特集論文

TRON EMULATORの開発

Development of TRON EMULATOR

天野晋弥*
S. Amano

概　要

TRON EMULATORとは、μITRONから組込みLinuxへのOS移行を支援するソフトウェアである。本ソフトウェアを使用することで、アプリケーションの作り直しが不要となり、OS移行に要するコストを大幅に削減することができる。

本稿では、TRON EMULATORが持つ機能および実装方法について記述する。

（なお、本稿ではμITRON標準仕様に準拠する一般OSを指してμITRONと記述する。）

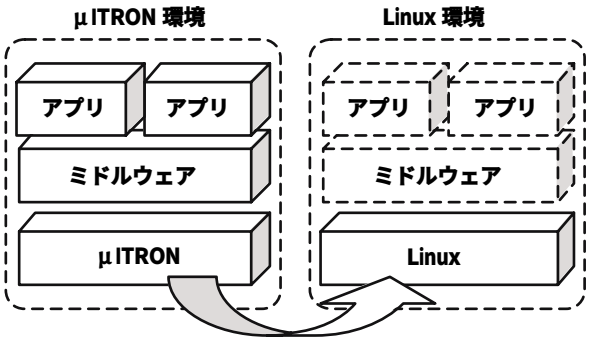
Synopsis

TRON EMULATOR is operating system migration support software to embedded Linux from μITRON. By using this software, customers can dramatically reduce a cost of the operating system migration, because the remaking of application is unnecessary. This paper describes the TRON EMULATOR function and an implementation method. And “μITRON” means the operating system in accordance with μ ITRON standard specifications, in this paper.

１．はじめに

近年、モバイル端末から家電機器まで、さまざまな製品で組込みLinux（以下、Linuxと記述する。）の採用に増加傾向が見られる。この背景として、これまで単一の機能だけを持っていた機器に対しても、複数の機能が求められるようになり、ソフトウェア資源や開発情報の豊富なLinuxへ移行が進んでいるなどの理由が考えられる。

しかし、実際には、OS移行には多大な作業時間および費用が必要であるため、移行を行うことは容易な作業ではない。そこで、このような課題を解決し、ユーザのOS移行に必要となる作業負担を軽減する目的でTRON EMULATOR（以下、TRONEMUと記述する。）を開発した。



※ベースOSは移行するが、上位のアプリケーションやミドルウェア資産は、できる限り流用できることが望ましい

図１　一般的なOS移行のイメージ

* （株）日新システムズ

２．OS移行の課題

OS移行を行う場合、ターゲットボード上のOSを移行しても、上位のアプリケーションやミドルウェアについてはそのまま動作しないことが大半である。

そのため、移行前後のOSの仕様について違いを把握し、状況に応じてさまざまな修正を行う必要がある。

そのひとつとして、アプリケーションタスクの状態を管理し、タスクスケジューリングを行うための情報がある。（表１ μITRONとLinuxの比較情報 参照）

表１　μITRONとLinuxの比較情報

項目	μITRON	Linux
タスク状態	実行状態 実行可能状態 待ち状態 二重待ち状態 強制待ち状態 休止状態 未登録状態	実行状態 待ち状態 デバイス待ち状態 スリープ状態 ゾンビ状態
スケジュール方式	FCFS	SCHED_OTHER SCHED_RR SCHED_FIFO
優先度	実装依存	0～99

他にも、タスクとして動作するユーザアプリケーションがカーネルと情報をやり取りするためのシステムコール（μITRONではサービスコールとも呼ばれる）にも互換性がない。そのため、例えば、タスクの生成や排他用オブジェクトの作成、などに使用するシステムコールおよび実行方法についても異なる。

表２　μITRONとLinuxのシステムコール比較

項目	μITRON	Linux
タスク生成	cre_tsk acre_tsk	fork pthread_create
セマフォ制御	cre_sem wai_sem sig_sem	sem_init sem_get sem_post
ミューテックス制御	cre_mtx loc_mtx unl_mtx	pthread_mutex_init pthread_mutex_lock pthread_mutex_unlock

