解　説

UNIXの概要とその設計思想†

斎　藤　信　男††

1. UNIX の設計思想とその特徴

UNIX は，使いやすい会話型システムとして設計され，現在多くのユーザに使われている．その起源はMIT で開発された Multics にさかのぼることができるが，UNIX の開発者の一人である D. Ritchie が ACM の Turing 賞の授賞講演[1] で述べているように，Multics に比べて UNIX は非常に低コストで使いやすい環境を実現できることがその特徴であった．したがって，時分割システム（TSS）の歴史を考えてみれば，TSS の第一世代である CTSS でその可能性を試し，TSS の第二世代である Multics でその徹底的な拡大を図ったのに対して，TSS の第三世代である UNIX でその普及を図ったことになる．たとえば，UNIX の一つの使い方であるソフトウェア開発環境の発展経緯を見ると，我が国のソフトウェア業界においては UNIX の出現により会話型環境の構築が初めて一般化したと言えるであろう．

開発者のもう一人であり Ritchie とともに Turing 賞を受賞した K. Thompson によれば，UNIX の設計・開発の最初の動機は，誰も使っていなかった PDP-7 に対してプログラマにとって使いやすく満足のいく計算機環境を実現したいという彼の個人的，自己満足的な要求であったという[2]．Bell 研究所という少なくとも形式的には一つの企業に属する組織において，そのような形態の研究が許されていたことに一つの感慨を覚えるが，文献 2) で述べているように他人の要求でなく開発者自身の要求に従ってシステムを設計し改良していくことが許されたことに，UNIX の成功の鍵があった．このような最初の設計目標をまとめると，つぎのようになる．

（1） 使いやすい会話型システムを低コストで実現すること．

† UNIX and Its Design Philosophy by Nobuo SAITO (Department of Mathematics, Faculty of Science and Technology, Keio University).

†† 慶應大学理工学部数理科学科

前述したように，Multics で考えられたユーザの使い勝手の良さを低コストで実現することが UNIX の最大の目標である．計算機の運用が，バッチ処理から会話型処理になってユーザの使いやすさは大幅に向上したけれども，システムの手軽さ，柔軟さ，透明度などの観点から見て，ユーザに本当に使いやすいものはなかなか実現されていなかった．特に，ソフトウェアの開発を担当するプログラマにとって，プログラムの記述，テスト，実行といった作業を満足して行える環境の実現を目指した．また，システムを低コストで実現するために，限られたハードウェアの規模や性能の制約のもとで目標とする機能を実現するように，設計に工夫をこらした．

（2） システムの構成，機能などを統一化してユーザから理解しやすくすること．

ユーザの使い勝手を向上させるためには，ユーザにわかりやすいシステムの機能や構造を提供しなければならない．そのためには，オペレーティングシステムの複雑な様相をなるべく少数の概念構成を使って統一的に表現し，簡単に操作や処理ができるようにする．ファイルの概念の統一化，簡略化はその良い例である．

（3） ユーザのカスタマイズが柔軟に行える仕組みを提供し，ユーザからのフィードバックによりシステムの成長をはかっていくこと．

オペレーティングシステムは，従来システムに固定した形でメーカから提供されてきた．ユーザの特別な要求は，メーカによってカスタマイズされることになっていた．UNIX では，ユーザの要求をユーザ自身でシステムに変更を加えてカスタマイズすることを期待されており，そのための仕組みをシステムとしてできるだけ用意している．たとえば，ソースコードにオンラインでいつでもアクセスできるようにし，簡単に新しいシステム生成ができるようにしている．そのようなユーザからのフィードバックをシステムの改良に反映させれば，よりよいオペレーティングシステムへ

成長させることができる．システムのソースコードを公開しているのは，この方針に基づいているからであろう．

上記のような設計思想に基づき開発された UNIX システムは，その後いくつかの変遷を経て現在のような状況になっている．その特徴は，つぎのような三つの視点から整理できる．

