示して構成と構能および今後の議論の理解に供することとする。表1-Aのカーソル移動については，コマンドの種類は，機能面からは多い程便利であり，覚えやすさの点では少い方が良い。ここでの工夫は，類似のコマンド表現を与えるという関連性であるが，表1-Aでは，それが読みとられるよう配置してある。

表1-Bは，探索・置換，領域管理およびマクロ操作，コンパイルといった多くの機能のうち，基本的なコマンドを表記したものである。特徴については3.2.1で論じる。表1-Cおよび表1-Dは，関連するいくつかのファイル，バッファを管理する基本コマンドである。特徴は3.2.2において論じる。

表1－A　カーソル移動と消去

character	word	line	paragraph	screen
↑b(一字左)	ESC-b(一語左)	↑a(左端)		
↑f(一字右)	ESC-f(一語右)	↑e(右端)		
		↑p(一行前)	ESC-[(冒頭)	ESC-V(前ページ)
		↑n(一行後)	ESC-](末尾)	↑V(次ページ)
↑d(右字消)	ESC-d(右語消)	↑k(右行消)		
↑h(左字消)	ESC－h(左語消)			

(↑はコントロール・キーとの同時操作，ESCはEscapeキーとの逐次操作を意味する)

表1－B　探索・置換・マクロ操作等

↑S　前方探索	↑ⓐ　マークセット	↑X（　キーボードマクロ開始
↑R　後方探索	↑X↑X　マークとカーソル入替	↑X）　キーボードマクロ終了
ESC-r　置換	↑W　領域消去	↑Xe　キーボードマクロ実行
ESC-g　確認置換	↑X↑F　ファイル書き込み	↑X↑E　"make"の実行
ESC-c　大文字化	↑C　Shellへの復帰	↑X↑N　次のエラーへ移動
ESC-l　小文字	↑－　Shellへリカーシヴに復帰	↑X!　コマンド実行

表1－C　ウィンドー操作

ESC－，　冒　頭	↑Xn　次のウィンドーへ	ESC－↑V(次ページを別のウィンドー)
ESC－・　末　尾	↑Xp　前のウィンドーへ	↑Z　(上スクロール)
↑X2　二　分	↑X↑Z　縮　小	ESC－Z　(下スクロール)
↑X1　統　一	↑XZ　拡　大	ESC－!　(冒　頭)

表1－D　バッファ及びファイル操作

↑Y　(消去からの復活)	ESC－↑Y(バッファをカーソル部へ挿入)
↑X↑V(ファイル呼び出し)	ESC－<　(現在バッファ冒頭へ)
↑X↑I(ファイルをカーソル部に挿入)	ESC－>　(現在バッファ末尾へ)
↑X↑R(ファイル読み・現在バッファ消去)	↑Xr　(メールを読む)
↑X↑B(使用可能バッファのリスト)	↑Xm　(メールを送る)

3.2　入力・編集・校正機能とその特徴

多様な利用者の要求に対応することができて，それぞれの利用者の労力をなるべく減少させるとともに，結果を早く知らせることが望ましいことであり，その立場から，挿入，消去，探索，置換といった，入力・編集・校正に必要な操作に対するEmacsの工夫の様相を検討する。

