Linux with RT-Preemptible Patchを用いたロボットシステムの実時間制御

○白井 伸明　羽鳥 裕樹　佐藤 大祐　金宮 好和（東京都市大学）

Real-Time Control of a Robot System by Using Linux with RT-Preemptible Patch

*Nobuaki SHIRAI, Yuki HATORI,
Daisuke SATO, and Yoshikazu KANAMIYA (Tokyo City University)

Abstract— We address the real-time (RT) control problem of a robot under the Linux OS. A RT-preemptible kernel patch for Linux is now under development. With this kernel patch the need for a special user API can be avoided. In addition, the sampling time can be decreased even under one millisecond. However, since the patch is still under development, we need to verify its capability for RT robot control. In this paper we examine real-time multitask operation under Linux with applied RT-preemptible patch. Under the experiment, we control the motion of a PA-10 robot arm and acquire force data from two force/torque sensors in real-time. We verify the possibility for robot control and memory management at the same time. Results from the experiments are presented.

Key Words: RT-preemptible patch, Linux, Real-time control, Robot arm, F/T sensor

1. 緒言

ロボットシステムを確実に制御するためにはハードリアルタイムでの実時間制御が不可欠である．現在，国内で使用されているリアルタイムOSには，商用OSではVxWorksやQNX，フリーOSではRTLinux，ART–Linuxなどがある．しかしVxWorksやQNXはライブラリや各種ソフトがLinuxほど豊富ではない．また，RTLinuxは無償版のサポートがkernel 2.4で止まっている．ART–Linuxは，最近になりkernel 2.6に対応したシングルCPU用途のものが開発された[1]．これらリアルタイムOSの欠点として，専用APIの使用が必要な点がある．また，カスタマイズしたLinuxのvanilla kernelを用いて実時間制御を行う試みもなされている[2]．しかし，この手法では制御周期を1 ms以下にすることができない．

以上のように，リアルタイムLinuxでは環境を整えやすいが，継続して使用する上で適当なものがなく，通常のLinux kernelではリアルタイム性に問題がある．しかし最近，kernel 2.6用に高精度なリアルタイム性を持たせるRT-preemptible patchが開発された[3]．

2. Linux with RT-preemptible patch

RT-preemptible patchを適用することで，Linuxは1 ms以下の精度で実時間制御が可能になる．このRTOSを使用することで，環境を開発から実行までLinuxに統一できる．また，POSIX準拠のAPIのみでリアルタイムプログラムが作成できる．さらにこのpatchは今後mainline kernelにマージされる予定であり，完全マージの目標は数年以内とされている．マージ後は，標準のkernelをカスタマイズするだけで使用できる．

以上よりLinux with RT-preemptible patchは今後最も導入が容易なLinuxベースのRTOSとなる可能性が高い．しかし現在は開発中のため，RTOSとしてどの程度の性能があるか，またロボット制御への適用が可能であるか検証する必要がある．これまで我々は本RTOSを用いて制御周期3 msでロボットアーム一本の制御を行い，リアルタイム性を検証してきた[4]．本稿では通常のLinuxでは不可能な制御周期1 ms以下の実時間制御と，マルチスレッドの制御におけるリアルタイム性について検証を行う．

Linuxにおけるリアルタイムプロセスは，静的優先度に従ってCPUを割り振られる．通常のプロセスは静的優先度が0であり，リアルタイムプロセスは1以上99以下の値に設定できる．プリエンプションはkernel内部のスケジューラが管理しており，ユーザはプロセスの優先度の決定のみ可能である．また，プリエンプションを行う際にはレイテンシが発生する．レイテンシを含めた処理時間が，デッドライン以内であれば，システムは正常である．しかしレイテンシはハードウェアに依存して変化する．よってプリエンプションおよび，レイテンシに関して検証を行う必要がある．

3. プリエンプション検証実験

静的優先度の異なる三つのスレッドを立ち上げ，周期実行を行った．その際の各スレッド処理の推移を測定し，正しくプリエンプションが行われているか検証した．静的優先度はそれぞれ99，98，97に設定した．このうち，二つのスレッドの実行周期と処理時間を長くし，残りの一つのスレッドは短くした．そのうえで周期の速いスレッドの静的優先度を97とし，実験を30回繰り返した．本実験では各スレッドの制御開始時間は同期していない．以下の実験で使用したコンピュータはPentium Ⅱ 450 MHzである．また，kernelは2.6.29.4を使用し，patchは2.6.29.4-rt16を使用した．

3·1 実験結果および考察

Fig. 1に，実験結果の一例を示す．図中での処理は以下のように遷移する．

1. **立ち上がり開始時間：** スレッドに設定されている起床時間．スレッドが割り込みを要求する．
2. **立ち上がり開始〜立ち上がるまでの区間：** スレッドが割り込みをかけてから実際に処理が開始されるまでの区間．この区間がレイテンシである．
3. **立ち上がり完了の時間：** スレッドの処理開始時間．CPU が割り振られ，処理を開始する．
4. **立ち上がり完了〜立ち下がり開始までの区間：** スレッドが処理を行う区間．ただし，線上に点がある区間のみ実際に処理を行っている．点がない区間は処理を中断している．
5. **立ち下がり開始の時間：** スレッドの処理終了時間．CPU を手離し次の処理開始時間まで休眠する．
6. **立ち下がり開始〜次の立ち上がり開始までの区間：** スレッドが処理をしない休止区間．

