ソフトウェア工学 20－1
（1981. 9. 21）

コンパイラの移植に関する一考察

徳永靖夫　　平塚芳隆
（富士通研究所）

1．はじめに

コンパイラの移植については，PASCALとCについての報告が多いようである。PASCALについては，国内でも詳細な報告がある1)。Cについても海外では，いくつかの報告がある2)3)4)し、国内においても移植した例があると聞いている。

本報告は，Cコンパイラのミニコンピュータ PFU-1500への移植に関するものである。

Cコンパイラの移植の目的は，PFU-1500へのUNIXの移植にある。UNIXは，ベル研究所で開発された優れたオペレーティング・システムであり5)，UNIXの殆んどすべてがC言語で記述されている5)6)。したがって，UNIXの優れた機能を自社製のミニコンピュータ PFU-1500で使おうとすると，まずCコンパイラを移植することが必要である。

移植の対象機種として，ミニコンピュータを選んだ理由は，UNIXを移植した際に，UNIXの特徴的な機能を充分に生かしうるのは，ミニコンであろうと判断したことによる。

2．Cコンパイラ

筆者等の知っている範囲では、現在、ウェスタン・エレクトリック社から，購入できるCコンパイラには，表1のようなものがある。

表1の1から4までは，デック社のPDP-11で稼動するUNIXに付属しているCコンパイラである。5番目のものも，やはり，デック社のVAX-11で稼動するUNIXに付属しているCコンパイラである。6番目のものは，IBM-360/370シリーズで稼動するものであり，7番目のものは，Honey-well 6000シリーズで稼動するものである。

1番目のものの keyword の数は表1にあるように，23箇であり，2番目のものは26箇で，unsigned，union，typedef が追加されている。3番目と4番目のものは，28箇で，更に，short と enum が追加されている。5番目と7番目については，筆者等には詳細は不明である。6番目は，1番目の中の long がなくなって，bliss，fortran，asm が追加されている。

本報告で，移植したCコンパイラは，3番目のものと4番目のものを折衷したものである。

Cコンパイラの構造は，ほぼ図1のようになっている。図1の右にあるアセンブラとローダとはCコンパイラではないが，Cコンパイラの主ルーチンから呼ばれている。

Cコンパイラは，殆んどC言語によって記述されている。表1の1番目のコンパイラでは，図1のPass1とPass2の一部でアセンブラで記述された部分があったが，表1の2番目と3番目になると，これらもC言語で記述されている。

なお，図1の右にあるアセンブラは，UNIXアセンブラで記述されており，ローダは，C言語で記述されている。

No.	Cコンパイラ	開発者	ハード	keyword数	備考
1	UNIX V6 の Cコンパイラ	Ritchie	PDP-11	23	
2	PWB/UNIX の Cコンパイラ	?	〃	26	No.1 に, unsigned, union, typedef を追加
3	UNIX V7 の Cコンパイラ	Ritchie Reiser	〃	28	No.2 に, short, enum を追加
4	〃	Johnson	〃	28	〃
5	UNIX V32 の Cコンパイラ	?	VAX-11	?	
6	C/370	Lesk	IBM 360/370	25	No.1 の long を削除 bliss, fortran, asm を追加
7	C/6000	?	Honeywell 6000	?	

表1. C コンパイラ の 比較

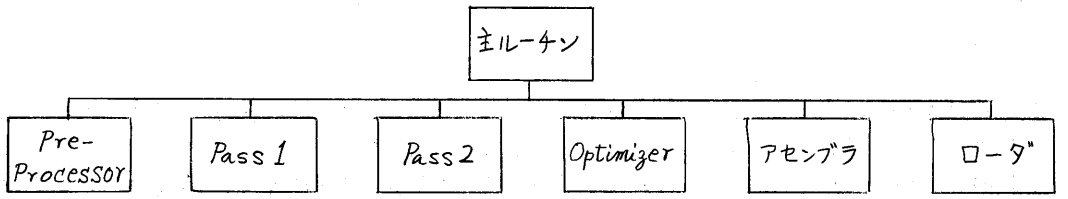

図1. C コンパイラ の 構成

3. 移植の方式

コンパイラを、ある機種から他の機種へ移植する方法は色々あると思われるが，筆者等がCコンパイラを，PDP-11/45 からPFU-1500へ移植するに先立って検討の対称とした移植の方式を図2に示す。