利用者が行う操作は，ディスプレイおよびキーボード操作である。

コマンドに対しては

(a)　覚えやすく，間違いをおかしにくい
(b)　操作が簡単である
(c)　機能が豊富である
(d)　多くの現用機種に対応できる
(e)　拡張が容易である

というような必ずしも併立するとはいえない要求がある。

使い方についてはグラフィクスを除外するとして

(f) 一つのスクリーンの中で，同じファイルの別の部分を同時に参照しながら編集ができる

(g) 一つのスクリーンの中で，別のファイルの一部分を同時に参照しながら編集ができる

(h) エディタの中で，コンパイルやエラー・チェックができる

(i) 書式制御の結果をエディタの中でスクリーン上に見ることができる

(j) 数式表現の結果をエディタの中でスクリーン上に見ることができる

(k) 使い慣れない利用者にとってサポートが豊富である

というような要求がある。UNIX-Emacsはこれらの要求すべてを満しているわけではない。

3.2.1 基本的な操作

カーソルの移動は，文字単位，語単位，行単位，段落単位，ページ単位，ファイル単位，のいずれもが必要となることがあり，Emacsは，これらに一字コマンドで対応する。消去についても，文字単位，語単位，行単位のものについては一字コマンドで対応し，領域消去については，事前処理を行ってから一字コマンドで処理する。挿入を指定するコマンドはない。この点は，Emacsの大きな特徴であって，通常のエディタでは，挿入の手続きの開始と終了を示すコマンドの使用が必要である。Emacsでは，制御キーを併わないすべてのキーボード操作が挿入とみなすので，あらためて挿入の開始や終了を示す必要がない。以上について表1-Aを参照されたい。

探索および置換も一字コマンドで処理する。探索・置換は，系統的な用語の変更あるいは誤りの修正等に対して力を発揮する。そのために

(1) 探索すべき文字又は文字列を早く処理できる状態におく

(2) 必要な置換を早く行う

(3) 探索・置換すべき文字又は文字列が網羅される

(4) 同じ文字又は文字列に対して，それが存在する場所によって置換を必要としたり，しなかったりする場合に対処する

(5) 文字列全体としては同じものではないが，その中に共通の文字列部分をもつ，すべての文字列を探索できる

(6) 探索対象を文字又は文字列で指定せずとも，ミスタイプのような形式的ミスを探索できる

等の要請が満されることが望ましい。Emacsは，(6)を除く要請に対して十分に対応している。(6)については，UNIX自体が対応するが，Emacsが後述するようなUNIXとのコミュニケーション機能を有するので，結果としては，Emacsの機能と考えてさしつかえない。この探索・置換機能を利用した入力操作簡単化のための略号モードの利用も容易になっている。

しかしながら，系統的な修正のための探索・置換のすべてが容易にできるようになっているというわけではなく，また，いくつかのファイルに共通部分があって，それぞれに同じ修正を施す必要が生じた場合，それぞれのファイルに個別に対応する必要がある，などの形式的処理の自動化が完成したとはいえないことを示す課題は，まだまだ残っている。

3.2.2 ファイル・バッファ・ウィンド処理

UNIX-Emacsは，ファイル，バッファ，ウィンド処理について，一字コマンドをたかだか二つ使う程度で，次の操作を行うことができる。

(1) 指定したファイルの内容をバッファに読み出し，バッファに名前をつけてUNIXにおけるファイルと同じように扱う。

(2) 現在処理中のバッファの内容をもとのファイルに書き込む。
(3) 現在処理中のバッファの内容を，あらためて指定したファイルに書き込む。
(4) 現在処理中のバッファおよび関連する一連のバッファの内容を，それぞれ，対応する元のバッファに書き込む。
(5) 現在処理中のバッファの内容を，あらためて指定したファイルの指定した場所に書き込む。
(6) バッファの数はソフトウェア上の制限はないと考えてさし支えない。
(7) ウィンドの数はソフトウェア上の制限はないと考えてさし支えない。
(8) どのバッファの内容もウィンドで見ることができる。
(9) ウィンド間のカーソル移動は容易である。
(10) エディタの中で，編集したいファイルを次々と呼び出して処理することができる。

したがって，(f)および(g)の要請に対しては十分対応すると言える。

なお，驚くほどのことではないが，長いファイル名や長いコマンドをキー・インするとき，途中までタイプした段階で，選択の余地がなくなっていれば，自動的にそのあとを判別して補う機能や，選択の余地が残っている場合で，そのあとの文字を忘れた，というようなときに，残された可能性として選択枝を列挙する機能も備わっている。

そのほか，ファイルの書き込みを行うとき，誤った書き込みをした場合の救済措置として，書き込みが行われる以前のファイルの内容を自動的にバックアップする機能や，今までに使用したバッファの中味をしまっておく checkpoint file なども備えられている。

3.2.3 コンパイルおよび UNIX との communication

大きなプログラムを作るときには，まず部分を分けて作成し，それぞれのチェックをしてかつ一つにまとめる，という手続きが必要である。UNIX のプログラムの一つである〝make〟は，各ファイルの間の関係を記述するデータ・ベースを持っていて，分割されたプログラムを個別にコンパイルすることができるので，大きなプログラムを作るときに有用である。Emacs はこの〝make〟を実行してエラー・メッセージをとり出すコマンドを有しているので，結局，エディタの中でコンパイルが可能ということになる。

