解　説

ベル研究所の軽装 OS——UNIX*

石　田　晴　久**

1. UNIX の開発経過

UNIX は，ベル研究所のコンピュータ・サイエンス部門で，PDP-11 シリーズ（主として 11/45）用に開発されたコンパクトな TSS オペレーティング・システムである．ベル研究所はかつて 1965 年から 1969 年にかけて MIT および GE と組んで，MIT の MULTICS 開発に参加したが，途中で MULTICS の OS が余りにも大きくなりすぎて，ユーザ・プログラムが走らないなど数多くの不満が出，MULTICS の完成を待たずに手を引いた．その後 MULTICS の方は MIT が GE（のちに Honeywell）とがんばっていいシステムに仕上げたが，ベル研究所の方では，MULTICS 流の巨大 OS への反省から，ミニコン用の OS に注目し，UNIX を開発したわけである．したがって UNIX の設計思想には MULTICS で開発されたアイデアが数多く入っており，UNIX はミニ MULTICS の性格をもっている．たとえば両システムとも，使用文字を大文字でなく小文字にするという珍らしい行き方をとっているが，大文字よりは小文字の方が人間にとっては読みやすく書きやすく，この点だけでも使い勝手のよいシステムを狙ったことがうかがえる．

さて UNIX の開発を始めたのは K. Thompson である．1969 年から 1970 年にかけて，PDP-7 や PDP-9，次いで 1970 年から PDP-11/20 用に単一ユーザ向けの UNIX が開発された．これでその有効性が実証されたらしく，1971 年 2 月からは D. M. Ritchie を中心として，PDP-11/40, 45, 70 について UNIX の開発が行われた．さらに 1973 年には，それまでアセンブラ語で書かれていた UNIX の大部分が新しい高水準言語 C で書き直された．今日では，UNIX はベル・システム内の 100 台以上の PDP-11 で使われ，外部でも，カリフォルニア大学（バークレイ）やウォータールー大学などあちこちで使われ，筆者の東大センターでもベル研と正式なライセンス契約を結び，ユーザに仲間入りしている．

広く使われていることからわかるように，UNIX は使いやすくてコンパクトな OS として非常に高い評価を受けている．筆者もベル研で試用して惚れこんだが，1976 年末に来日したウォータールーの E. Manning 教授も UNIX のよさを盛んに宣伝していった．UNIX には，現在 TSS 用の UNIX と，単一ユーザ用の MINI-UNIX があり，ベル研とライセンス契約*** を結べば使うことができる．UNIX が使える最小機器構成を**表-1** に示す，表でわかるように，UNIX は高級ミニコン用の OS である．2 人年という少ないマンパワーで開発された OS としても知られている．UNIX の評価の 1 例としては，**表-2**(次頁参照) にあげた Datamation 1976 年 12 月号 (p. 108〜139) に出たソフトウェア・パッケージのアンケートによる評価点が参考になる．UNIX の場合，表のでは回答数が少ないのでこの評価は余りあてになる数字ではないが，他の定評ある優良ソフトウェア・パッケージに劣らぬほど，UNIX が高い評価を受けているのは確かである．ただ表-2 にもあるように，ユーザ・サイドに対しては，実行システム，ソース・プログラム，マニュアルの入った磁気テープやカートリッジ・ディスクを提供して勝

表-1 UNIX に必要な最小機器構成

	UNIX	MINI-UNIX
CPU (PDP-11)	40 (EIA 付), 45, 70	11, 20, 40
コア (16ビット)	48 kW	28 kW
オプション	KT セグメンテーション	—
OS 領域	20-22 kW	12-16 kW
プログラム	リエントラント	リエントラントでない
言語 C コンパイラ	最少 14 kW	左に同じ
ディスク	カートリッジ/パック	カートリッジ
クロック	KW 11-L か -P	左に同じ
端末機(全2重通信)	DEC ライタなど	左に同じ

* UNIX——An easy-to-use operating system developed by Bell Telephone Laboratories, by Haruhisa ISHIDA (Computer Centre, University of Tokyo)

