B-031

コンシューマ向け組込み機器に対する組込み Linux の評価

Evaluation of embedded linux as an operating system for consumer devices

撮津　敦　　伊藤　孝之　　落合　真一

Atsushi Settsu　Takayuki Ito　Shinichi Ochiai

１．はじめに

デジタル家電や携帯電話など、コンシューマ向け組込み機器(以下、コンシューマ機器)は、システムの複雑化に伴い、アプリケーションの規模が年々増大する傾向にある。それに伴い、小型ながら機能が限定されている従来のリアルタイム OS から、メモリ保護やインターネット接続などが可能な OS が適用されてきている。このような高機能 OS の選択肢として組込み Linux に対する興味が高まっている。

一方、家電や携帯電話などのコンシューマ機器は、ユーザからの指示(ボタン押しなど)に対し、即座に対応(電源 ON、画面表示)してきた。組込み Linux の場合でも、即座に応答できない、ユーザはストレスを感じ、不快に思う。

Linux の場合、OS 起動に時間が掛かることは知られており、電源 ON に対する OS 起動の高速化開発が活発に行われている([1],[2],[3])。しかしながら GUI を含んだアプリケーションの起動性能もキー応答に対するユーザ操作感に直結するものであり、通常 1 秒未満の応答が求められる。

今回、筆者らはこの部分に着目し、組込み Linux における GUI アプリケーションの起動性能について測定し、評価を行った。本稿では、この評価結果について述べる。

２．アプリケーション起動性能測定

２．１ 測定方法

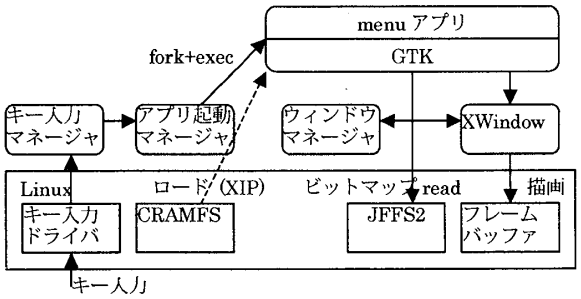

図 1：S/W 構成

測定対象となる S/W 構成を図 1 に示す。

今回、測定対象の GUI として、X/GTK を採用した。X/GTK は PC-Linux で標準的に利用されている GUI であり、アプリケーションの流用性が高く、コンシューマ機器用 GUI の 1 候補として考えられるのが理由である。

測定対象のアプリケーションは、コンシューマ機器で頻繁に起動／終了が実行される可能性が高いメニュー画面処理アプリケーションを想定した。本アプリケーションは、他のアプリケーションよりも GUI 処理の依存性が高い(その他の処理は選択されたアプリケーションを起動するのみ)のも選択した理由の 1 つである。実際に測定したアプリケーションは 19 個のビットマップ(合計 45KB)を表示する。

測定を行った H/W および S/W の仕様を表 1 に示す。

三菱電機（株）情報技術総合研究所, Mitsubishi Electric Corporation Information Technology R&D Center

表 1：H/W・S/W 仕様

H/W		
	CPU	ARM ベースコア 168MHz
	メモリ	RAM 32MB, フラッシュ 32MB
S/W		
	Linux バージョン	2.4.20
	XIP 機能	カーネル/アプリとも ON
	ファイルシステム	CRAMFS:13MB(XIP(Read Only)用) JFFS2:16MB(root file system(R/W)用)
	GUI	XFree86 4.3.0＋GTK2.2

XIP 機能は、フラッシュメモリ上にあるプログラムコードを直接実行するものであり、カーネルコードを実行するカーネル XIP と、ユーザプログラム／共有ライブラリを実行するアプリケーション XIP の２つの機能がある。今回の測定では、双方とも ON の状態で測定している。

測定では、システムコールの開始／終了、プロセス(スレッド)スイッチをトレースし、そのトレース結果を/proc にて読み出せるカーネルを用意した。また測定アプリケーションには、初期化の各フェーズにマーカとなるシステムコールを発行することで、アプリケーション内の動作を追跡できるようにした。