Emacsでは，任意の段階で，UNIX shellに移行し，UNIXのコマンドを実行した後にその結果をバッファに入れて，ウィンドで内容を見るとか，あるいは別のプログラムを実行して，それが終了したらエディタに戻る，というようなこともできる。とくに，Emacs を呼んだときのファイルをそれが属するディレクトリから別のディレクトリに移したいというようなとき，あるいは，現在時刻を入れたい，というときなどには，この機能はしばしば使われる。

4. Nroff/Troff Formatter

フォーマッタとは作成したファイルの書式を整えて印刷するのに必要なソフトウェア・ツールであって，エディタと合わせて使用することによって俗にいうワードプロセッシング機能を果すことになる。

UNIX-VAX システムで使用するフォーマッタは Nroff/Troff であり，その概要は次の通りである[5)7)]。

4.1 Input Format

Nroff/Troff では必要な命令をリクエフトと呼ぶ。2 文字の英文字でリクエスト名を表わし，

コントロールラインにコントロール文字を置くことによって，このリクエスト名はリクエストとして機能することになる。コントロール文字は，通常はピリオドであり，バック・スラッシュもしばしば使われる。

リクエストには，リクエスト名を指定すればそれで用が足りるものもあるが，通常は，スペースであれば何行分であるか，文字の字体や大きさであればそれぞれどれにするか，というようないくつかのパラメータを指定する必要がある。このパラメータをリクエスト引数と呼ぶ。

4.2 基本機能

基本的なリクエスト名とリクエスト引数を表2～表4に示す。書式の表現で使うことのできる画面の分解能は，水平方向に1/432インチとなるように設計されている。したがって，これ以上の細かな動きは指定できないし，リクエスト引数として長さを小数点のついたインチ数で指定したとき，この分解能の倍数からはみ出た端数は，切り捨てられて処理される。

表2 基本的なリクエストとその内容（その1）（＊1～＊5は本文中に説明）

内　　容	リクエスト	備　　考
filling	.fi	元へ戻すには .nf とする
adjusting	.ad A	A は l(左)，c(中央)，r(右)，b(左右)，空白(左右)
		元へ戻すには .na とする
break	.br	
spacing	.sp FK[*1]	F は K を単位とする長さ
	.sp N	N は行数
line length	.ll Fi	F はインチを単位とする長さ
indent	.in ±Fi	F は上と同じ。＋は右へ，－は左への意。
temporary—	.ti ±Fi	F は上と同じ。1行についてのみ機能
page offset	.po ±Fi	F は上と同じ。line length をかえない。
centering	.ce N	N は行数。N＝0 は解除を意味する。
tabset	.ta M_1，M_2[*2]，…	M_1，M_2…単位はインチ
underlining	.ul N	N は行数。N＝0 は解除を意味する。
continuous—	.cu N	スペースにも underling を行う点が .ul とちがう
hyphenation	.hy N[*3]	N は後述するような整数
	空　　白	自動的に行う。解除は .nh
font selection	.ft N[*4]	N は1～6の整数又は p
point size	.ps N[*5]	N は6ポイントから36ポイントまでのポイント数
embolden font	.bd NM	N は font 番号，M は M－1ユニットに2回，プリントするの意。光学的出力装置のみ。

(＊1)の K は scale indicator であって，次の8通りのいずれかの文字を指定する。i：インチ，C：センチ，P：パイカ，p：ポイント(1/72インチ)，m：現在のポイント値，n：現在のポイント値の半分の値，u：基本ユニット (1/432インチ)，v：ラインスペース数。

(＊2)のM1，M2，…はストップを置く場所をインチを単位として示す数字で，数字のみの場合は左端からの長さを意味し，プラス記号と数字の場合は左隣りのストップからの長さを意味する。ストップは35まで置くことができる。

(＊3)の N は整数。N＝0 は off，つまり hyphenation を行わないこと，N＝2 はページの最終行については off，N＝4 は各語の末尾2字を分けることはしない，N＝8 は各語の語頭の2字を分