優先度 98 のスレッドの処理中に，優先度 99 のスレッドが実行可能状態になったためにプリエンプションが発生していることが分かる．また，優先度 97 のスレッドは他の二つのスレッド処理が終わるまで処理が行われない．よって正しくプリエンプションが行われている．

4. 周期実行検証実験

我々のシステムでは三菱重工業株式会社製 7 自由度マニピュレータ PA–10 およびニッタ株式会社製 6 軸力覚センサ IFS-67M25A 50-I 40 を使用する．そこで，これらの制御の周期実行が可能であるか確認した．力覚センサの制御にはドライバを使用し，メモリマップを用いてデータの取得を行っている．そのため OS はメモリの管理やセンサからの割り込みなどの処理も行う必要がある．本 RTOS がこれらの処理を行いながら周期実行を行えるか確認した．

4·1 実験条件

本実験では本 RTOS に PA–10 コントローラ 1 台および力覚センサを 2 台接続した．PA–10 を 1 本，および力覚センサ二つを制御するスレッドをそれぞれ一つ立ち上げ，その際のレイテンシおよび実行時間を測定した．制御は 600 s 行った．

PA–10 制御スレッドでは，PA–10 コントローラ 1 台に対して ARCNET を用いて角度指令を送信し，エンコーダの値を取得する．本実験では制御周期を 2000 μs とした．測定した 1 回の通信時間は，1.6×10^3 μs だった．力覚センサスレッドの制御周期は 250 μs とした．測定した 1 回の力データ取得に要する時間は，5 μs だった．力覚センサ制御スレッドの制御周期が短いため，優先して実行される必要がある．そこで，力覚センサ制御スレッドの静的優先度を 99 に設定し，PA–10 制御スレッドは 98 とした．

4·2 実験結果および考察

Fig. 2 に両スレッドのプリエンプションを行う際に生じたレイテンシを示し，Fig. 3 には両スレッドのレイテンシおよび実行時間を示す．Table 1 にはそれらの最大値と平均値を示す．

PA–10 制御スレッドのレイテンシおよび実行時間には，力覚センサ制御スレッドのレイテンシおよび実行時間が含まれる場合がある．Table 1 より二つのスレッドは，レイテンシおよび実行時間が最大の場合でも周期内で処理が終了している．よって我々のシステムでは本 RTOS によって周期実行が可能であると分かった．

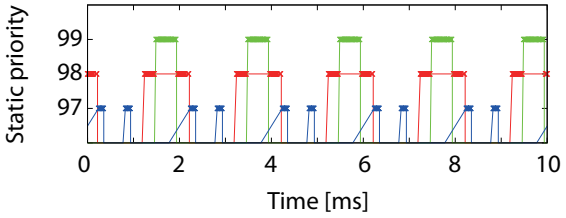
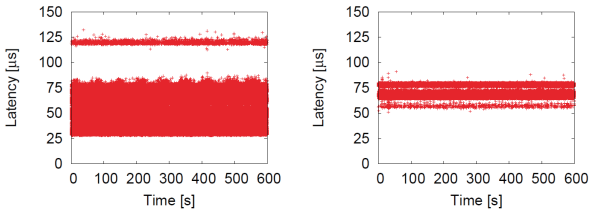
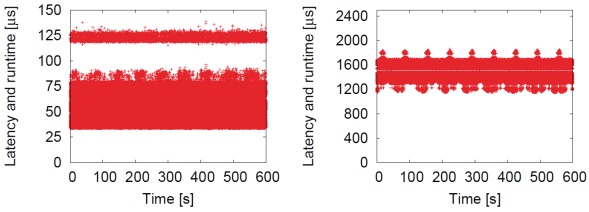


Fig. 1 Result of the experiment.



(a) (b)

Fig. 2 Latency. (a) F/T sensor control thread. (b) PA–10 control thread.



(a) (b)

Fig. 3 Latency and runtime. (a) F/T sensor control thread. (b) PA–10 control thread.

Table 1 Latency and runtime of two threads.

	FTS	PA–10
	Max (average) [μs]	
Latency	132 (33)	91 (70)
Latency and runtime	139 (38)	1863 (1476)

5. 結言

Linux with RT-preemptible patch によるアームおよび力覚センサからなるロボットシステムの実時間制御の検証を行った．実験結果より，本 RTOS が我々のシステムに必要なリアルタイム性を実現可能であると分かった．

参考文献

[1] 石綿 陽一，加賀美 聡，西脇 光一，松井 俊浩，“シングル CPU 用 ART–Linux 2.6 の設計と開発”，日本ロボット学会誌，26，6，pp.78-84，2008.
[2] 熊谷正朗，“汎用 Linux によるロボット制御 -KNOPPIX + kernel 2.6 による開発環境の構築-”，日本機械学会ロボティクス・メカトロニクス講演会’06，1P1–C40，2006.
[3] “Real-Time Linux Wiki”，http://rt.wiki.kernel.org/index.php/Main_Page
[4] 久米修平，金宮好和，佐藤大祐，“Linux におけるリアルタイムプリエンプションパッチに基づいたロボットのリアルタイム制御”，日本機械学会ロボティクス・メカトロニクス講演会’08，1P1–E16，2008.