２．２ 測定結果

図２に測定結果を示す。各処理の内容は以下の通りである。

(1) キー応答
キー入力から測定アプリケーションの起動開始(execve()実行)直前の区間。割込みハンドラ/ドライバ/マネージャプロセスが介在。

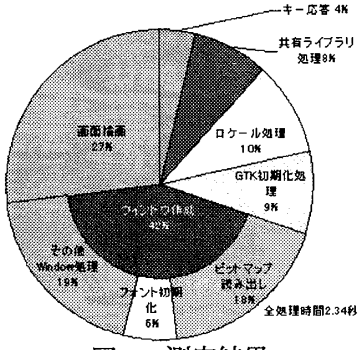

図 2：測定結果

(2) 共有ライブラリ処理
測定アプリケーションが execve()にて起動されると、main()を実行する前に必要な共有ライブラリのリンク処理を行う。このリンク処理を行う区間。

(3) ロケール処理
日本語など、アプリが英語以外の言語を使うための初期設定を行う区間。EUC/Shift-JIS/Unicode などのロケールデータのロードが行われる。

(4) GTK 初期化処理
GTK+を使用するための初期設定が行われる区間。

(5) ウィンドウ作成
アプリケーションの初期画面として使用するウィンドウが生成される区間。この区間には、ウィンドウ処理の他、ビットマップファイルの読み出しやフォントの初期化処理が含まれる。この処理はアプリケーション単体で処理される(描画そのものは行われていない)。

(6) 画面描画
画面上にアプリケーションの初期画面を描画する区間。アプリケーション/X サーバ/ウィンドウマネージャのスイッチが頻繁に起こる。

３．考察

測定結果では、起動に 2.34 秒程度掛かることが判明した。全体処理の内、実際の描画処理は全体の 27%であり、また、キー入力からアプリケーションの起動までは 4%程度であるため、結果としてアプリケーションの初期化処理に全体の 70%近く消費していることがわかる。またアプリケーションの初期化処理の内、共有ライブラリなど OS レベルでの初期化処理は 8%程度であり、残りの 61%は GUI レベルでの初期化処理であることが判明した。

プロセッサ能力や XIP による ROM/RAM 実行性能差などの差異はあるものの、X/GTK を利用した場合、各処理の時間消費率はほぼ同様になると考えられ、起動に 1 秒以上掛かることが予想される。これはコンシューマ機器では致命的な問題である。

アプリケーションの初期化処理を削減する方法の 1 つとして、予め必要なアプリケーションを起動しておき、キー入力が入ると、選択されたアプリケーションの画面に切り替えるという方法が考えられる。しかし、この場合ではアプリケーションを起動することによるメモリ使用量の増加が問題となる。XIP 化によって、メモリ削減は期待できるもの、データ/ヒープ/スタックはメモリに配置するため、メモリ消費量は増加(測定では 2MB 程度増加)し、メモリ資源が乏しいコンシューマ機器には推奨できない。よって、通常の起動による改善策を検討する必要がある。

４．改善案検討

今回の測定結果を元に、改善が可能な箇所について検討した。以下、検討した内容について述べる。

４．１　ＯＳレベルでの改善

(1) ファイル読み出しの高速化
ウィンドウ生成処理(図３)においてビットマップファイルの読み出しに全体の 42%が消費されている。これらファイルは処理の重い JFFS2 に置かれており、read()に時間が掛かっている。そこで、これらファイルの置き場所をより高速なファイルシステムに変更する。例えば、ファイルイメージを ROM 化し、フラッシュメモリから直接参照できるようなファイルシステムを構築すれば、この読み出し時間をほぼ 0 にすることができる。コンシューマ機器用アプリケーションでは、メニュー画面等で使用されるビットマップファイルは基本的に固定であり、より簡易なファイルシステムでも対応できると考える。