●技術的特徴

（1） 使いやすいファイルシステムを提供する．

（2） 使いやすいプロセスシステムを提供する．

（3） ユーザコマンドは，コマンドシェルで一括して解釈する．

（4） プログラムの組合せによる新しいユーティリティの開発が容易に行える．

（5） 行入出力型の端末に対して柔軟なインタフェースを持っているので，多くの種類の端末を支援することができる．

（6） 柔軟なネットワーク機能のインタフェースを持つ．

（7） ファイルの性能および機密保護に問題がある．

（8） 実時間システムの処理には適さない．

（9） 行入出力，文字入出力の端末機器に向いており，ビットマップディスプレイなどの機器には向いていない．

●システムの運用上の特徴

（1） システムが高級言語で記述してあり，公開されている．

（2） システムの運用や保守は，スーパユーザがオンラインで行う．

（3） 玄人向きのシステムであり，エンドユーザ向きにはできていない．

●社会的意味としての特徴

（1） ソフトウェアの流通の基盤として優れている．

（2） 標準のオペレーティングシステムとして採用することができる．

これらの特徴の中で，技術的特徴の(7)，(8)，(9)，運用上の特徴の(3)はその短所とも考えられる．これらは，その長所とちょうど裏腹の関係にある．たとえば，一般的，統一的なファイル構造を採用しているので，その性能は必然的に低下する．また，ユーザにオペレーティングシステムのカスタマイズを大幅に許してそのフィードバックをシステムの成長の基礎としているので，アプリケーションを能率よく実行することを最大の目的とするエンドユーザにとってはシステムの使用にわずらわしさを感じることがでてくる．

これらの欠点は，いくつかのシステムでその改良が試みられており，それらの実績もあがっている．たとえば，ファイルシステムの性能の改良が Berkeley バージョンや AT & T バージョンで行われている．また，実時間用の UNIX システムの開発や商品化が行われている．ビットマップディスプレイに対する制御システムやインタフェースは最近のシステムの最大の課題であり，SUN マイクロシステムズの SUN View[3] や SUN NeWs, MIT で開発した X ウィンドウ[4]，CMU の ITC プロジェクトで開発した ANDREW システム[5] など多くのウィンドウ管理システムの開発が UNIX 上で行われている．

以下には，UNIX の機能や構造の特徴について詳しく説明し，その利用場面についていくつかの考察を加え，また UNIX システムの将来の動向についてその特徴との関連性について述べる．

2. オペレーティングシステムとしての機能と構造

2.1 UNIX の基本構造[6),7)]

UNIX は，時分割用オペレーティングシステムである．オペレーティングシステムは，ユーザと計算機ハードウェアとの間に介在し，ハードウェア資源に対するユーザの多重の要求を管理してそれらの要求を満足させる．通常，複雑なシステムを設計し記述するためには，そこに構造を見いだすように整理する方法が採られる．オペレーティングシステムでよく見られる方法は，ユーザからハードウェアの間に階層構造を設けることである．UNIX においても，機能や実現の仕方から整理すると，次のような階層が見られる．

（1） ユーザインタフェース層

（2） ユーティリティ層

（3） ライブラリルーティン層

（4） カーネル層

一方，UNIX はユーザの要求を容易にシステムに組み込めるような機能を用意している．それらは，上記の階層構造に対応して，それぞれ整備されている．これらをまとめて図示すると，図-1 のようになる．

この階層構造を最上位のレベルから使う立場を，UNIX ユーザという．これに対して，システムを改

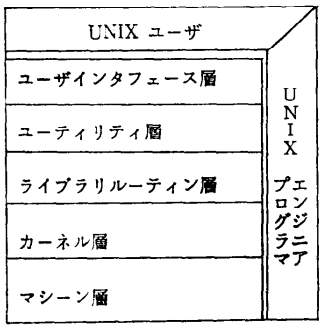

図-1 UNIX の構造(B)

