第1章

実測ツールの導入方法からデバイス無効化/サービス停止まで…
チューニングを一通り体験する

起動高速化

小林 明

Linuxは，OSの拡張性を実現するために動的なアドレス解決の処理などを行います．そのため，RTOSに比べると起動に多くの時間がかかります．

少しでも起動時間を短縮するためにはどのようなアプローチがあるのか，実際に試しつつ掘り下げてみたいと思います．

1-1　起動時間チューニングの進め方

起動時間の短縮について，一般的なチューニングの考え方を紹介します．

まずは，ブート・シーケンスと起動時間の現状把握を行い，チューニング対象が何なのかをじっくりと解析します．

● ステップ①…ブート・シーケンスの把握

最初に対象とするBSP（Board Support Package）のブート・シーケンスを理解する必要があります．例えば，通常のArm64 Linuxのブート・シーケンスは**図1**のような流れとなります．

● ステップ②…現状の起動時間の把握

次に，ブートローダ，Linuxカーネル，Linuxシステムの起動時間を把握します．どの部分にどのくらいの時間がかかっているかを明らかにすることで，大まかにチューニング対象を把握します．このとき，なるべくシリアル・コンソールのログにタイムスタンプを付けるなどして，時間がかかっている部分を特定できるように工夫できると良いでしょう．

簡単な方法としては，例えばTera Termなどのターミナル・ソフトウェアの機能で，シリアル・コンソールのログにタイムスタンプを付加する方法があります．

● ステップ③…ブートローダ/Linuxカーネルの最適化

▶起動時間的には小サイズの方が有利

一般的にLinuxカーネルは，ブートローダがカーネルをメモリに読み込んだ後から起動を開始します．このため，Linuxカーネルのサイズは小さい方が起動時間短縮の面で有利です．

ブートローダ（U-Bootなど）やLinuxカーネルには機能を取捨選択してカスタマイズできるようにコンフィグレーションしてビルドできる機能（`make menuconfig`）が備わっています．Linuxカーネルは，汎用的に作成されているので，この機能を利用すれば，不要な機能を削除することでカーネル自体のサイズも小さくできます．

▶すぐ使わない機能はモジュール化も選択肢に

デバイス・ドライバやファイル・システムなどで起動中に初期化する必要がない機能をモジュール化して，後で読み込む機能も備わっています．モジュール化した機能は，最初にロードされるカーネルからは除かれ，ルート・ファイル・システムへ移動できるので，カーネルのサイズが小さくなります．また，初期化処理を起動プロセス全体の後方へ移動できます．

例えば，一番早く起動しなくてはならない重要なアプリケーションが必要とするデバイス（例：ディスプ

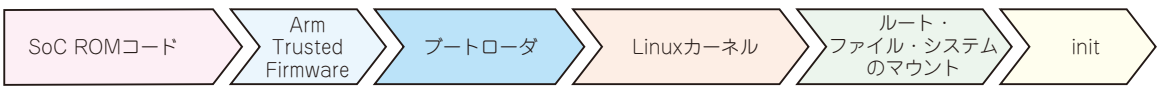

図1　Arm64 Linuxのブート・シーケンス