けることはしない，の意味がある。N＝6は上のN＝2と4の組み合せ，N＝12は上のN＝4と8の組み合せ，N＝14は全部の組み合せを指定する。

(＊4)のNは次のFontの番号を意味するが，( )内に示す記号を用いてもよい。1. Times Roman(R), 2. Times Italic(I), 3. TimesBold(B), 4. Roman Bold Italic(BI), 5. News Gothic(G), 6. News Gothic Italic(GI)。このほかSpecial Mathematical Fontのために7が用意されている，とマニュアルには記載されているが，実際には7は使える状態にはない。Fontの指定を解除して，元のFontに戻すためには，Nとして数字ではなく，文字Pを使う。

(＊5)のNは，6～12の整数，14～24の偶数，28，36であってポイント数を意味する。

表2は，基本的なリクエストのうち，その結果が印刷面の上に具体的にあらわれるものを扱っている。似たような機能に対して異なる方法がこまかく指定されているが，さほどの繁雑さを感じないのは，現場の要求がとり入れられているためであろうか。(＊3)のhyphenationはかなりこまかい指定があるが，それぞれの機能に割り当てた数字とその組合せ方は，記憶しやすいという意味で，工夫のあとが見られる。

表3 基本的なリクエストとその内容（その2）

内　　容	リクエスト	備　　考
strings	. ds xx(string)	xxは与える名前で1又は2字の英文字。行がかわるとstringの終り。長いときは，行末に＼をおく。*6
append to strings	. as xx(string)	. dsのstringのうしろに続けたいとき。
macros	. de xx ⋮ ..	新しくリクエスト機能を定義して使う。入力テキストの行数に制限はない。name listはstringsと同じものを使用する。
macros arguments	. de xx (＼ $ n) ..	nの値は1～9まで9変数の使用が可能。同じ変数は何回でも使用できる。
append to macro	. am xx ..	macroの. deへの追加
remove request	. rm xx	xxは通常のリクエスト，マクロ，ストリングのいずれも可。
rename requset	. rn xx yy	リクエスト，マクロ，ストリングのname xxをyyに。
number register	・nr xx N	number registerの値をNとする。
auto-increment	. nr xx N 1	number registerの値を増やす。
assign format	. af M C	register Rにフォーマット Cを与える。*7
remove register	. rr R	register Rをとり除く

表3は直接的に書式を決めるためのリクエストではなく，書式を与えるためのフォーマッタ使用を便利にするためのリクエストである。

(＊6)stringの終りは行の終りで判断されるので，2行以上にまたがる長いstringの場合は，行末にback slashをおくことで〝続き〟をあらわす。

(＊7)フォーマットCはC＝1ならレジスターは0，1，2，…に，C＝01ならば00, 01, 02, …を，C＝001ならば，000, 001, 002, …をレジスターに入れることを意味する。

stringsは指定した表現を短い(name)で入力できるようにするもの。.dsで定義したものを使

うときは＼＊ｘまたは＼＊(xxを用いる。

macrosは，基本リクエストを組み合せて新しいリクエスト機能を定義して使うものである。書式の複雑な操作をくり返し行うときの入力テキスト表示を簡単にすることができるものであり，フォーマッタの使い易さを支配する重要な機能である。

number register xxは.nrで定義した値をもつが，これを使用するときは，＼nxまたは＼n(xxの形を用いる。auto incrementを定義したときは，＼n＋xを用いるとレジスターの値が1だけ増大する。

表4　基本的なリクエストとその内容（その3）

内　　容	リクエスト	備　　考
setting traps	.wh N xx	xxはマクロ名であり，Nはトラップを置く場所。0はページのトップ，－は底から数える。
no-break control character	▼(single quoto)	マクロリクエストがbreakを起さないようにする。
three-part titles	.lt F i .tl ▼M 1▼M 2 ▼M 3	タイトルの長さをインチ単位で指定。 M 1は左を，M 2は中央，M 3は右を指定する。
page length	.pl N	インチで指定する。通常11インチ。
page numbering	.pn N 空　白	次のページをNとする。 自動的に行う。
changing trap position	.ch xx N	マクロxxの位置をNに動かす。
unpaddable space	＼＼	adjustingによる語間のあき過ぎを防ぐ。
zero-width filler	＼&	コントロール文字を印字するときの無効命令。
break and spread	＼p	これをつけた単語の終りでbreakとする。
continuation	＼c	2行以上に分れている単語をつなぐ。
constant character width	.cs N, L, H	Nは表2のfont selectionの番号。Lは字幅でL/36 emsで数える。Hはfont sizeをポイントで指定するものであり，省略すれば現在のsizeを意味する。

