IoT eラーニング

IoTで使われるOS

（具体例、特徴）

国立大学法人　琉球大学

目次

IoT組込みOS

● リアルタイムOS（RTOS）の登場

IoT組込みシステムの開発には当初、OSを使わないシステムが普通であった。
※OSを使わないのでベアメタル（Non OS）と呼ばれている。

IoT組込みシステムの複雑化に合わせて、

- リモコン等の入力情報の受付／表示制御／計算処理／接続機器の制御などの多数の処理を行う。
- 用途によっては、各種シリアルやイーサネットを使用した通信を行う。

といった機能が求められるようになった。

このような背景から、機能別に処理を分割してソフトウェアの開発が進められるようになると、組込みOSが導入されるようになった。

このOSは、

- 割り込みサービスルーチン（ISR：Interrupt Service Routine）により一定の時間内で特定のタスクを終了させる。
- タスクの実行順序については決定論的なプロセスが採用され実時間処理の管理が行える。

ことが特徴で、リアルタイムOS（以下RTOS）と呼ばれている。

RTOSのタスク管理

RTOSでは、次のようにタスク管理を行う。

- タスクをその優先度に基づいてスケジューリング・管理する。
- 実行が可能な状態にあるタスクのうち優先度がもっとも高いものを実行する。
- タスクが実行可能状態になったとき、実行中のタスクよりも優先度が高い場合、即座にタスクの切り替えを行う。

そのため、単一のハードウェア資源等のリソースに対して、複数のタスクのアクセスが重ならないかなど、開発者の設計が重要である。

開発者は、

- 各タスクの実行可能となるタイミング
- 資源アクセスに対する要求

などを考慮し、消費される時間が最小となるよう、スケジューラ等の設計を行う必要がある。

RTOSのタスク間通信

RTOSは複数のタスクを処理するため、データやハードウェア資源を共有するという課題に対処する必要がある。

複数のタスクが、同じタイミングで同じデータやハードウェア資源にアクセスする必要が発生した際に、一般的には次の手法により解決を行う。

- 割り込みマスク（イベントの制御）
- セマフォ（ポーリングの制御）
- 共有メモリなどでメッセージ渡し（メッセージの制御）

開発者はマイコンの性能を考慮しながら、リソースの共有が図られデッドロックが発生しないよう解決方法を考慮し設計する必要がある。

RTOSのメモリ管理

RTOSでは、個別のタスクがメモリを必要とした際に、固定サイズのメモリブロックを割り当てる方式が採用されている。

情報系OS等では、不定長のフリーなメモリブロックの中から必要なサイズを探索・確保する。
この場合、メモリを確保するメモリアロケーション処理の処理時間は想定しずらくなり、また空き領域の断片化が発生することによるメモリ確保の失敗でプログラムが停止する可能性がある。

メモリ空き領域の断片化は、情報系OS等の場合はリブートがある程度の頻度で行われるため許容されているが、IoT組込みシステムは長期間リブートしないため許容されない。

RTOSでは、固定サイズのメモリアロケーションを行うことにより断片化の回避を行い、さらに一定時間内でメモリーアロケーションが行われることも行う。

RTOSと情報系OS

- **情報系OSとの違い**

RTOSは、タスク（処理のまとまり）を優先度やハードウェアなどのリソース共有を制御する設計を行うことでリアルタイム性を確保する。

開発者が厳密にスケジュール設計を行うことにより、リアルタイム性が確保され、発生するイベントや処理時間があらかじめ予測でき、高速に応答できるシステムを開発できる。

一方で情報系OSは、一定時間ごとに実行可能状態にあるプロセスを順番に切り替える方式を採用していた。

この方式では、即時実行したい処理の場合でも、他の処理の状態によっては待たされる可能性があり、優先度を考慮することができない。
最近の情報系OSでは優先度を考慮したスケジューリングを採用するようになってきたが、処理の状態により自動的に優先度が変更されることがあるため、処理時間を予測することが困難であることは変わっていない。