修し成長させていく立場を，UNIX エンジニアあるいは UNIX プログラマという．UNIX は，そのエンジニアあるいはプログラマによって常に成長しているシステムとも考えられる．ユーザとエンジニアあるいはプログラマは同一人であってもよく，その仕事の要求によって立場が異なってくるだけである．

ユーザの立場から見て，これらの階層構造の各層のインタフェースは，あらかじめ決められたインタフェース言語によってアクセスされる．それらは，次のようになる．

（1） ユーザインタフェース層に対し……シェルコマンド

（2） ユーティリティ層に対し……おのおののサブコマンド

（3） ライブラリルーティン層に対し……サブルーティンコール

（4） カーネル層に対し……システムコール

（5） マシーンに対し……マシーン言語

一方，UNIX エンジニアがシステムを成長させていくときには，それぞれの階層に対してやはりインタフェース言語を使う．これは，次のようになる．

（1） ユーザインタフェース層に対し……コマンドスクリプト

（2） ユーティリティ層に対し……各ユーティリティのカスタマイズ言語

（3） ライブラリルーティン層に対し……C言語

（4） カーネル層に対し……C言語

一方，オペレーティングシステムをユーザに対するサービスを提供するシステムと見なせば，その機能はそのサービスを実現するために用意されていると考える．これにより，オペレーティングシステムの機能が明確になる．次に UNIX で提供されているサービスを列挙する．

（1） プロセス機能のサービス

（2） ファイル機能のサービス

（3） 記憶割り当てのサービス

（4） ユーザインタフェース機能のサービス

（5） 端末制御機能のサービス

（6） 周辺装置へのアクセス機能のサービス

（7） ネットワーク機能のサービス

（8） システム管理機能のサービス

UNIX ユーザから見れば，これらのサービス機能が横断的に提供されており，そのサービスごとに階層構造が存在していると考えられる．

2.2 UNIX の基本サービス

UNIX の基本的な機能，あるいはユーザから見れば基本的サービスについて，特に重要なものについてその概略を説明する．

（1） プロセスの機能[8]

UNIX における重要な資源としてプロセス (process) とファイル (file) がある．プロセスはプログラムの実行をする実体であり，ファイルはあらゆる情報を格納する実体である．両者を巧みに利用すれば，UNIX の上で効率のよい応用ソフトウェアが実現できる．両者は，見かけ上の性質が異なるので同一に論じることはできないが，密接な関係を持っている場面もあり，それらをよく理解しておくことが重要である．

プロセスは，独立な制御を持ったプログラム実行のための実体で，それはユーザのみならずシステムの機能の実現のためにも使われている．カーネル層から上位にプロセスは存在し，ユーザとシステムとの区別なくプロセス機能のサービスが提供されている．したがって，カーネル層に変更を加えずにシステムの機能を実現しようと思えば，システムのためのプロセスを使えばよい．このようなプロセスの中には，システム稼働時には必ず存在していなければならないものがあり，それらのプロセスを特にデーモンプロセス (daemon process) と呼ぶ．ネットワークのユーティリティを実現するためには，このようなデーモンがよく使われている．

プロセスは，図-2 に示すような要素から構成されている．

- 独立したプロセッサ利用のためのデータベース
  プロセッサ状態語，各種レジスタなど．
- 独立した記憶空間
  異なったプロセス間には，基本的に共通のアドレス

