V60 UNIX System V 移植と機能強化について

1C-3

川又 滋* 水橋 由紀子* 和田 良彦** 赤羽 健治*** 寺本 雅則*

*日本電気㈱ ﾏｲｸﾛｺﾝﾋﾟｭｰﾀｿﾌﾄｳｪｱ開発本部
**日本電気技術情報ｼｽﾃﾑ開発㈱
***日本電気アイシーマイコンシステム㈱

1. はじめに

近年、32ビットマイクロプロセッサ(μP)をCPUとした多品種のUNIXシステムが開発されている。こうしたUNIXの大量生産に対応するためにはUNIXの移植技術の見直しも必要になる。本論文では日本電気オリジナルμPのV60用に現在開発中であるUNIXについて、その設計方針を述べる。

2. 背景

μPの高機能、高性能化に伴い、チップ内にメモリ管理機構(MMU)を取り込んだり、専用のMMUチップが用意されたμPが多くなっている。MMUの仕様はOSを開発する際に重大な影響力を持つ。それがチップに取り込まれることは、CPUの選択がOSのMMUまわりの実現方式を規定することになる。一般的に移植性が良いと言われているUNIXでも移植する際の改造は、このMMUを制御をする部分に集中している。しかし、一度V60に移植したUNIXのMMU制御部は、それ以降開発されるV60 UNIXシステムに共通にV60汎用部として使える。

一方、システム固有である周辺装置では、ディスク等の二次記憶装置の高速、大容量化、またマウス等の新しい装置の接続が進んでいる。UNIXではこれらを制御するドライバはシステム毎、装置毎に新規に開発しなければならない。

このような状況を総合すると、UNIXの移植を大きく二つのフェーズにわけられる。一つは、あるCPU(MMU)のための移植(汎用移植)と、もう一つはそのCPUを使ったあるシステムのための移植(固有移植)である。V60を使ったUNIXシステムの大量生産とは、固有移植の繰返しであり、今後はこのフェーズが益々重要になることが予想される。(図 1 参照)

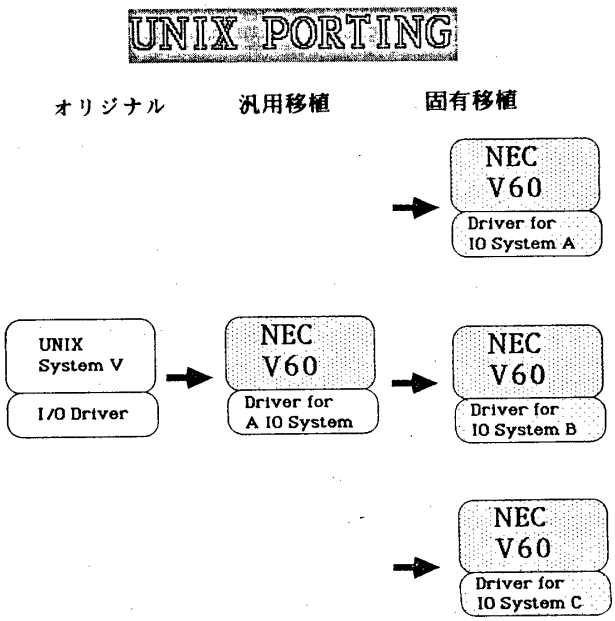

図 1 汎用移植と固有移植

3. V60汎用部

3. 1 特徴

V60 UNIXの汎用部は System Vを用いている。これはデマンドページングを採用したり、シェアードメモリ等のMMUに依存した機能も提供している。V60はページ方式の多重仮想空間をチップ内のMMUがサポートしており、命令再実行機能も持っていて、System Vの特徴を活かせるアーキテクチャとなっている。

3. 2 標準化

汎用部はUNIX自身の外部仕様を決定する部分でもある。最近System V Interface Definition(SVID)が発表され、AT&T版のUNIXは標準仕様が決まりつつある。これはUNIXを名乗るための条件ともなる。そこでこの仕様を満しているか、どうかを検証するシステムを開発した。これは標準マシン上で動作させた結果と、ターゲットマシンの結果を比較して検証するというものである。これにより

PORTING UNIX SYSTEM V TO THE V60 SYSTEMS

Shigeru Kawamata*, Yukiko Mizuhashi*, Yoshihiko Wada**, Kenji Akahane***, Masanori Teramoto*
*NEC Corporation Microcomputer Software Development Lab.
**NEC Scientific Information System Development Co. Ltd.
***NEC IC Microcomputer Systems, Ltd.