表4はページコントロールを中心としたリクエストの表である。トラップの定義は通常はheaderやfooterのために必要であり，それぞれマクロ命令として与えられているので，トラップの場所と記述内容を与えれば用が足りるしくみになっている。

書式については，英語はかなり単純化かれており，科学論文，文芸関係，手紙など書式はおおむね決まっているが，日本語は漢字の問題や縦書き横書きの問題がある上に，それぞれの分野で書式が定形を持つに至っていない，という問題もあって，書式の制約が日常感覚と一致するまでには時間がかかりそうである。

5．数式処理プログラム EQN

科学技術論文には不可欠な数式を，数学（および／又は）タイプセットに通じていない人が，容易に表現できるように作られたもので，UNIXおよびGCOSの上でのグラフィックシステムphototypesetterのためのプログラムである[8]。この処理には，出力機器としてレーザープリンタの

ような光学的装置を必要とする。EQN は，前項で述べたフォーマッタである Troff の前処理プログラムとして機能するので，数式と通常のテキスト表現の両方が混じり合うテキストに対しても対応できる。数式部分は EQN が，通常の表現部分は Troff が担うこととなる。UNIX の上で EQN を使用するためには，一行のコマンド

eqn ファイル名 | troff

を送るだけで良い。

使用する文字は，英文字，数字のほか，数学記号のほとんどすべてとギリシア文字である。subscript または superscript は何重にも使用可能であり，総和記号や積分記号のように変数の変域を指定する場合は，通常の英語の表現からの類推が容易な表現を用いるように工夫している。数学記号やギリシア文字は，所定の英単語表現を用いるのであるが，それは，記憶しやすい反面，タイプ操作の手間はそれほど簡単ではない。

Emacs エディタの諸機能を活用して手間をはぶく工夫が必要であり，この処理プログラムを活かすには，慣れが必要である，という点で，冒頭に紹介した意図が十分に満たされているとは言い難い。

5.1 数式処理の例

数式として，行間に独立に記述されるものについては，そのはじまりを，.EQ，その終りを，.EN で指定する。その間におかれたものが数式処理を受けることになる。一方，本文中に簡単な数式を引用する場合は，そのはじまりと終りを，いずれも $（ドル記号）で指定する。この場合は，冒頭で

```
.EQ
delim$$
.EN
```

というリクエストによって，delimiter をセットしておかねばならない。以上で用意は終りであり，あとは式および本文を入力すれば良い。

式の中に登場するギリシア文字は，$\alpha \rightarrow$ alpha，$\beta \rightarrow$ beta，$\gamma \rightarrow$ gamma のように，通常の英単語で表す。ギリシア語の大文字は，対応する英単語を大文字で表す。

表5 簡単な記号のキーワード

Σ	sum	∂	partial	$\geqq$	>=
$\int$	int	∞	inf	$\leqq$	<=
Π	prod	lim	limit	$\equiv$	==
U	union	X	times	$\rightarrow$	->
$\cap$	inter	$\neq$	!=	$\leftarrow$	<-

例として，n 次元のニュートン法公式をタイプセット用に表現したものと，その印刷結果を示す。

入力テキスト

```
.EQ (1)
x sub (m+ 1) sup kappa ^^=^^
x sub (m) sup kappa ^^-^^
sum from { mu = 1} to n^^
[ partial over { partial x sub (m) sup kappa }
g sup mu ( bold x sub (m) ] sup {-1}
g sup mu ( bold x sub (m) )^^^^
( kappa ^=^1,...,^n),
.EN
```