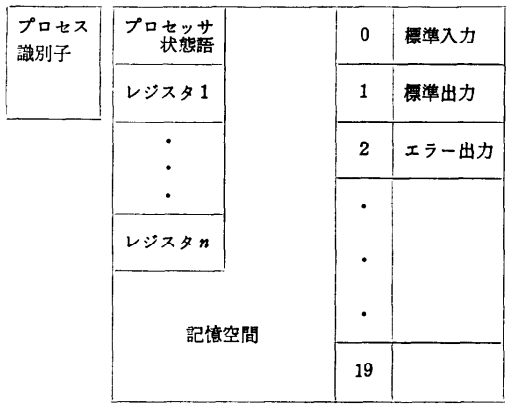

図-2　プロセスの構成要素（A）

空間は存在しない．

●独立した入出力チャネル

ファイルのアクセスのためのファイルデスクリプタを，プロセスごとに 20 個まで使用できる．特に最初の三つのデスクリプタは，特定の入出力用（標準入力，標準出力，および標準エラー出力）に確保されている．

●システム内で一意的な識別子

プロセスの管理のためにプロセス生成時に整数値の番号を与える．

プロセスの機能には次のようなものがあり，そのためのシステムコール，ライブラリコール，あるいはシェルコマンドが提供されている．

●プロセスの生成および消滅

プロセスは基本的に親子関係により管理している．子プロセスを生成する fork，自分自身を消滅させる exit が基本機能である．fork により生じた子プロセスは，親プロセスの入出力チャネルなどの属性を引き継ぐ．

●プロセスにおけるプログラムの起動

プロセス上で新しいプログラムの実行を開始するために，execute がある．execute を実行すると，指定したファイル内の実行形式プログラムがプロセスにロードされ，全く新しいアドレス空間に切り替わる．

●プロセスの同期

子プロセスが消滅する時期を待つ wait が用意されている．

●プロセス間通信

プロセス同士で情報の交換をすることは，複数個のプロセスを使ったプログラムにとって重要な機能である．UNIX では，いろいろのレベルで情報の交換機能が用意されているが，システムのバージョンごとに異なっているので，プロセスの機能の中では，最も標準化の遅れているところである．

標準的なプロセス間通信として pipe 機能があり，ユーティリティの結合による機能の追加などによく利用される．また，4.2 BSD でネットワーク機能の中で提供された socket 機能[9] はプロセス間通信として優れたものである．また，System V では，共有記憶と semaphore 機能とを提供しており，それにより，プロセス間の情報交換が効率よくできる．また，System V Release 3.0 で標準的に提供されている stream I/O[10] を利用すれば，より一般的なプロセス間通信が可能となるであろう．

●プロセスへの割り込み機能

プロセスの実行を中断する機能は，暴走しているプログラムを停止するために必須の機能である．割り込み処理のルーティンを登録する signal，割り込み信号を送る kill がユーザに提供されている．

●プロセスの状態の検索

システム全体，あるいはユーザごとに各時点において存在しているプロセスとその状態とを検索し，それを表示するコマンド ps が用意されている．ユーザは，これを使って，プロセスに関する情報を得る．

（2）　ファイルの機能[11]

ファイルは情報の格納の実体として重要な役割を果たす．UNIX のファイルシステムは，なるべく簡潔な構成を採用し，ユーザの使いやすさ，システムの管理のしやすさ，ファイル空間の効率のよい利用などを目指している．

●UNIX のファイルの型

UNIX においては，次の三つの型のファイルがある．

◎ノーマルファイル

通常のファイル．ユーザから見たとき，UNIX のノーマルファイルはすべてバイトストリーム（byte stream）と見なせる．

◎ディレクトリファイル

ファイル管理用のファイル．

◎スペシャルファイル

ハードウェア機器．これには，ブロック型およびキャラクタ型の二つの種類がある．

●ファイルの管理

UNIX においては，すべてのファイルを木構造の

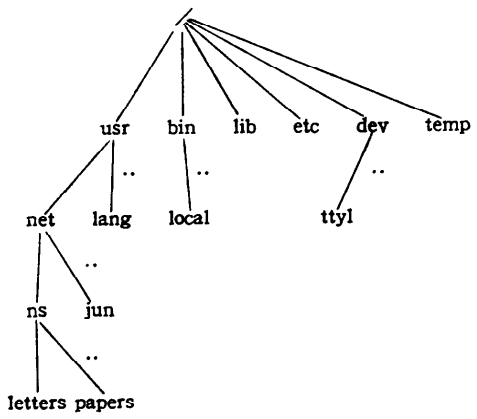

図-3 木構造ファイル(B)

