UNIX のワークステーションへの移植性について†

多田 好克††

OA (Office Automation), CAD (Computer Aided Design), プログラム開発の各分野でワークステーションの活用が始まり，専用ハードウェアの開発が盛んになってきた．新しいハードウェアの製造にともなって，今後，ワークステーション用オペレーティング・システムの開発，移植の必要性も増すと考えられる．このような背景を考えて本論文ではミニコンピュータ用のオペレーティング・システムとして開発された UNIX* をワークステーションへ移植する際の作業を概観し，①作業に要した時間，②プログラムの変更行数，の2点からその作業量を測定する．測定の結果，UNIX のカーネルのメモリ管理，プロセス管理に変更点が多いこと，一方，ファイルシステム，パイプ，tty ドライバには変更点がほとんどないことが明らかになった．また，C コンパイラの移植にカーネルの移植と同程度の時間が必要であることも明らかになった．さらに，本論文では UNIX 移植時の問題点についても言及し，ワークステーション用オペレーティング・システムとして UNIX に欠けている機能をも示した．

1. はじめに

近年のハードウェア技術の進歩によって，①ミニコンピュータ並みの処理能力，②良質のマン・マシン・インタフェース機能，③10 Mbit/sec 程度の高速通信機能，を合わせ持ったワークステーション[1),2)]の大量生産が可能になった．これらのワークステーションは，現在 OA (Office Automation), CAD (Computer Aided Design), プログラム開発の各分野で活用されており，ハードウェアの改良とソフトウェアの開発も引続き行われている．

ワークステーション用のシステム・ソフトウェアとしては，現在，ALTO OS[3)]，PILOT[4)]，AEGIS[5)]，Lispmachine Lisp System[6)]，Interlisp-D[7)]，Smalltalk-80[8)]，UNIX[9)] などが有名である．しかし，新しいハードウェアの開発やウィンドウ・システム等の新概念の実現にともなって，今後，ますますワークステーション用システム・ソフトウェア作製の必要性が高まるものと考えられる[10)]．

ワークステーション用オペレーティング・システムを新しく開発すべきか，それとも，既存のオペレーティング・システムをワークステーション用に移値・拡張すべきかは議論の分かれるところであるが，①作業量が少ない，②蓄積されたソフトウェアをそのまま活用できる，といった観点からは後者の方が有利である．ミニコンピュータ用オペレーティング・システムとして開発された UNIX は移植性が高く[11),12)]蓄積ソフトウェアも豊富なので[13),14)]，今後，さまざまなワークステーションへの移植が予想される．

我々の研究室では 1983 年 3 月に Sun ワークステーション[15),16)]を設置したが，当時これには後述の基本ソフトウェア（モニタ）しかなく，マウス・コントローラ，ディスク・ドライバ等のあらゆるプログラムを自主開発しなければならなかった．そこで，上述のような考察を経て，これに UNIX (32 V) を移植することにし，約 18 人月を掛けてその移植を完了した．本論文では，この経験に基づき，UNIX を Sun ワークステーションへ移植した際の手順をまとめ，その作業量を評価する．また，ワークステーション用オペレーティング・システムとして UNIX に欠けている機能についても論じる．

以下，第2章では移植を行った環境を簡単に説明する．続く第3章では移植作業を概観し，その作業量を作業時間と変更行数の2点から議論する．また，第4章では移植時の問題点を論じ，ワークステーション用オペレーティング・システムとして UNIX に欠けている機能についても言及する．

2. 移植環境

2.1 ターゲット・マシン

今回 UNIX の移植を行ったのは，Sun Microsystems 社の Sun ワークステーションである．Sun ワークステーションには現在3種類のハードウェアがあるが，今回使ったのはその初期のものである．ハードウェアの特徴は以下のとおりである．

i) CPU は MC 68000．データバスは 16 bit 転

† Transportability of the UNIX to a Workstation by YOSHIKATSU TADA (Department of Mathematical Engineering, University of Tokyo).

†† 東京大学工学部計数工学科（現在 電気通信大学電子情報学科）

* UNIX is a trademark of AT&T Bell Laboratories.