** 東京大学大型計算機センター

*** 大学の場合は磁気テープ代のみで $150. 民間会社の場合はウエスタン・エレクトリック社経由で $20,000 (600 万円).

表-2 他のパッケージとの比較からみた UNIX

評価項目	UNIX	WATFIV	SPSS	IMSL
用途	TSS用OS (PDP-11用)	教育用 Fortran	社会科学用統計計算	Fortran用数学ルーチン
発売元	ウエスタンエレクトリック	ウォータールー大	SPSS社	IMSL社
アンケート回答数	5社	11社	24社	18社
総合的満足感	4.0	3.8	3.5	3.5
スループット/効率	4.0	3.7	3.4	3.2
移植のしやすさ	3.4	3.4	3.4	3.4
使いやすさ	3.6	3.7	3.5	3.2
ドキュメント	2.6	3.3	3.7	3.6
技術サポート	2.0	3.1	3.1	3.5
ユーザ教育	1.0	3.8	2.7	2.5
価格	$20,000	$1,200	$750 -$5,000	$980 -$1,220/年

(注) 4=excellent, 3=good, 2=fair, 1=poor

手に使ってくれという形であるため，ユーザに対する面倒見は余りよくない．ベル研の外で UNIX を使っている有名なシステムとしては，カリフォルニア大バークレーの INGRES（リレーショナル・データ・ベース）やウォータールー大の WIDJET（多数端末のフロント・エンド）などがある．

2. UNIX の特徴

他のシステムに余りみられない UNIX の特徴をあげてみると，次のようになる．

（1） コマンド体系が一種のプログラム言語になっており，Shell と呼ばれるコマンド・アナライザが強力にできている．これにはユーザごとのインターフェースが設定できる（エディタの中にいきなり入る，など）．コマンドにはパラメータが多く（て困る面もあるが），多様な使い方ができる．また Shell を通してシステム全体のアカウント・データをとることもできる．

（2） ファイル・システムがハイアラーキになっており，非常に使いやすい．使用するディレクトリ（カタログ）も簡単に変えられる．

（3） コマンドは入力を出力に変換する一種のフィルタとしてとらえられており，端末・ファイル・プロセスにおける入出力が統一的に扱える．このため端末での入出力をファイルへの入出力に切り換えることがすぐできる．使用頻度の低いファイルを磁気テープに入れる archive 機能もある．

（4） ユーザ・レベルで非同期プロセスの起動が容易にできる．いくつかのコマンドを直列に実行させる場合に，あるコマンドの出力はパイプラインと呼ばれるファイル（小さいときはコア）に入り，次のコマンドへの入力となる．いくつかのコマンドの同時並行処理も容易に指定できる．

表-3 UNIX 特有のソフトウェア

言語Cのコンパイラ
TROFF …写植機用のフォーマット編集プログラム
EQN ……TROFF 用数式編集プログラム
TBL ……テーブルのレイアウトや内容を編集するための TROFF プリプロセッサ
MS ……原稿レイアウトのための TROFF プログラム
NROFF …一般端末機用ランオフ (runoff) プログラム・これに関連して NEQN（数式用）がある
TYPO …タイプミスを見付けるためのプログラム
INDEX …英単語のクロスレファレンスを作るプログラム
YACC …LR(1) にもとづくコンパイラ記述システム
TMG ……古典的トップダウン・コンパイラ・コンパイラ
M6 ……汎用マクロプロセッサ
FORM …可変部分をもつ手紙の作成プログラム
MERT …UNIX にもとづく実時間 OS

（5） テキスト編集（ワード・プロセッシング）の機能がきわめて強力である．QED という使いやすいテキスト・エディタがあり，Runoff を強化した清書プログラムもある．写植機がつなげるようになっており，数式の取り扱いを含む強力なテキスト編集プログラムが用意されている．ユーザ・マニュアルはすべてファイル化されている．

（6） いろいろなプログラム・ジェネレータがある．東工大や東大にも移植されている Ratfor (Rational Fortran) はじめ，表-3 に示したような YACC, TMG その他がある．

（7） コンピュータ・ネットワーク機能があり，ベル研究所の中では SPIDER ネットワークに接続されている．端末間の mail 機能も便利にできている．

（8） UNIX の大部分は高水準言語 C で書いてあり，保守や変更や勉強が比較的楽である．カリフォルニア大学などでは，UNIX のソース・コードを教材として使っている．

（9） 端末は大部分が 300 ボーの電話用端末で，全2重方式をとっている．したがって，コマンドをどんどん入れるタイプアヘッドや，パスワードをエコー(印字) させないなどの手がとれる．