図2の一番上の行は，ウェスタン・エレクトリック社から購入したUNIXをDEC社のPDP-11/45で稼動させて，C言語で書かれたプログラムをCコンパイラでコンパイルして，PDP-11/45で実行させるときの概念図である。

図2の2行目以下の三つの方式は，当社のPFU-1500へUNIXを移植するに先立って，C言語で記述されたプログラム（Cコンパイラも含めて）をPFU-1500上で稼動させるための変換の方式であり、これらが検討の対称として候補に上った。

右端に1と書かれている方式は，PFU-1500のファーム・ウェアの修正を主とした方式で，この場合，コンパイラ，アセンブラ，ローダも修正が必要なことは勿論であるが，これを最少とするような方式である。ただし，ファーム・ウェアの右にあるPFU-1500には，一切修正を加えないこととする。ここでは，PFU-1500のオペレーティング・システムの機能は使わないで，PDP-11/45の上で，PFU-1500で稼動するロード・モヂュールを作り，これを，PFU-1500に逐次構築して少

く方式である。

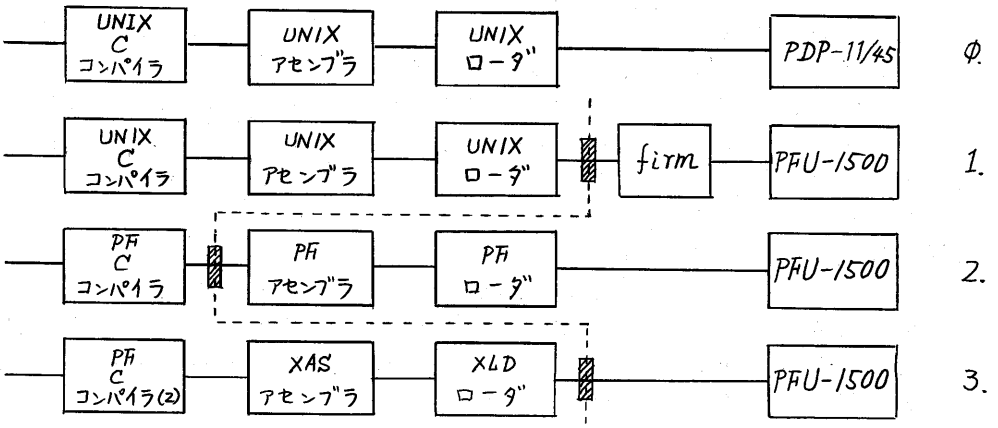

図2. 移植の方式

図2の右端に2と書かれている方式は，Cコンパイラを主として修正し，PFアセンブラとPFローダは若干の修正に止める方式である。ここでは，ファーム・ウェアもPFU-1500も一切修正を加えないこととする。この方式はPFU-1500のオペレーティング・システムの機能を有効に利用して，UNIXを逐次，PFU-1500に構築してゆく方式である。

図2の右端に3と書かれている方式は，1の方式と同じく，PFU-1500のオペレーティング・システムの機能は殆んど使わないで，Cコンパイラとともに UNIXアセンブラ，UNIXローダの修正を行ない，裸のPFU-1500で稼動するロード・モヂュールを，PDP-11/45 上のUNIXで開発して，PFU-1500に逐次構築して行こうという方式である。

ベル研究所で行なわれたCコンパイラとUNIXの移植は，図2の3に近い方式を採用している。2) すなわち，裸の Interdata 8/32 に，UNIXアセンブラを作ることから始めて，最終的には，UNIXシステムが Interdata 8/32 で稼動するまでの開発に成功している。

