5C－5

リアルタイムUNIXでのプログラム起動制御の一手法

二村 祐地，竹並 春佳
三菱電機(株) 情報技術総合研究所

1. はじめに

産業分野に適用されるリアルタイムOSでは近年オープン性への要求が高まっているものの，その基本価値として高速性や定応答性，動作安定性が重要であることに変わりはない．

本稿では，リアルタイムコンピュータ向けUNIX互換OSの開発にあたり，大規模システムへの適用のため，定応答性と動作安定性の実現を目的に採用したプログラム起動制御手法について報告する．

2. 背景，課題

数百ミリ秒以下の応答性を達成するリアルタイム応用システムでは一般に，リアルタイム処理を行うプログラムは主メモリ上に常駐させて動作させる．しかしUNIXのスタイルではプログラムは実行時毎にディスクからローディングする．このためUNIX互換のOSの上でリアルタイム応用システムを実現する場合，リアルタイム処理プログラムは終了しないプロセス(daemon)として動作させ，プロセス間通信(IPC)によりその処理の起動を制御するのが定石である．

ただし同一計算機上に数百のプログラムが置かれる場合，全てプログラムを常駐動作させることは同時動作プロセス数の制限などにより実現困難なことが多い．このような場合，応答時間の厳しいプログラムは常駐動作とし，応答時間に余裕のあるプログラムは非常駐動作として対処する．このときシステムは，UNIXのスタイルに沿って単純に実装すると，常駐動作プログラムはdaemonとして動作させIPCにより起動し，非常駐動作プログラムはUNIXのスタイルに沿ってfork及びexecシステムコールを用いて起動するかたちとなる(図1)．

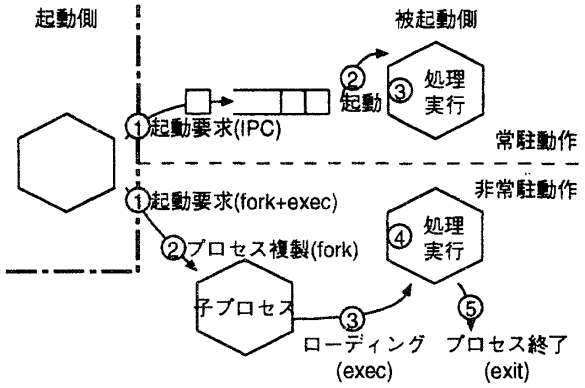

図1：単純な実装

しかし本方式には次の課題がある．

定応答性に対する課題：

1) 起動元の処理は，常駐動作プログラム起動時はIPCによる起動要求通知(≪1ミリ秒)であるが，非常駐動作プログラム起動時はfork処理(≫1ミリ秒，プロセスサイズ依存)となる．両者の処理時間には大差があるため，起動元のプログラム動作時間設計が困難となる．

2) プログラムローディング(exec)の処理時間は状況により大きく変動する．このため起動元でのプログラム起動処理から非常駐プログラムの動作開始までに要する時間が予測困難となる．

動作安定性に対する課題：

3) 計算機内での同時動作プロセス数の制限から非常駐プログラムの起動失敗が簡単に発生しうる．

3. アプローチ

これら課題の原因は次の2点に集約される．

a) 定常稼働時にfork処理を行う

b)exec処理(プログラムローディング)所要時間が大きく変動する

本稿で報告するプログラム起動制御は両者の各々の解決を図るものである．なお前者はミドルウェア層で対処し，後者もexecシステムコールのプログラミングインタフェースを変更しない範囲で対処した．これは動作安定性とオープン性への配慮からカーネルの改修は極力避けるべきとの判断による．

A method of driving programs on Real-Time UNIX

Yuji Nimura, Haruka Takenami
Mitsubishi Electric Corporation
Infomation Technology R&D Center
5-1-1 Oofuna, Kamakura, Kanagawa 247, Japan

4. 実装

4.1 プロセスプールを用いたプログラム起動

定常稼働時の fork 処理を避けるため，プログラム起動はすべて IPC で行う方式を採用した．非常駐動作プログラムの起動は，起動要求を検知するプロセス broker とプログラムローディングを行うプロセス loader により実現した．非常駐動作プログラム宛の起動要求が発行されると broker がそれを検知し loader へ該当プログラムのローディング要求を発行，loader では同要求を受け，そのうちの1つが直接非常駐動作プログラムをローディングする (図 2).

