T-Kernelソースコード公開

豊山 祐一（とよやま ゆういち）
YRPユビキタス・ネットワーキング研究所

T-Kernelのソースコードがついに一般に公開されました。希望される方は誰でも所定の手続きを踏めば、T-Kernelのソースコードを手にすることができます。

T-KernelはT-Engineから生まれた新世代のリアルタイムOSです。T-Kernelのオープン化のプロジェクトについては、すでに本誌にて紹介してきましたので、ここでは実際にオープン化されたT-Kernelを入手し、動作させるまでを説明していきたいと思います。

T-Kernelの入手方法

T-Kernelのソースコードは、T-Engineフォーラムのウェブページより以下の手順で利用申し込みを行い、入手することができます（図1）。

①T-Engineフォーラムのウェブページ（http://www.t-engine.org/japanese.html）から［T-Kernel利用申し込み］を選んでください。ページ上の左側のメニューの中にリンクがあります。

②T-Kernel利用申し込みのページに移動します。ここで、T-Kernel利用についての説明をよく読んだら、ページ下の［利用申込み］のボタンを押してください。

③T-Kernelのライセンスのページに移動します。ここで、T-License「T-Kernelのソースコードのライセンス契約」の全文が表示されますので、これをよくお読みください。T-Kernelのソースコードを申し込むには、このライセンスに同意し、承諾する必要があります。承諾される方は、ページ下の「承諾」ボタンを押してください。

④T-Kernelを利用されるのが、個人か法人かを選択するページに移動します。個人で申し込まれる方は［個人契約］のボタンを押してください。会社など法人として申し込まれる方は［法人契約］のボタンを押してください。ここでは個人契約

図1　T-Kernel公開Webページ

について説明を進めていきますが、法人契約でも大きな違いはありません。

⑤個人契約の入力ページに移動します。ここで必要な事項を入力します。必須の入力事項は、「氏名」「住所」「E-mailアドレス」です。必須ではありませんが問題がなければ、プロフィールや使用目的も入力しましょう。入力された情報の扱いについては、前のページから個人情報の取り扱い（プライバシーポリシー）のページへのリンクがありますので、そこに記載されています。入力が終わったら、ページ下の［次へ］ボタンを押します。

⑥申し込みの確認のページの移動します。表示された内容に間違えがなければ、ページ下の［この内容で申請する］ボタンを押してください。これで利用申し込みの申請は終了です。

以上の申し込みが終わりますと、T-Engineフォーラムで利用者登録が行われ、ユーザIDとパスワードが電子メールで返ってきます。登録されるまで場合によっては1週間ほどかかる場合もあるとされていますが、通常はすぐに登録が行われるようです。申し込みの手続きが少々堅苦しく感じられたり、厳しく思われる方もいると思いますが、T-Kernelは実際の組込み機器などの製品にも使用されることを前提にライセンスを考えていますので、このような形になっています。T-Kernelのライセンスについては別の記事で説明していますので、ここでは詳しく書きませんが、GPLとは異なった独自のライセンスであることには注意してください。大きな違いのひとつは、T-Kernelのソースコードの再配布が禁止されていることです。T-Kernelを含むT-Engineのプロジェクトは、ミドルウェアの流通プラットフォームを作ることをひとつの目標としています。この実現のため、OSはソースコードのレベルで一本化し、改変物の流通を原則として許していません。なお、禁止されているのはソースコードの再配布であり、T-Kernelを改造して使用したり、改変物を製品に組み込んだりすることは許されています。逆に、製品にT-Kernelを組み込んで使用しても、ソースコードを公開しなくてはいけないといった制約はありません。このあたりは、組込み機器などでの使用に適したライセンスになっています。

T-Kernelの利用者登録が済み、ユーザIDとパスワードが手に入ったら、いよいよT-Kernelのソースコードのダウンロードです。ダウンロードページのURLは、ユーザIDといっしょに送られてきているはずですので、早速ダウンロードしてみましょう。執筆時点で公開されているファイルは、表1のとおりです。