送，内部では 32 bit 処理．汎用レジスタとしてデータレジスタ 8 個，アドレスレジスタ 8 個を持つ．カーネルとユーザの実行モードがあり，7 レベルの割込み制御機能を持つ．

ii) 2 Mbyte×16 コンテクストのアドレス空間．各コンテクストは 32 kbyte のセグメント 64 個で構成され，各セグメントは 2 kbyte のページ 16 個で構成される．セグメントはメモリ保護の最少単位，ページはメモリ管理の最少単位である．

iii) 実メモリは 1.25 Mbyte．1 Mbyte がオンボード・メモリ(アクセスタイム 400 nsec)，残り 256 kbyte がマルチバス・メモリ(アクセスタイム 1 μsec)．I/O 装置の DMA はマルチバス・メモリに対してだけ可能．

iv) キーボード，3 キーの光学マウス，マルチバスに接続された 1024×800 ドットのビットマップ・ディスプレイ．

v) 容量 20 Mbyte のハードディスク．

vi) 10 Mbit/sec のイーサネット．

vii) ふたつの RS 232 C ポート．

また，32 kbyte の ROM にレジスタやメモリの参照と変更，例外処理，文字入出力などを行う基本ソフトウェア(モニタ)が入っている．

2.2 ホスト・マシン

ホスト・マシンには東京大学大型計算機センターの VAX-11/780 UNIX (4.1 bsd) を使用した．

移植には UNIX の提供している機能を最大限に利用した．階層的なファイル管理，UNIX の環境に準拠した C クロスコンパイラ，そして，make (maintain program groups)，ar (archive and library maintainer) などのコマンドを使うことにより，UNIX 移植に要する作業量を軽減した．

2.3 データ転送路

ホスト・マシンからターゲット・マシンへのデータ転送には，9600 baud の端末回線 (RS 232 C) を使った．転送データはすべて Motorola の S-format に変換し ASCII 文字列として扱った．データ転送速度の実測値は約 430 byte/sec であった．

この転送路は移値の進行状況に応じて，①UNIX カーネルのオブジェクトコード (約 70 kbyte)，②一部のコマンドのオブジェクトコードを含むファイルシステム(約 500 kbyte)，③コマンドのソースコード(約 3Mbyte)，など多くのデータを運ぶ．したがって，磁気テープ装置または取外し可能なディスク装置等が使える場合はそれらの併用も考えた方が良い．

3. 移植作業

この章では，移植に際して決定した変更方針を議論する．また，UNIX の移植作業を変更行数の観点から定量的に考察する．

変更部分の議論に先立ち，変更手順と変更に掛かった時間を**表 1** に示す．なお，ひとつの番号に複数の作業を書いてあるのはそれらの作業を並行して行ったことを示している．

変更方針の決定から実際のコーディングまで，移植作業のほとんどの部分を筆者がひとりで行った．ただし，C コンパイラは移植の途中でマサチューセッツ工科大学の nu プロジェクトのもの*が利用可能となりり，以後，カーネル等の移植にはそのコンパイラを使用した．表 1 に示した C コンパイラ移植の所要時間は，途中まで行った作業から推定したものである．

以下，項目別に移植作業を論じる．

3.1 移植オペレーティング・システムの決定

Sun ワークステーションに移植するオペレーティング・システムを UNIX にしたのは，①移植性が高い．②マン・マシン・インタフェースが良い．③ユーティリティ・ソフトウェアが多い．④拡張性が高い，といった理由による．また，UNIX の各バージョンの中から 32 V を選んだのは，①32 bit アーキテクチャなので 2 Mbyte のアドレス空間を直接指定できる，

表 1 移植手順と移植に要した時間
Table 1 Transporting steps with required time.

	作業内容	所要時間(人月)
I.	a) ホスト・マシン使用法の習得	0.3
	b) ターゲット・マシンの CPU の理解	0.3
II.	a) C コンパイラの移植	3.5
	b) データ転送路の確立	0.5
III.	a) UNIX カーネルの理解	2.0
	b) VAX-11 のアーキテクチャの理解	0.5
	c) ターゲット・マシンのアーキテクチャの理解	1.0
IV.	移植方針の決定	0.5
V.	カーネルの書換え	3.5
VI.	a) ライブラリの書換え	1.0
	b) インクルード・ファイルの書換え	0.3
VII.	コマンド／ユーティリティの作成	0.7
VIII.	ファイルシステムの作成	0.4
IX.	システムのデバッグ	3.0

* MIT Lab. for Computer Science の Christopher J. Terman による．

表 2 Cコンパイラの変更行数
Table 2 Code changes of C compiler.

