社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2008－SLDM－134／2008－EMB－8（8）
2008 / 3 / 27

AnT における入出力機器動作履歴を考慮したドライバプログラム起動制御法

滝口 真一† 乃村 能成†† 田端 利宏†† 谷口 秀夫††

† †† 岡山大学大学院自然科学研究科 〒700–8530 岡山県岡山市津島中三丁目1番1号
E-mail: †takiguchi@swlab.cs.okayama-u.ac.jp, ††{nom,tabata,tani}@cs.okayama-u.ac.jp

あらまし 計算機が提供するサービスの多様化に伴い，入出力機器の種類が増加している．このため，オペレーティングシステムは多くの入出力機能を提供しており，これらの入出力機能をドライバプログラムとして実現している．一方，一つの計算機システムが必要とする入出力機能は限られており，利用環境に応じて変化する．しかし，必要な入出力機能のみを有する OS を作成するには OS に関する専門的な知識を必要とするため、多くの計算機システムでは、入出力機能の絞込みは行われない．そこで，OS は多くのドライバを保有することなり，占有メモリ量の増加，および立ち上がり時間の長大化が問題になる．そこで，入出力機器の動作履歴を考慮したドライバの起動制御を行う方法を提案する．具体的には，既存 OS の問題点を示し，この問題に対処する入出力機器の動作履歴を考慮したドライバの起動制御法を提案する．また，*AnT* オペレーティングシステムにおける提案手法の実現について述べる．

キーワード ドライバ，マイクロカーネル，適応制御

A Method for Control of Driver Program Invocation based on Device Operation History on *AnT* Operating System

Shinichi TAKIGUCHI†, Yoshinari NOMURA††, Toshihiro TABATA††, and Hideo TANIGUCHI††

† †† Graduate School of Natural Science and Technology, Okayama University 1–1, Tsushima-Naka, 3–Chome, Okayama, Okayama 700–8530 Japan
E-mail: †takiguchi@swlab.cs.okayama-u.ac.jp, ††{nom,tabata,tani}@cs.okayama-u.ac.jp

Abstract Recently, operating systems have become increasingly multi-functional by the providing of various devices. Its functions are provided as driver programs. Even though average users do not use all functions, they do not customize and shrink their operating systems because of its job difficulty. Accordingly, they usually use wasteful and fat operating systems: much memory, slow down bootup time. In this paper, we propose an effective method for controlling invocation timing of device driver programs. As a strategy for the control, we use operation history on device drivers. And we present this design on *AnT* operating system.

Key words driver, microkernel, adaptability

1. はじめに

入出力機器の増加により，オペレーティングシステムは多くの入出力機能を提供しており，これらの入出力機能はドライバプログラムとして実現している．一方，一つの計算機システムが必要とする入出力機能は限られており，利用環境に応じて変化する．しかし，必要な入出力機能のみを有する OS を作成するには OS に関する専門的な知識を必要とするため、多くの計算機システムでは、入出力機能の絞込みは行われない．このため，OS は多くのドライバを保有することなり，占有メモリ量の増加，および立ち上がり時間の長大化が問題になる．

これらに対処するため，入出力機器の動作履歴を考慮したドライバの起動制御を行う方法を提案する．具体的には，既存 OS の問題点を示し，この問題に対処する入出力機器の動作履歴を考慮したドライバの起動制御法を提案する．また，*AnT* オペレーティングシステム [1] における提案方式の実現について述べる．