一方，Wollongong 大学では，図2の2に近い方式を採用している4)。ここでは，Interdata 7/32 のオペレーティング・システムの下で稼動するCコンパイラを開発し，このCコンパイラを使って，UNIXを少しづつ，Interdata 7/32 に移して行っている。

二つの論文を比較してみると，Wollongong 大学の方が，短期間に，ベル研究所よりも早い時期に移植を完了している。

もとに返って，図2の3つの方式について筆者等の比較検討の結果を述べると次のようになる。

4. 移植方式の比較

図2の3つの移植方式を比較検討してみると，1の方式は，まず，誰れもが考えつく方式であり，しかもメーカーが行う移植方式としては魅力がありそうであるが，どう考えても，ファーム・ウェアのデバッグにかなりの工数がかかりそうで断念せざるを得なかった。

次に，第2の方式であるが，この方式は，文献4）によると，約6週間で，Cコンパイラのブートストラップを終了している。このことは，第3の方式を採用したと考えられる文献2）等にある2～3ケ月に比べると約半分であり，一見易しそうであるが，筆者等の場合は，問題があることが判った。

問題の第1は，最終目標であるUNIXの移植，特に，第七版のUNIXの移植を考えると，16ビットのアドレス空間に，UNIXのオペレーティング・システムの常駐部分を納めることができず，どうしても，データとテキストを別の空間に納める必要があるが，現在のPFU-1500のアセンブラには，その機能がなく，UNIXアセンブラの機能を使わなくてはならなかった。

次に，UNIXのオブジェクト管理と，PFU-1500のオペレーティング・システムのそれとの整合性が悪るく，PFU-1500のオペレーティング・システムの下で少しづつ作られたファイルをUNIXのファイル・システムとしてまとめる場合，可成りの困難があると考えられた。

更に，文献4）のように，移植対称マシンが32ビットのアドレス空間をもっているInterdata 7/32等であれば，Interdataのオペレーティング・システムの下で，同じ空間に，一時的にでも，shellとか，入出力ルーチン等を同居させることも可能であり，種々のテストもできると思われるが，16ビット・マシンで同じことをするには無理があると考えた。

最後に，C言語の特徴の一つであるinclude文を制限なしに使おうとすると，アセンブラも，ローダもUNIXに似た機能をもったものが必要であると考えた。

結局，第3の方式を採用したわけであるが，開発工数についても予測をしてみた。予測の結果は，図3のようになるのではないかと思われる。

第1の方式は，修正しなければならない量としては最も少ないと予測されるが，デバッグ等の開発環境は最も悪るいものと予測される。

第2の方式は，修正量としては，第1の方式に比べれば多くなるが，開発環境の方は良くなることが予想される。

第3の方式は，修正しなければならない量としては，3つの方式の中で最大であると考えられるが，一方，開発環境というか，デバッグのためのツールは，UNIXという良い環境の下で，大部分を行うわけであるから最も良いと考えられる。