(10) 使用文字は小文字をベースとしている．このため大文字に比べ，書きやすく読みやすいが，ネットワークで他の大文字ベースのシステムとつなぐときには，コード変換が必要となる．

3. UNIX のコマンド・アナライザ (Shell)

TSS のコマンド体系はシステムの使いやすさを左右する重要な要素である．UNIX では，使い勝手を

よくするため，コマンドは原則として小文字で表現し，ユーザによるキーインの手間を省くため，いろいろな省略表現（たとえば正規表現）を使うなど，非常によく工夫されている．またコマンド・アナライザとしてはユーザ番号ごとに任意のプログラムがパスワード・ファイル内に登録できるようになっており，TSS使用開始と同時にいきなりエディタに入る（そしてエディタしか使えない）とか，ゲームしかやれないユーザ ID を設けるといったことが可能になっている．

UNIX の標準的なコマンド・アナライザを Shell と呼ぶ．これは言語Cで書かれた約 20 ページのプログラムである．Shell では，コマンド名の扱いは次のようになっている．

（1） 実行可能なロード・モジュール名を表すとき，そのモジュールを新しいプロセスとしてすぐ実行させる．

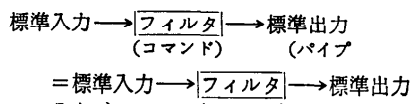
（2） コマンド・ファイルを指すとき，Shell 自体の子プロセスを (fork 操作により) 発生させ，コマンド・ファイルの中から順次コマンドを呼び出しては，それを実行させる．このファイル（名を file とする）を探すとき，UNIX ではファイルのディレクトリ (カタログ) が自由に選べるので，まず現在使用中のディレクトリの中で，次に /bin/file (システム・ファイル)，それでもなければ /usr/bin/file (ユーザ usr のファイル) について探索が行われる．コマンド・ファイルは単なる順編成ファイルである．

Shell で面白いのは，端末のキーボード (file 0) やプリンタ (file 1) と一般のファイルとが全く同格で使えることである．たとえば，次のような使い方ができる．

ls	(現ディレクトリ中のファイルの名前をリストせよ)
ls >there	(there というファイルを作り，その中にファイル・リストを入れよ)
ls ≫there	(リストを there ファイルに追加せよ)
ed	(QED エディタを呼べ)
ed <script	(エディタに対するコマンド群を script ファイルから読め)

したがって，コマンドの出力をいったんファイルに入れて，それをエディタで編集して出力するといったことが簡単にできる．この>と<の記号は，Shell 自身が解釈して標準入力 (file 0) と標準出力 (file 1) に指定されたファイルを割当ててくれるので，ユーザ作成の自家用コマンドの中では，単に file 0 から入力し，file 1 へ出力するものと考えておけばよい．

次にコマンドの連続（直列）実行については，Shell では，コマンドは



のように一種のフィルタであると考え，コマンド間で情報を伝送するファイル（情報が少ないときは主メモリ）をパイプライン・ファイルと呼ぶ．たとえば

ls|pr −2|opr

というコマンド行では，たて棒が直列実行を表し，

ls >temp1	(ファイル・リスト temp 1 へ)
pr −2 <temp1 >temp2	(temp 1 の内容を2カラムに分けて temp 2 へ)
opr <temp2	(オフライン (バッチ) でプリント)

という命令列と同じはたらきをする．こうしたフィルタおよびパイプラインの考え方は，コード変換，ソート，暗号化・復号化，出力編集などを行うときにはとくに大きな威力を発揮する．ただし UNIX でもすべてのコマンドがフィルタとなるわけではない．

コマンドはパイプラインでつながらないときには，; で切って一行にいくつでも書ける．もうひとつの区切り記号としては，平行処理（マルチタスキング）のための & 記号がある．これは & の前のコマンドの終了を待たずに (wait 操作をせずに)，次のコマンドを並行して実行せよという意味である．たとえば，

as source >output & ls >filelist &

は，「source ファイルをアセンブルして，output ファイルに出力を出す一方で，同時にファイル名リストを filelist に入れよ」という意味になる．また