2. 既存 OS の問題点

2.1 問題点

利用環境に合わせて，必要最小限のドライバを保有する OS を構成するには，適切なドライバの組み込みが必要である．ド

表 1　占有メモリ量

	サイズ (Byte)
標準カーネル	3,470,924
小カーネル	2,036,636

ライバの組み込みにはカーネル構築などを必要とし，専門的な知識を必要とする．また，設定工数も小さくない．このため，多くの場合，標準的な構成の OS を利用する．

したがって，利用者の環境においては，不要なドライバを保有する OS を利用することになってしまう．この結果，以下の問題点がある．

（1）　ドライバによる占有メモリ量の増大

（2）　ドライバの初期化処理に伴う立ち上がり時間の長大化

立ち上がり時間を短縮するためのいくつかの技術 [2] があるが，特定の環境に合わせる手法のみで，利用環境の変化に合わせた入出力機能の絞込みは行われていない．

2.2 定 量 化

2.2.1 環　　境

FreeBSD 4.3 を用いて，占有メモリ量と立ち上がり時間を評価した．評価には，CPU : Intel Pentium4 2.8GHz，メモリ : 512MB の計算機を用いた．OS として，標準カーネル以外に小カーネルを用意した．標準カーネルは，GENERIC としてインストールキットに存在するカーネルで，多くの計算機環境で動作するよう構成されており，ドライバを含め 175 個のモジュールが組み込まれている．小カーネルは，評価に用いた計算機には不要なドライバを省いたカーネルであり，55 個のモジュールが組み込まれている．

2.2.2 占有メモリ量

コマンド kldstat を利用して，ログイン直後の占有メモリ量を測定した．結果を 表 1 に示す．表 1 より，標準カーネルは，小カーネルに比べ，約 1.43MB 占有メモリ量が大きい．この値は，標準カーネルの 41.32%，小カーネルの 70.42%に相当し，ドライバが OS の利用するメモリ量に占める割合は大きい．

2.2.3 立ち上がり時間

CPU のタイムスタンプカウント値を取得する rdtsc 命令を用いて，電源投入から login 起動までの時間 (以降，立ち上がり時間と名付ける) を測定した．測定結果は，測定回数 10 回の平均である．結果を 図 1 に示す．

図 1 より，標準カーネルは，小カーネルに比べ，非 ISA ドライバ設定時間と ISA ドライバ設定時間が大きい．ここで，立ち上がり時間とドライバ初期化 (ドライバ登録，非 ISA ドライバ設定，ISA ドライバ設定) 時間の関係を 表 2 に示す．表 2 より，標準カーネルのドライバ初期化にかかる時間は，小カーネルに比べ約 6.10 秒長い．この値は，標準カーネルの立ち上がり時間全体の 20%に相当する．つまり，不要なドライバの初期化処理に使用される時間は大きい．

3.　入出力動作の状態

入出力の動作、特に動作開始を効率的に行えるようにするため、入出力の動作を分析する。

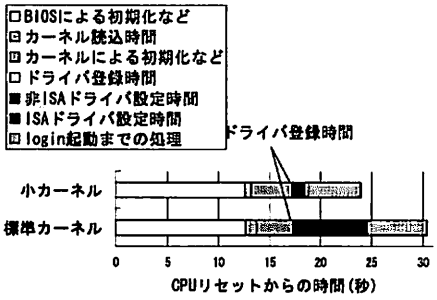

図 1　立ち上がり時間

表 2　立ち上がり時間とドライバ初期化時間の関係

	立ち上がり時間	ドライバ初期化時間 (比率)
標準カーネル	30.42 秒	7.31 秒 (24.04%)
小カーネル	23.94 秒	1.21 秒 (5.06%)

表 3　ドライバ制御状態

状態名	デバイス	ドライバ	AP からの利用状況
(1) 無し	無	非読込	非利用
(2) 装荷待ち	無	読込済み	非利用
(3) 装荷	有	非読込	非利用
(4) 初期化待ち	有	読込済み	非利用
(5) 利用待ち	有	初期化済み	非利用
(6) 利用中	有	初期化済み	利用中

入出力の動作に関与する要素として、デバイス、ドライバ、および AP からの利用状況がある。この 3 つの要素は、次の状態を持つ。デバイスは、計算機に接続されているか否かに相当する「存在の有無」の 2 状態を持つ。ドライバは、メモリ上に読み込まれているか否か、およびメモリ上に読み込まれた後に初期化処理が終了しているか否かにより、「非読込」、「読込済み」、および「初期化済み」の 3 状態を持つ。AP からの利用状況は、利用中か否かにより、「非利用」と「利用中」の 2 状態を持つ。これらの状態により 12 個の組み合わせが考えられるが、実際に存在する状態は 6 個であり、その組み合わせを 表 3 に示す。ここでは、表 3 に示した各組み合わせを「ドライバ制御状態」と名付ける。