表1のtkernel.1.00.00.tar.gzがT-Kernelのソースコードそのものです。tkernel.txtはT-Kernelソースコードの説明書で、ソースファイルの構成や、カーネルの構築方法、カーネル起動処理の説明などが記載されています。T-Kernelのソースコードを読む際には一読しておくとよいでしょう。その他のファイルは、T-Kernelの実装仕様書です。現在公開されているT-Kernelは、表2に記した5機種の標準T-Engineと2機種のμT-Engineに対応しており、それぞれのT-Engineについて実装仕様書があります。実装仕様書には、ハードウェアに依存する仕様や、T-Kernelが実際にどのように実装されているかが説明されています。T-Kernel上でシステム寄りのプログラムを作成する場合や、T-Kernelのしくみ自体を知りたい場合などには役に立つかと思います。

表1　公開されているファイル

tkernel.1.00.00.tar.gz	T-Kernel ソースコード バージョン 1.00.00
tkernel.txt	T-Kernel ソースコード説明書 Rev.1.00.00
imp_sh7727.txt	標準 T-Engine/SH7727 実装仕様書
imp_sh7751r.txt	標準 T-Engine/SH7751R 実装仕様書
imp_vr5500.txt	標準 T-Engine/VR5500 実装仕様書
imp_arm720.txt	標準 T-Engine/ARM720-S1C 実装仕様書
imp_arm920.txt	標準 T-Engine/ARM920-MX1 実装仕様書
imp_m32104.txt	μT-Engine/M32104 実装仕様書
imp_vr4131.txt	μT-Engine/VR4131 実装仕様書

表2　対応T-Engine一覧

標準 T-Engine/SH7727	μT-Engine/M32104
標準 T-Engine/SH7751R	μT-Engine/VR4131
標準 T-Engine/VR5500	
標準 T-Engine/ARM920-MX1	
標準 T-Engine/ARM720-S1C	

表3　ソースコードのディレクトリ構成

kernel	T-Kernel 本体
lib	ライブラリ
include	各種定義ファイル（ヘッダファイル）
config	rominfo、SYSCONF、DEVCONF など設定ファイル
etc	make ルール、スクリプトなど

T-Kernelの構築と実行

T-Kernelのソースコードが手に入ったら、カーネルを構築し実際に動かしてみましょう。公開されているT-Kernelをそのまま動かすには、表2に記した標準T-Engineまたはμ T-Engineのいずれかの開発キットが必要です。開発キットには、T-Engineのハードウェアとともに、Linux上で動作するGNU開発環境が含まれています。T-Kernelのソースコードは、この開発キットの環境で構築することを前提に、makefileやスクリプトファイルが付属されています。これ以降の具体例は、Shellにはbashを使用し、標準T-Engine/SH7727のカーネルの構築手順を説明していきます。他のT-Engineでも一部パス名が変わる以外は基本的に同じ手順です。また、bash以外のShellでは一部コマンドを置き換える必要があります。

まず準備として、T-Engine開発キットのGNU開発環境を、PCにインストールします。ちゃんとインストールできたかどうか、開発キットに付属のサンプルプログラムをいくつかコンパイルしてT-Engineで動かしてみてください（開発環境のインストールやサンプルプログラムの動かし方は開発キットのマニュアルをご覧ください）。

開発環境の動作が確認できたら、T-Kernelのソースコードファイル「tkernel.1.00.00.tar.gz」をどこか適当なところに展開します。ここでは、ホームディレクトリの下に展開してみましょう。Shell上で以下のコマンドを実行します。

```
% cd
% tar xvfz tkernel.1.00.00.tar.gz
```

tkernel_sourceというディレクトリが作成されたと思います。このディレクトリの下には、表3に示す5つのディレクトリがあります。kernelディレクトリにT-Kernel本体のソースコードが入っています。libディレクトリは、T-Kernelが提供するライブラリのソースコードです。この中にシステムコールのインタフェースライブラリも含まれます。includeディレクトリにはC言語のヘッダファイルが入っています。このファイルは、kernelやlib中のファイルやユーザプログラムから使用されます。configディレクトリには、rominfo、SYSCONF、DEVCONFなどT-Engineのシステムの設定ファイルが入っています。これらの設定ファイルの内容は、各T-Engineの実装仕様書に記載されています。etcディレクトリにはソースコードはありません。makeルールやスクリプトファイルなど、T-Kernelを構築する上で使用するファイルが入っています。