(date; ls)>x & as source &

は，「ファイルxにまず現在日時，次いでファイル名リストを出力し，その間にアセンブル結果を端末にプリントせよ」という意味になる．

Shell はまたそれ自身がコマンドで，その再帰呼び出しが可能である．たとえば

sh <commandfile

というコマンドは，commandfile 中のコマンドを Shell に実行させよという意味で，この sh コマンドはもち

ろん Shell で解析される．Shell ではコマンドのパラメータの書き方に正規表現が許される点も面白い．正規表現で使われる記号は次の通りである．

?	任意の1文字
*	(／以外の) 任意の文字列 (null も可)
[…]	セット中の任意の1文字
＼	エスケープ (\new-line は空白として扱われるので，複数行からなるコマンドが可)

これらの記号を使うと，たとえば [af]*.s は a.s, f.s, file.s, first1.s などのいずれにもマッチすることになり，マッチしたパラメータ（n番目のを $\$n$ で表わす）はアルファベット順にソートされて，コマンド・パラメータとしてプログラムに渡される．次に示すのはそうしたコマンド・プログラムの1例 (loopdump) である．

```
: loop
if "$1"="  " exit   (空なら飛び出す)
echo $1 $2 $3 $4 $5 $6   (印字する)
shift    (パラメータを1個分左へずらす)
goto loop
```

これを loopdump *a b c* というコマンドで呼び出せば，プリント結果は

a b c
b c
c

のようになる．この例が示すように UNIX のコマンド体系は完全なプログラム言語(goto のある)である．

4. UNIX ファイル・システムの構造

UNIX のファイル・システムは，非常に使いやすく設計されている．以下のような特徴の一部〔(4)など〕は GCOS や ACOS などにもみられるが，一応あげておく．

(1) ファイルの構造はシステムで制御するのではなくプログラムで制御する．主に使われるファイルはソース (ASCII)・ファイルと2進（オブジェクト）・ファイルである．

(2) ファイルのサイズは伸縮自在で，普通ユーザはサイズの指定をする必要がない．

(3) 他システムでのカタログに対応するツリー状のディレクトリがあり，現用（カレント）ディレクトリの概念がある．このためファイルには次のような指定ができる．

/alpha/beta/gamma	システムのルート(1)から始まるパス名
alpha/beta	現用ディレクトリ中のサブディレクトリ alpha の中の beta
alpha	現用ディレクトリ中のファイル
.	ディレクトリ自身
..	ディレクトリの親（上位ディレクトリ）

UNIX のコマンドの中には，change directory というのがある．たとえば，上の例で，

chdir alpha

とやると，現用ディレクトリが alpha に変るので，ファイル名は alpha/beta といわなくても，単に beta といえばよくなる．これは複雑なディレクトリをもつファイルや他人のもつファイルを(許可をえて)使う(ディレクトリ間を渡り歩く)場合に非常に便利である．

(4) コマンドやプログラムの中でファイル名が指定されたとき，もしもそのファイルがまだ存在していなければ，その名前のファイルがその場で作られる．したがって，ファイルはあらかじめ作っておく必要はない．ただ反面，既製のファイルの名前をスペリングをまちがえて指定したりすると，そのまちがった名前のファイルが作られてしまうという危険もある．またファイルが余りにも作りやすいため，各ユーザがついファイルを作りすぎるという問題もある．

(5) 入出力装置を特殊ファイルと同じに扱う．これはある程度は他システムでも行われているが，UNIX のはかなり徹底している．入出力機器はシステムの /dev ディレクトリに登録されており，紙テープ・パンチなら /dev/ppt のようなファイル名となる．こうした扱いのためすでにのべたように，

```
pr          (このコマンドへの標準入力を印字)
pr <file1 >>file2
```

の使い分け（後者はファイル1の中味を印字形式に直し file2 に追加せよの意）ができる．

(6) ファイル保護は，自分および他のすべてのユーザに対し，read, write, execute のいずれを許すかという単純な機構で行われる．このほか，execute に関連しては，もしオンであれば，当のユーザ ID をそのプログラム・ファイルの所有者のユーザ ID に切り換えるというビットがある．つまりプログラムの所有者がこのビットをオンにしておいてくれば，ユーザはその所有者の資格でそのプログラムが実行できる（その中で所有者のもつ他のファイルをアクセスしたりできる）というわけである．

