UNIX ディスクドライバ カスタム化インタフェースの実現

3P-4

国松 浩司*, 川又 滋**, 高橋 久**, 宮内 哲夫***
(*日本電気技術情報システム開発 **日本電気 ***日本電気マイコンテクノロジー)

1. はじめに

UNIX システムの中でデバイス・ドライバは各周辺装置に固有な物であり、OS の移植、周辺装置の追加などによるドライバの作成機会は多い。

本論文では、周辺装置の多種・多様化に伴なうデバイス・ドライバの開発を支援する機構のうち、ディスク・ドライバ作成支援に関する報告を行なう。

2. 背景

ドライバ作成支援は、次の様な要求から考えられた。

(1) 一般的にUNIXは移植性の良いOSだと言われているが、移植の際にはターゲットシステムのメモリ管理機構 (MMU)、例外・割込みのハンドリング機構や、I/O システム構成により、かなりのソースコードに改造を加えなければならない。しかし、MMUや例外・割込みなどは、そのシステムの CPU に依存している部分であるため、一度それ用の制御モジュールを開発すれば、同一 CPU を使用する限りは汎用モジュールとして使用することが出来る。これと同様に周辺装置を制御するためのドライバも一度開発すれば、I/O システム構成が変わらないかぎり使用することが出来る。しかし、同一 CPU を用いて様々な (I/O) システム上に UNIX を展開して行く時、その度に周辺装置に準じてドライバを開発する事はなるべくなら避けたいものである。

(2) 近年の様々な周辺装置の出現に伴なって、周辺装置の追加・入れ換えの要求が高まっている。しかし、そのためのドライバを作成する為にはドライバのモジュール構成やデータ構造、 OS とのインターフェイス部分、装置の仕様などについての知識が必要であり、その事がユーザにとって大きな負担となる事が多い。この負担をなるべく軽減したい。

このような要求からドライバ作成支援システムの開発を始めた。要求される機能としては『ドライバを作成する際に、局所化された部分のみをコーディングすれば、周辺装置に準じたカスタム仕様のドライバを作成することが出来ること。』である。

我々は、まず、ディスク・ドライバに対する支援システム DDCS (ディスク・ドライバ・カスタマイズ・サポート・システム) の開発を行なった。これは、まず、従来のディスクドライバを比較・検討し, ドライバの処理を共通的な処理の部分と装置に依存した処理の部分とに分類する事から始めた。

3. 実現方法

3.1 構成

DDCSの構成は図1で表わされる。(図1参照)

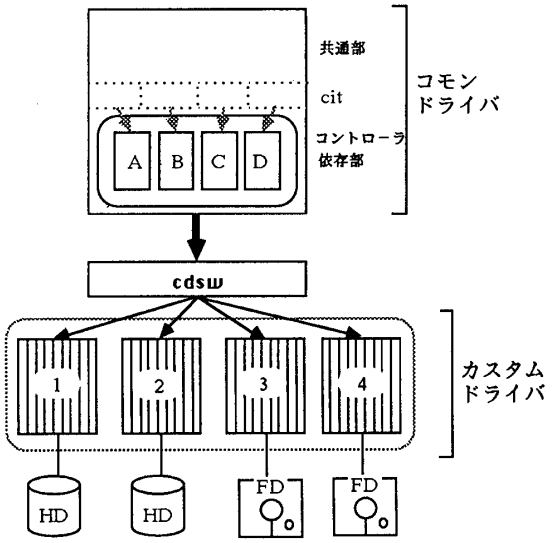

図1 : DDCSの構成

DDCSでは各ディスクコントローラ毎にユニークな ID - コントローラID - を持つ。また、コントローラIDの分だけ各装置の仕様・コントローラの I/O ポートアドレス・フォーマットテーブルのアドレス他などの情報をもつテーブル cit (コントローラ・インフォメーション・テーブル) があり、装置の識別を行なう。コントローラIDは cit へのインデックスである。

DDCSのモジュール体系は、(1) コモンドライバ, (2) カスタムドライバの2つに分けられる。

(1) はDDCSで全ディスクドライバに共通なものであり, ディスクドライバの共通的な処理を分担する部分である。

コモンドライバは更に大きく2つの部分、仕様の異なる多種のディスクコントローラに対応するためのコントローラ依存部分 (図1 : A, B, C, D) と真の共通部分とに分けられる。コントローラ依存部分の分類 (図1 : A, B, C, D) の要因としては、「 R/W 時にソフトウエアでシークが必要」, 「ユニット間の R/W 時排他制御が必要」, 「複数ユニットのパラレルシークが可能」等々が考えられ、これらの要因で処理を切り分けることによって多種のコントローラに対応する汎用モジュールを構成することができた。

コモンドライバ・共通部からコモンドライバ・コントローラ依存部分の処理 (例. 図1 : A) を選択する時のインデックスとして各 cit 内にコントローラタイプ情報を保持しているフラグを持っている。