(2)共有ライブラリアドレス検索処理の高速化
共有ライブラリは、アプリケーション実行時に必要となるライブラリがロードされ、アプリケーション内で未定義になっているライブラリ関数のリンク処理が行われる。特に GUI を利用するアプリケーションは大量の X/GTK ライブラリをリンクするため、リンク処理に時間が掛かる。家電などのコンシューマ機器では、使用するアプリケーションは決まっており、使用する共有ライブラリも固定的である。よって、プロセス空間上で各共有ライブラリ位置を固定化し、そのアドレス情報をアプリケーションの共有ライブラリリンク情報領域に予め格納すればこのリンク処理を省くことが可能となると考える。

４．１　ＧＵＩライブラリレベルでの改善

(1) ロケール処理の共有化
ロケール処理では、各種言語用にロケール情報からロケールデータを作成するため、処理に時間が掛かっている。基本的に言語は出荷時に固定的であり、またアプリケーション毎で異なることはないと考えられる。よって、出荷時に予め作成したロケールデータを、各アプリケーションで共有化しておけば、アプリケーション毎のロケール処理を省くことが可能となると考える。

(2) フォント処理の共有化
ウィンドウ生成処理(図 3)では、フォントの初期化も行われている。これもロケール処理と同様にフォント情報からフォントデータをアプリケーション毎に生成するため、時間が掛かる。フォントも各アプリケーションで共通に利用することが可能であり、フォント処理を全プロセスで共有化することで、各アプリケーション毎に行っているフォント初期化処理を削除することが可能と考える。

４．３　改善効果

上記改善策を施した場合の効果を図 3 に示す。前述した改善策が実現できれば、41%の削減効果が期待でき、処理時間を 2/3 まで短縮できる可能性がある。

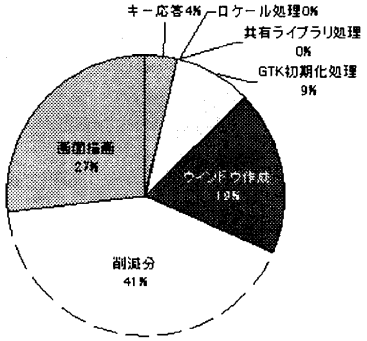

図３：改善策の効果

５．おわりに

本稿では、コンシューマ向け組込み機器に対し、組込み Linux が適用可能かという観点のもと、GUI を備えたアプリケーション起動性能の測定結果と、起動の改善策について述べた。評価の結果、全体の 70%近くがアプリケーションの初期化処理に費やされ、かつ 61%が GUI レベルの初期化処理であることが判明した。またこれら初期化処理に幾つかの改善を施すことで、2/3 まで起動処理を短縮できる可能性があることを述べた。

今回検討した改善策だけでは、依然として起動時間は 1 秒を切れておらず、さらなる起動時間短縮が必要である。その方策の 1 つとして XIP をやめることが考えらる。実際に測定した結果では XIP を実施しないほうが起動性能が向上する結果がでている。これにより実行性能が 1.6 倍(RAM/フラッシュの性能差 3 倍)になるため、これを見込めば起動時間が 1 秒以下になる可能性がある。しかし、XIP をやめることは、メモリリソースを消費することになり、搭載できる RAM 量の問題から採用すべきか判断できていない。

今後は、これら改善策の検証を行うとともに、XIP の効果についてもより詳細に検証していく予定である。

参考文献

[1] 町田、田中、篠原、大和田、水山、天野、関口：
「Linux をデジタル家電向けに改良、起動時間と応答時間を半分以下に」、日経エレクトロニクス 2003.7.7 号、p.131～142

[2] 丸山、八木、森田、吉本：
「情報家電における OS 起動時間解析」、情報処理学会第 66 回全国大会(2004)

[3] CE-Linux Forum 1.0 Specification：
http://www.celinuxforum.org/PublicSpecifications/CELF_Specification_V_1_0_R2.pdf