ソースコードが展開できたら、環境変数BDにソースコードを展開したディレクトリのパス名を設定します。この環境変数BDは開発環境のベースディレクトリを示しています。

```
% export BD=~/tkernel_source
```

まず各ライブラリファイルを作ります。T-Kernel本体もこのライブラリを一部使用しますので、必ずライブラリから作る必要があります。ライブラリは、ターゲットシステムのディレクトリに移動し、makeすることにより作成することができます。ターゲットシステムのディレクトリは、libディレクトリ下のbuildディレクトリ下にあります。標準T-Engine/SH7727の場合は以下のようにコマンドを実行します（他のT-Engineではターゲットシステムのディレクトリ名が異なります）。

```
% cd ~/tkernel_source/lib/build/
std_sh7727
% make
```

続いて、T-Kernel本体を構築します。カーネルのターゲットシステムのディレクトリに移動し、makeすることにより作成します。ターゲットシステムのディレクトリは、kernel/sysmain/build/のパスの下にあります。標準T-Engine/SH7727の場合は以下のようにコマンドを実行します（ライブラリと同様に、他のT-Engineではターゲットシステムのディレクトリ名が異なります）。

```
% cd ~/tkernel_source/kernel/
sysmain/build/std_sh7727
% make
```

カーネルの構築に成功すると、カレントディレクトリ上にkernel-rom.motというファイルが生成されます。これがT-Kernelのオブジェクトファイルです。

最後にRomInfoのオブジェクトを作成します。config/build/のパスの下のターゲットシステムのディレクトリに移動します。標準T-Engine/SH7727の場合は以下のようにコマンドを実行します（繰り返しますが、他のT-Engineではターゲットシステムのディレクトリ名が異なります）。

```
% cd ~/tkernel_source/config/
build/std_sh7727
% make
```

カレントディレクトリ下に、rominfo.motという名前でRomInfoのオブジェクトファイルが生成されます。以上でカーネルの構築は終了です。

いよいよ構築したT-Kernelのオブジェクトを T-Engineで動かしてみましょう。これにはまず、構築したT-Kernelのオブジェクトファイルkernel-rom.motをT-Engineのフラッシュ ROMに書き込みます。フラッシュROMの書き込みは、通常T-MonitorのFlashLoad（FLLO）コマンドにより行うことができます。フラッシュROMへの書き込み方法は、T-Engineの種類やバージョンによっても違う場合がありますので、T-Engineに付属のマニュアルをお読みください。フラッシュROMへの書き込みを行うと、それまでフラッシュROMにあった開発キットのソフトウェアは上書きされてしまいますのでご注意ください。元のフラッシュROMのソフトウェアに戻すには、開発キットに付属しているROMイメージのオブジェクトファイルを書き戻します。この方法も、各T-Engineに付属のマニュアルに記載され

ていますので、そちらをお読みください。

kernel-rom.motの書き込みが終了したら、続いてRomInfoのオブジェクトファイルrominfo.motを同様の手順でフラッシュROMへ書き込みます。このRomInfoにはシステムの起動に必要な情報が設定されていますので、システムのソフトウェアを変更した場合はいっしょに変更が必要です。

2つのファイルのフラッシュROMへの書き込みが終わったら、T-Engineの電源を入れ直しますと、新たに書き込んだT-Kernelが起動します。この際、T-EngineのシリアルI/Fにパソコンを接続し、ターミナルソフトを動かしておくと、T-Kernelからの出力が表示されます。通信設定はT-Monitorとの通信の設定と同じです。

構築したT-Kernelが正常に起動すれば、以下のメッセージが表示されるはずです。

```
T-Kernel Version 1.00.00

Push any key to shutdown the
T-Kernel.
```

ここで、パソコンのターミナルソフトから何らかのキー入力を送ると、T-Kernelは終了します。T-Engineのハードウェアによりますが、T-Monitorから電源を切る機能のあるT-Engineならば、電源も切られます。

