

ポータブルOSの移植と評価

松本 均 平塚芳隆 徳永靖夫
(富士通研究所)

1. はじめに

Bell研究所で開発されたオペレーティング・システム UNIX は、移植性がよいとの評判であり、近年、海外では各種マシンに移植が実現されている。著者等は、ポータブルOSの研究の一環に UNIX を取り上げ、その移植性を検討するために PDP11/45 からミニコンピュータ PFU-1500 への移植を試行した。

本報告は、移植に際し考慮した留意点及び UNIX の移植性に関するものである。

2. UNIX の構成

UNIX は top-down 設計原理を応用し、C 言語という高級言語でそのほとんどが記述されている。図1に示すように、ユーザインターフェイスとしてシステム・コールと呼ばれる基本ルーチンがある。そしてこの下に UNIX の核の大部分があり、その機能として、ファイル管理、プロセス管理、記憶域管理、及び I/O 処理等の基本的なオペレーションをサポートしている。最も低いレベルは、ハードウェアとのインターフェイスを制御するルーチンであり、割込み処理、プロセススイッチング、メモリマッピングの切り換え、及びユーザプログラムとシステムプログラムとの情報の転送等の処理を行う。この部分はアセンブリ言語で記述され、特にマシンに依存したレベルである。UNIX の核のその他の部分はすべて C 言語で記述されている。

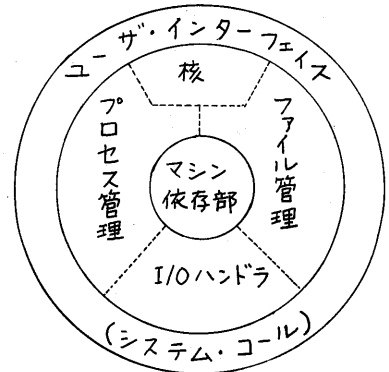


図1 UNIX の構成

3. UNIX の移植方法

上述したように、UNIX 核のほとんどが C 言語で記述されているため、まず、その主言語である C コンパイラの移植を行い、それに引き続き UNIX 核自身を移植するという過程で進められる。

一般に、OS の移植には次の 2 つの方法が考えられる。

- (i) 対象マシンの既存 OS のもとに移植する方法
- (ii) 裸のマシンに移植する方法

Wollongong 工学で方法 (i) により Interdata 7/32 に UNIX を移植した報告がある^[1]。一方、Bell 研究所では方法 (ii) により Interdata 8/32 に UNIX 移植を実現した^[2]。いずれの方法にも長所・短所があり、また、移植作業環境によっても一概に最適な移植方法を決定することは難しい。つまり、方法 (i) では、対象マシンの既存 OS の下で UNIX 移植の際の副産物としてソフトウェアの再利用が可能となるが、一方その既存 OS との整合性に十分な考慮が必要となる。方法 (ii) では、移植過程における他 OS との整合性は考える必要はない反面、言語処理系の修正部分の増大、及び対象マシン上でのデバッグの手法に難がある。

そこで、著者等が選定したのは、後者の方法に 1 つの機能を追加した方法であ

る。即ち、PDP11/45 UNIX上で対象マシンU-1500用UNIXのファイルシステムをディスク上に構築し、U-1500上でデバッグする際、初期段階としてUAS (U-1500既存OS) を利用するという方法である。以下、この方法を選定した理由について述べる。

(1) 移植環境

図2に示すように移植作業環境としてPDP11/45とU-1500にPF6032という当社製のディスクが装備されている。つまり、ファイルシステムのディスク渡しが可能である。

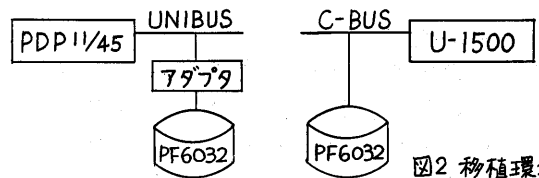


図2 移植環境

(2) 移植道具

豊富なユーティリティをもつUNIX自身を1つのツールとして利用し、使い易いUNIX環境下で移植ツールの効率よい開発ができる。また、U-1500上でのデバッグツールとしてUASのシステムデバッグを積極的に利用することができる。