RTOSと情報系OS

- **組込みLinux**

先に述べた理由から、機械制御などマイクロ秒の応答性を求められるシステムには情報系OSでは技術的な限界がある。

しかし、

- IoT組込みシステムへ求められる機能の高度化
- 提供する機能の処理に集中できること
- ライブラリ等の資産の活用しやすさ

などから、情報系OSである組込みLinuxの採用も進んでいる。

組込みLinuxは、IoT組込みシステム用にLinuxをベースに開発されたため厳密にはLinuxとは違う。

また、タスクスケジューリングについては、OS（組込みLinux）に任せており、この点はRTOSとは異なる。

RTOSのメリット・デメリット

ベアメタル（Non OS）のシステムとRTOSを使用したシステムを比較した際の代表的なメリットは次のようになる。

- 複数の機能があっても、処理時間の分配が容易
- タスクに優先順を容易に持たせられる
- リアルタイム性が確保される

デメリットは次のようになる。

- RTOSの処理時間・メモリ消費が増える
- スケジューリング設計などのRTOSを使用するための習得が必要

デメリットはあっても、

- リアルタイム性が必要
- 要求される機能の増加

などから、IoT組込みシステムでは、RTOSを採用するものが多い。

IoT組込みOSの事例

- **RTOSの事例**

  マイコンベンダ企業や非営利組織などから統合開発環境が提供されており、ターゲットとなるマイコンボードや要求される機能により使用するRTOSが選択されている。

  次にいくつかのRTOSを説明する。

IoT組込みOSの事例

- **組込みLinux①**

Linuxはフリーの OSとして公開されており、様々なユーザーによる利用と改善が行われ、またライブラリやドライバ等の資産の蓄積も行われている。

初期のIoT組込みシステムでもLinuxの利用が図られていたが、マイコンの性能が低くMMUによるシステム制御が困難であった。

しかし、
- マイコンの性能の向上と低価格化
- IoT組込みシステムに求められる機能の高度化

が進むと、LinuxをIoT組込みシステムへ利用する動きが増えてきた。

Linuxは無償でありながら
- ネットワークやファイル管理などが高機能で搭載
- ミドルウェアが豊富
- ソースコードを自由に書き換えられる

ことから、IoT組込みシステム向けに最適化が図られるようになった。

IoT組込みOSの事例

- **組込みLinux②**

IoT組込みシステム向けに最適化Linuxは、組込みLinuxと呼ばれている。

現在は、
- 携帯電話製品のプラットフォーム
- テレビやカメラなどの家電
- ネットワーク機器
- 医療機器

などの様々なIoT組込みシステムに利用が進んでいる。

PCで動作するLinuxはBIOSによりハードウェアの違いを補っているが、組込みLinuxではBIOSがないハードウェアが対象となり、これを考慮するコストが必要である。

一方で、開発環境や蓄積されたミドルウェアの利用など、開発時の大きな強みとなっている。

IoT組込みOSの事例

- **Nucleus RTOS**

Nucleus RTOSは米国のメンター・グラフィックスが各CPUプラットフォーム向けに開発したRTOSであり、30億台以上のIoT組込みシステムに搭載されてきた。

Nucleus RTOSは、
- 家電製品
- ケーブルTV等の映像受信機器
- 携帯電話

などのIoT組込みシステムに利用されている。

コードとデータを合わせて13KBで動作可能で、このメモリ使用量の少なさが重要な特徴の１つである。

世界中の1000以上の企業に提供されており、様々な製品に利用されている。

IoT組込みOSの事例

- **ITRON（μITRON）**

ITRONはTRONプロジェクトが策定したRTOSである。

日本では長年シェアのトップを占めており、業界標準のRTOSとして採用されている。
海外であまり知名度はないが、日本製家電に搭載され世界に輸出されていることからOSのシェアは高い。