プリント結果

$$x_{(m+1)}^{\kappa} = x_{(m)}^{\kappa} - \sum_{\mu=1}^{n} [\frac{\partial}{\partial x_{(m)}^{\kappa}} g^{\mu}(x_{(m)}]^{-1} g^{\mu}(x_{(m)}) \quad (\kappa = 1,\ldots, n), \qquad (1)$$

ここにギリシア文字は κ (kappa) および μ (mu) であり，ベクトルを意味する太字は，boldを変数の前に付すことによって指定している。

文中に式を引用する例を次に示す。

入力テキスト

```
as one of the iterative methods obtaining approximate solutions
of the system of equations $g sup kappa ( bold x)^=^0^ $( kappa^=^1,...,^
n)$, where $bold x sub (m)$
denotes a point of the m-th approximation, it is well-known that
the difference $bold epsilon sub (m)$ between the m-th approximation
and the exact solution $bold x bar$ can be represented asymptotically as the
homogeneous quadratic form of $bold epsilon sub (m-1)$:
```

プリント結果

as one of the iterative methods obtaining approximate solutions of the system of equations $g^{\kappa}(x) = 0$ $(\kappa = 1,\ldots, n)$, where $x_{(m)}$ denotes a point of the m-th approximation, it is well-known that the difference $\epsilon_{(m)}$ between the m-th approximation and the exact solution $\bar{x}$ can be represented asymptotically as the homogeneous quadratic form of $\epsilon_{(m-1)}$:

5.2 エラー処理

数式を表すべく，所定の約束に従って数式ではない表現をプログラムするのであるから，その表現を見ただけで目的を達しているか，エラーが含まれるかを判別するのは楽ではない。したがって，誤りであると判別できるものは，なるべく処理プログラムに判別してもらう必要がある。現在のところ，括弧の数が左と右で合わない場合，superscript が指定されていながら，それを肩にのせるはずの変数がない場合，など形式上あきらかに誤りと判別できるものが含まれる場合，syntax error のメッセージを発する。メッセージは，ファイル名とその誤りが所在する場所の行番号であって，誤りの内容は指示されない。

次に，syntax error とはならない誤りについては，この処理プログラムは何らの働きもしない。ミスタイプであって syntax error とはならないものは，視察によって発見するか又は，実際にプリンターに印字した結果を見て判別するしかほかに方法はない。1節で述べたように，レーザープリンターのターンアラウンドタイムは，かなり長いので，後者の方法による誤り発見はまだ不便であり，今後の課題である。

6. 問題点の検討および結言

UNIXシステムを，英語を母国語とする人たちが使うときの問題点と日本語を母国語とする人たちが使うときの問題点，さらに，日本語を母国語とする人たちが，UNIXが英語で果している役割とほぼ同じ機能を，日本語で処理するためのシステムを考えるときの問題点，これらは関連性および共通性が大きいものの，それぞれ独自の問題をかかえていて，大きな相違があることは言うまでもない。本稿は，上の分類で言えば，第二の問題を扱っており，第三の問題において利用されることを期待している。動機は，イリノイ大学において，UNIXシステムを日常的に使い得る環境の中に滞在する間に，UNIXについて感じたことを表現しようということである。

検討は，UNIXの利用状況についての客観的な数字をもとに進めることが望ましいのである

が，アメリカの大学で，計算機利用に関するデータを入手することはかなり困難である。イリノイ大学を含めて，アメリカの多くの大学は，計算センター等の計算機利用時間という最も単純なデータでさえ公表していないからである。これは，意識的に秘密にしているからでもなく，統計をとらない政策をとっているからでもない。事実，筆者が1983年9月から11月まで，CYBERの利用状況を示す毎日のデータを入手することは可能であった。公表しない理由は，単にそのための手間と費用を節約するためである，とのことであった。月刊もしくは隔月間で計算機利用報告を発行している大学は，筆者の調べた範囲では，Purdue大，Minnsota大，Washington大など10大学ほどであった。Illinois大では，データは所定の会議のメンバーに回覧報告するところまで行っている。筆者が調べた計算機システム利用状況を月毎にまとめると表6のようになる。