	移植前	削除	挿入	移植後
プリプロセッサ	1553	0	0	1553
コンパイラ	11668	1150	1472	11990
マシン依存	3293	991	858	3160
マシン非依存	8375	159	614	8830
アセンブラ				3267
ローダ				1051
オプティマイザ	2071			
制御プログラム	464*1			152*2

*1 Cプログラム
*2 シェル・スクリプト

②デマンド・ページングを行わない，③システムが安定している，といった理由による．

3.2 Cコンパイラ

Cコンパイラは，

i) Cプリプロセッサ
ii) Cからアセンブリ言語へのコンパイラ
iii) アセンブラ
iv) ローダ
v) オプティマイザ
vi) 上記 i)から v)の各プログラムを必要に応じて起動する制御プログラム

に分けて考えることができる．**表 2** に各プログラムの行数と移植に際して変更した行数とを示す．なお，コンパイラ，アセンブラ，ローダは前述したように nu プロジェクトで作製したものである．以下，各プログラムの変更部分について述べる．

Cプリプロセッサのプログラムは，変更を加えることなしに移植できた．

コンパイラは portable C compiler[17] を利用して移植してある．表 2 には，ターゲット・マシンに依存する部分とターゲット・マシンに依存しない部分の変更行数も合わせて示した．

アセンブラ，ローダは VAX-11 用を参考にして新しく作成してある．オブジェクトのフォーマット ("a.out" format) は 32 V のものと同一ではなく，シンボル名が可変長で格納できるように変更してある．

オプティマイザはコンパイラが出力したアセンブラ・コードを扱う．したがって，移植に際しては新しく作りなおす必要があるが，現段階ではまだオプティマイザを作製していない．表 2 には参考として 32 V の VAX-11 用オプティマイザの行数を示した．

32V本来の制御プログラム "cc" はCを使って記述してある．しかし，今回の移植では，①変更が容易，②テストが簡単，といった理由からこれをシェル・スクリプトで記述した．現在は "cc" のすべてのオプションを実行でき，その行数は 152 行である．

表 3 カーネルの変更行数
Table 3 Code changes of kernel.

	移植前	削除	挿入	移植後
全行数	14190	4116	4731	14805
Cプログラム	12777	2703	2630	12704
アセンブラ・プログラム	1413	1413	2101	2101*

* 327 行の 32 bit 乗除算ルーチンを含む

3.3 カーネル

表 3 に，カーネルの変更行数を示す．以下，カーネルの変更部分をさらに細かく分類し，その変更方針を述べる．

a) アセンブラ

アセンブラで記述してあるのは，①割込み処理，②割込みレベルの制御，③システム・コール時のスタックフレームの扱い，④プロセス切替え，⑤アドレス空間の割当て，⑥アクセス権のチェック，である．また，移植後は，⑦32 bit 乗除算ルーチン (327 行) が加わった．

ほとんどの部分は，VAX-11 のアセンブラから MC 68000 のアセンブラへの一対一の機械的な変更である．VAX-11 の，①コンテクスト切替え命令 (LDPCTX, STPCTX)，②アクセス権チェック命令 (PROBER, PROBEW)，③32 bit 乗除算命令 (MUL, DIV)，は MC 68000 のサブルーチンを作りその機能をシミュレートした．

割込み処理ルーチンは Sun ワークステーションのモニタを流用し，必要なルーチンだけを書き換えた．こうすることにより，

i) ダイナミック RAM のリフレッシュのような，モニタのソフトウェアがやっている処理をそのまま流用できる；
ii) デバッグ中はモニタの機能を活用できる；

といった利点が生れる．移植が完了すると，割込みはカーネル内で処理するようにし，予期しない割込みによってシステムが止まってしまうのを防止した．

b) メモリ管理

VAX-11 はメモリを 512 byte 単位で管理をするが，Sun ワークステーションは 2 kbyte 単位で管理する．また，カーネルはメモリのアロケート／フリー要求を 512 byte 単位で行う．移植にあたっては，①従