(3) U-1500 既存OS との非整合性

移植方法(i)を考えた場合、U-1500の既存OSのもとにUNIXの常駐部分を納めることが不可能であり、また、逐次作られたオブジェクトをUNIXのファイルシステムに構築する際、かなりの困難があると予想された。

4. UNIX 移植プロセス

この節では、U-1500用UNIXファイルシステム構築法及びU-1500上でのデバッグ法、また準備したツールについて述べる。著者等の移植方法はUNIXの機能を最大限に利用しようとするものであり、それ故、ほとんどの移植作業はPDP11/45 UNIX上で実施された。

まず、ハーフブートストラップによりPDP11/45上にクロス処理系(クロス・コンパイラ、クロス・アセンブラ、クロス・リンカ・ジェネレータ)が作成され、これと並行してUASのシステムデバッグを利用するため、UAS上にUNIXの基本ブートプログラムに相当するプログラム(boot1)を作成し、デバッグを行った。

図3に示すようにUNIX移植作業中は、最初UASをIPLし、システムデバッグをメモリに常駐させ、boot1プログラムによりUNIXファイルシステムとインターフェイスをとっている。この方法を用い、2段階めのブートプログラム、UNIX核及び各種コマンド等のデバッグを進めた。

デバッグの為、PDP11/45上で構築したファイルシステムをU-1500上で利用する際、1つの変換が必要である。つまり、次節で述べるcharacter(バイト・データ)の扱いが両者のマシンで異なるためである。そこで、1つのツール・ファイルシステム変換プログラム(swab)を作成した。このプログラムは、ファイル制御情報中の3

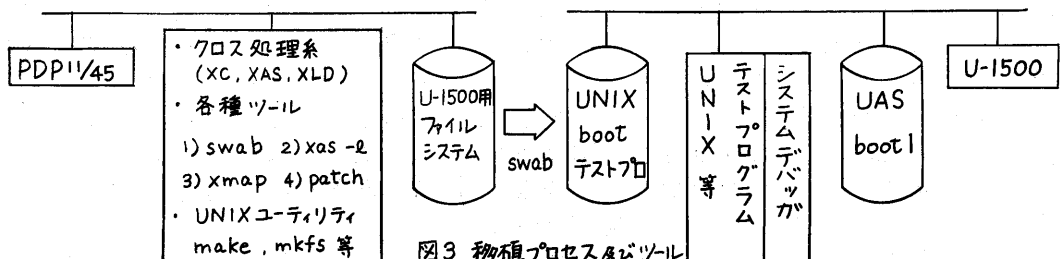


図3 移植プロセス及ツール

バイトからなるブロックアドレス、及びディレクトリ・エントリ名のバイト・スワップを行うもので、両マシン間のファイルシステム変換に利用された。

次に、UNIX 核を修正し I/O ハンドラを作成して、システムとしての各機能（I/O 処理、割込み処理、スタック拡張機能、スワッピング、プロセススイッチング及びファイル管理等）のデバッグを行う。数種のテストプログラムにより単体テストを行った後、各種 UNIX コマンド等を利用した結合テストに移った。しかし、全モジュールを *bottom-up* 的にデバッグするという方法ではない。その理由として、システムデバッグという強力なツールを利用し、システム全体を作り上げた後でも機能ごとのデバッグを進めることができたためである。

そして、バグを発見する度にディスクを PDP 11/45 にマウントし、ファイルシステムを PDP 11/45 用に変換してソースの修正あるいは C コンパイラの修正等を行う。新しいモジュールを作成した後ファイルシステムに登録し、ディスクを U-1500 用に変換して再び U-1500 上でデバッグを行うという繰り返しである。

また、UNIX 移植に際し作成したツールとして、図 3 に示すものがある。1) は前述したファイルシステム変換ルーチンである。2) は U-1500 上でのデバッグにおいてプログラム内の相対位置や機械語等の情報を利用するため、クロスアセンブラにリスト出力機能を付加したものである。3) はロードモジュール内でのシンボルの絶対位置を示すネームリスト出力ルーチンであり、2) と同様 U-1500 上でのデバッグに利用される。更にデバッグ効率を上げるためのパッチルーチン 4) 等がある。また、利用した UNIX ユーティリティとしてはファイルシステム構築ルーチン *mkfs* や変更ファイルに対し自動的に設定手続きを実行してくれる *make* ユーティリティ等を利用し、効率よく作業を進めた。