μITRONはITRONのサブセットとしてまとめられた仕様であるが、全ての機能が引き継がれる形になり、μITRON3以降ではITRONはμITRONのことを指している。

スマートフォンが登場する前の携帯電話（ガラケー）の時代にはシェアをほぼ独占していたが、スマートフォンの登場によりITRONのシェアは次第に少なくなっていった。

しかし、映像録画サーバーや自動車などの機器の場合、構造の複雑さから複数のOSを組み合わせて構成されている場合があり、メインとなるOSではLinuxなどを採用しても、制御用機器にはITRONが搭載されていることがある。

例えば、Nintendo Switchの本体にはメインOSとしてFreeBSDが使われ、コントローラとの無線通信制御用OSにμITRON4.0が採用された。

IoT組込みOSの事例

- **T-Kernel**

  T-KernelはμITRON3.0仕様の発展版としてT-Engineフォーラムによって策定された。

  ITRONの思想の１つにある「弱い標準化」による問題点から、T-Kernelは「強い標準化」の思想でITRONの標準化が図られたものである。

  互換性や移植性の高いミドルウェアの流通を推進し、高度なシステムの開発の効率向上を目的としている。

  2013年9月に打ち上げられた日本のイプシロンロケットとそれに搭載された観測衛星ひさきにT-Kernelが使用され、また、2014年12月に打ち上げられたH-IIAロケットで打ち上げられた「はやぶさ２」の制御システムにも使用されている。

IoT組込みOSの事例

- **OS-9**

OS-9は、モトローラの8ビットマイクロプロセッサ（6809）用に開発されたRTOSである。

開発元のマイクロウェアはモトローラの依頼により開発していたプログラミング言語の実行環境としてこのOSを開発した。
その後、 680X0／x86／PowerPC／SH／ARM等様々なCPUに移植された。

OS-9の特徴の１つにモジュール構造がある。統一された構造（モジュール）の組み合わせで構成されており、必要な機能だけを選び使用することが出来る。

OS-9が採用された機器は以下の通り。

- 電子楽器のシンセサイザー
- TCP/IP用ルーター
- レーザープリンター用エンジン
- 銀行用ATM
- カーナビゲーションシステム

誕生から40年近くなるが現在も開発が進められている。

IoT組込みOSの事例

- **Windows 10 IoT**

Windows 10 IoTは、マイクロソフトが開発したIoT組込みシステム向けのRTOSである。

従来、マイクロソフトではIoT組込みシステム向けに「Windows Embedded」を提供していたが、Windows 10のリリースと合わせて、IoT組込みシステム向けにWindows 10カーネルが動作するOSとして「Windows 10 IoT」をリリースした。

「Windows 10」と「Windows 10 IoT」では、ユニバーサルWindowsプラットフォーム（UWP）によりアプリケーションの効率的な開発が行える。

また、マイクロソフトが提供するクラウドサービスにAzureがあるが、「Windows 10 IoT」に対応する概念としてAzure IoTというクラウド側の考えを提唱している。

マイクロソフトは、IoT組込みシステムからクラウドまでWindows 10カーネルよるプラットフォームの共通化やVisual Studioによる開発環境の共通化を図り、高品質・高安全性のサービスが提供できるとしている。

IoT組込みOSの事例

- **FreeRTOS**

FreeRTOSは英国のReal Time Engineersが開発したIoT組込みシステム用のフリーのRTOSである。

FreeRTOSは、修正条項付きGPLv2で配布されていたが、Amazonに買収され、バージョン10からMITライセンスに変更され配布されるようになった。

フリーで商用利用も可能であったため、多くのプラットフォームで利用されており、新しいライセンス形態もMITライセンスであることから、これからも様々なプラットフォームで利用されると思われる。

代表的なプラットフォームは次のようになる。

- ARM(ARM7／Cortex-M3等)
- Atmel AVR
- AVR32
- PIC
- ルネサスH8/S
- x86