来どおり 512 byte で管理する，②2 kbyte で管理する，のふたつの方法を検討したが，

ⅰ） カーネル内には 512 byte に依存したプログラムが多数存在し，また，各所に分散している；

ⅱ） 2 kbyte 単位で管理しても，Sun ワークステーションは 32 kbyte 単位でメモリ保護を行うのでプログラムが簡単にならない；

といった理由から 512 byte 管理を採用した．実際には，アロケート／フリーは 512 byte 単位で要求を出し，その処理ルーチンが 2 kbyte 単位にまとめている．この方法を使えば移植は簡単になるがメモリ管理の処理速度は遅くなる．

また，Sun ワークステーションには 16 個までのコンテクストを高速に切替えるためのハードウェア（コンテクスト・レジスタ）がある．しかし，今回の移植では，

ⅰ） UNIX は同時に多数のプロセスを実行し，16 コンテクストでは足りない；

ⅱ） 各コンテクストはセグメント・マップを共用し，大きなプロセスの実行時には効率が上がらない；

ⅲ） ひとつのコンテクストにカーネル／ユーザの両プログラムを混在しても 1.5 Mbyte のユーザ空間が確保できる；

といった理由から，このコンテクスト・レジスタは使用しなかった．

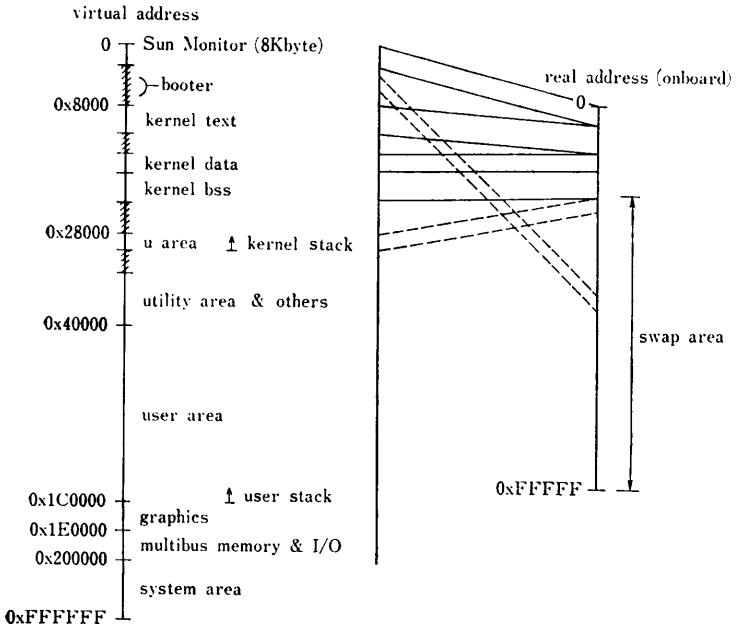

図 1 移植後のアドレス割当て
Fig. 1 Address mapping after transportation.

図 1 に移植後のアドレス割当てを示す．

c） コーリング・シーケンス

ここでは，①関数呼出し，②システム・コール，のふたつのコーリング・シーケンスについて述べる．

関数のコーリング・シーケンスは，C コンパイラの実現方法によって決まる．今回の移植では，引数をスタックに積んで関数を呼出し，関数の実行終了後に呼出し側が積んだ引数を取り除くというシーケンスを採用した．また，A0, A1, D0, D1 の各レジスタを演算用レジスタとして使い，残り（A2～A5 と D2～D7）をレジスタ変数用として使った．この場合，割込み処理の開始時には A0, A1, D0, D1 の各レジスタを退避し，終了時にはそれらを回復しなければならない．

VAX-11 でのシステム・コールは，

ⅰ） 引数をスタックに積む；

ⅱ） 関数呼出しと同じフレームを作る；

ⅲ） CHMK (CHange Mode to Kernel) 命令を使ってカーネル・モードに移行する；

という手順で実現してある．MC 68000 でのシステム・コールは，このようなフレームを作らずに実現した．また，カーネル・モードへの移行には TRAP 命令を使い，CHMK 命令のオペランド部で指定していたシステム・コールの種類は D0 レジスタを使って指定した．

d） I/O ドライバ

I/O ドライバは，VAX-11 用のソース・コードを参考にしながら新しく書き直した．

I/O ドライバの記述は，I/O 装置からの割込みと密接に関係する．Sun ワークステーションは 7 レベルの割込みを制御でき，これらの割込みはレベルの数値が大きいほど優先順位が高い．また，レベル 7 の割込みは禁止状態にはできない．