方式	修正部分	開発環境	開発工数
1	小	×	大
2	中	△	中
3	大	○	小

図3．移植方式の比較

これらを総合すると，開発工数としては，むしろ第3の方式が最も少ないと考えられる。

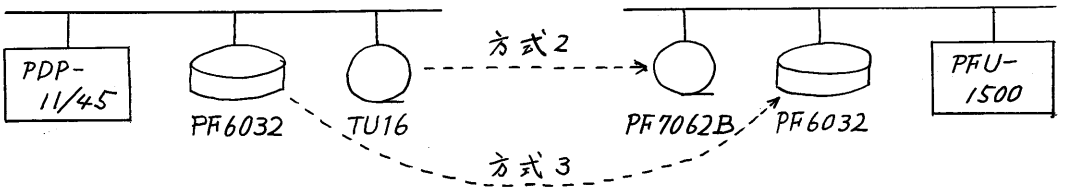

図4．　移植のためのシステム構成図

最後に，図4に方式2と方式3を実施するためのシステム構成図を示した。方式2では，コンパイル出力であるアセンブラ・ソースをASCIIからEBCDICに変換して磁気テープ経由でPFU-1500に渡し，PFU-1500のオペレーティング・システムの下で，これを入力として，アセンブルし，オブジェクト・モジュールを作って行く。　一方，方式3では，PDP-11/45とPFU-1500に同一形式の当社製のディスクであるPF6032が装備されていることを利用して，PDP-11/45で作り上げたロード・モヂュールを，PDP-11/45のディスク上に，UNIXのディスク・ファイル・システムと同じ形式で構築して，このディスク・パックをPFU-1500に移して，PFU-1500でブートし，稼動させようとするものである。　もちろん，方式3においても磁気テープ渡しができないわけではない。2)　たゞし，この場合，上位バイトと下位バイトの入れ替えは，磁気テープ渡しの場合と逆になるようである。すなわち，磁気テープ渡しでは文字列はそのまゝで，その他は上位バイトと下位バイトを入れ替えねばならぬが，ディスク渡しでは，文字列を入れ替えて，その他はそのまゝで良いと考えられる。

ついでながら，図2の方式3のローダは，前述のようにC言語で書かれたUNIXのローダの数行の変更で，PFU-1500のロード・モヂュールを出力するクロス・ローダ（図2の方式3のXLD）を作ることができる。　また，方式3のアセンブラは、アセンブラで記述されているが，一般のアセンブラに比べて大きさが小さく変更は容易なようであるし，分岐命令も分岐先との距離により1語命令を出すか2語命令を出すかの判断をする等の便利な機能が方々にある。

要するに，図2に提案した3つの移植方式を考えた場合，移植対称マシンの構造と移植作業環境とを勘案して，よい方式を撰ぶべきだと考える。　筆者等の場合は，方式3を撰んだ。

5.　PDP-11/45　と　PFU-1500　との主な相異点

コンパイラの移植に関係のあるPDP-11/45とPFU-1500との主な相異点は次のようになる。

（1）　簡略記号の相異

オペレータの機能としては全く変らないが簡略記号が異るものがある。　これらは、種類は少ないし，修正も単純であるが，修正の量としては最も多い。

（2）　レジスタの相異

レジスタの相異の主なるものはレジスタ修飾にある。　PDP-11では，レジスタ修飾のために3ビットが使用されているが，PFU-1500では2ビットしか使われていない。　このため，PDP-11のレジスタは8種類の修飾が可能であるが，PFU-1500では4種類しか可能でない。　このため，PDP-11にあるオート・デクリメント，すなわち，−(R)，ならびに，デファード，すなわち，@(R)+，@−(R)，@X(R)　は，PFU-1500では使えない。

また，PFU-1500では，4番目のレジスタが特殊で，このレジスタに限り，インデックス・モードがない。　その他，命令カウンタと7番目のレジスタは別に利用できるようになっているので，使い方に制限はあるが，PDP-11に比べてレジスタ数としては勝えている。　更に，PFU-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，これを代用するために，汎用レジスタを1ケ割り当てなくてはならない。

(3) 比較命令の相異

PDP-11とPFU-1500の比較命令の相異点は，PDP-11では，左オペランドから右オペランドを引いた差が評価されるのに対し，PFU-1500では，右オペランドから左オペランドを引いた差が評価される所にある。 この相異は，PFU-1500の方が一般的で，PDP-11の方が特殊であると言われている。

(4) 演算と条件分岐

ある演算を行って，その結果によって分岐する場合，PDP-11では，演算には論理的（符号なし）と算術的（符号つき）の区別はなく，分岐命令の方に論理的と算術的の区別がある。 一方，PFU-1500では逆で，演算に論理的と算術的の区別があって，分岐命令には区別がない。

