5K-6

リアルタイムスケジューリングのための実験基盤の開発

黒井崇史† 早川栄一‡
拓殖大学 工学部 情報工学科†‡
r38439@st.takushoku-u.ac.jp† hayakawa@cs.takushoku-u.ac.jp‡

1. はじめに

現在利用されている組込み機器の多くは低消費電力を目指すものや，リアルタイム性の向上などの点に力を注いでいるものが多い．これらの要求は年々増してきており，それによってタスク管理の重要性も増してきている．

しかし，現在はこれらの要求を解決するためにハードウェアによる解決や，新たな組込み OS を定義することでの解決は頻繁に行われているが，スケジューラ単体のタスク管理による解決はこれらの要求に対して決して都合のよいものではなくなってきている．そのためには，要求を満たすタスク管理が可能なスケジューリングアルゴリズムを定義する必要がある．

そこで本研究では，リアルタイム性に重点を置いたスケジューリングアルゴリズムの定義を実験によって行うための，実験データを取得する実験基盤の開発を行う．さらに取得した実験データから，リアルタイム OS においてリアルタイム性の低下の原因となる，優先度逆転やデッドラインオーバーの検出を行うツールの開発を行った．

2. 特徴

(1)2 台の組込み機器を用いたログの取得

本システムではリアルタイム性に重点を置いたスケジューラ定義することを目的としているので，イベントの発生と検出においてリアルタイムに行う必要がある。そこで本システムではログの取得部だけでなくイベントの発生部にも独立した組込み機器を用いてそれぞれにリアルタイム OS を実装した．

これは，実際のイベント発生は組込み機器側で発生することであり，それを同じ時間単位を扱うリアルタイム OS 用いることで一つの機器と考えられるようにするからである．イベントの発生部をログの取得部と一つにしてしまうと，OS の実行とログ取得の妨げになってしまい，PC から直接イベントを発生させる方法を取ると，イベントの発生にリアルタイム性を持たせることができない． イベントの発生とログの取得の両方にリアルタイム性を確保させるために，2 台の組込み機器を用いる方法を取った．

(2)実行時のデータサイズの縮小

組込み機器をターゲットとしたシステムを開発する上では使用できる記憶領域の都合上，システムの規模には限りがある．本システムでは使用するデータをビット単位で削減し，不必要なデータサイズを取らずデータサイズを小型化している．

(3)スクリプト言語を用いた解析

本システムでは，ログの解析部の開発にスクリプト言語である Perl を用いている．Perl を用いる利点として，インタプリタ型の言語であるためコンパイル等の処理が不要であることが挙げられる．これにより，ログの取得先や取得内容を追加や変更した際に，即座に実行が可能になる．

Development of Experiment Environment for Real-time Scheduling
Takashi KUROI, Eiichi HAYAKAWA
Faculty of Engineering, Takushoku University

3. 構成

本システムは 2 台の組込み機器とリアルタイム OS を用いて，I/O への操作や割込み処理を行うことでのタスク遷移などの変化を監視する．全体の構成を図 1 に示す．

システムは大きく分けてイベントの発生部，ログの取得部，ログの解析部の三つに分けられ，イベントの発生部とログの取得部にはそれぞれ組込み機器を用いる．

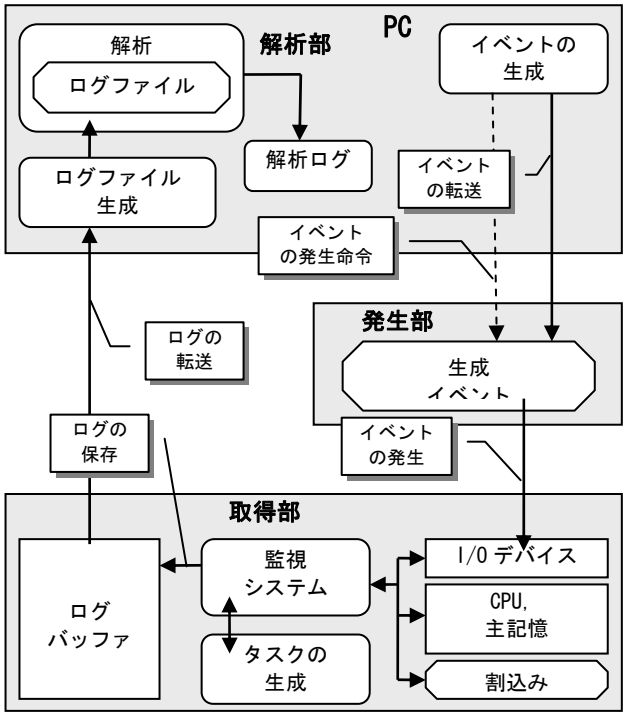

図 1 全体構成

3.1. イベントの発生部

1 台目の組み込み機器は，I/O へのイベントの発生用に用いる．発生させるイベントの生成は PC 上で行い，事前に転送しておくことで任意のタイミングでイベントの発生を行う．

3.2. ログの取得部

2 台目の組込み機器は，発生したイベントや割込みに対してのタスク遷移など変化の監視やログ情報の取得に用いる．取得したログ情報はバッファに保存され，イベントの終了やすべての実行タスクが終了するとバッファに保

存したログ情報はPCへ転送される.