OS移行では、これらの違い全てに対策を行い、移行前と同じ機能を持つシステムとして再構築する必要があるが、システムの規模に応じて発生する膨大な移行コストを、いかに軽減するかがOS移行成功の鍵となる。

３．TRONEMUの機能

TRONEMUは、μITRONからLinuxへのOS移行支援ツールとして、前述の課題を軽減する目的で開発されており、次のような機能を持つ。

TRONEMUの主な機能
１．μITRON 3.0/4.0 システムコールライブラリ
２．μITRONタスクスケジューリングのエミュレート
３．簡易シェル
　（タスク情報等の表示、イベントトレース）

３．１　μITRON 3.0/4.0 システムコールライブラリ

μITRONには、数多くのシステムコール（標準仕様3.0および4.0）があるが、TRONEMUに含まれるシステムコールライブラリをリンクすることで、Linux上でμITRONのシステムコールを、そのまま呼び出して使用することが可能となる。つまり、μITRONのアプリケーションをLinux用に修正する必要がない。（図２　TRONEMUを使用した場合の移行イメージ）

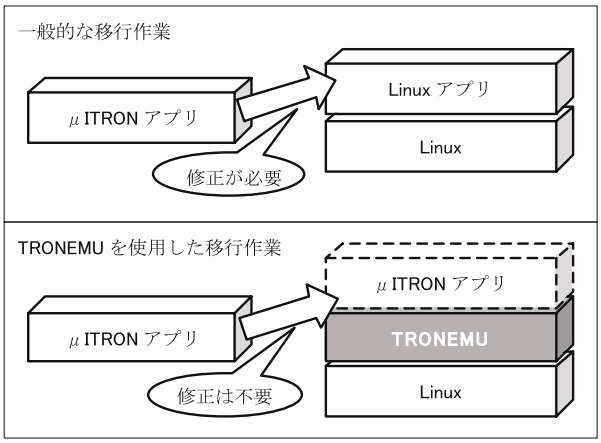


図２　TRONEMUを使用した場合の移行イメージ

なお、対応するμITRONシステムコールは以下の通りである。

表３　μITRON 3.0 システムコール対応一覧

機能	対応状況
タスク管理機能	○
タスク付属同期機能	○
同期・通信機能	○
拡張同期・通信機能	○
メモリプール管理機能	○
時間管理機能	○
割込み管理機能	未対応
システム管理機能	一部未対応

※1 割込み管理機能には対応していないため、Linuxデバイスドライバでの対応が必要
※2 システム管理機能の一部システムコールで取得する情報は、実装依存であるため（ユーザによる変更が可能）

表４　μITRON 4.0 システムコール対応一覧

機能	対応状況
タスク管理機能	○
タスク付属同期機能	○
タスク例外処理機能	○
同期・通信機能	○
拡張同期・通信機能	○
メモリプール管理機能	○
時間管理機能	○
システム状態管理機能	○
割込み管理機能	未対応
サービスコール管理機能	一部未対応
システム構成管理機能	一部未対応

※1 割込み管理機能には対応していないため、Linuxデバイスドライバでの対応が必要
※2 サービスコール管理機能およびシステム構成管理機能の一部システムコールで取得する情報は、実装依存であるため（ユーザによる変更が可能）

３．２　μITRONタスクスケジューリング

μITRONのタスクスケジューリングは、FCFS（First Come First Served）と呼ばれる方式で行われる。

また、タスク状態についても、Linuxの5種類に対し、μITRONでは異なる7種類の状態を持つ。（表５ LinuxとμITRONのタスク状態 参照）また、タスク優先度範囲にも違いがある。

表５　LinuxとμITRONのタスク状態

Linux (5種類)	μITRON (7種類)
実行状態	実行状態
待ち状態	実行可能状態
デバイス待ち状態	待ち状態
スリープ状態	二重待ち状態
ゾンビ状態	強制待ち状態
	休止状態
	未登録状態