今回の移植では，

レベル 7：内部タイマ（メモリ・リフレッシュとキーボード，マウスのポーリング）
レベル 6：RS 232 C 回線 (tty ドライバ)
レベル 5：システム・タイマ
レベル 4：ディスク装置

の 4 レベルを使用した．

e） スケジューラ

プロセスのスケジューリング・アルゴ

表 4 カーネルの機能別変更率
Table 4 Code changes of kernel. (Classified according to functions)

	ファイル数	全行数	変更率(%)
I. 初期化，プロセス理管，メモリ管理	24	4044	50.5
II. 例外処理，シグナル，システム・コール	13	2695	35.5
III. スワップ処理，基本 I/O，block device	7	1324	17.9
IV. ファイル，パイプ	18	2582	2.9
V. tty ドライバ，character device	8	2679	3.0
VI. I/O ドライバ	5	1154	88.0
VII. その他 (32 bit 乗除算)	1	327	100.0

表 5 インクルード・ファイルの変更率
Table 5 Code changes of include file.

	ファイル数	全行数	変更率(%)
カーネル関係	31	1588	12.4
ユーティリティ関係	28	900	21.4
追加 (グラフィクス，マウス)	2	186	100.0

リズムは変更を加えずそのまま移植した．ただし，スケジューリングは VAX-11 のソフトウェア割込みを使って実現してあったので，その部分はアルゴリズムを変更した．

f) その他

上記 a)から e)以外の，ファイルシステムの管理，パイプの処理等は変更を加えずに済んだ．また，各 tty ドライバに共通の I/O キューの操作，バッファの管理も変更の必要はなかった．

表 4 にカーネルの機能別変更率を示す．システムの初期化とプロセス管理やメモリ管理とは区別が困難であったので項目 I としてひとつにまとめた．また，項目 III の基本 I/O とはブロック I/O のバッファ管理のことを，項目 VI の I/O ドライバとは実際のディバイス・ドライバのことを意味している．

3.4 インクルード・ファイル

インクルード・ファイルは**表 5** に示したようにカーネルに関係するものが 31 ファイル，ユーティリティに関係するものが 28 ファイルある．カーネルに関係するファイルはカーネル変更時に書き換えたものをそのまま流用した．一方，ユーティリティに関係するものは，①オブジェクト・ファイルのフォーマット，②ロング・ジャンプの環境格納場所，③アーカイブ・ファイルのフォーマット，を記述するファイルをそれぞれ変更した．ユーティリティに関係するファイルの変更率が高いのは今回の移植においてオブジェクト・ファイルのフォーマットを変更したためである．

また，今回の移植では Sun ワークステーションのハードウェアを記述するファイルとして，①ビットマップ・ディスプレイ，②マウス，のふたつのインクルード・ファイルを追加した．

表 6 ライブラリの変更率
Table 6 Code changes of library.

	ファイル数	全行数	変更率(%)	備考
システム・コール	68	1359	100.0	アセンブラ
システム・サポート				
32bit 乗除算ルーチン	12	557	100.0	アセンブラ(追加)
実数演算ルーチン	1	692	100.0	アセンブラ(追加)
その他(エラー処理)	1	11	100.0	アセンブラ
入出力	44	1330	0.0	C
	6	577	100.0	アセンブラ
ユーティリティ	47	2319	2.6	C
初期化	1	41	100.1	アセンブラ

3.5 ライブラリ

ここでいうライブラリとは，Cのプログラムの実行に必要なルーチン群のことである．ライブラリは，

i) システム・コール
ii) システム・サポート (32 bit の四則演算ルーチンなど)
iii) 入出力
iv) ユーティリティ
v) 初期化 (プログラム実行環境の初期化)

に分類できる．

表 6 にはそれぞれのファイル数，移植後の行数，変更率を示した．

システム・コールは平均行数 20 行の短いアセンブラプログラムの集まりで，ほとんどのファイルは機械的に書き換え可能である．

システム・サポートもすべてアセンブラプログラムである．このうち 12 ファイル (557 行) は 32 bit の整数乗除算ルーチン，また，1ファイル(692 行)は実数四則演算用ルーチンで，移植に際して MC 68000 用に新しく作成した．