3.3. ログの解析部

PCへと転送されたログ情報は，それぞれのログ情報の種類ごとにログファイルの生成を行う.生成されたログファイルは解析ツールに読み込まれ、優先度逆転やデッドラインオーバーの検出が行われる.検出が行われると，優先度逆転やデッドラインオーバーの発生に関連するタスクのログファイルを作成する.

4. 設計

4.1. イベントの発生部

イベントの発生部は主記憶にマップされた I/O デバイスのアドレスのレジスタ値を書き換えることで仮想的にI/Oの動作を与える．与えるイベントの動作は次の表1に示す.

表1　発生イベント動作一覧

I/Oデバイス	動作		
LED(二つ)	両方OFF	LED1 ON	LED2 ON
	両方ON		
モータ(左右)	停止	前進	後退
	左折前進	右折前進	左折後退
	右折後退	左回転	右回転
タッチセンサ(左右)	左右OFF	左ON	右ON
	左右ON		
光学センサ(明度:0～255)	黒(85以下)	灰色(86～169)	白(170以上)

4.2. ログの取得部

(1)CPUの監視

CPU の監視ではタスクの遷移ログを取得する．取得はタスクの遷移時に行い，取得するデータは遷移時刻，実行タスクのID，最優先度の実行待ちタスクのID，実行タスクの優先度，最優先度の実行待ちタスクの優先度である.

取得するログはいずれも整数である.

(2)主記憶の監視

主記憶の監視では書換え内容のログを取得する.取得は書換え時に行い，取得するデータは主記憶の書換え時刻，書換えページアドレスである.

(3)I/Oデバイスの監視

I/Oデバイスの監視では使用I/Oデバイスのログを取得する．取得はI/Oデバイスの使用時に行い，取得するデータはI/Oデバイス使用時刻,I/Oデバイス使用タスクのID,使用したI/Oデバイスのマップアドレスである.

(4)割込みの監視

割込みの監視では発生のログを取得する.取得は割込み発生時に行い，取得するデータは割込み発生時刻、割込みタスクである.

(5)生成タスクログ

(1)～(4)のログとは別に生成されたタスクのログを作成する．作成するログの内容はタスクの ID，タスクの起動時優先度，タスクの絶対デッドライン，タスクの相対デッドラインである.

4.3. ログの解析部

取得したログはCSV形式のファイルのようにカンマで区切る形式にファイル化して，生成されたログファイルの内容の前後関係を元に優先度逆転とデッドラインオーバーなどの解析を行う.

優先度逆転やデッドラインオーバーなどの解析の Perl による例を次に示す.

```
#前のタスク優先度の取出し.
tsk_pri = get_tsk_pri( tsk_log );

#後ろのタスク優先度の取出し.
nxt_pri = get_nxt_pri( tsk_log );

#優先度逆転のチェック.
check_pri_inv( tsk_pri, nxt_pri );
```

以上のライブラリを用いることで優先度逆転の解析が行える.

5. 実現

(1)実現環境

実現において使用した環境は，組込みハードウェアにはシリコンリナックス社製の CPU ボード CAT709[1]，実装したリアルタイム OS には μITRON4.0 仕様[2]に準拠した OS である TOPPERS/JSP[3]を用いた.

(2)実現規模

本研究で作成した各ログ取得ライブラリのソースコードのプログラムサイズは，CPU のログ取得が 1702 バイト，I/O デバイスのログ取得が 3384Byte，割込みのログ取得が 821Byte，生成タスクのログ取得が 953Byte である．ログ取得ライブラリの合計は 6860Byte と 10kByte 以下と小規模なものにできた.

6. おわりに

本研究ではOSの実行時のタスクの状態とI/Oデバイスの監視を実験によってデータ取得を行い，そのデータからリアルタイム性の低下原因を解析する実験基盤の開発を行った．これにより，リアルタイム性を重要視した新たなリアルタイムスケジューラの定義に役立てることができる.

今後の課題を次に示す.

(1)設計における未実装部分の実現

(2)新たな監視先の追加

(3)取得内容の追加

参考文献

[1] シリコンリナックス株式会社 CAT709 ホームページ
http://www.si-linux.co.jp/index.php?CAT/CAT709

[2] μITRON4.0 仕様ホームページ
http://www.ertl.jp/ITRON/SPEC/mitron4-j.html

[3] TOPPERS/JSP ホームページ
http://www.toppers.jp/jsp-kernel.html