ディレクトリを使って管理している．この木構造の深さには制限はない．木構造内の各節点に対してそれぞれ名前付けができ，それが木構造の同列（兄弟）内で一意的であるならば，根から各節点へのパスを表す名前はシステム全体で一意的になる．この合成した名前を，パス名 (path name) という．木構造とそのパス名とにより，柔軟で使いやすいファイルの管理が実現できる．

また，ファイル機器を表すスペシャルファイルも木構造のディレクトリに含ませている．これにより，ファイルの統合的な操作と管理とが可能となる．木構造ディレクトリの例を，図-3 に示す．

● ファイルの機密保護

木構造ディレクトリによりユーザごとに勝手な名前付けをしても混同を避けることができる．UNIX では公開を原則としているので，このパス名を使えば，木構造のどれでも指定できる．そこで，ファイルの機密の保護機構を用意しておく必要がある．

UNIX では，ファイルごとにアクセス権を設定できる．一般に，ファイルの所有者，同一のグループのユーザ，および一般のユーザの三種類に分け，それぞれに対して読み出し，書き込みおよび実行のアクセス権を設定できる．これに従い，プロセスからファイルアクセスの操作の要求が示されたときに，そのプロセスの所有者によって要求が許されるかどうかのチェックを行う．

● ファイルの操作

ファイルの操作は，シェルコマンドのレベルの操作，およびユーザプログラムにおけるライブラリとシステムコールを介した操作とがある．コマンドでは，ファイル全体の操作が主なものであるのに対して，ライブラリおよびシステムコールではファイル内の各情報の操作が主になる．

ファイル操作のコマンドとして，ディレクトリ操作の cd, pwd, ls, mkdir, rmdir など，ファイル全体の操作として rm, cp, mv など，ファイルの内容の表示操作として cat, more, pr, lpr などがある．

ファイル操作のライブラリあるいはシステムコールとして，ファイルの open, close, 内部情報のアクセスの getc, putc, read, write, lseek などがある．

● UNIX のファイルの物理構造

UNIX のファイルは，ファイル空間の使用効率を向上させるために，統一した索引構造になっている．この索引を inode とよび，この識別子が UNIX のファイルの物理的アドレスとなっている．inode は，大規模なファイルにも対応できるよう間接的な索引構造を使っている．

以上のほかに，UNIX の基本サービス機能として，ネットワーク機能，記憶管理機能，端末制御機能などがあるが，ここでは省略する．

3. ユーザから見た UNIX の特徴

UNIX は，使いやすい会話型システムとして広く普及している．その一つの理由は，優れたユーザインタフェースを持っていることである．UNIX は，もともと行入出力型の端末装置を対象として開発されてきた．それに対し，CRT ディスプレイ，あるいはビットマップディスプレイなど端末装置のハードウェアの発達が見られ，それらに対応したユーザインタフェースの開発が UNIX の問題点となっている．UNIX におけるユーザインタフェースの発達は，次のようになる．

（1） 行向きユーザインタフェース

UNIX のユーザインタフェースは，ユーティリティの一つとして実行されるコマンドシェルにより実現されている．これは，ユーザとの会話をコマンド行を単位として行い，ユーザの要求の授受とその実行を司っている．

コマンドの追加や修正をユーザが容易に行えるようにコマンドスクリプト言語[12] を用意してあるので，コマンド行を含んだシェルスクリプトプログラムを作成すれば，新しいコマンドが追加できる．