入出力は全 45 ファイルの内の 44 ファイルがCプログラムであり，これらは変更の必要がなかった．残るひとつのファイルはフォーマット化出力のためのアセンブラプログラム (577 行) で，このファイルは MC 68000 用に書き換えた．

ユーティリティはすべてCプログラムであり，その

移植性は高かった．ただし，型変換に関するルーチンのバイト・スワップを防ぐためにふたつのファイルを変更した．

初期化はアセンブラプログラムで，移植後のユーザプログラム環境に合わせて変更した．

3.6 コマンド／ユーティリティ

UNIX (32 V) には約 180 個のコマンド／ユーティリティがある．これらのプログラムはほとんどの部分をCで記述してあり，移植用のライブラリを使って再コンパイルすれば大部分のプログラムはそのまま利用可能であった．

そのままでは移植できないプログラムには，

ⅰ） ハードウェアに依存したもの
デバッガ("adb", "sdb") など

ⅱ） システム構成の違いによるもの
"df" のディフォールトI/O装置名など

ⅲ） 環境の違いによるもの
"date" の出力が日本標準時(JST)でないなど

ⅳ） オブジェクト・フォーマットの違いによるもの "nm", ps" など*

があった．

4. 移植の問題点

UNIX は移植性を考慮して作られたオペレーティング・システムではあるが，実際の移植に際していくつかの問題点が明らかになった．この章では，それらの問題点をターゲット・マシンに注目して分類し，それぞれについて議論する．なお，分類は以下の基準に従った．

ⅰ） 移植一般に関する問題点
UNIX の構成法に起因するもの（すなわち，大型計算機，ミニコンピュータ，ワークステーション等，どのようなターゲット・マシンに移植する場合にも問題となるもの）；

ⅱ） ワークステーションに関する問題点
マウス，ビットマップ・ディスプレイ等のワークステーションに特有の原因によるもの；

ⅲ） Sun ワークステーションに関する問題点
Sun ワークステーションに固有の原因によるもの；

4.1 移植一般に関する問題点

a） バイト・スワップ

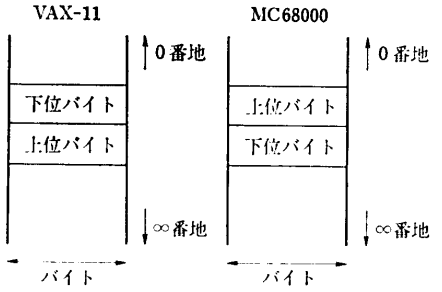

図 2 バイト・スワップの例（ワード・データをメモリに格納した場合）
Fig. 2 Example of byte swapping.

VAX-11 と MC 68000 では long (4 byte), short (2 byte) のメモリ上での格納方法が違う（図2）．すなわち，VAX-11 ではデータの下位バイトをメモリ内の低番地（0番地）側に格納し，MC 68000 では逆に高番地側に格納する．したがって，同じデータをあるときは long (または short) として扱い，またあるときは4 byte (または2 byte) のchar として扱うプログラムは，そのままでは移植できない．

今回の移植では，①カーネル内のファイルシステムの処理，②クロスコンパイラのオブジェクト作成，③ライブラリ内の型変換ルーチン，④ホスト・マシン上でのファイルシステムの作成，の各部分でバイト・スワップが問題となった．

バイト・スワップの問題は CPU のアーキテクチャに即したコードを出力するためには避けようのない問題である．移植に際しては，問題を正しく認識しプログラムの該当部分にコメントを入れて注意を促すとか，コンパイラのマクロ機能を使ってターゲット・マシンの CPU アーキテクチャに合わせるとかの工夫が必要である．

b） 構造体の違い

Cコンパイラの仕様は，構造体のメモリ上での配置法を規定していない．したがって，

ⅰ） 構造体内の long データを2 byte 境界に置くか4 byte 境界に置くか；

ⅱ） 構造体内のビット・フィールドを最上位ビットから割り振るか最下位ビットから割り振るか；

等の仕様は，CPU のアーキテクチャによって異なることが多い．

今回の移植に使用した MC 68000 用クロスコンパイラと UNIX の VAX-11 用コンパイラとでは，ⅰ），ⅱ）共に仕様が異なっており，各種構造体の定義部やカーネル内のメモリ管理ルーチン等を書き換える必要

* 移植したCコンパイラのオブジェクト・フォーマットが 32V のものと同じ場合には，この問題は生じない．

があった.