(5) バイト命令の相異

バイト命令で直接数値を扱う場合、PFU-1500では直接数値のあとに，*256を書かなければならないが，PDP-11では，このようなことは不要である。

また，バイト命令で，レジスタに転送する場合，PDP-11では，下位バイトに転送されて，そのバイトの符号が上位バイトに拡張されるが，PFU-1500では，下位バイトに転送されることに変りはないが，上位バイトは，もとのまゝで変化しない。

(6) 論理積の相異

PDP-11では，直接，論理積を作る命令はなく，bic(bit clear)とコンプリメントを使って論理積を作っている。 PFU-1500では，このような面倒なことをしなくても直接論理積を作ることができる。

(7) 排他論理和の相異

PDP-11では，排他論理和を作る場合に，オ1オペランドをレジスタにロードしておく必要があるが，PFU-1500ではその必要はない。

(8) シフト命令の相異

PDP-11のシフト命令は4つしかなく，1ビットの右シフト命令と1ビットの左シフト命令，および任意ビットの左シフト命令が1ワード用と2ワード用とである。 いずれも，arithmeticシフトで，logicalシフトを行いたいときは厄介である。 また、任意ビットの右シフトを行いたいときは，シフト数を負の値で与える必要がある。 これに比べて，PFU-1500のシフト命令は8つあり，logicalシフトも簡単である。

6. コンパイラの修正

前節で述べたような相異点を考慮して，コンパイラの修正を行った。 まず，図1の主ルーチンとしては，表1のNo.4のものを変更して用いた。 これは表1のNo.3のものと比べて，実行速度がおそいという難点はあるが修正が容易であり，開発段階では，この方が有利であると判断したからである。 最終的には，No.3のものでまとめる予定である。 なお，この主ルーチンの修正は，もともとのコンパイラと開発中のコンパイラが同じ環境に雑居しているために必要となったもので最終的には不要なものである。

次に，図1のPre-Processorであるが，これは，表1のNo.3もNo.4も同じものものを使っている。 そして，これについては，修正は全く行っていない。

次に，図1のPass1であるが，これは，表1のNo.3のものを修正して利用して

いる。 表1のNo.4のものは，大きさが大きいため，今回は採用しなかったが，大きさを小さくする方法も検討してみる必要がある。 Pass 1で修正したのは，前節の(2)のレジスタの相異によるものだけである。 この修正で，ユーザが実質的に利用可能なレジスタは3ケから1ケになっている。 この減少に対処するため，インデックス修飾のできないレジスタの使用も考えられるが，レジスタ2ケ減少による実行速度の減少を数ケのプログラムで実測した所では，実行速度の変化は認められなかったので，対処していない。 なお、ユーザは，この減少に気がつかないようになっている。 修正行数は1行である。

次に，図1のPass 2の修正であるが，前節で述べたPDP-11とPFU-1500の主な相異点と，その他，細かい相異点のすべてがここで吸収される。 これらについては、表4にまとめた。 修正行数の最も多い簡略記号の修正は量は多いが，修正は単純である。修正要因の項目にはないが，文字列のアドレスが，PDP-11とPFU-1500では異っており，注意を要する。

	修正要因	修正行数
1	簡略記号	525
2	シフト	150
3	レジスタ	67
4	論理積	50
5	その他	26
	計	818

表4. 修正要因と修正行数

さらに、図1のOptimizerの修正であるが，ここでは，簡略記号の相異による修正が殆んどであるが，この他に，PFU-1500には，MCタイプという便利な命令があり，0から15までの直接数値の転送，加減算，比較が1ワードでできるので，これを有効に利用するための修正を行った。 Optimizerをかけることにより，オブジェクトは平均して，約92%位になる。 この数値は，もともとのOptimizerより若干良くなっている。

7. まとめ

移植方式として，代表的と考えられる3つの方式を想定し，これらを比較検討して，その一つを選定した。 この選定は，移植対称マシンと移植作業環境等によって変ると考えられる。 また，この選定された方式につき，移植を実施するにあたり配慮すべき主要な点を述べた。