このため、LinuxとμITRONではタスクのスケジューリングに関する条件が異なり、OS移行の際にはタイミングなどの調整が必要となる。

しかし、TRONEMUはこのタスク状態や優先度の違いを吸収し、μITRONと同様のスケジューリングを実現する。これはユーザアプリケーションなどで意識する必要はなく、TRONEMUのシステムコールライブラリに含まれる関数を用いてタスクを制御するだけで、μITRONと同じスケジューリングポリシ、そしてタスク状態、優先度に応じたスケジュールが行われる。

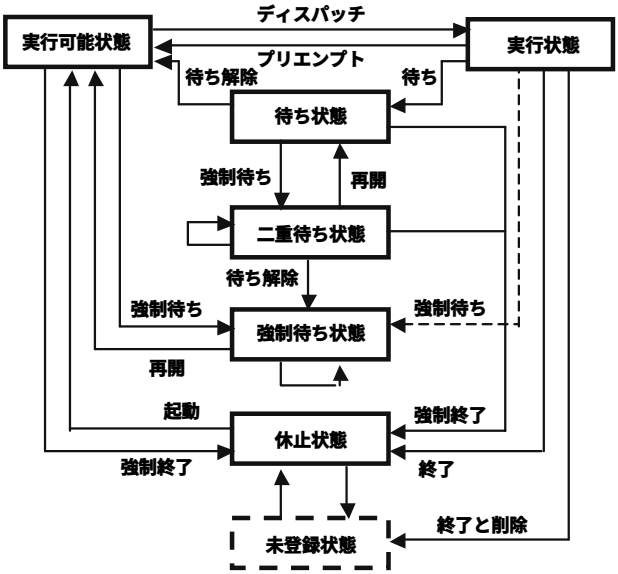


図３　μITRONタスク状態遷移図

３．３　簡易シェル

TRONEMUは、簡易版のシェル機能を持つ。タスクや各オブジェクト（イベントフラグ、セマフォなど）情報の表示や変数の参照、システムコールの実行状況をトレースする機能を持ち、デバッグ用途に使用可能である。これらのデバッグ機能は、アプリケーションから独自のデバッグ関数を呼ぶことでも利用できる。

例えば、現在実行中のタスク情報を確認する場合、shw_tskコマンドを実行する。（図４　タスク状態表示コマンドの実行イメージ 参照）

```
N-Shell version 1.00 (uITRON Ver 4.02)
Copyright(C) Nissin Systems Co.,Ltd.

->
-> shw_tsk                          <-- タスク管理情報参照コマンドの実行

[ TSK ] 0x8107da0 { 258, 420, 0x810c920, 0x0 }
ID   ATTR EXINF    TASK     PRI STACK    STKSZ    STAT WAIT WOBJ TIMEOUT
---- ---- -------- -------- --- -------- -------- ---- ---- ---- --------
   1 0000 00000000 08049384  10 b755ef44     1024 DMT             0        0
   2 0000 00000000 08049384  10 b6b5df44     1024 DMT             0        0
   3 0000 00000000 08049384  10 b615cf44     1024 DMT             0        0
   4 0000 00000000 08049384  10 b575bf44     1024 DMT             0        0
   5 0000 00000000 08049384  10 b4d5af44     1024 DMT             0        0
   6 0000 00000000 08049384  10 b4359f44     1024 DMT             0        0
   7 0000 00000000 08049384  10 b3958f44     1024 DMT             0        0
   8 0000 00000000 08049384  10 b2f57f44     1024 DMT             0        0
   9 0000 00000000 08049384  10 b2556f44     1024 DMT             0        0
  10 0000 00000000 08049384  10 b1b55f44     1024 DMT             0        0
 258 0002 0804953a 080739a0   1 b7f5e344     8192 RUN             0        0
value = -18 (0xffffffee)
->
```