（2） CRT ディスプレイ向きのユーザインタフェース

会話型システムの端末装置は，行端末型から CRT ディスプレイ型へと変遷しており，現在ほとんどこの

型が端末として使われている．CRT ディスプレイは数行にわたって画面に同時に表示することが可能であるから，文書の編集などに利用すると便利である．実際，ed をディスプレイ型に変更した vi[13]，teco をディスプレイ型に変更した emacs[14] は，この型の端末装置によくなじんでおり広く普及している．

コマンドシェルは独立したユーティリティとして実行されるから，CRT ディスプレイ向きに変更したものを開発し利用することが行われてきた．メニュ式の会話に基づいたメニュシェル，コマンド行を emacs 風に編集できる newcshell などは，その例と考えられる．

（3） ビットマップディスプレイ型ユーザインタフェース

CRT ディスプレイに対し画面内にある密度でアドレス付けできるビットを設定し，それを利用してウィンドウの操作，グラフィックスの表示，イメージデータの表示などを行うビットマップディスプレイが，ワークステーション，PC あるいは通常の端末にまで普及しつつある．これを利用したユーザインタフェースの実現は，UNIX にとって新しい問題である．これは，ウィンドウの利用とアイコンの利用が鍵となる．たとえば，X-ウィンドウや SUN-View ではウィンドウの操作が主な機能であり，各ウィンドウ内では，通常のシェルが働く．また，ANDREW システムのユーザインタフェースは，ウィンドウの管理とともに，アイコンを利用した操作が可能となっている．

上記の問題以外に，我が国では日本語とユーザインタフェースとの関係の問題がある．日本語の入力は，最近仮名／ローマ字漢字変換が主流を占めつつあるが，この機能をユーザインタフェースに持たせるのが UNIX としては最も適切である．UNIX は，そのような付加機能を追加するのが比較的容易にできる点がオペレーティングシステムとして優れている．AT & T では，UNIX の国際化を狙って多言語の支援ができるようにカーネルの設計を行っている[15]．日本語機能を備えた UNIX はすでにいくつか商品化されており，その普及も急速に進むであろう．

4. システム運用から見た UNIX の特徴とその問題点

4.1 システムのセキュリティ

UNIX は，その設計思想に基づきユーザからのフィードバックを得てシステムの成長を図ることを目指している．したがって，すべての情報はユーザに公開することを原則としている．たとえば，ファイルのパス名を指定すれば，ディレクトリのどの節点に対しても原則としてアクセスできる．任意の時点において存在しているプロセスとその状態を，ps コマンドで検索することができる．last あるいは lastcomm コマンドを使えば，ほかのユーザの login 時間，使用コマンドの履歴などを，検索することができる．

このような原則は，メインフレームの採っている原則とは正反対であり，そのようなシステムに慣れているユーザにとっては違和感を持つであろう．しかし，そのような情報公開によって得られる利点を考慮すると，この原則は変更できない．むしろ，そのような環境の中でどうしても実施しなければならないセキュリティの管理の機能を提供しておくことが重要である．

ファイルの内容の読み出し，書き込みおよび実行のアクセスに対しては，前述のようにファイルの所有者，グループ，および一般ユーザの三つの対象のプロセスからの要求を制御できる．また，ディレクトリファイルに対しては，その下のファイルに対する検索の要求を制御することができる．これらのアクセス権の設定は，セキュリティについて特に関心を払わなければならない場合にユーザの責任において行わなければならない[18]．

UNIX システムの保守運営は，オンラインで行えるようにするために，特別なユーザとしてスーパユーザを設け，そのユーザ識別子を用いてログインした場合，任意のファイルの操作をそれが持つアクセス権とは無関係に行える．スーパユーザは，そのためのパスワードを持ち，セキュリティの保護をしているが，一般にパスワードの洩れはよく見られることであり，スーパユーザを委託されたユーザの自覚を待たねばならない．