汎用移植はもとより固有移植の正当性も迅速にチェックできる。

4．ドライバ作成支援

今、ドライバの作成支援が求められる理由には

1）前述の固有移植を効率的に行う。

2）小規模なUNIXを個人ベースで使用するようになり、多様な周辺装置を使いたい、そのためのドライバの書きかたを知りたい、という要求が高まっている。

がある。これは現在のUNIXのドライバを書くのに

a）OS部とのインタフェースを理解する。

b）ドライバのモジュール構成及びデータ構造を理解する。

c）装置の仕様を理解する。

の三つの作業が必要であり、ドライバの中身を知らない人には難しいという問題があるからである。a）、b）はUNIX上の話であり、ドライバを書く人には関係がない。そこで、最終的には装置のマニュアルがあれば、UNIXを知らなくても、さらにはC言語を知らなくても、ドライバが書けることを目標にし、典型処理の多いディスク系ドライバから作成支援の検討を開始した。具体的にはドライバ中の共通部分、固有部分を分類した。(図 2 参照)

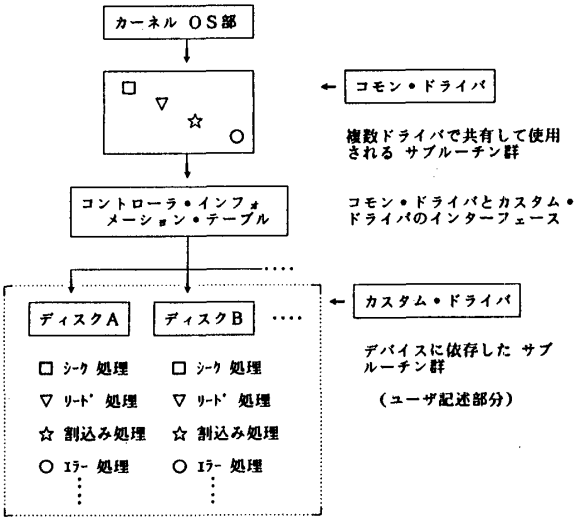

図 2　ドライバの切り分け

i）共通部分の切り出し(コモンドライバ)

ディスク系ドライバに共通な処理(IO要求を待ち行列化する機構、ヘッド移動の最適化等)を汎用ルーチン化する。

ii）装置への処理要求の関数化(カスタムドライバ)

装置固有の処理(リード・ライトのコマンド、エラーステータスの見方等)を関数化し、その仕様を規定する。

これによってユーザはii）の仕様に従って関数を記述するだけでディスクドライバができあがることになる。これに加えて、以下のメリットが得られる。

- 各ドライバの共通部分を一本化することにより、ドライバ全体のプログラムサイズを小さくできる。
- 装置の増設に伴うドライバ以外(割込みベクタ等)の変更が少い。

また問題点としては、

- 現在、五つ程のディスク系ドライバを基に検討しているが、これ以外の装置についてii）の仕様で充分なのかの確認が難しい。
- サポートできるディスクの種類を増やすことは、種類識別のオーバーヘッドを増すことになり、これはディスク種類数とのトレードオフとなる。

がある。これについては、より多くのドライバを参考にし、本質的な情報を抽出したい。

4．まとめ

V60システムへのUNIXの移植を通して移植作業のフェーズを見直し、ドライバ作成支援の重要性を認識した。ディスク系ドライバの分析からユーザ記述部インタフェースの規定を行った。これによって固有移植、周辺装置の追加作業の簡便化を図った。

今後はサポートできる装置(ディスク系以外も含めて)の範囲を広げていく予定であるが、この方式で全てのドライバをサポートするのは不可能なので、知識ベースの援用、ドライバ記述言語の仕様等を検討していくつもりである。

参考文献

[1]　米田他：V60 OS開発用デバッグシステム、本予稿集 1C-2

[2]　広屋他：PC-UXを用いたデバッグ環境の改善 第31回情処全国大会 6F-5, 1985

[3]　"System V Interface Definition", Spring 1985 AT&T