図４　タスク状態表示コマンドの実行イメージ

同じくsta_evt、stp_evtコマンドで、取得したイベントトレース情報を、shw_evtコマンドで出力し、タスクのイベント実行状況を確認することも可能である。（図５　イベントトレースコマンドの実行イメージ 参照）

```
N-Shell version 1.00
Copyright(C) Nissin Systems Co.,Ltd.

->
-> def_evt 0,4096                          <-- バッファサイズ 4096 バイト
->
-> fil_evt "TSK", 1                        <-- タスク管理関数をロギング
->
-> sta_evt 0,4                             <-- 開始＆リアルタイム表示
00:18.420|LC---|IRQ|WUP_TSK |016|0x0 (0)
00:18.420|LC--R|IRQ|WUP_TSK |016|[E_OK (0)]
00:19.450|LC---|IRQ|WUP_TSK |016|0x0 (0)
00:19.450|LC--R|IRQ|WUP_TSK |016|[E_OK (0)]
-> stp_evt                                 <-- 停止
->
-> shw_evt                                 <-- ロギングの表示
00:17.440|LC---|IRQ|WUP_TSK |016|0x0 (0)
00:17.440|LC--R|IRQ|WUP_TSK |016|[E_OK (0)]
      :
->
```

図５　イベントトレースコマンドの実行イメージ

このように、システム全体の情報や各タスクの動きを確認し、μITRONからLinuxへのアプリケーション移行直後のデバッグを容易に行うことができるよう実装されている。

４．TRONEMUの実装方法

TRONEMUは、Linuxユーザ空間で動作するアプリケーションプログラムとして実装を行っている。Linuxカーネル空間は使用しない。

また、μITRONのタスクやスケジューリングをエミュレートするため、Linuxの標準C言語ライブラリ（glibc）に含まれるPOSIXスレッドライブラリを利用した実装となっている。（図６　TRONEMU使用時のLinuxイメージ 参照）

TRONEMUから生成されたタスクは、TRONEMUプロセス下で動作するスレッド(※1)として生成されており、μITRONのスケジューリングポリシであるFCFSは、タスクが独自にTCB情報を用いてスケジュール管理する実装となっている。

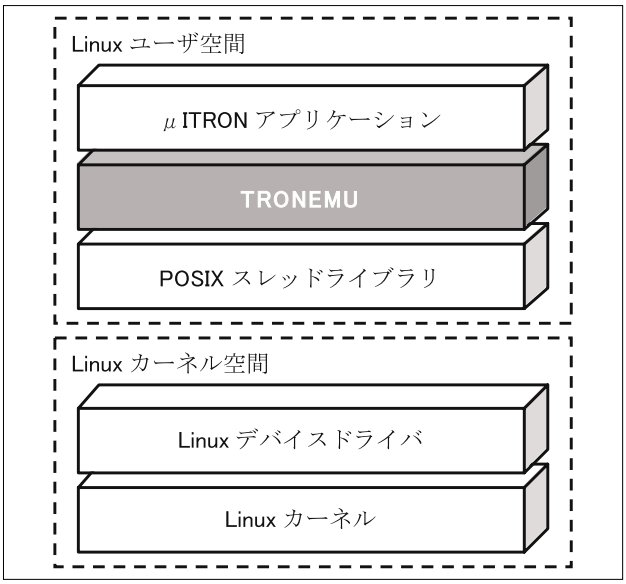


図６　TRONEMU使用時のLinuxイメージ

※１　POSIXスレッドライブラリ（pthread_create）の使用で生成されるプログラムの実行単位

４．１　システム構成

TRONEMUを起動すると、はじめに各スレッドの親となるプロセスが生成される。このプロセスからは、μITRONアプリケーション（ユーザスレッド）と簡易シェルスレッドが生成されるという構成になっている。親プロセスは、そこまでの処理が終わると他のスレッドが削除されるまで待ち状態に入る。（図７ TRONEMUシステム構成 参照）

また、TRONEMUは独自に動作時間を取得するため、Linuxのシグナルを利用しており、定期的に入るシグナルから、タイムアウト付きシステムコールからの復帰タイミングを判断している。