次に、ドライバ制御状態の遷移を 図 2 に示し、説明する。OS が立ち上がった時点でのドライバ制御状態は、「無し」または「装荷」である。両者の状態間の遷移は、デバイスの取付けあるいは取外しで起こる。ドライバプログラムのメモリ上への読み込みにより、「無し」は「装荷待ち」、「装荷」は「初期化待ち」に状態遷移する。「初期化待ち」は、初期化処理の実行により、「利用待ち」の状態に遷移する。「利用待ち」と「利用中」の間の状態遷移は、AP からの利用開始要求および利用終了要求で起こる。なお、ドライバプログラムが OS カーネル内に存在する場合は、ドライバプログラムの終了は起こらない。一方、マイクロカーネルのように、ドライバプログラムがプロセスとして存在する場合は、ドライバプログラムの終了（つまり、ドライバプロセスの終了）が起こる。

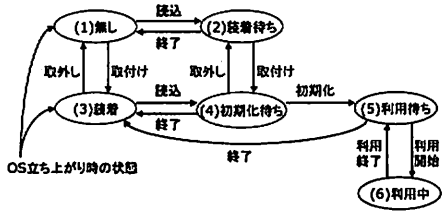

図 2　ドライバ制御状態の遷移

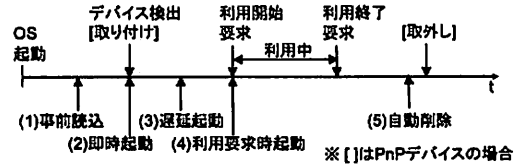

図 3　制御内容の開始契機

4. 入出力機器動作履歴を考慮したドライバプログラム起動制御法

4.1 基本方針

図 2 において状態遷移を起こす要因は、デバイスの取付け、デバイスの取外し、ドライバプログラムのメモリ上への読み込み、初期化処理の実行、利用開始要求、利用終了要求、およびドライバプログラムの終了である。デバイスの取付けとデバイスの取外しは利用者（人）の行為であり、利用開始要求と利用終了要求は AP の行為である。そこで、OS の行為であるドライバプログラムのメモリ上への読み込み、初期化処理の実行、およびドライバプログラムの終了に着目して、ドライバプログラムの起動を制御する。

提案する制御法の基本方針は以下である．

（方針） 必要な時に必要なドライバプログラムのみを動作させる

必要な時に必要なドライバプログラムのみを動作させるとは，ドライバプログラムが必要の無いときはドライバプログラムを組み込まず，必要なときには組み込まれている状態にすることである．必要の無いときにドライバプログラムを組み込まないようにする単純な方法として，ドライバプログラムを利用を開始した時点で組み込みを行うことが考えられる．しかし，利用開始時点で必要とされるドライバプログラムのみ組み込みを行うと，利用するときに読み込みと初期化処理が行われ，利用者から見た待ち時間は増大すると考えられる．このため，よく利用されるドライバプログラムに関しては利用したいときにはすぐに利用できることも重要となる．

必要な時に必要なドライバプログラムのみを動作させるために以下の課題に対処する必要がある．

（課題 1） 多様な起動方法の導入

（課題 2） 起動方法の選択方法

必要な時に必要なドライバプログラムのみを動作させるためには，個々のドライバプログラムの利用の形態に合わせたドライバプログラムの起動制御を行うのが良いと考える．

利用の形態に合わせたドライバプログラムの起動制御のため，まず多様な起動方法を導入する．既存 OS は必要のないドライバプログラムの読み込みが行われており，OS 立ち上がり時に，計算機に取り付けられているデバイスに対応する全てのドライバプログラムを組み込んでいる．しかし，計算機に取り付けられているデバイスに対応したドライバが存在しても必要ないのであれば，そのドライバプログラムを組み込まなくてもよい．また，デバイスに対応する全てのドライバプログラムを組み込まなくても必要になった時点で組み込みを行えばよい．

次に，個々のドライバプログラムはその利用の形態に合わせ，導入した起動方法の中から 1 つ選択し，利用の形態に合わせた制御を行う必要がある．このため，過去のドライバプログラムの利用の形態を動作履歴として収集する．そして動作履歴の情報から，起動方法を選択する．