一方，プログラムの中では，それを使うユーザのIDを知るルーチンが使えるから，それでわかるユーザIDを調べて，いろいろなアクセス制御を行う（あるユーザにはあるファイルをいじらせないなど）ことが可能となる．

UNIXにおける（伸縮自在な）ファイル領域の割当ての仕方を**図-1**に示す．ディレクトリから指される各ファイルのインデックス・リストの中には，所有者ID，ファイル・サイズ，保護の8ビット，リンク数（いくつのディレクトリから指されているかを示す）などの情報のほかに，8個のアドレス項目（各2バイト）がある．一方，ディスク領域の方は，512バイトのブロックに分かれていると考え，図-1(a)のようにファイルの大きさが4kB以下のときは，8個のアドレスで8ブロックを直接指定する．しかしより大きなファイルについては，(b)のように，512 Bのブロックに256個のアドレスが入ることを利用して，間接アドレスおよび2重間接アドレスを使い，合計で(7+256)×256×512バイトまで指定できる（実際有効なのは16 MBまで）ようにしている．ブロックのガーベジ・コレクションはもちろん別に行われる．

5. UNIX のテキスト処理系

ユーザからみたときのUNIXの大きな特徴はテキスト処理（ワード・プロセッシング）系が充実していることである．とくに低価格であるが割合いよいオンライン写植機（グラフィック・システム社製*，**表-4**参照）がよく使われている．これを使ってKernighanがプログラム書法やソフトウェア・ツールの本を出版しているほか，ソフトウェア工学会議などの報文集の中で，ベル研からUNIXで編集して出された論文は刷り上がりがきれいなことで光っている．とにかく驚くべく安いコストでオンライン写植機を使って論文を書いたり，本を作ったりしているのは注目に値する．

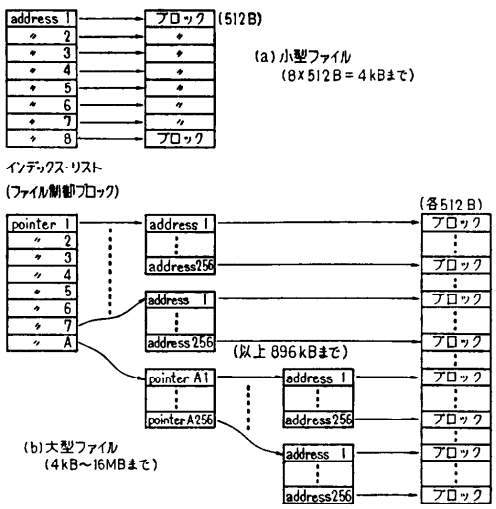


図-1 UNIXにおけるファイル領域割当て法

表-4 グラフィック・システム1の概要

活字フォント(フィルム)	4種（ターレット・レンズ切換え）
字　種	102字/フォント
ポイント・サイズ	6,7,…,24,28,36 (15種)
ピッチ	1/6 ポイント （1ポイント=1/72 インチ=0.35 mm）
行の長さ	0～47 パイカ （20 cm，1パイカ=1/6 インチ）
スピード	50 行/分（8ポイント，11パイカのとき）
紙　幅	2,3,4,6,8 インチ (20.3 cm)

この写植機を使うためのソフトウェアとしては，いわゆるRunoff（清書）プログラムを拡張したTroffがある．さらに数式の多い論文の編集にはKernighanの作ったeqnと呼ばれるパッケージがある．これを使って，たとえば

$$f(t)=2\pi\int \sin(\omega t)dt$$

と印刷したいときには

$$f(t)=2 \text{ pi int sin (omega } t)^{\sim}dt$$

のように指定すればよい．

さらに複雑な例を**図-2**に示す**．こうした複雑な指定をしたり，一般に論文や手紙やマニュアルや本の

$$\int \frac{dx}{ae^{mx}-be^{-mx}}=\begin{cases}\dfrac{1}{2m\sqrt{ab}}\log\dfrac{\sqrt{a}e^{mx}-\sqrt{b}}{\sqrt{a}e^{mx}+\sqrt{b}}\\ \dfrac{1}{m\sqrt{ab}}\tanh^{-1}\left(\dfrac{\sqrt{a}}{\sqrt{b}}e^{mx}\right)\\ \dfrac{-1}{m\sqrt{ab}}\coth^{-1}\left(\dfrac{\sqrt{a}}{\sqrt{b}}e^{mx}\right)\end{cases}$$