4.2 システムの性能

UNIX は，プロセスやファイルの実現法を一般的，統一的に行っているので，その性能は必ずしも優れているとは言えない．システムの性能は，システムの持ついくつかのパラメタによってある程度決まってくる．システム生成時に，それらのパラメタの設定を注意深く行って，与えられたハードウェア環境において最も性能が高くなるように考慮しなければならない．

たとえば，ファイルデバイス（ディスク）をいくつかのパーティションに分割してシステムの生成を行うが，スワッピング領域の設定場所は，仮想記憶のアク

セスの性能を大きく左右する．一般に，UNIX ではファイルの実体（物理的空間）を統合的索引構造に基づいて割り当てており，性能を向上させるように連続領域を割り当てることができない．これは，時間的性能を犠牲にして空間使用効率を向上させるという設計方針に従っているためであろう．

4.3 システムの運用，移植，拡張

UNIX は，システムに関する情報をできるだけ公開するという原則に基づき AT&T で運用されている．ソースの公開は，その最もよい例である．これにより，いくつかの新しいバージョンが開発された．また，ハードウェア的にも，多くの計算機システムの上に移植されている．これは，メインフレームのプラグコンパチブルマシンとは考え方が異なり，なるべく多種類の計算機システム上にソフトウェアを実装する方針を採っている．このほうが，より高性能の機種にシステムを実現でき，またシステムの普及の推進力ともなる．

オペレーティングシステムを移植することは必ずしも容易なことではないが，UNIX はソースコードが公開されていること，システムの大部分が高級言語 C で記述されていることなどから，移植は比較的容易であると言える．この場合，まずシステムのカーネル層を移植する．このとき，入出力装置のドライバは機器や計算機システムに依存することが多いので，その部分は新しく作成することが必要である．カーネル層が移植できれば，ユーティリティは再コンパイルにより大部分が移植できる．ただし，端末装置に大きく依存するようなユーティリティ，たとえばビットマップディスプレイのウィンドウ管理システムなどは，やはり改修する必要があるだろう．いずれにしろ，スーパコンピュータ，メインフレームからワークステーション，PC のレベルまで同一のオペレーティングシステムが稼働している例は，ほかにはない．

UNIX を使用することの一つの利点は，その上で流通しているソフトウェアシステムが非常に多いことである．システム自体が広く普及していること，オペレーティングシステムが改修しやすくソフトウェアシステムを開発する土台として十分信頼に足り，また柔軟に対応できること，UNIX 自体が広くユーザに公開してフィードバックを得ることを考えていたのでそのユーザたちがソフトウェアを公開することによく慣れていることなどが，ソフトウェアの流通が非常によくなった理由である．特に，大学や研究所で開発したソフトウェアシステムをパブリックドメインとして公開する習慣が広まっていることは，今後の計算機ソフトウェアのあり方に大きな影響を与えずにはおかないであろう．

5. UNIX の利用

UNIX システムは，汎用のオペレーティングシステムとしていろいろの場面に適用できる．その設計思想からくる特徴により，特に適している場面と，あまり得手でない場面とがある．

（1） UNIX の適している場面

● ソフトウェア開発環境

ソフトウェアの開発のツールを多数揃え，また優れたユーザインタフェースを提供しているので，ソフトウェア開発環境としてよく利用される．

● 研究教育環境

UNIX は，各種プログラミングツール，優れた文書処理ツール，使いやすいネットワーク用ユーティリティを備えているので，計算機に関する研究あるいは教育環境として利用することがよく見られる．

● オフィス環境

UNIX は，優れた文書処理ツール，ネットワークツール，各種プログラミングツールなどを備えており，また，データベース管理システム，日本語処理機能なども実現されているので，オフィスオートメーションを推進するオフィス環境として利用しても十分有効である．

（2） UNIX の不得手な場面

● 実時間処理環境

UNIX は，統合的なファイルシステム，一般的なプロセスシステムを採用しているので，応答時間に制限を課せられる実時間システムにはあまり向いていない．これに対しては，連続したファイル空間の割り当て，プロセススケジューリングの改良などにより，実時間処理に適した UNIX システムが開発されている．UNIX の優れたソフトウェア開発環境の面を考慮すると，多少の性能の低下が見られても，このシステムを導入する動機は十分あると言える．