4.2 対　　処

4.2.1 多様な起動方法の導入

既存 OS では，ドライバプログラムの読み込みと，初期化処理の実行はデバイスの検出時に行われる．しかし，組み込まれたドライバプログラムは必ずすぐに利用されるわけではない．また，全く使われない可能性もある．そこで，ドライバプログラムの制御契機を調節した多様な起動方法により，必要なときに必要なドライバプログラムのみ動作させるようにする．ドライバプログラムの読み込み，初期化，終了により行う多様な起動方法 (以降，制御内容と名付ける) として以下の 5 つを考える．

（1） 事前読込

デバイス検出前にドライバプログラムを読み込み，デバイス検出時にドライバプログラムを初期化する．事前読込を行う前に，デバイス検出された場合は，即時起動を行う．

（2） 即時起動

デバイス検出時にドライバプログラムを読み込み，初期化する．

（3） 遅延起動

デバイス検出時にドライバプログラムを読み込まず，その後の空き時間に読み込みと初期化を行う．

（4） 利用要求時起動

AP から利用開始要求が来た時にドライバプログラムを読み込み，初期化する．

（5） 自動削除

利用終了要求後にドライバプログラムを終了させる．自動削除後，利用開始要求が来た場合，利用開始要求時起動を行う．

5 つの制御内容の開始契機を 図 3 に示す．ドライバプログラムは OS 起動から利用開始要求までの間に 4 つの契機のいずれかで読み込みと初期化を行う．利用終了要求後，そのドライバプログラムが必要なければ自動削除を行う．PnP デバイスは [取り付け] と [取外し] が，利用者により行われる．

制御内容は，OS 立ち上がり時間短縮のため，遅延起動，利用開始要求時起動，自動削除を行い，利用したいときにすぐに利用可能にするため，事前読込，即時起動を選択する．

4.2.2 制御内容の選択に必要な情報

適切な制御内容を選択するため，ドライバプログラムの動作履歴を収集する．動作履歴として，立ち上がり時間の短縮と利用時の待ち時間が短くするのに有効な動作の情報を収集する．このため，デバイス検出の時刻やデバイス利用の時刻とその回数などを収集する．また，ドライバプログラムの動作履歴の少なさによる不適切な制御を抑制する．動作履歴は制御内容を選択するために数値化し閾値を設ける．

以下に収集する情報を示す．個々の情報には英字1文字の名前を設けている．英字1文字の名前は以降で制御内容の選択基準を示すのに用いる．

（1） 利用開始時間 (S)

いつ利用され始めるのかを表す．デバイス検出されてからの最初の利用開始要求が来るまでの時間を表す．

（2） 利用度 (U)

過去一定期間内の利用の程度を表す．具体的には過去の時間を一定の区間に区切り，過去のN個の区間のうち利用中であったか区間がいくつあるかで表す．

（3） 非利用時間 (N)

最後に利用されてから現時刻までの利用されていない時間

（4） デバイス種別判定 (D)

OS立ち上がり時，存在デバイスか非存在デバイスか判定する．OS立ち上がり時刻からデバイスが検出時刻までの時間で表す．

（5） 履歴収集期間 (H)

動作履歴を収集している期間を表す．制御内容を選択するために十分な情報を持っているかを判断するために用いる．

4.2.3 制御内容の選択方法

ドライバプログラムの利用の形態にあわせ制御内容を選択するため，動作履歴による選択の基準について述べる．

動作履歴を用いた制御内容の選択基準を 表4 と 表5 に示し，以降で説明する．制御内容を選択するため閾値を用いる．表中では閾値を H に対して H_{th} といった形で th をつけることで示している．

OS立ち上がり後一定時間経過してから起動する遅延起動は，よく利用されているが，OS立ち上がり時に利用されていないという場合に用いる．遅延起動を用いることにより，OS立ち上がり時間を短縮でき，利用時のドライバプログラムの読み込みと初期化処理による待ち時間も発生させないことが期待できる．遅延起動を選択するための条件は利用度が高く，利用開始時間が長い場合とする．

事前読込は，デバイス検出より前にドライバプログラムの読み込みを行うため，デバイス検出より前の時間を有効に利用できる可能性がある．事前読込は，よく利用されており，デバイス検出後，早い時刻に利用されるドライバプログラムに対して用いると有効であると考えられる．