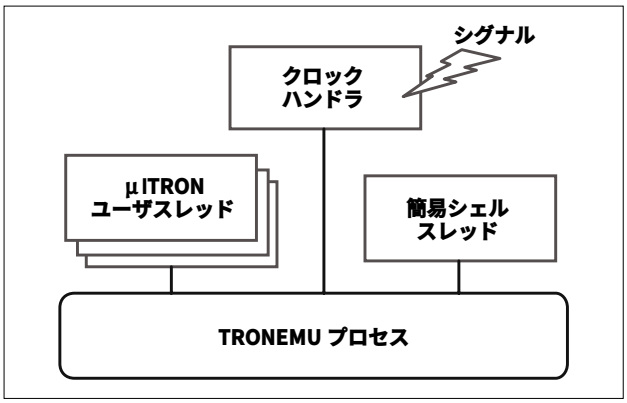


図７　TRONEMUシステム構成

４．２　タスクスケジューラ

μITRONアプリケーションは、TRONEMUプロセスの下で動作するスレッドとして生成される。

生成されたタスクは、TRONEMUのスケジューラ、ディスパッチャにより、FCFS方式に従って実行される。

スケジューラの実装としては、FCFS方式に応じた複数のタスク実行状態管理キューにより行っている。このキューにより、実行可能状態のタスクと待ち状態（休止状態を含む）のタスクを管理し、それらをPOSIXスレッドライブラリの関数を用いて、意図的に停止・実行させることでμITRONと同様のスケジューリングを実現している。（図８　TRONEMUのスケジューリングイメージ）

また、μITRONは7つのタスク状態を持つが、Linuxが持つ5つの状態では、これらの全ての状態をカバーすることができない。そのため、スレッドが独自にμITRONのタスク情報を持つことで機能を実装している。

