B-032

AnT オペレーティングシステムにおける Linux の FD ドライバのプロセス化手法
A Method of Executing Linux FD Driver as a Process for *AnT* Operating System

島崎 泰† Yutaka Shimazaki　谷口 秀夫† Hideo Taniguchi　田端 利宏† Toshihiro Tabata　乃村 能成† Yoshinari Nomura

1. はじめに

入出力機器の多様化に伴い，デバイスドライバの種類は増加傾向にある．このため，オペレーティングシステム(以降，OS)は多数のデバイスドライバが必要になり，OS 開発の際，デバイスドライバの新規開発は大きな工数を要する．

このような背景から，デバイスドライバの開発工数を削減する研究がなされている[1][2]．我々は，***AnT*** オペレーティングシステム [3] (以降，***AnT***)におけるデバイスドライバの開発工数を削減するため，Linux の LKM ドライバを利用する．***AnT*** はマイクロカーネル構造であり，デバイスドライバのプログラムはプロセスとして動作する．

そこで，Linux の LKM ドライバをプロセスとして動作させる．本稿では，LKM ドライバのプロセス化の方針と手法について述べる．また，具体的な LKM ドライバとして，フロッピーディスク(以降，FD)の LKM ドライバを取り上げ，プロセス化の課題と対処を述べる．

2. 方針

Linux の LKM ドライバを ***AnT*** 上でプロセスとして動作させる際に，以下の 2 つを方針とした．

(方針 1) LKM ドライバのソースコードを変更しない

(方針 2) ***AnT*** のドライバプロセスのモデルに合わせた形態で動作させる

LKM ドライバのソースコードを変更することは，工数の増大につながる．そこで，(方針 1)により，最大限の開発工数の削減を狙う．また，制御機構の単一化のため，LKM ドライバをプロセス化したドライバプロセス(以降，LKM ドライバプロセス)と ***AnT*** 独自のドライバプロセスを同様に扱いたい．そこで，(方針 2)により，両者を区別しない形態とする．

3. 設計

3.1 *AnT* のドライバプロセスのモデル

AnT のドライバプロセスの動作を図 1 に示す．ドライバプロセスの対外インタフェースには，以下の 3 つがある．

(1) 処理依頼受け取りと結果返却
(2) 内コア機能の利用
(3) 割り込み処理の実行

処理依頼受け取りは，***AnT*** のサーバプログラム間通信機能[4]を利用する．AP の処理依頼は，ドライバプロセスの依頼キューへ登録される．ドライバプロセスは，get()システムコールで処理依頼を受け取る．

結果返却は，処理依頼受け取りと同じ get()システムコールを利用する．get()システムコール処理では，結果を

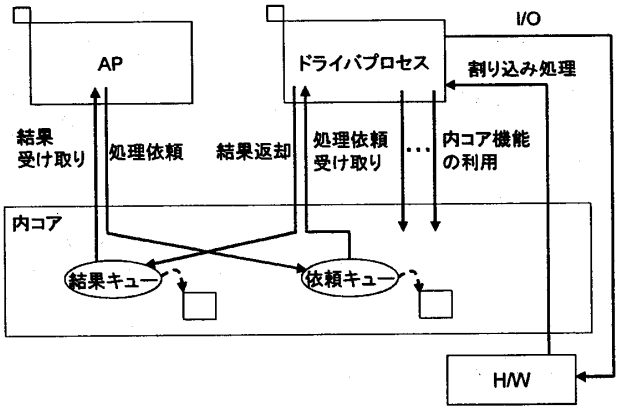

図1 ***AnT*** のドライバプロセスの動作

AP の結果キューへ登録する．AP は，システムコールを発行して結果キューから結果を受け取る．サーバプログラム間通信機能が提供する get()システムコールは，返却すべき結果があれば返却を行った後に，処理依頼を受け取るものである．