利用要求時起動は，ドライバプログラムを利用しようとした時点で読み込みと初期化処理を行うため，使われていないドライバプログラムに対して用いるのが有効と考えられる．

即時起動は，以上の3つの制御のどれが適切か判断できない場合に用いる．判断するのに十分な動作履歴を持っていない場合に用いる．

表4 起動制御の選択条件

条件		説明	制御
$H \leqq H_{th}$	無し	動作履歴の情報が十分でない	即時起動
$H > H_{th}$	$S \leqq S_{th}$ $U \leqq U_{th}$	デバイス検出後短期間で利用し，利用度は低い	
	$S \leqq S_{th}$ $U > U_{th}$	デバイス検出後短期間で利用し，利用度は高い	事前読込
	$S > S_{th}$ $U \leqq U_{th}$	デバイス検出後，利用まで間があり，利用度は低い	利用要求時起動
	$S > S_{th}$ $U > U_{th}$	デバイス検出後，利用まで間があり，利用度は高い	遅延起動

表5 自動削除の選択条件

条件		説明	制御
$H \leqq H_{th}$	無し	動作履歴の情報が十分でない	無し
$H > H_{th}$	$N \leqq N_{th}$ $U \leqq U_{th}$	最後に利用されてから間が無く，利用度は低い	無し
	$N \leqq N_{th}$ $U > U_{th}$	最後に利用されてから間が無く，利用度は高い	無し
	$N > N_{th}$ $U \leqq U_{th}$	最後に利用されてから間があり，利用度は低い	自動削除
	$N > N_{th}$ $U > U_{th}$	最後に利用されてから間があり，利用度は高い	無し

自動削除は，使われていないドライバプログラムを削除するのに用いる．ただし，ドライバプログラムの利用はある程度連続的に行われるものと考えられる．このため，利用終了要求後一定時間経過してから自動削除を行うのが適切と考えられる．

4.3 制御内容の選択を含めた制御状態の遷移

図2を基に制御内容の選択を反映させた状態遷移を 図4 に示す．制御の開始からデバイスの利用までの流れを考えるため，外部から与えられる制御契機としてドライバ生成要求，APからの利用開始要求と利用終了要求がある．そして，起動制御プログラム自身が提供する制御契機として，制御の開始時とタイマを加えた5つの契機で読み込み，初期化，終了の制御を行う．個々のドライバプログラムの制御は (1-a) 無しの状態より始まる．ドライバ生成要求が制御契機として与えられた場合でも，制御内容の選択結果によって遷移先は異なるため，状態遷移は制御内容の選択を含んだものとなっている．また，事前読込は読み込みを行う契機によってAとBに分ける．事前読込AはOS立ち上がり時に直ちに，ドライバプログラムを読み込む．事前読込BはOS立ち上がり後一定時間経過後に読み込む．そして，個々の遷移では読み込みと初期化が行われる．ここではドライバプログラムの終了に関する遷移は省いている．