クロス・コンパイラ，クロス・アセンブラ，クロス・ローダができた時点で，コンソールとだけ入出力動作のあるプログラムをいくつか作成し，これが，PDP-11/45の上でも，PFU-1500の上でも同じ動作をすることを確認した。

現在、このクロス・コンパイラ等を用いて，PFU-1500の上で動作するUNIXを構築しつつある。 UNIXがPFU-1500で稼動し始めれば，文献1) 等で紹介されている方法を用いて，PFU-1500の上で動作するコンパイラやアセンブラを作成することが可能であると考えている。

コンパイラの修正は、先にも述べたが、主ルーチンが71行中26行，Pre-processorが，1417行中修正なし，Pass 1が4085行中1行，Pass 2が6123行中818行、Optimizerが1618行中246行で，移植工数は約3人月である。 この工数の中には，中間ファイルの表示プログラム等のデバッグ・ツールの作成工数も入っている。

図1のアセンブラの修正も，コンパイラの移植で，方式3を採用した以上、密接な関連をもっている。 ここでは，約3600行中約1400行を修正している。 修正の率から言うと，アセンブラが一番大きいが，修正の工数は約3週間であった

。図1のローダは，1300行のうち7行が修正された。修正工数は1日もかかっていない。

以上，開発工数について述べたが，先にも述べたように，C言語の外部仕様に制限を加えることなく，しかも，コンパイラ，アセンブラ，ローダの分担をうまく行うためには，開発工数もさることながら，方式3がもっとも良いと考えられる。特に、条件分岐の場合、分岐先との距離によって1ワード命令を使うか2ワード命令を使うかの判断は，アセンブラが行うのが好ましく，これをコンパイラが行なうのは，コンパイラ，アセンブラを通して考えたとき良いとは考えられない。このことは，UNIXアセンブラではうまくできていると思う。また，アセンブラを使用する際に，フラグの選択によって，未定義シンボルをexternalにする機能をもたせることは，アセンブラでは，わずかな変更で可能となるようであるが，この機能がないと，コンパイラは，すべての未定義シンボルに対して，external宣言を出力しなくてはならず，コンパイラ出力はもとよりコンパイラ自身も大きくなる。更に，include fileの記述に制限を加えないとかC言語の記述を容易にするためには，アセンブラやローダの機能に頼わなくてはならない所が多いと考える。

要するに，同一の開発者によって開発されたと考えられるUNIXのコンパイラ，アセンブラ，ローダの機能分担の良さをできるだけ保存した形で移植が行なわれるのが望ましいが，環境条件でこれができないこともありうるようである。

8. おわりに

UNIXの導入の当初からご指導をいただいている東京大学の石田晴久助教授ならびに，コンパイラのご指導をいただいた渡辺勝東京大学名誉教授に厚く御礼申し上げるとともに，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君，加藤君，奥津君ならびに富士通研究所ソフトウェア研究部の久保君，松本君，村上君に感謝の意を表します。奥津君には，短期間にクロスアセンブラの開発をあわせてお願いした。最後に，本研究に終始ご指導ご鞭撻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深甚なる感謝の意を表します。

参考文献

1) 疋田輝雄，"コンパイラのキットを用いたPASCALの移植"，
日経エレクトロニクス，1976，12，13，pp100
2) S.C.Johnson and D.M.Ritchie, "Portability of C Programs and the UNIX System", BSTJ, 1978, 7-8, pp2021
3) S.C.Johnson, "A Portable Compiler: Theory and Practice", Proc. 5th ACM Symp. on Principles of Programming Languages, January, 1978
4) R.Miller, "UNIX - A Portable System?", Australian Universities Computing Science Seminar, February, 1978
5) 石田晴久，"ベル研究所の軽装OS — UNIX"
情報処理学会，Vol. 18，No.9（Sep. 1977） p.942
6) 徳永靖夫，平塚芳隆，"コンパイラの移植性に関する一考察"，
昭和56年度電子通信学会総合全国大会，6-66