内コア機能の利用は，内コアに対してシステムコールを発行する．例えば，メッセージ出力やメモリ領域確保がある．

割り込み処理の実行は，H/W から内コアに割り込みが通知され，内コアがドライバプロセスの割り込み処理を呼び出すことで開始される．

3.2 LKM ドライバプロセス

ドライバプロセスのモデルに合わせて LKM ドライバをプロセス化するため，LKM ドライバの対外インタフェースをドライバプロセスのモデルの対外インタフェースに変換する．

LKM ドライバの対外インタフェースと LKM ドライバプロセスの構造を図 2 に示す．LKM ドライバの対外インタフェースには，以下の 3 つがある．

(1) 処理呼び出しと結果返却
(2) Linux カーネル機能呼び出し
(3) 割り込み処理の実行

上記の 3 項目は，ドライバプロセスのモデルの対外インタフェースに対応する．したがって，以下の変換処理を実現する．

(1) 要求変換部は，「処理呼び出しと結果返却」を「処理依頼受け取りと結果返却」に変換する．
(2) カーネル変換部は，「Linux カーネル機能」を「内コア機能の利用」に変換する．
(3) 割り込み変換部は，割り込み呼び出しの形式を変換する．

上記の要求変換部と割り込み変換部は，主に呼び出し形式の変換である．一方，カーネル変換部は，Linux と

† 岡山大学大学院自然科学研究科
Graduate School of Natural Science ant Technology, Okayama University

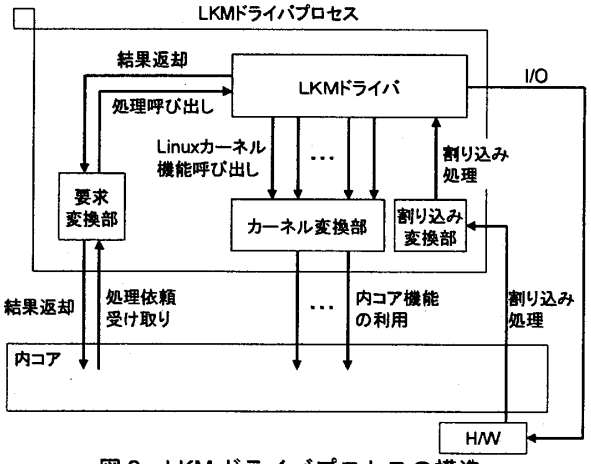

図 2 LKM ドライバプロセスの構造

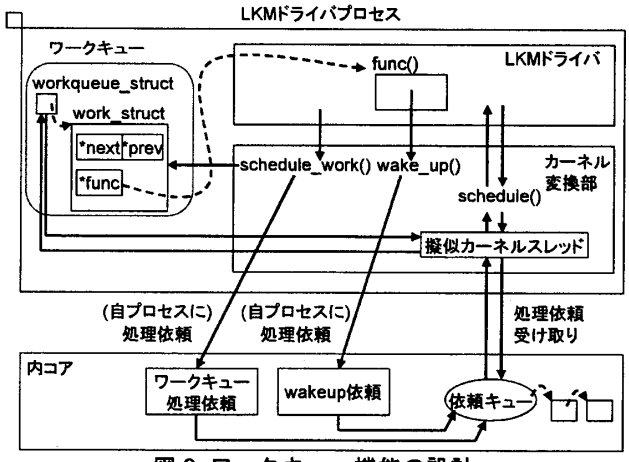

図 3 ワークキュー機能の設計

AnT 内コアでは，提供する機能が大きく異なるため，対応する処理実現も含めた変換を行う．

4. LKM 形式の FD ドライバのプロセス化

4.1 各変換部の処理

要求変換部は，処理依頼を受け取り，処理依頼の内容を判断し，処理依頼に応じた LKM ドライバの処理を呼び出す．LKM 形式の FD ドライバの処理呼び出しと結果返却の形式は 16 種存在する．そのうち基本的な，open, close, read, write について実現する．処理が終了すると，LKM ドライバから結果返却され，結果を ***AnT*** の形式に変換し，結果返却する．