5. 実現方式

5.1 *AnT* オペレーティングシステム

AnT オペレーティングシステム (*AnT* と呼ぶ) は，プログラム構造としてマイクロカーネル構造を採用し，デバイスドラ

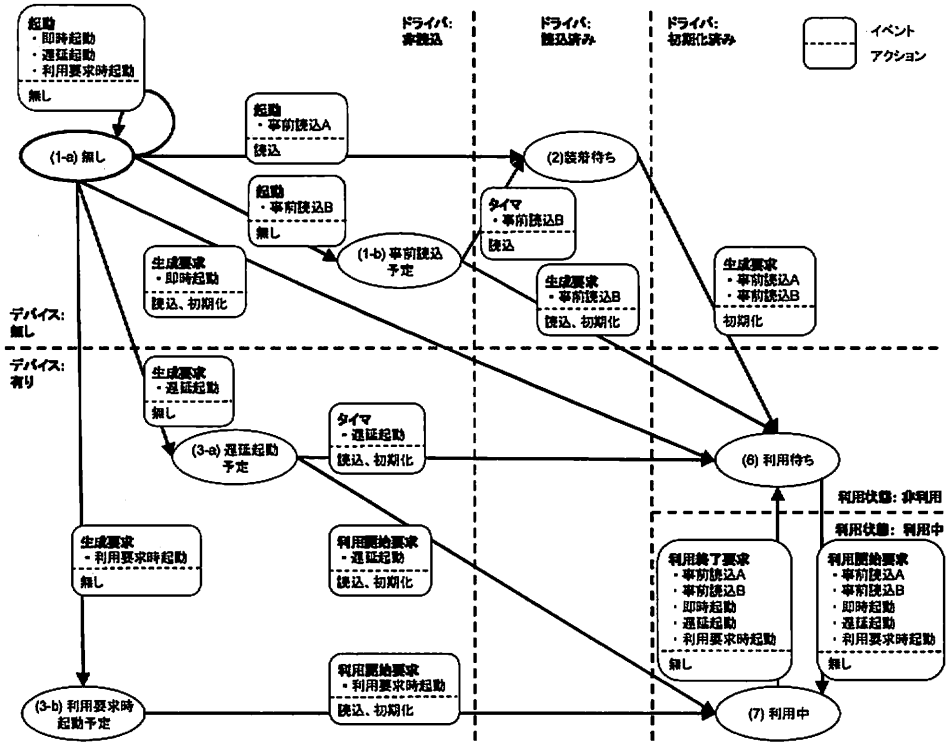

図 4　制御内容の選択を含めた制御状態の遷移

イバやファイルシステムといった機能をプロセスとして動作させる．プログラムは，OS とサービスからなる．OS は，内コアとプロセスとして動作する外コアからなる．サービスは，利用者の要求するサービスを提供するプロセスからなる．内コアは，システムの最小限な機能の動作を保証するプログラム部分である．主な機能として，メモリ領域管理，プロセスの実行制御部がある．外コアは，適応したシステムに必須なプログラムであり，動的に再構成可能な構造を有する．主な機能として，デバイスドライバやファイル管理がある．サービスは，利用者の要求する機能を提供する部分である．

5.2　*AnT* でのドライバプロセス制御機構

AnT におけるドライバプロセスの制御の流れを 図 5 に示す．提案制御法はドライバ構成制御てして実現する．***AnT*** でのドライバプロセス生成は主に以下の手順で行われる [3].

（1）　ドライバ生成要求

プログラム決定部がドライバ生成要求をドライバ構成制御に送る．プログラム決定部は H/W の検出などに合わせ，計算機が利用される環境に合わせてプログラム間で関連の深いものを同時に起動する．プログラム決定部とドライバ構成制御は利用可能なドライバプログラムをドライバ情報表として共有している．

（2）　プロセス生成要求

ドライバ構成制御がプロセス生成要求を適応制御機能制御部に送る．プロセス生成要求は，ドライバプログラムの読み込みに相当する．適応制御機能制御部は，プロセスを生成するためプロセス生成システムコールを呼ぶ．

（3）　プロセス起動

ドライバ構成制御がプロセス起動システムコールを呼び，ドライバプロセスの走行を開始させる．プロセス起動はドライバ制御状態の遷移の初期化処理の実行に相当する．

（4）　コア登録

走行を開始したドライバプロセスは初期化を行い，コア登録システムコールを呼ぶ．コア登録が完了すると外コアは外部からの要求を受けることが可能となる．

（5）　コア登録の可否確認

不許可なドライバプロセスの起動を禁止するため，内コアはコア登録の要求を受けた時，実際のコア登録を行う前に，許可されている登録なのかどうかをドライバ構成制御に確認する．そして，ドライバ構成制御は自身が生成したドライバプロセスだった場合のみ許可する．コア登録を確認することで，ドライバ構成制御は走行しているドライバプロセスを管理する．

また，ドライバ構成制御より必要に応じてドライバプロセスへ終了要求が送られる．

5.3　ドライバプログラム起動制御機構

ドライバ構成制御は外部からの要求を受け，ドライバプログラムの制御状態を扱い，起動制御を行うため，制御構造として状態遷移機械を用いる．また，ドライバ構成制御は制御内容を選択するための動作履歴を収集する．そして，***AnT*** においての生成したドライバプロセスの管理する仕組みを持つ．