表6 Illinois大学における最近のCYBER175利用状況

	JOB数	CPU時間(分)	connection 時間(分)
1983年 9月	66,719	6,584	1,620,028
10月	66,399	6,961	1,674,764
11月	48,417	4,834	1,220,743
11月推定	(72,600)	(7,250)	(1,831,000)

※11月は20日までのデータ

アメリカのacademic yearは9月から始まって5月に終るので，9月あるいは10月という月は，日本で言えば4月と5月に相当する。計算機利用が最も少い時期であるが，この段階での計算機利用時間がCPU時間で1日平均3時間半をこえている，という使用頻度は，7 days 24hours利用をたてまえとするデータであるとはいえ，かなり高いものといえよう。

さらに，表6の数字のうちでとくに注意をひくのは，connectiontimeの大きさである。1日を24時間で平均したとしても，実時間の40倍以上のconnectiontimeがあるということ，つまり，つねに平均40個所以上からのアクセスが維持されていることになろう。これは，計算機が科学計算を主要用途としていたときには考えられなかった数字である。CPU時間に比べてconnectiontimeが大きいことを示すこの表からも，計算機利用の中に文書処理がしめる割合の増大をうかがい知ることができる。

日本語は3000あまりの漢字を使用するため，わずか26文字の基本文字を有するに過ぎない英語に比べて，入力操作の不便が著しいと指摘されて久しい。それを補うための工夫がなされてつつあるとはいえ，同音異義語の多様性，文字の読み方の多様性という日本語の特性に妨げられて，簡便に行われるための道のりは遠いと言わざるをえない。ここで〝簡便に〟というのは重要なことであって，用語そのものは相対的な概念であるが，その簡便さのわずかな相違が，この種のシステムの日常的な利用をもたらすか否かを分ける重要な要因となると思われる。

〝はじめに〟の節で述べたように，話をIllinois大に限定したとして，どの研究室にも置かれているコンピュータ端末を通してUNIXを使い始めてからは，論文の作成，印字，討論，会合等の通知および計算のいずれもが，UNIX-VAXシステムに集中してきている。一日に一回は端末機操作を行って，情報を確認することが日常化している。従来の研究生活習慣を維持してもさしつかえないにもかかわらず，結果として，日常的に端末機を利用することになっているのは，それ自体の便利さおよび気楽さのゆえにほかならない。

2．2．1節で述べたuser communicationが良く利用されていることは，mailの交換，notefilesの利用の頻度から理解されるが，日本語を母国語とする人が，この流れの中に入って，英語

でmailを交換する場合，用件は果すことができるとしても，表現自体に楽しさを見出すような気楽さを感じることは稀であろう。UNIXを日本に直輸入しても，アメリカにおける日常的利用と同じレベルでの日常的利用がなされるとは考えにくい所以である。

当初から指摘されてきたように，文書処理の機械化・自動化という問題は，単なる技術上の問題ではなく，その国の言語習慣あるいは教育に深く依存する問題でもある。手書き文字を使用することが習慣であって，そのための教育に力をそそぎ，その成果を得ていることが，日本の場合，この問題に直面した場合には，マイナスの要因となって作用していることが，UNIXおよび関連するソフトウェアツールの機能と使用実態の検討からも結論することができる。

参考文献

1. "UNIX Programmer's Manual", Seventh Ed., Virtual VAX-11 Version, Univ. of California, Berkeley (1981).
2. James A. Deckert : "The UNIX System on the VAX Computer" Computer Research Laboratory, Univ. of Illinois at Urbana-Champaign (1983).
3. "UNIX Time-Sharing System", B.S.T.J., 57, No.6, Part2 (1978).
4. R. Thomas and J. Yates: "A User Guide to the UNIX System", McGraw-Hill (1982).
5. Joseph F. Ossanna: "Nroff/Troff User's Manual", Bell Laboratories Murray Hill (1977).
6. James Gosling: "UNIX Emacs", Carnegie-Mellon Univ. (1983).
7. Edmund DeWan: "Nroff/Troff Tutorial", Computer Service Office, Univ. of Illinois at Urbana-Champaign (1979).
8. Brian W. Kernighan and Lorinda L. Cherry: "Typesetting Mathematics User's Guide (Second Ed.)", Bell Laboratories Murray Hill (1979).