具体的には、TRONEMUではLinuxの待ち状態だけ

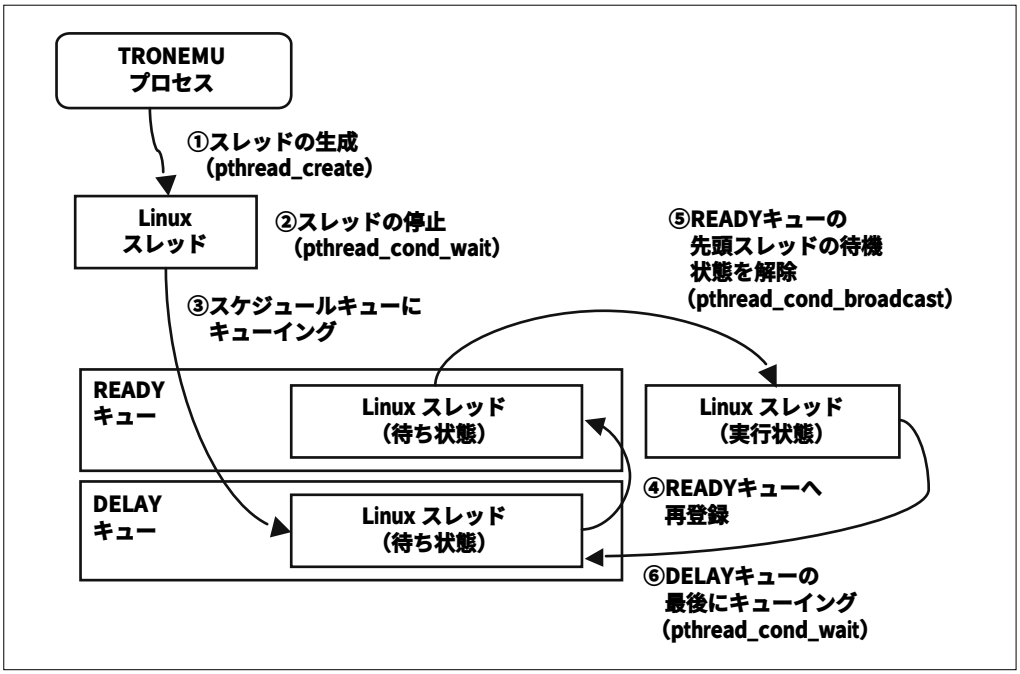


図８　TRONEMUのスケジューリングイメージ

を使用して、μITRONの状態のうち、5つの状態（4つの待ち状態と休止状態）をエミュレートしている。（表６　μITRONとLinuxのタスク状態 参照）

表６　μITRONとLinuxのタスク状態

μITRON	Linux
実行状態	実行状態
実行可能状態	待ち状態
待ち状態	
二重待ち状態	
強制待ち状態	
休止状態	
未登録状態	（タスク削除）

４．３　タスクコントロールブロック

各タスクの状態は、μITRONと同様に、それぞれのタスクがタスクコントロールブロック（TCB）情報を持ち、スケジューリング時にそれらの情報をタスクID番号によって確認することで管理を行う。（図９ タスクコントロールブロック（TCB）参照）

また、一般的なTCB情報以外に、各タスクのスレッド情報など、TRONEMU独自の拡張情報も持ち、各タスクの制御時に使用している。

Linuxの待ち状態だけで、μITRONの5つの状態を判断するための情報も、このコントロールブロックに格納される。

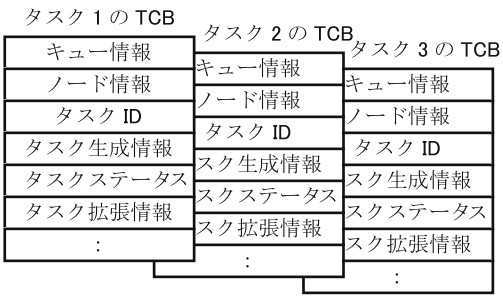


図９　タスクコントロールブロック（TCB）

４．４　各種オブジェクト

TRONEMU上で使用される、イベントフラグ、セマフォ、メールボックス、ミューテックス、メモリプールなどは全てオブジェクトという単位で共通管理を行う実装としている。（図10　各種オブジェクトの実装イメージ参照）

ただし、機能はそれぞれで異なるため、オブジェクトが持つ属性によって、そのオブジェクトが持つ機能を判断し、切り替えている。

また、タイムアウト付きの各種オブジェクトは、TRONEMUがLinuxのシグナルを用いて独自にカウントしている経過時間から、その復帰タイミングを判断し、タスク状態を「待ち状態」から「実行可能状態」へと切り替える実装としている。

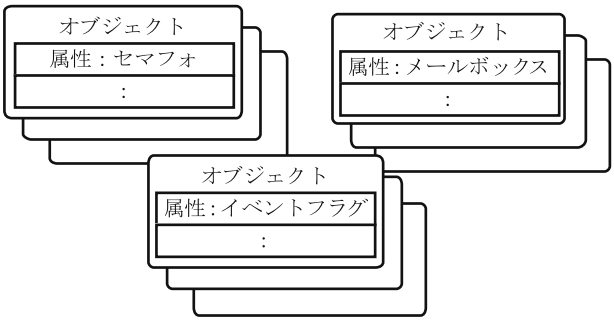


図10　各種オブジェクトの実装イメージ

４．あとがき

本稿では、μITRONからLinuxへの移行支援ソフトウェアであるTRON EMULATORの機能およびその実装について記述した。ここに記述した内容はソフトウェア全体の一部分ではあるが、OS移行作業に要する作業およびTRON EMULATORを使用したイメージは掴んでいただけるものと考える。

TRON EMULATORのような製品は、新規プロジェクトで、最初から採用されるような製品ではなく、どのようなお客様にも紹介できる、といった製品ではない。

しかし、お客様の既存システムに含まれる多くの有効資産を次のプロジェクトへとつなげることで、お客様のプロジェクト開発に貢献できる製品であると確信している。

今後とも、お客様のプロジェクト成功に貢献できるソフトウェアの開発・支援に積極的に取り組んでいく所存である。

参考文献

トロン協会（トロンプロジェクト）ホームページ：
μITRON3.0仕様、μITRON4.0仕様

＼執筆者紹介

天野晋弥　Shinya Amano
（株）日新システムズ
エンベデッド・ソリューション・ビジネスユニット
プロダクト技術部