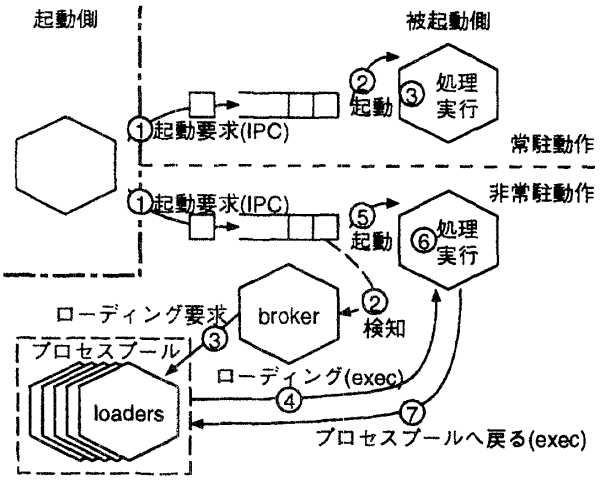

図 2：プログラム起動動作

4.2 プログラムローディング処理の定時間化

UNIX のプログラムローディング (exec) 処理において，処理時間の主な変動要因はファイルアクセスに関係した次の点にある．

i) ファイルの内容が不連続なブロックに保持され，その位置が制御困難．

ii) ローディング処理毎にファイルブロック配置情報 (inode 情報) を得るためディレクトリ探索する．

よって処理の定時間化には連続配置されたブロックからのファイル内容読み込みと，ディレクトリ探索の省略が有効である．このためまず連続ブロックファイルを実現し，続いて同ファイルからのプログラムローディングを可能とした．さらにファイルの inode 情報をあらかじめ key 値と共にカーネル内に登録することで，ディレクトリ探索を省く key 値指定でのプログラムローディングを実現した．

なお key 値は図 3 に示す形式の文字列であり，exec システムコールに対しファイル名に代え指定する．このため exec システムコールのプログラムインタフェースは保たれており，通常のプログラムからでも key 値を用いたプログラムローディングは利用可能である．

"/"+ マジックパターン +"/"+ 数文字列

図 3：key 値の形式

5. 評価

プロセスプールを用いたプログラム起動の採用により次の効果を得た．

1) 起動元は起動先の常駐動作 / 非常駐動作に限らず同じ方式 / 処理時間で処理ができる．

2) 定常稼働状態で動作するプロセス数を一定に保つことで動作の安定化が図れる．

またプログラムローディング処理の定時間化では実測により以下を確認，処理時間変動の抑制と高速化の効果を得た．

・連続ブロックファイルからのローディングで raw モードのディスクアクセスに近いファイル読み込み性能を達成．

・key 値指定では処理時間は対象ファイルのあるディレクトリの深さの影響を受けず，さらにディレクトリの深さ 0 の場合でもファイル名指定より処理が速い (図 4).

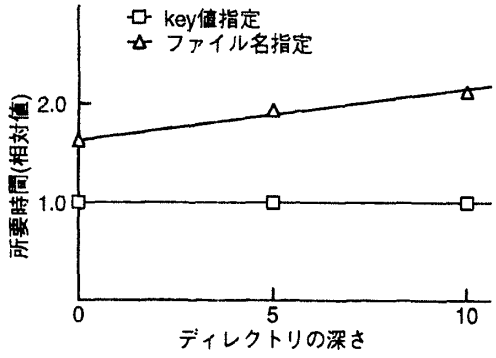

図 4：ローディング所要時間

6. おわりに

UNIX 互換リアルタイム OS の開発において，定応答性と動作安定性を目的に実装したプログラム起動制御の一手法について述べた．今後は実証した有用性をもとに，実システムへの展開を図っていく予定である．