c) VAX-11 の強力な命令

VAX-11 には, ①コンテクスト切替え (LDPCTX, SVPCTX), ②アクセス権チェック (PROBER, PROBEW), ③サブルーチン呼出し (CALLS), ④32 bit 乗除算 (MUL, DIV), などの強力な命令が用意されている. これらの命令はターゲット・マシンの命令群を使ってシミュレートすることになるが, それが間接的にカーネルの移植性を下げる原因ともなる. 特に CALLS 命令の procedure entry mask* は効率的にシミュレートする方法がないにもかかわらず, カーネル内のシグナル・ハンドラでその情報を利用しているため, 移植に際してはライブラリの書換えをも含めた大幅な変更が必要であった.

d) メモリ管理

UNIX のカーネルは VAX-11 のメモリ管理法 (基本的には 1 レベル・ページング) を前提にして記述してある. したがって, 多レベル・ページングのメモリ管理機構を持つハードウェアには比較的簡単に移植できるがセグメンテーション方式のメモリ管理機構を持つハードウェアへは容易に移植できない. また, メモリ管理ルーチンはソースコードの各所に分散しており, それがカーネルの移植性を下げる原因になっている.

4.2 ワークステーションに関する問題点

a) tty ドライバ

UNIX の tty ドライバは, 文字出力終了時に割込みを発生する端末を想定している. すなわち, tty ドライバの文字出力ルーチンは端末からの割込みによって起動され, 出力バッファ内の次の文字を端末に送り出す. ところが, 一般にワークステーションでは, 文字出力終了時にも割込みは発生しない. したがって, ワークステーションへの移植に際しては tty ドライバの文字出力ルーチンのアルゴリズムを変更しなければならない.

今回の移植では, まず, 出力バッファに文字がある限り端末 (ワークステーションの画面) に出力を繰り返すという方法を採用したが, この方法ではワークステーションの画面への文字出力が通常の端末への文字出力よりも優先されるという不都合が生じた. 現在は一定の文字数**を出力するたびにプロセッサを放棄し, ソフトウェアで割込みをシミュレートするというアルゴリズムを採用している.

b) マウスとキーボード

ワークステーションには, ①マウス, ②キーボード, のふたつの入力源がある. ところが, UNIX の read システム・コールはデータがないときにはプロセッサを放棄するように作ってあり, ふたつの入力を同時に待つことができない. マウス用, キーボード用にそれぞれプロセスを作り, プロセス間通信を使ってふたつの入力を待つことはできるが, UNIX のプロセス間通信機能はこの目的には貧弱すぎる.

ふたつの入力を扱う方法としては,

i) ふたつの入力を単一の queue につなぐ;

ii) マウス入力によってシグナルを発生させる;

iii) システム・コールを拡張しプロセッサを放棄しないようにする;

などがあるが, それぞれ,

i) queue につながるデータが大きくなる;

ii) システムの負荷が増える;

iii) CPU 資源を浪費する;

という問題点もある.

今回の移植では, マウスの read システム・コールを拡張しプロセッサを放棄しないようにした. 一方, キーボードは既存のソフトウェアがそのまま使えるようにという配慮から, ioctl システム・コールを拡張しキーボード・バッファの文字数を確認できるようにした.

4.3 Sun ワークステーションに関する問題点

a) オンボード・メモリとの DMA

Sun ワークステーションでは, マルチバス上の I/O 装置はマルチバス・メモリに対してだけ DMA が可能で, 主記憶(オンボード・メモリ)に対しては DMA ができない. すなわち, データを読み書きするには主記憶⇄マルチバス・メモリ⇄マルチバス I/O 装置という 2 段階の手続きを経る. 今回の移植では, システムのバッファをマルチバス・メモリ上に割付けて処理効率の低下を防いだ. ただし, ユーザ・プロセスのスワップ処理は主記憶とマルチバス・メモリ間のコピー操作が必要でありシステムのオーバヘッドの原因になっている.

b) グラフィクス・ボード

Sun ワークステーションのグラフィクス・ボードは, 高速のラスタ・オペレーションを実現するために内部にさまざまなレジスタを持っている. これらのレ

* CALLS 命令でスタックに積まれるビット列. 各ビットはレジスタ (r0 から r11) に対応し, それらは, サブルーチン呼出し時にスタックにセーブしたレジスタを示している.