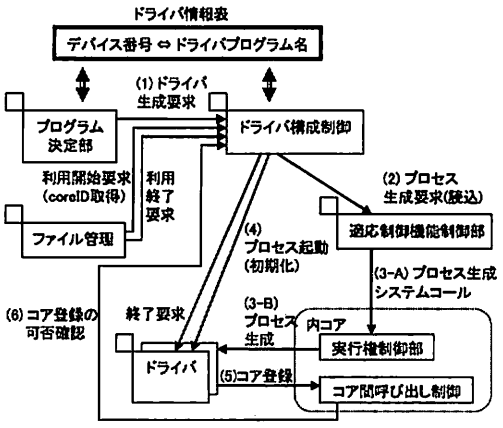

図 5　ドライバプロセス制御の流れ

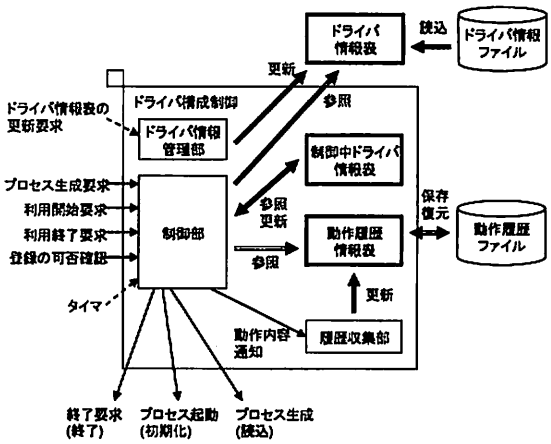

図 6　ドライバ構成制御の内部構成

ドライバ構成制御の構成を 図 6 に示し，以下で説明する．ドライバ構成制御を実現するために必要な処理は大きく以下に示す 3 つの部分に分けられる．

（1） 制御部

（2） 動作履歴収集部

（3） ドライバ情報管理部

制御部は制御中のドライバプログラムの状態を管理する．4.3 節 に示した状態遷移を元に外部からの要求を契機としてドライバプログラムの読み込み，初期化，終了を制御する．制御部は利用可能なドライバプログラムが記録されているドライバ情報表を参照することで起動してよいドライバプログラムを確認する．そして，動作履歴が記録されている動作履歴情報表を参照し，制御内容を決定する．外部からの要求を契機に制御中ドライバ情報表から現在のドライバプログラムの制御状態を確認し，次の状態へ遷移するための制御を行い，制御中ドライバ情報表を更新する．

ドライバ情報管理部は起動可能なドライバプログラムをドライバ情報表により管理する．ドライバ情報表はファイルに記録されており，ドライバ構成制御起動時にはドライバ情報表をファイルより読み込む．また，OS で利用するドライバプログラムを追加するにはドライバ追加用のコマンドを用いてファイルとしてのドライバ情報表を更新する．その後，ドライバ構成制御にドライバ情報表の更新を通知して，ドライバ構成制御のドライバ情報管理部は自身のメモリ上にあるドライバ情報表を更新する．

動作履歴収集部は制御部からの通知を受けドライバプログラムの動作履歴を収集する．また，動作履歴情報表を定期的に保存する．

6. おわりに

利用環境に応じた OS の構成は難しいため，多くの OS はシステムにとって不要なドライバプログラムを含む．このため，FreeBSD の場合，標準カーネルでは，不要なドライバプログラムが利用メモリの約 4 割を占有し，立ち上がり時間が 2 割以上遅くなっている．そこで，入出力動作の状態について，要素と状態を整理し，入出力機器の動作履歴を考慮したドライバプログラム起動制御法を提案した．提案手法は，ドライバの動作履歴を利用し，ドライバプログラムのメモリ上への読み込み，初期化処理の実行，およびドライバプログラムの終了に着目し，制御する．***AnT*** オペレーティングシステムにおいては，提案手法をドライバ構成制御として実現する．

残された課題として，実装と評価がある．

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．

文　献

[1] 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，Jul，2006.

[2] Tim R. Bird，"Methods to Improve Bootup Time in Linux，" Linux Symposium 2004，Volume One，pp.79-88，Jul，2004.

[3] 仁科匡人，野村裕佑，田端利宏，乃村能成，谷口秀夫，"***AnT*** オペレーティングシステムにおけるプロセス生成制御機構，"情報処理学会研究報告，2007-OS-105，vol.2007，No.36，pp.7-14，Apr，2007.