（a） 印刷すべき数式

```
define emx ""{e sup mx}""
define mab ""{m sqrt ab}""
define sa ""{sqrt a}""
define sb ""{sqrt b}""
int dx over {a emx −be sup −mx}~=~
left {lpile}
    1 over {2 mab{~log~
        {sa emx−sb} over {sa emx+sb}
  above
    1 over mab~ tanh sup −1 (sa over sb emx)
  above
    −1 over mab~ coth sup −1 (sa over sb emx)
```

（b） 上記数式の指定法

図-2 数式編集における指定の例

* 日本ではハイゼ商会（電話 03-402-8291）で扱っている．価格は$12,000（数百万円）から．

** 木村：カーハン氏とオンライン写植機，bit, Vol. 8, No. 5, pp. 414～420 (1976)

原文をオンラインで作るには，使いやすいエディタが必要である．UNIX には QED 系のエディタがある．QED 型テキスト・エディタの特徴は文字列の指定に正規表現が使えること，「ある文字列を含む行」という指定ができること，ファイルの扱いが簡単なこと，QED のコマンド体系が一種のプログラム言語になっていて，コマンド群からなるサブルーチン的なものが定義できること，などである．QED で許される正規表現やコマンドの例を次に示す．

/^first/	文字列 first で始まる文字列
/last$/	文字列 last で終る文字列
/*x* . *y*/	*x* と *y* の間に任意の 1 文字のある文字列
/*a* . * *z*/	*a* と *z* の間に任意の文字列を含む文字列
/*a* . * *z* $/	*a* で始まり，まん中に何文字かあり最後が *z* である文字列
/*a*1/*s*/*b*1/*c*1/*p*	*a*1 を含む行の中で *b*1 を *c*1 に置きかえ (substitute)，結果をプリントせよ
//*s*///	前に指定した文字列を含む次の行の中でその文字列を消せ

この QED は UNIX の中でもっともひんぱんに使われるプログラムであり，最近はソフトウェアのドキュメンテーションにもよく使われている．

6. ネットワーク機能とワークベンチ的利用

UNIXにはインハウス・ネットワークの機能がある．これを利用して，マレーヒルのベル研内では SPIDER と呼ばれる一種のデータハイウェイ的ループ・ネットワークに対して，数台の UNIX が接続され，他のミニコンや大型機との間で，周辺機器の共用やデータ交換が行われている．基本通信速度は 1.544 メガビット/秒である．このループは論理的には，64 個のスロット (チャネル) に分れたベルト・コンベア状になっており，そこに長さ 386 ビット (うち有効情報 256 ビット =32 バイト) のパケットが毎秒 4,000 個の速さで投入され，伝送される．

この SPIDER ネットワークとは別に，他の UNIX システムの中には，ワークベンチとして IBM 370 などの大型機にデータ通信線で結ばれて使われているものがある．ここでワークベンチ (work-bench, 仕事台) という概念は，ソース・プログラムの入力と編集，ファイルの保守，ソフトウェア開発工程管理上の処理など，要するに，大型機でしかできない仕事を除く他の準備作業や雑用は一切ミニコンでやるという考え方をさす．プログラムを作成してランさせる場合であれば，ソース・プログラムの編集は UNIX の QED などで行い，完全なソースができたら大型機に転送し，コンパイル・ラン後の結果は，UNIX に返送させて出力するという形がそれである．ウォータールー大学では，100 台程度の CRT ディスプレイをつないだ UNIX を IBM 370 に 9600 ボーの回線で接続し，ワークベンチ的に使っている．