移植プロセスとして、次に各種 UNIX ユーティリティやコマンドを移植する際、C 実行時ライブラリの修正、及びデバッグを同時に行った。

そして、最終的に U-1500 上で利用できるセルフ処理系の作成（ブートストラップによるセルフ C コンパイラの作成、セルフアセンブラの開発等）、及び UAS に依存しないセルフブート法を実現するという工程で進めた。

5. PDP 11/45 と U-1500 のアーキテクチャの相違点

UNIX 核の移植に関係のある PDP 11/45 と U-1500 との主なアーキテクチャの相違点及び問題点は、次のようになる。

(1) データ表現の相違

両者のマシンにおいてはバイトアドレッシングという性質は同じであるが、図 4 に示すように、マシンのデータ表現の 1 つであるワード内でのバイトデータのアドレス順序が異っている。PDP 11/45 では下位バイトが偶数アドレスをもつが、一方 U-1500 では全くその逆である。両者ともワードデータをアクセスする場合は偶数アドレスを指定する。また、メモリに格納されているオペランドアドレスは、PDP 11/45 の場合偶数アドレスという制限だけであるのに対し、U-1500 では一般にそのデータの整数倍境界、即ち 2 バイトのデータを指定する場合 2 の倍数、4 バイトのデータの場合は 4 の倍数、という制限がある。

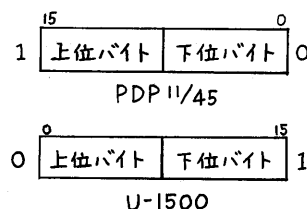


図4 ワードとバイトのアドレス

(2) レジスタの相違

レジスタの相違の1つにレジスタ修飾がある。PDP11/45ではレジスタ修飾のために3ビット使用されて8種類の修飾が可能であるが、U-1500では2ビットしかないために4種類の修飾だけである。更に、汎用レジスタr0, r7はインデックスモードがないといった制限がある。図5にUNIXが利用しているPDP11/45の汎用レジスタとU-1500の汎用レジスタを示す。

PDP11/45では汎用レジスタのセットが2組あり、UNIXはその内の1組(r0~r5)と2つのスタックポインタ(r6)を利用している。PDP11/45では、r6はサブルーチンリンケージに関する命令や割込み時に「ハードウェアスタック」として自動的に利用される。U-1500には8個の汎用レジスタがあるが、このような機能をもつレジスタはない。また、PDP11/45のr7はプログラムカウンタとして使用されるが、U-1500ではICというレジスタがその目的のために別に用意されている。

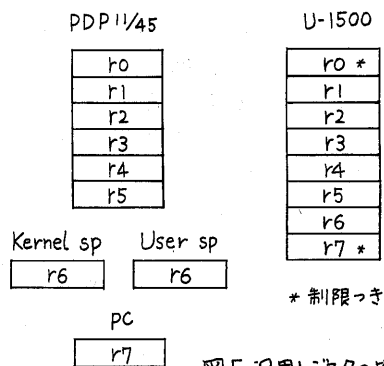
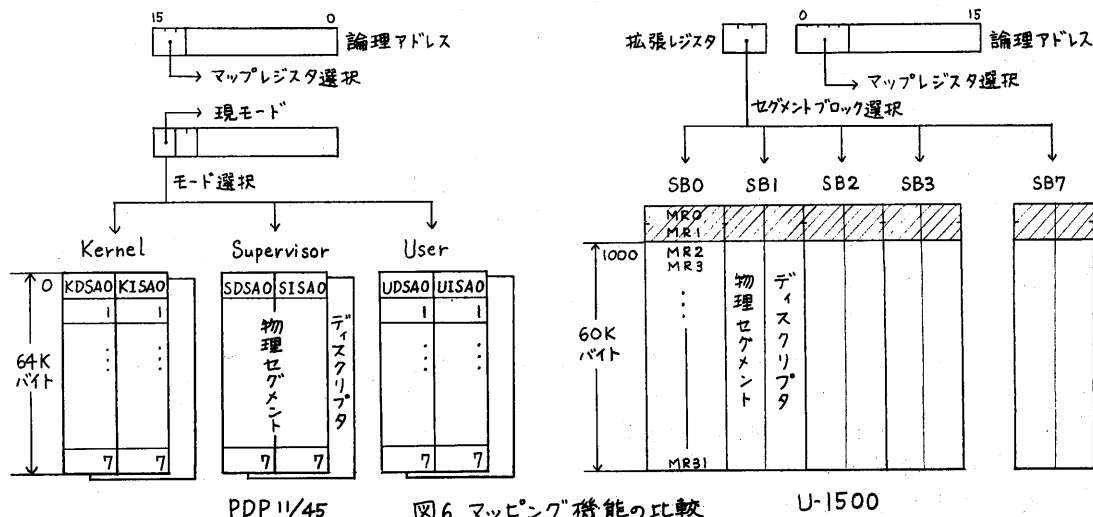