● 大容量計算

上記の場合と同じように，UNIX は性能を第一義にして設計されていないので，計算機システムの性能を余すところなく使いたい大容量計算には UNIX は適していない．しかし，Cray のようなスーパコンピュータにも UNIX が移植されている．これは，や

はり使いやすいユーザインタフェース，豊富なツール群などの利点をスーパコンピュータでも生かしたいからであろう．

6. おわりに

UNIX の設計思想とそれから生まれてくる特徴について，簡単に紹介してきた．計算機システムの小型化，それに基づく分散化がこれからの技術動向になるのは確かである．UNIX は，そのような計算機システムの上で稼働されれば，さらにその良さを発揮できる．

いくつかの組織やグループでその標準化を検討している[17] のは，その普及がさらに推進されることを予測しているからであろう．

UNIX は，まだ改良すべき機能，追加すべき機能がある．ビットマップディスプレイを利用したユーザインタフェース，ファイルシステムの高速化とセキュリティの強化，日本語処理のユーティリティの強化などが当面の課題であろう．

参 考 文 献

1) Ritchie, D. M.: Reflections on Software Research, Comm. ACM, Vol. 27, No. 8, pp. 758-760 (1983).
2) Ritchie, D. M. and Thompson, K.: The UNIX Time-Sharing System, Comm. ACM, Vol. 17, No. 7, pp. 365-375 (1974).
3) SUN View Programmer's Guide, SUN Microsystems Inc. (1985).
4) X window System-MIT, User Contribution Software (UCS), 4.3 Berkeley Software Distribution, University of California, Berkeley (1986).
5) Morris, J. H. et al.: ANDREW: A Distributed Personal Computing Environment, Comm. ACM, Vol. 29, No. 3, pp. 184-201 (1986).
6) UNIX System-V, Release-2.0 User Reference Manual, AT&T Technologies, Inc. (1984).
7) UNIX Programmer's Manual, 4.2 Berkeley Software Distribution, University of California, Berkeley (1983).
8) Thompson, K.: UNIX Implementation, The Bell System Technical Journal, Vol. 57, No. 6, pp. 1931-1946 (1978).
9) Leffler, S., Joy, N. W. and Fabry, R. S.: 4.2 BSD Networking Implementation Notes, 4.2 BSD UNIX Programmer's Manual, Vol. 2C-Supplementary Documents (1983).
10) Ritchie, D. M.: A Stream Input-Output System, AT&T Bell Laboratories Technical Journal, Vol. 63, No. 8, Part 2(1984).
11) Ritchie, D. M.: Retrospective, The Bell System Technical Journal, Vol. 57, No. 6, pp. 1947-1969 (1978).
12) Bourne, S. R.: The UNIX Shell, The Bell System Technical Journal, Vol. 57, No. 6, pp. 1971-1990 (1978).
Joy, W.: An Introduction to the C Shell, 4.2 BSD UNIX Programmer's Manual, Vol. 2 C-Supplementary Documents (1983).
13) Joy, W.: An Introduction to Display Editing with Vi, 4.2 BSD UNIX Programmer's Manual, Vol. 2 C-Supplementary Documents (1983).
14) Gosling, J.: Unix Emacs, Department of Computer Science, Carnegie-Mellon University (1982).
15) What are International Functions and Native Language Supplements on UNIX System V?, $ echo, Asia/Pacific Edition, AT & T (1986).
16) たとえば Work Station 500, Mass Comp.
17) Portable Operating System Environment (Posix) P 1003/D6, IEEE (1985).
18) Wood, P. H. and Kochan, S. G.: UNIX System Security, Hayden Book Company.

(昭和 61 年 10 月 1 日受付)