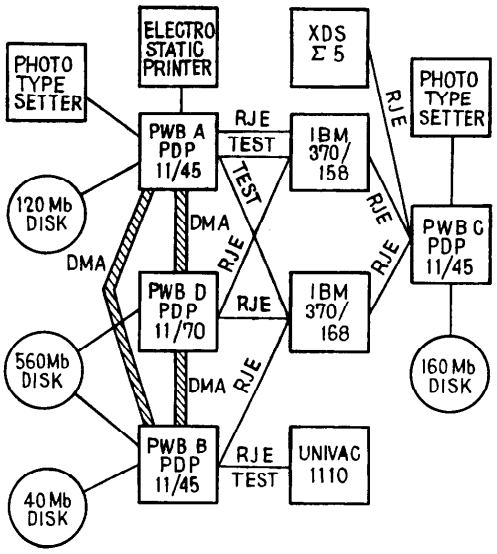


図-3 ワークベンチを含むネットワーク構成の例

ワークベンチとして UNIX を使うときに，ものをいうのは，すでにのべた UNIX のさまざまな特徴である．**図-3** にワークベンチ (PWB) を含むネットワーク構成の実例を示す．図には写植機（フォトタイプセッタ）や静電プリンタが含まれており，各種ドキュメントが能率よく素早く作成できるようになっている．

7. プログラミング言語 C

UNIX の主力言語 C は，BCPL 言語を参考にして設計されたインタプリタ言語 B をさらにもとにして D. M. Ritchie によって設計されたコンパイラ言語である．一見して Algol 風の小文字ベースの言語で，システム・プログラミング向き，構造的プログラミング向きになっている．データ型としては，文字，整数，実数(単・倍精度)，ポインタがあり，配列，関数，構造体 (structure) が使える．この C 言語はとくに際立ってすぐれた言語とはいえないかもしれないが，いく

```
main ( ){char c[5]; int m,n;
    struct jfile {int a[4]; char b[12]; ifile[200];
  printf ("hellow/n");
  *c='z';
      for (n=0; n<11; n++) {for (m=0; m<11; m++) ifile [n]. b[m]='a'+n+m;
        for (m=0; m<4; m++) ifile [n]. a[m]=n+m;
printf ("%d %d %d %d/n", ifile [n]. a[0], ifile [n]. a[1], ifile [n]. a[2],
            ifile [n]. a[3]);
       printf ("%s %c/n", ifile [n]. b, ifile [n]. b[1]);}}
```

```
hellow
0 1 2 3
abcdefghijk b
1 2 3 4
bcdefghijkl c
2 3 4 5
cdefghijklm d
3 4 5 6
```

実行結果の一部

図-4 言語Cによるプログラムの例

つかの特徴がある．図-4 にCによるプログラムの例を示す．その中で使われている記号の意味は次の通りである．

$n++$	$n=n+1$ と同じ．$n--$ も可．$++n$ なら先に増やしてから演算．
ifile [n]. $b[m]$	jfile で定義された構造をもつ配列 ifile の n 番目の要素中の $b[m]$
$\%d$	10進数に変換
$\%s$	文字列として処理
$\%c$	1文字として処理
$\backslash n$	復帰改行 (new line)
$*c$	c のポインタ（$*$は間接指定）

これらのほか，キーワードとしては，register, break, else, do, while, case, switch, default などがある．また演算子としては ≫ と ≪ で表されるシフトや多数の2進演算子がある．代入文では，

$$a=a+b \rightarrow a=+b$$

などの略記法も許される，cの強みは，UNIXやCコンパイラ本体を始めとして，cで書かれたソフトウェアが多いこと，コマンドに対応する数多くのシステム・サブルーチンが用意されていることである．

8. 使いながら設計されたシステム

最後に，UNIX を使ってみて，他のシステムとの比較でいえることは，UNIX は，MULTICS の設計哲学とそれへの反省を強力なバックボーンとして，ユーザの立場から，すなわち設計者自らが使い手となって設計が進められ，使いやすさを主眼としたシステムになっていることである．さらにそれほど大きなシステムではないため，少数の設計者の設計方針が貫かれ，決して委員会や大組織による設計のシステムではないということである．こうしたことは，設計・開発陣とユーザ側とがはっきり分かれてしまい，設計者が自分では使ってみない大多数のシステムでは大いに反省すべきことであろう．

以上本稿では UNIX のよい点を中心に紹介した．もちろん UNIX は決して理想的なシステムとはいえないが，マン・マシン・インターフェースのよさが中心課題となる TSS の設計にあたって参考になる面を多くもつシステムである．筆者自身も手許にある TSS のコマンド体系やテキスト編集処理系の設計には UNIX から多大のヒントをえている．終りに UNIX に接する機会を与えて頂いたベル研究所の藤村靖氏や Peter Denes 氏を始めとする関係者の方々に謝意を表したい．