図5 汎用レジスタの比較

(3) メモリマッピング機能の相違

マッピングレジスタは図6に示すように、PDP11/45では各モード（カーネル、スーパーバイザ、ユーザ）にそれぞれ8個ずつインストラクション及びデータ用のマッピングレジスタがある。UNIX環境下ではカーネルとユーザの2つのモードが利用される。メモリマッピングを利用した16ビットの論理アドレスから18ビットの物理アドレスへの変換は、CPUの現モードによりカーネルあるいはユーザが選択され、参照すべき空間（I-空間、D-空間）によりマッピングレジスタ群が決定される。一方、U-1500のマッピングレジスタにはカーネル、ユーザというモードの区別はなく、16ビットの論理アドレスに3ビットの拡張アドレスを付加し、この拡張部により利用すべきセグメントブロック（SB0~SB7）を決定する。各々のセグメントは32個のマッピングレジスタからなる。また、メモリマップにはCPU



のアドレス変換に用いるCPUマップとチャネル装置のためのDMAマップがある。

このように、物理アドレスに変換する方式に違いがある。また、PDP11/45における物理セグメントは64バイト境界のアドレスがセグメントされるが、一方U-1500では2Kバイト境界である。更に図6に示すように論理空間のプログラムの先頭アドレスはPDP11/45の0番地に対し、U-1500では16進1000番地である。

また、16ビットのアドレス空間にUNIXの常駐部分を納めることができないため、PDP11/45では2つの空間（I-空間、D-空間）を利用しているが、U-1500ではデータ領域専用のマッピングレジスタがないので1つの汎用レジスタを代用して同様の機能を実現している。

(4) 割込み処理法の相違

PDP11/45における割込みは各事象に対しベクタアドレスを個別にもつが、U-1500では割込み事象が数種のレベルに分割され、各事象とベクタアドレスが1対1に対応していない。そのため、割込み処理においてソフトウェアによりその原因を判定し、各々の処理ルーチンに制御を渡す必要がある（図7）。また、割込みによるマッピングレジスタの切り換えも拡張レジスタをソフト的に変更することにより行う。

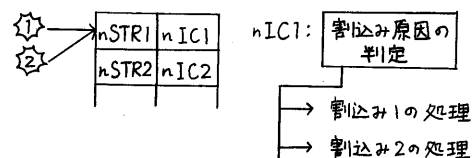
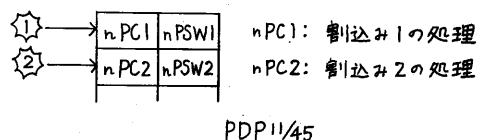


図7 割込み処理 U-1500

6. UNIXの修正点

前節で述べた主なマシンアーキテクチャの相違点を考慮して、UNIX核の修正を行った。表1に示すように、プロセス管理のモジュールの修正がその大部分を占める。修正点の1つは前節(3)項に関連して、U-1500での4つのセグメントブロック（SBO~SB3）をそれぞれPDP11/45のカーネル、ユーザのI-空間・D-空間に対応させたことである。またメモリ管理では、物理セグメントに対応しPDP11/45では64バイト単位であったのをU-1500では2Kバイト単位に変更した。更に、U-1500ではマッピングレジスタは特別なシステム制御命令により扱われるため、オリジナルUNIXにおいてC言語で記述されているモジュールの一部をアセンブリ言語で記述した。また、PDP11/45ではスワッピング等のI/O処理において物理アドレスを指定しI/Oハンドラを起動するが、U-1500ではDMAマップを利用し論理アドレスで処理するため、DMAマップの割当て・解放のルーチンを追加した。このように、大部分がメモリマッピング機能に関連する変更である。