以上でT-Kernelの構築から起動までは終わりです。

ユーザプログラムの組み込み

さて、うまくT-Kernelは起動したでしょうか。起動はしたけど、起動メッセージが出る以外は何もできない？ これはそのとおりで、公開されているソースコードは、T-Kernel本体のみですから、その上で動作するアプリケーションプログラムは存在しません。本当にT-Kernelだけが動いている状態です。T-Engineに何かさせるには、T-Kernel上で目的のユーザプログラムを動かさなくてはなりません。

まず、T-Kernelの起動からユーザプログラムが実行されるまでの流れを簡単にを説明します。

T-Engineに電源が投入されてると最初にT-Monitorが起動します。T-Monitorは起動時に必要なハードウェアの初期化を行った後、T-Kernelの起動プログラムを呼び出します。今回構築したT-Kernelは、フラッシュROM上に置いて直に起動していますので、特別な起動プログラムは不要です。よって、T-MonitorからT-Kernelの開始アドレスが直接呼ばれます。なお、起動プログラムを作成することにより、T-Kernelをディスクから起動したりすることも可能です（システムの起動処理の詳細なシーケンスは、T-Kernelとともに公開しているソースコード説明書に記載されていますのでご覧ください）。

さて、T-MonitorからT-Kernelが開始されると、T-Kernelはまず内部の初期化処理を行います。この初期化処理が終わると、初期タスク（init_task）と呼ばれる特別なタスクを生成し実行します。初期タスクは、初めにT-Kernel起動の最後の処理である、タスクコンテキストで実行する各種の初期化処理を行います。これらの処理が終わるとT-Kernelの初期化処理はすべて終了です。初期タスクはいよいよユーザプログラムを呼び出します。

初期タスクから呼び出されるユーザプログラムは、初期タスクメイン処理usermainです。このusermainプログラムが、ユーザプログラム全体の開始処理を行います。

T-Kernelのソースコードの以下のパスのファイルを見てください。

```
/tkernel_source/kernel/
usermain/usermain.c
```

このファイルには、usermain()という関数が1つ定義されています。これがusermainプログラムです。リスト1にこの関数のリストを示します。見たとおり、実行行は3行だけのとても短いプログラムです。1行目のtm_putstring()という関数は、T-Monitorのサービス関数で、コンソールへの文字列出力を行います。T-Kernelを起動した際に、パソコンのターミナルソフト上に表示された「Push any key to shutdown the T-Kernel.」という文字はここで出力していたわけです。2行目のtm_getchar()という関数も、T-Monitorのサービス関数で、コンソールからの1文字入力を行います。引数に-1が指定されていると入力があるまで待ち続けます。入力があると、3行目が実行されusermain()関数は終了します。戻り値の0は正常にusermain()が終了し、システムを終了することを示します。

以上の処理は、前に述べたT-Kernelを起動したときの動作そのものだということがわかると思います。公開されているT-Kernelにはユーザプログラムがありませんので、とりあえずこのような簡単なプログラムが記述されています。自分が動かしたいユーザプログラムは、このusermain()関数の内容を書き換えることによって実行することができます。

ユーザプログラムをT-Kernelに組み込み、動作させるのに、一番単純な方法はこ

リスト1　usermain()関数のソースコード

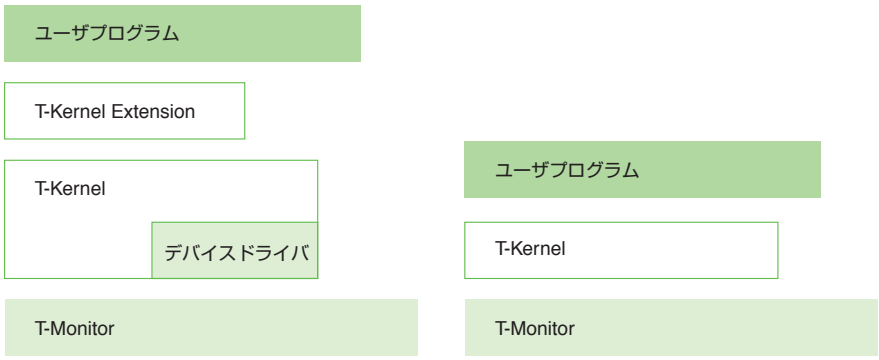
```
EXPORT  INT usermain( void )
{
    tm_putstring("Push any key to shutdown the T-Kernel.\n");
    tm_getchar(-1);

    return 0;
}
```

