4A-7

AnT オペレーティングシステムのドライバ構造

乃村能成 † 野村裕佑 † 仁科匡人 †† 谷口秀夫 †
† 岡山大学大学院自然科学研究科 †† 岡山大学工学部

1 はじめに

マイクロプロセッサや入出力ハードウェアの進歩には目覚しいものがある．また，様々な場面で計算機が必要となり，提供するサービス種別も飛躍的に増大している．このような背景から，新しいハードウェアやサービスへの適応制御機能を持つ ***AnT*** オペレーティングシステムを開発している [1]．

AnT の特徴である「外コア」と呼ばれるモジュールは，特権モードで高速に実行可能な特別なプロセスである．本稿では，***AnT*** オペレーティングシステムの外コアによるデバイスドライバの実現について述べる．

2 *AnT* の基本構造

2.1 コアと走行モード変更機能

AnT の基本構造を図 1 に示し，以下に説明する．

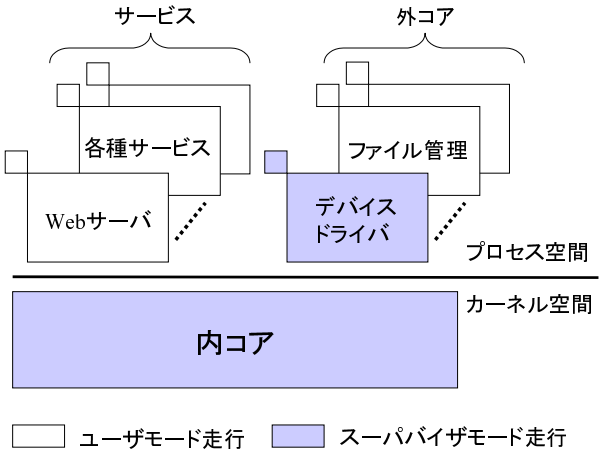

図 1 基本構造

内コア: 従来のカーネルに相当する部分で，OS の動作に必要な基本的な処理を担う．

サービス: Web サーバなどの一般的なサービスを提供するプロセス．

外コア: デバイスドライバやファイル管理など，システムの構成に応じた適応制御を実現するための特別なプロセス．

サービスは，内コアを通じてシステムコールに相当する「コア呼び出し」を要求する．内コアは，処理に応じて最適なコアを決定し，要求をディスパッチする．この内，プロセス生成などの基本的な要求は，内コア内で処理される．

AnT では，プロセスの高速な実行を可能にするため，走行モード変更機能 [2] を備えている．これは，他プロセスの走行モードをユーザモードからスーパバイザモードへ，あるいはその逆へ動的に変更する機能である．

走行モード変更機能を利用して外コアでデバイスドライバを実装した場合，以下の利点がある．

(1) 一般のサービスプロセスと同じ方式で実装可能なため，実装にかかる工数削減が期待できる．
(2) スーパバイザモード走行が可能なため，内コアとの効率的な連携が図れる．
(3) 走行モードを動的に変更できるので，効率性と安定性の両面を考慮した柔軟な運用が可能になる．

2.2 コア識別子

コア識別子は，サービスからの要求を処理するコアを特定するための ID で，32 ビットの整数で表される．サービスは，コアを呼出す際に処理の内容や種別に応じたコア識別子を指定して，内コアのコア呼出し処理部に要求転送を依頼する．コア識別子の形式を図 2 に示し，それぞれのフィールドについて以下に説明する．

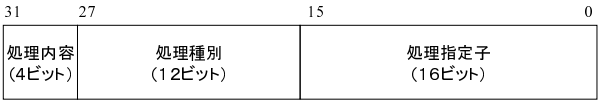

図 2 コア識別子

処理内容: コアが行うべき read/write などの処理の内容を示す．

処理種別: ディスクの入出力やメッセージ通信など，デバイスの種別を表す．

処理指定子: UNIX 系 OS におけるデバイスのマイナー番号に相当し，デバイスの種別毎に異なる処理の詳細を付加的に指定するのに利用する．

以降では，このコア識別子を媒介として，外コアがサービスからのデバイス操作要求を処理する手順と，そのための API について説明する．

Device Driver of ***AnT*** Operating System
Yoshinari Nomura†, Yusuke Nomura†,
Tadato Nishina†† and Hideo Taniguchi†
† Graduate School of Natural Science and Technology, Okayama University
†† Faculty of Engineering, Okayama University

3 デバイスドライバ実現方式

3.1 提供インタフェース

内コアが外コアとサービスに対して提供する主な API を以下に示して説明する．

(1) **callcore(coreid, op)**
callcore は，サービスからコアへの処理要求を行う操作である．要求を受け取った内コアは，coreid に基づいて担当する外コアを決定後，要求を転送する．coreid は，2.2 節で示したコア識別子である．op は，パラメータや返却値の受渡しに利用するメモリ領域を指す．