<修正>	修正ファイル / 全ファイル	総行数	修正	追加	削除	計
プロセス管理	9/14	3352	164	71	73	308
ファイル管理	2/11	2920	17	1	0	18
I/O関連モジュール	2/5	1421	24	3	6	33
ヘッダファイル	6/30	1642	34	19	10	63
合計	19/60	9335	239	94	89	422

表1 UNIX修正点

単位: 行

<作成>	U-1500用ソース
I/Oハンドラ { ディスク コンソール ラインプリンタ }	751 { 285 242 224 }
マシン依存部 (アセンブリ)	1104
コンフィグレーションテーブル	74
合計	1929

表2 作成ソース

次にファイル管理においては、前節(1)項に関連した3個のバイトデータをロング(32ビット)長のディスクブロックアドレスに変更するルーチンの修正、及び前節(3)項で述べたプログラムオリジンの変更である。

各ファイルに組み込まれるヘッダファイルの修正点は、前節(2)項に関するレジスタの変更、及び主にハードに依存したパラメータの設定に関する修正である。

そして、表2に示すマシン依存部では、割込み処理、マッピングレジスタの設定等、前節(1)(2)(3)(4)項目全てに関連している。この部分のアセンブリ言語による作成に多くの時間が費された。

このように UNIX 核の修正部分において、総行数9335行中約4.5%の422行が変更されただけである。また、作成部分は1929行である。

7. UNIX 移植性評価

この移植プロジェクトは、デバッグに入ってから2ヶ月後にはほとんどの移植が終了し、UNIX が U-1500 上で各種コマンドと共に稼働している。発生したバグの多くはマシン依存部とCコンパイラのコード生成によるものである。

UNIX 核等の移植工数は、プロセス管理に4人月、ファイル管理に1人月、マシン依存部に2人月、デバイスハンドラ等 I/O 関連に1人月、C 実行時ライブラリ及びブートプログラムに1人月である。この工数の中には各種ツールの作成工数も入っている。

このような短期間で UNIX の大部分が移植できたことは、まさに UNIX 自身がポータブルな OS であることを実証している。しかし、PDP11/45 と対象マシンである U-1500 では、アーキテクチャは異なるが類似する重要な性質をもっている。このことも大巾な変更なしに UNIX を移植することができた要因の一つである。

また、移植方式は、移植対象マシンの構造と移植作業環境により異なると考えられるが、その環境の整備とツールの開発が移植効率における重要な役割をもつと考える。特に主言語であるCコンパイラのテストプログラムを実現すれば、更に効率化を計ることができると思う。現在、U-1500 上での UNIX の定量的性能評価はまだ行っていないが、パフォーマンスには十分満足している。

最後に、UNIX は新しいマシンの環境に、重大なパフォーマンスの損失なしに移植しやすいことから、充分強力で汎用性のあるポータブルな OS であるといえる。

8. おわりに

UNIX の移植作業に協力していただいたパナファコム株式会社の西嶋氏、奥津氏、加藤氏ならびに富士通研究所ソフトウェア研究部の久保氏、村上氏、相川氏に感謝の意を表します。最後に、本研究に終始ご指導をいただいた富士通研究所の遠藤副所長ならびに山田部門長に感謝の意を表します。

参考文献

- [1]. R.Miller, "UNIX - A portable System?", Australian Universities Computing Science Seminar, February, 1978
- [2]. S.C.Johnson and D.M.Ritchie, "Portability of C Programs and the UNIX System" BSTJ, 7-8 1978, pp.2021
- [3]. 徳永靖夫, 平塚芳隆, "コンパイラの移植に関する一考察" ソフトウェア工学 20-1, 1981