のusermain()関数の中にユーザプログラム自体を書いてしまうことです。試しに、リスト1のtm_putstring()関数の引数の文字列を変えたり、別のtm_putstring()関数を付け加えたりして、再びT-Kernelを構築し、実行してみてください。変更が反映されることが確認できます。

実際には、ユーザプログラムはusermain()関数の中に記述するのではなく、usermain()関数からユーザプログラムを構成するタスクを生成、起動するというのが一般的な使い方です。ある程度の規模のユーザプログラムは通常、複数のタスクから構成されますので、これらのタスク自身やタスクが使用する他のカーネルオブジェクトを生成するのが、usermain()関数の役割となります。その際、usermain()関数が終了するとT-Kernelの終了処理が始まり、システム全体が終了してしまうことに注意してください。システムの終了までusermain()関数を終わらせないしくみが必要です。

ところで、ここまで説明した方法ですと、ユーザプログラムを変更するたびにT-Kernelを含めたシステム全体から構築し直さなくてはならず、大変だと思われた方もいると思います。ユーザプログラムの開発を効率的に行うには、開発を支援する各種のツールが必要です。今回説明したシステムはT-Kernelだけで、これら開発ツールは含まれていませんので、このようになっています。通常は、T-Kernelにユーザプログラムを直に組み込むのは、そのユーザプログラムのデバッグ等の開発作業が終わって最終段階で行うものです。開発キットのもともとのシステムには、T-Kernelに加えてIMSやCLIなどのツールが組み込まれており、ユーザプログラムを簡単にロードする手段が提供されています。また、プログラムのデバッグにはgdbというデバッガが使用できます。ユーザプログラムの開発には、開発キットを使用したほうがよいでしょう。

開発キットの上で開発したプログラムを、公開されているT-Kernelの上で動かすにはいくつかの注意点があります。



（a）開発キットのシステム　（b）本稿で構築したシステム

図2　システム構成の概略

開発キットのシステムには、図2（a）に示すようにT-Kernel以外にも多くのプログラムが含まれています。逆に今回構築したシステムは図2（b）に示すようにシンプルなものです（T-MonitorはROM上のものをそのまま使用しています）。まずT-Engine上の各種ハードウェアを制御するには、それぞれのハードウェアに応じたデバイスドライバが必要です。デバイスドライバはハードウェアに依存し、つまりハードウェアごとに作成する必要があるものですので、T-Kernel本体には含まれません。T-Kernel Extensionは、ひと言で言えばT-Kernelの機能を拡張し、より高機能なシステムを作るためのソフトウェア群です。開発キットでは、PMC T-Kernel ExtensionというBTRON仕様に準拠した強力なExtensionが含まれています。このExtensionにより、ユーザプログラムは、ファイルシステムやプロセス管理、仮想メモリ管理といった機能を使うことができます。T-Kernelのみのシステムでは、当然これらの機能は使うことができません。

また、現段階での開発キットに含まれるT-Kernelは、一般公開される前のバージョンのT-Kernelです。よって、若干異なる部分もあるかと思われます。この点についてはおいおい統一されていくことと思われます。

最後に

公開されたT-Kernelのソースコードは、単なるリファレンスや実験レベルではく、実際の製品における使用を前提としたリアルタイムOSのプログラムです。また、CPU間の移植性も高く、すでに7種のCPUに対応したソースコードとなっています。このような本格的なリアルタイムOSのソースコードが無償で公開されたことは、非常に意義のあることだと思います。

個人のレベルでも、リアルタイムOSを勉強したい人には格好の教材になるでしょう。また、自分で作ったハードウェアにリアルタイムOSを移植したい場合にも格好のものでしょう。うまくT-Kernelが移植できれば、今後続々と登場してくると思われるT-Engine用のソフトウェアも利用できます。

本稿を読まれてT-Kernelに興味を持たれた方は、ぜひチャレンジしてみてください。