** 経験的には 30 文字前後.

ジスタの値はCPUのMOVE命令などを使って変更できるがその値を読み出すことはできない．ウィンドウ・システムのように複数のプロセスが並行して画面に出力する場合には，このハードウェアでは効率的な処理ができない．

もし，グラフィクス・ボードのレジスタの値を読み出すことができれば，それらの値をCPUのレジスタ同様プロセス・コンテクストの一部として扱うことが可能となる．すなわち，これらのレジスタの値をプロセス制御ブロックに格納し，プロセス切替え時にカーネル内でレジスタの値を設定することによって，効率的なウィンドウ・システムが実現できる．

5. おわりに

本論文では，UNIX (32 V) をSunワークステーションに移植した手順を概観し，その作業量を，①作業に要した時間，②プログラムの変更行数，の観点から評価した．また，その結果，UNIXの移植には約18人月の時間が掛かったこと，カーネル・プログラムの変更行数は約4400行（全行数の31%）に及んだことを明らかにした．新しいOSを構築しコマンド，ユーティリティを新しく作製する場合に比べ，このUNIX移植の作業量は格段に少ないと考えられる．

UNIXの移植性は高く，今後もワークステーション等のコンピュータのオペレーティング・システムとして広く使用されると思われる．しかし，マウスやビットマップ・ディスプレイなどのワークステーションに特有なハードウェアを管理するには不十分な面もあり，それらを拡張し移植することは容易でない．

今回移植したUNIXのプロセス間通信機能を拡張し，ウィンドウ・システム，ローカルエリア・ネットワーク等のワークステーションに特有な機能を実現するのが今後の課題である．

参考文献

1) 鈴木則久：高機能ワークステーションについて，情報処理，Vol. 25, No. 2, pp. 83-92 (1984).
2) 坂村 健：高機能ワークステーションのアーキテクチャ，情報処理，Vol. 25, No. 2, pp. 93-102 (1984).
3) Lampson, B. W. and Sproull, R. F.: An Open Operating System for a Single-user Machine, *Operating System Review*, Vol. 13, No. 5 (1979).
4) Redell, D. D. et al.: Pilot: An Operating System for a Personal Computer, *Comm. ACM*, Vol. 23, No. 2, pp. 81-91 (1980).
5) Nelson, D. L.: *Apollo Domain Architecture*, Apollo Computer Inc. (1981).
6) Weinred, D. and Moon, D.: *LISP Machine Manual*, 4th Ed., Symbolics Inc., Cambridge, Massachusetts (July 1981).
7) Interlisp-D Users Guide, Xerox (Sep. 1982).
8) Goldberg, A. and Robson, D.: *SMALLTALK-80: The Language and Its Implementation*, Addison-Wesley, Reading, Mass. (1983).
9) Ritchie, D. M. and Thompson, K.: The UNIX Time-Sharing System, *Comm. ACM*, Vol. 17, No. 7, pp. 365-375 (1974).
10) Wilkes, A. J. et al.: The Rainbow Workstation, *The Comput. J.*, Vol. 27, No. 2, pp. 112-120 (1984).
11) Johnson S. C. and Ritchie D. M.: Portability of C Programs and the UNIX System, *B. S. T. J.*, Vol. 57, No. 6, pp. 2021-2048, July-August (1978).
12) Paul J. J. and Thomas S. H.: Transporting a Portable Operating System: Unix to an IBM Minicomputer, *Comm. ACM*, Vol. 26, No. 12, pp. 1066-1072 (1983).
13) *UNIX Programmer's Manual*, VOLUME 1: Bell Telephone Laboratories, New Jersey (1979).
14) Bill Joy: Berkely 4.2 Gives UNIX Operating System Network Support, *Electronics*, July 28 (1983).
15) Programmer's Reference Manual for the Sun Workstation, Sun Microsystems, Inc., Santa Clara, CA (1982).
16) 多田好克：Sunワークステーション，情報処理，Vol. 25, No. 2, pp. 132-137 (1984).
17) Johnson, S. C.: A Tour through the Portable C Compiler, *UNIX Programmer's Manual*, VOLUME 2, Bell Telephone Laboratories, pp. 554-568 (1979).

(昭和59年10月15日受付)
(昭和60年4月25日採録)