# 4H-05 UNIXベースシステムにおける起動高速化の検討

菅井 尚人　山口 義一
三菱電機 (株)　情報技術総合研究所

## 1. はじめに

ソフトウェアの複雑化やネットワーク機能への対応の容易さから、Linuxをはじめとする UNIX ベースの OS の適用が拡大しており、組み込み機器などの高速な立ち上がりを要求される用途に使用される場合も増えてきた。

本稿では、UNIX ベースのシステムにおいて、アプリケーションプログラムの起動やサービスの提供が可能となるまでの時間を最小化することを目的とした、システムの起動高速化の手法を検討する。

## 2. UNIXベースシステムの起動処理

PC/AT 機上で動作する Linux を例に取り、起動処理の概要を示す。

(1) BIOS ROM

CPU はリセット後、BIOS ROM 内の初期化コードを実行する。BIOS ROM の初期化コードは、主記憶サイズの取得、ブートデバイスの選択、PCI の拡張スロットへのカード実装の有無の確認などを行う。初期化コードの実行終了後、設定されたブートデバイスの先頭にあるブートセクタを主記憶にロードし実行する。

(2) カーネルロード

ブートセクタには、ブートローダが書き込まれており、カーネルファイルの主記憶へのロードを行う。Linux の場合、カーネルファイルは通常は圧縮形式を使用しているため、主記憶へのロードの完了後、まず復元処理が行われ、その後に実際のカーネルコードの実行が開始される。

(3) カーネル初期化

カーネルの初期化コードは、カーネル内の共通データ構造、カーネルに組み込まれているファイルシステムや各種のデバイスドライバなどの初期化を行う。カーネル初期化の最後に init プロセスを生成し、ファイルシステム上の init プログラムを起動する。

(4) init と rc による初期化

init プログラムは、設定ファイルの記述に従い、ローカルファイルシステムのマウント、ネットワークインターフェースの初期化、各種デーモンの起動などをおこなう初期化プログラム rc を実行する。

これらの起動処理の所要時間を測定した。

測定は、PentiumII 266MHz、64MB メモリ、IDE HDD の H/W 上で動作する Linux を使用して行った。

表 1 初期化処理時間の内訳

| 処理 | 処理時間 (秒) |
|---|---|
| (1) BIOS | 22.0 |
| (2) カーネルロード | 0.6 |
| (3) カーネル初期化 | 5.3 |
| (4) init および rc 初期化 | 24.6 |
| 計 | 52.5 |

この測定結果では、BIOS ROM の実行時間が処理起動の 42%を要している。BIOS ROM での処理内容は、メモリサイズの取得とブートデバイスの検索がその主なものであるが、これらは H/W 構成や BIOS ROM の実装方法に依存する。

カーネル初期化の処理時間 5.3 秒のうち 4.3 秒はデバイスの初期化処理によるものである。測定に使用したカーネルでは、組み込むデバイスドラ

---

A method of fast booting mechanism for UNIX-based system

Naoto SUGAI, Yoshikazu YAMAGUCHI
Mitsubishi Electric Corporation

イバを今回の測定環境における必要最小限の構成とすることにより、処理時間の短縮を図っている。

init と rc の実行による初期化処理は、今回の測定の中で最も長時間を要しており、全体の47%を占めている。rc の処理内容は、システムの構成に依存するが、初期化処理の枠組みとしての rc 処理に着目した高速化手法を次に検討する。

## 3. rc 処理の高速化手法

rc 処理の高速化手法として、rc のバイナリ実行形式化、条件判断の事前実行による rc 処理の簡略化を提案する。

### 3.1.rc のバイナリ実行形式化

rc は通常、shell スクリプトで記述されている。これは、処理内容の変更が容易であるという利点がある反面、実行速度の面では不利である。

rc の処理を C 言語で記述、コンパイルし、バイナリ実行形式にすることにより処理の高速化が期待できる。バイナリ実行形式化による効果としては fork、exec 回数の削減、shell によるスクリプトの解析処理が不要となることが挙げられる。shell スクリプト中では、ファイル削除などの単純な処理でも外部コマンドの起動によって行われている。C 言語による記述では、これらの処理は単なるシステムコールの呼び出しなどに置き換えることができるため、shell での外部コマンド実行時に行っていたプロセスの生成・実行を削減することが可能となる。

表 2 削減可能なコマンド

| コマンド | 処理 | 回数 |
|---|---|---|
| rm | ファイル削除 | 13 |
| touch | タイムスタンプの更新 | 9 |
| uname | システム名の設定・取得 | 4 |
| hostname | ホスト名の設定・取得 | 13 |

表 2 は、今回測定に使用した Linux の rc 処理において、バイナリ実行形式化により削減可能なコマンドとその使用回数を示したものである。これらの総数は、rc 全体でのプロセスの生成・起動回数の約 10%を占めており、バイナリ実行形式化による効果が期待できる。

### 3.2.条件判断の事前実行

rc 中では、設定ファイルに対して grep コマンドなどによるパターンマッチングを行い、ある条件を満たす時にのみ必要な初期化処理を行う場合がある。このような条件判断のための処理は、設定ファイルの変更が行われない限り、起動のたびに行う必要はない。そこで、あらかじめ設定ファイルの内容が必要な条件を満たすかどうかを判断し、その時点での設定ファイルのタイムスタンプを取得しておく。rc の実行時には設定ファイルのタイムスタンプの比較を行い、更新されていなければ以前に行った条件判断に基づいて初期化処理を行う。

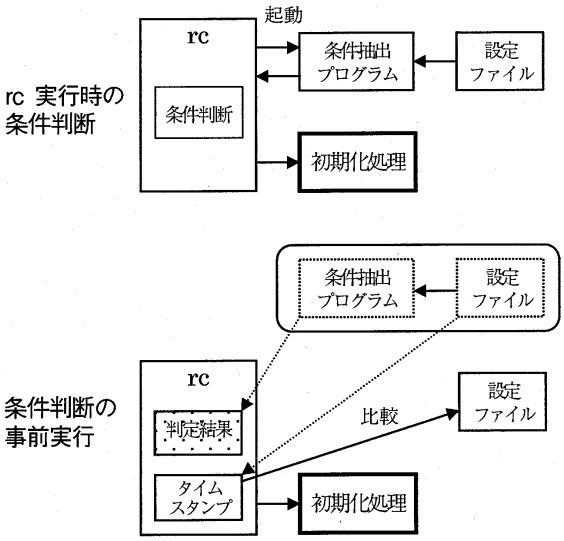


図 1 条件判断の事前実行

## 4. おわりに

UNIX ベースシステムの起動高速化の手法として、rc のバイナリ実行形式化、条件判断の事前実行による rc 処理の簡略化を提案した。

今後、本手法による起動高速化の効果を確認するとともに、rc での処理内容の変更時の対応方法などについても検討していく予定である。