カーネル変換部は，FD ドライバが呼び出す Linux カーネル関数(78 種)を実現する．

割り込み変換部は，***AnT*** の形式 ifunc(int intrno)を LKM の形式 handler(int irq, void *dev_id, struct pt_regs *regs)に，必要な引数の作成を用いて変換する．

4.2 ワークキュー機能の実現

ワークキュー機能は，ワークキューと呼ばれるキューに関数を登録し，登録した関数を専用のカーネルスレッドで実行する機能である．FD ドライバでは，ワークキュー機能を利用するために以下の関数を呼び出している．

(1) schedule_work()
work_struct 構造体をワークキューに登録する．また，ワークキュー実行カーネルスレッドを wakeup する．

(2) schedule()
スケジューラを呼び出す．このとき，LKM ドライバの処理は wait 状態となる．ワークキュー実行カーネルスレッドが wakeup されている場合，ワークキュー実行カーネルスレッドに処理が切り替わる．

(3) wake_up()
schedule()呼び出しによって wait 状態となった LKM ドライバの処理を wakeup する．

ワークキュー機能の設計を図 3 に示し，以下に説明する．LKM ドライバプロセスでは，ワークキュー機能で用いられるワークキュー実行カーネルスレッドをドライバプロセス内部で擬似カーネルスレッドとして実現する．

schedule_work()は，work_struct 構造体をワークキューに登録し，処理依頼システムコールを用いて，自プロセスに対してワークキュー処理依頼を発行する．

schedule()は，擬似カーネルスレッドを呼び出す．擬似カーネルスレッドは，処理依頼受け取りシステムコールを発行し，依頼キューから処理依頼を取り出す．取り出された処理依頼がワークキュー処理依頼である場合，ワークキューを実行し，wakeup 依頼であるならば schedule()を終了する．

wake_up()は，自プロセスに対して wakeup 依頼を発行する．

5. おわりに

Linux の LKM ドライバを ***AnT*** のプロセスとして実現する方式について，方針と設計を述べた．また，LKM ドライバの対外インタフェースをドライバプロセスのモデルの対外インタフェースに変換する手法について述べた．さらに，具体的な LKM ドライバとして FD の LKM ドライバを取り上げ，プロセス化の課題と対処を示した．ワークキュー機能の実現では，ワークキューを実行するカーネルスレッドを LKM ドライバプロセス内で擬似的に実現することを可能にした．実現において，***AnT*** のサーバプログラム間通信機能を利用した．

今後の課題として， LKM 形式の FD ドライバのプロセス化と評価があげられる．

謝辞 本研究の一部は，科学研究費補助金基盤研究(B)「適応性と頑健性を有する基盤ソフトウェアのカーネル開発」(課題番号：18300010) による

参考文献

[1]奥野 幹也, 片山 徹郎, 最所 圭三, 福田 晃, "UNIX 系 OS におけるデバイスドライバの抽象化と生成システムの実現," 情報処理学会論文誌, Vol.41, No.1, pp.1755-1765(2000.01).

[2]Qing-Li Zhang, Ming-Yuan Zhu, Shuo-Ying Chen, "Automatic Generation of Device Drivers," ACM SIGPLAN Notices, Vol.38, Issue 6, pp.60-69(2003.06).

[3]谷口 秀夫, 乃村 能成, 田端 利宏, 安達 俊光, 野村 裕佑, 梅本 昌典, 仁科 匡人, "適応性と堅牢性をあわせ持つ ***AnT*** オペレーティングシステム," 情報処理学会研究報告, 2006-OS-103, Vol.2006, No.86, pp.71-78(2006.07).

[4]岡本 幸大, 谷口 秀夫, "***AnT*** におけるサーバ間通信の高速なプログラム間通信機構," マルチメディア通信と分散処理ワークショップ論文集, Vol.2007, No.9, pp.61-66(2007.10).