(2) **registcore(coreid, ifunc, ufunc, retbuf)**
registcore は，外コアが自身を内コアに登録する操作である．coreid は，2.2 節で示したコア識別子である．ただし，処理内容と処理指定子の全ビットを 1 にすると，ある処理種別について全ての機能を担当することを示す．ifunc は担当デバイスからの割り込みを処理するルーチンのアドレス，ufunc は callcore による要求の受け付けアドレス，retbuf は callcore の結果を返却するバッファの先頭アドレスである．

3.2 処理概要

図 3 にコア呼出し処理の流れを示して，以降で外コアがデバイス操作要求を処理する手順について説明する．

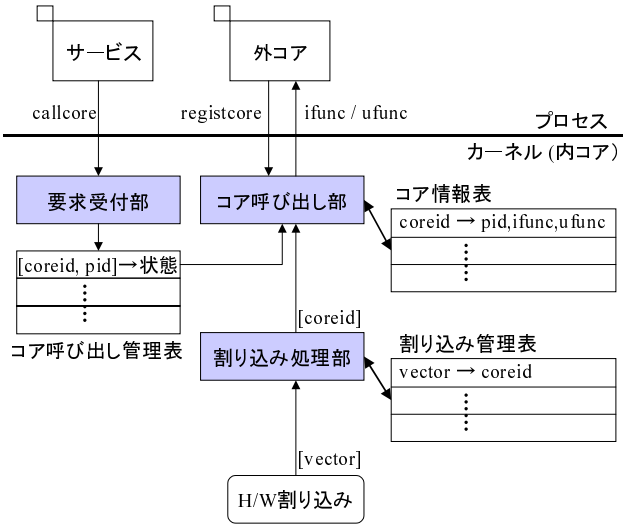

図 3 コア呼び出し処理の流れ

(1) デバイス操作を担当する外コアは，デバイスに対応する coreid を引数として，registcore を呼出す．

(2) 内コアのコア呼び出し部は，コア情報表に coreid と外コアの pid, ifunc, ufunc を登録する．また，割り込み処理部を通じて割込み管理表に，割込みベクタから coreid への変換規則を登録する．最後に，外コアを要求待ち状態に移行させる．

(3) デバイスを利用するサービスは，デバイスに対応する coreid を引数として，callcore を呼出す．

(4) 要求受付部は，要求元サービスの pid と要求内容をコア呼出し管理表に登録し，サービスを処理待ち状態に移行させる．

(5) コア呼出し部は，コア呼出し管理表から未処理の呼出し要求を取り出し，coreid とコア情報表から処理を担うコアを決定する．処理担当コアが外コアの場合，外コアの要求待ち状態を解除して，外コアのコンテキストで ufunc を実行させる．

(6) 外コアは，ufunc 内で処理が完了した場合，完了を示す戻り値を返却する．ハードウェアの処理待ちなどの事象が生じた場合は，処理中を示す戻り値を一旦返却する．

(7) コア呼出し部は，ufunc からの戻り値を検査し，処理完了の場合は，コア呼び出し管理表に完了状態を設定し，サービスを処理待ち状態から復帰させる．処理中の場合は，コア呼び出し管理表に処理中の状態を残して処理を完了する．以降の処理は，ハードウェアからの割込みを契機として継続される．

(8) ハードウェアの処理完了を契機として，割込み処理部が起動されると，割込み種別を表すベクタと割込み管理表から，処理を担う coreid を決定する．コア呼び出し部を通じて，外コアの仮想空間に切り替えた後，ifunc を実行する．以降の動作は，ufunc を実行した場合の動作と同様である．

(9) callcore から復帰したサービスは，要求を完了する．

4 おわりに

本稿では，***AnT*** オペレーティングシステムにおけるデバイスドライバの実現方式について述べた．デバイスドライバは，外コアとして実現されており，一般のサービスプロセスと同じ方式で実装可能なため，実装にかかる工数削減が期待できる．一方で，走行モード変更機能による効率的なコア間連携や割込み処理の直接受付を利用することによって，効率的な処理も期待できる．

今後の課題として，実装及び評価がある．

参考文献

[1] 谷口秀夫, 乃村能成, 田端利宏, “***AnT*** オペレーティングシステムの設計,” 情報処理学会第 68 回全国大会, Mar.2006.

[2] 横山和俊, 乃村能成, 谷口秀夫, 丸山勝巳, “応用プログラムの走行モード変更機構,” 情処研報 2004-OS-95,pp.25-32, Jeb.2004.