(2) はコントローラID 毎にあって、イニシャライズ・R/W ・シークなどのコントローラのプログラミング, 各ドライバ特有の処理などを行う。カスタムドライバの作成は、後述の

UNIX disk driver custom interface

*Hiroshi Kunimatsu**, *Shigeru Kawamata***, *Hisashi Takahashi***, *Tetsuo Miyauchi****
**NEC Scientific Information System Development, Ltd.*
***NEC Corporation ****NEC Microcomputer Technology, Ltd.*

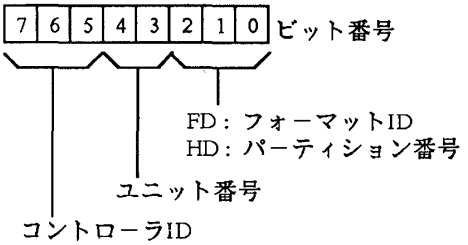

図２：マイナ番号のビットアサイン

```
(*init)()
(*seek)()
(*read)()
(*write)()
(*clrint)()
(*recal)()
(*sense)()
(*dmaopen)()
(*dmaclose)()
(*hrstr)()
(*timer)()
(*cylinder)()
(*format)()
(*senseintr)()
(*uioctl)()
(*open)()
(*exit)()
(*volchange)()
(*err)()
(*idat)()
(*rqsense)()
(*statuscheck)()
```

図３：cdsw

カスタムドライバの関数仕様に従う。

実際には、装置に対するサービスの要求がほとんどであるので、装置の仕様に応じ、装置のマニュアルに準じてコントローラのプログラミング・シーケンスを書き並べたものである。

コモンドライバはディスクドライバの共通的な処理を行ない、装置に対して（装置に依存した）処理要求を行なう場合はカスタムドライバ内の処理関数を呼び出すことによって各周辺装置の仕様に応じた処理を行なうことが出来る。ドライバ開発者は、この局所化されたカスタムドライバのみを作成することになる。

3.2 両ドライバのインターフェイス

DDCSでは従来のメジャー番号は意味を持たず、マイナ番号に装置のコントローラID ,ユニット番号、フォーマットID、パーティション番号などの情報を持たせるようにしている。（図２参照）

コモンドライバとカスタムドライバの間にはデバイスコントローラと１対１に対応したカスタムドライバスイッチテーブル（ cdsw ）があって，コモンドライバがカスタムドライバのサービスを受けたい場合は，コントローラ ID をインデックスとして cdsw からカスタムドライバ関数のアドレスを参照して間接呼出しを行う。　cdsw は図３の様な構造となっている。

各カスタムドライバ関数の関数仕様は呼び出し時のパラメータ・返却値などが決められており、カスタムドライバは各装置毎に独立したものではあるが、各関数は統一されたインターフェイスを持つ。

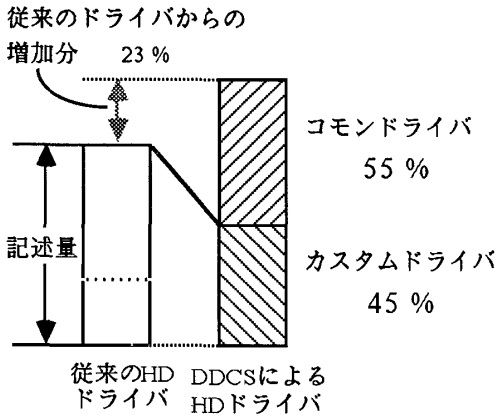

図４

4. 効果

我々は、まず、V60MPU ＋ PC9800 をターゲットとしたUNIX システムにDDCSのインプリメントを行なった。更に、システムバスにマルチバスをもちいた V60MPU ＋ ディスクコントロールボードの構成でハードディスクドライバを開発した。

従来の HD ドライバとDDCSによって開発された HD ドライバとの『ドライバ記述量』、『コモンドライバとカスタムドライバのライン数の比率』、『従来のドライバからの全ライン数の増加率』を図４に示す。棒グラフの長さは絶対コーディングライン数を示しており、従来の HD ドライバに比べてDDCSを採用することにより、ドライバ全体のライン数は増加しているが、実際のコーディングライン数は減少している。更にDDCS上でディスクドライバを開発する場合、コモンドライバが全カスタムドライバで共通な分だけ従来に比べてドライバ全体のオブジェクトサイズが減少する効果を持つ。

5. おわりに

DDCSは，ディスクドライバ作成時のカスタム化に対するインタフェースを提供するものである。現段階のDDCSは、ある程度ドライバの知識を持っている事を前提としたドライバ開発サポートであるが、将来的には、周辺装置（デバイス・コントローラ）のマニュアルがあって、その仕様が理解できれば誰にでも容易にドライバが作成出来るような仕様にしていくつもりである。

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、知識ベースの活用、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。

謝辞

日頃御指導いただいている日本電気（株）壷屋課長、日本電気技術情報システム開発（株）加藤課長に感謝いたします。

参考文献

"System V System Administrator's Reference Manual", 1986 AT&T