〔UNIX に関する文献〕

1) D. M. Ritchie & K. Thompson: The UNIX time sharing system, CACM, Vol. 17, pp. 365～375 (1974)
2) D. M. Ritchie & K. Thompson: UNIX programmer's manual, Bell Labs (1975)
3) B. W. Kernighan: UNIX for beginners, Bell Labs (1974)
4) UNIX summary, Bell Labs (1976)
5) B. W. Kernighan: A tutorial introduction to the UNIX text editor, Bell Labs (1974)
6) B. W. Kernighan: Programming in C－A tutorial, Bell Labs (1975)
7) D. M. Ritchie: C reference manual, Bell Labs (1975)
8) B. W. Kernighan & P. J. Plauger: Software tools, Proc. of 1st Nat. Conf. on Software Engineering, IEEE, pp. 8～12 (1975)
9) B. W. Kernighan & P. J. Plauger: Software

tools, Addison-Wesley (976)
10) B. W. Kernighan & L. L. Cherry: A system for typesetting mathematics, CACM, Vol. 18, No. 3, pp. 151～156 (1975)
11) J. F. Osanna: NROFF User's Manual, Bell Labs (1974)
12) K. Thompson: The UNIX Command Language, Infotech, pp. 375～384 (1975)
13) T. A. Dolotta & J. R. Mashey: An introduction to the programmer's workbench, Proc. of 2nd Int. Conf. on Software Engineering, IEEE, pp. 164～168 (1976)
14) J. R. Mashey: Using a command language as a high-level programming language, Proc. of 2nd Int. Conf. on Software Engineering, IEEE, pp. 169～176 (1976)
15) J. R. Mashey & D. W. Smith: Documentation tools and techniques, Proc. of 2nd Int. Conf. on Software Engineering, IEEE, pp. 177～181 (1976)
16) T. A. Dolotta et al.: The LEAP load and test driver, Proc. of 2nd Int. Conf. on Software Engineering, IEEE, pp. 182～186 (1976)
17) B. W. Knudsen et al.: A modification request control system, Proc. of 2nd Int. Conf. on Software Engineering, IEEE, pp. 187～192 (1976)
18) M. H. Bianchi & J. L. Wood: A user's viewpoint on the programmer's workbench, Proc. of 2nd Int. Conf. on Software Engineering, IEEE, pp. 193～199 (1976)
19) D. L. Bayer & H. Lycklama: MERT—A multi-environment real-time operating system, ACM SIGOPS Conf. (1975)
20) A. Snyder: A portable compiler for the language C, MAC TR-149, MIT Project MAC (1975)
21) 土井: ウォータールー大学, bit, Vol. 8, No. 12, p. 1217 (1976)
22) G. D. Held, M. R. Stonebraker & E. Wong: INGRES—A relational data base system, Proc. of NCC, Vol. 44, pp. 409～416 (1975)
23) S. C. Johnson: YACC-Yet another compiler-compiler, Bell Labs (1975)
24) B. B. Bunt: Scheduling techniques for operating systems, IEEE Computer, Vol. 9, No. 10, p. 16 (1976)
25) B. W. Kernighan and P. J. Plauger: Software tools, Addison-Wesley (1976)
26) A. D. Hall: The M6 macroprocessor, Bell Labs (1969)
27) A. G. Fraser: A virtual channel network, Datamation, pp. 51～56 (February 1975)

(昭和52年4月12日受付)

〔追記〕

本稿を書いた後，1977 年度全米コンピュータ会議 NCC で次の発表があった．

H. Lycklama: UNIX on a micro-processor, Proc. of NCC, Vol. 46, pp. 237-242 (1977)

これはマイクロコンピュータ LSI-11 用の UNIX で，MINI-UNIX の機能をさらに落して，OS 部分を 8 キロ語にし，C コンパイラに必要な 12 キロ語と合わせて計 20(ないし 28)キロ語のメモリと，フロッピー・ディスク 1～2 台のシステムで動くようにしたものである．PDP-11 と互換性のある LSI 11 は，アメリカの大手キット・メーカーであるヒース社から，ヒース・キット H 11 として，基本価格 $ 1295 で発売され始めた．したがって，PDP-11 用の UNIX のようにすぐれたソフトウェアは，今後は H 11 を通して，ホビイスト・レベル（個人レベル）でも使われる可能性が強くなっている．

(昭和52年8月23日)