# 第１章
# RTLinux への招待

RTLinux は，マルチメディアシステム，テレスコープ，FA システム，ロボティクス，コミュニケーションデバイスなど多くのシステムで利用されています．

しかし，そんな状況の中にあって中級から上級者レベルのプログラマ用の参考文献が基本的に不足しています．

本書はそのギャップを埋めるために RTLinux を使ったハードリアルタイムアプリケーションの構築について解説します．RTLinux キットに添付されているドキュメントファイルと本書を参考にしてください．本書にはコンピュータ専門家の間ではすでによく知られていることも含まれていますが，リアルタイムプログラミングについて総合的に理解できるよう，その一般的な背景についても説明します．

すでに C プログラミングやリアルタイムコードおよびカーネル作業に精通している読者には，API に関する解説などを参照することで RTLinux 実用化への近道となります．一方，まだ学習中だという人は，その背景や重要点を学ぶ事によって RTLinux がより身近に感じられるようになるでしょう．

ただし，本書では C 言語やスレッドプログラミングまたハードウェアの内部処理について知識があることを前提としています．

まずは RTLinux の基本概念について詳しく説明します．次に API の概要，そしてリアルタイムスレッドと非リアルタイムプロセス間におけるデータ交換機能モデルを紹介し，これらのメカニズムの適用方法を解説していきます．

## 1.1　きまりごと

ソースパスは特記されていない限り RTLinux ディレクトリのトップです．

たとえばシステムの`/tmp` にある `rtlinux.tar.gz` を開封して得られる`/tmp/rtlinux/examples/measurements/rt_process.c` というファイルは，`examples/measurements/rt_process.c` と書かれています．

本書ではリアルタイムカーネルのアイドルスレッドとして機能する汎用的 OS が Linux である場合を基本としていますが，このコンセプトは必ずしも Linux だけに限定されているものではありません．RTLinux は BSD もサポートしています．

## 1.2　RTLinux の概要

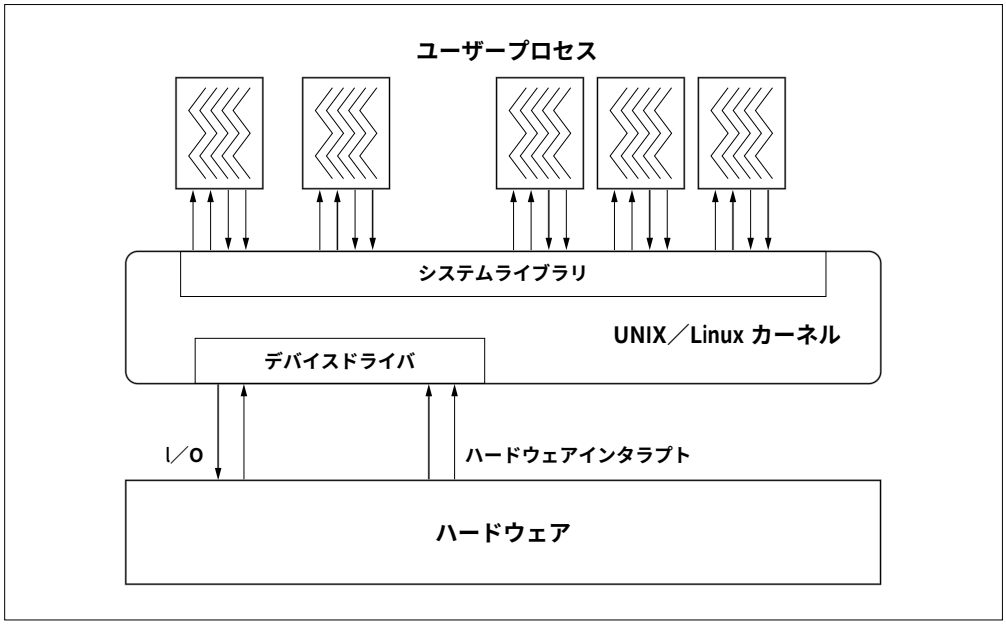


**図 1-1　Linux カーネルの基本構造**

**図 1.1** はハードリアルタイムサポートなしの基本 Linux カーネルを表しています．これを見ると Linux カーネルがユーザーレベルのタスクをハードウェアから切り離していることがわかるでしょう．一度，そのタスクが CPU によって割り当てられた時間枠を上回ってしまうと，カーネルはどんなユーザーレベルのタスクも一時停止させる機能をもっています．たとえば，ユーザータスクがロボットアームをコントロールすることを仮定してみてください．標準 Linux カーネルは潜在的にタスクをプリエンプトし，CPU へあまりクリティカルでないものを渡します（たとえば Netscape の起動）．したがって，アームは厳格なタイミングの要求に反応しないでしょう．このように全てのタスクに対し，「公平」とすることにおいては，カーネルは時間通りに起こる厳格なイベントを妨げる可能性があります．

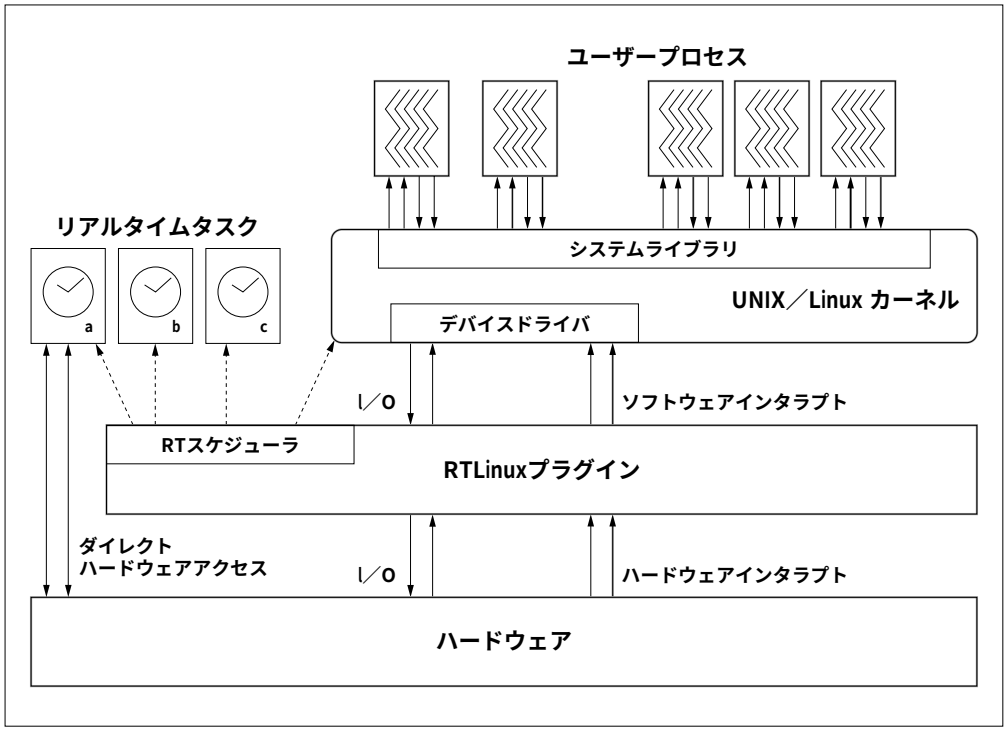


**図 1-2 RTLinux カーネルの構造**

**図 1.2** はハードリアルタイムをサポートする RTLinux の構造を表しています．文字通り「バーチャルマシン」と呼ばれるアブストラクションの付加レイアは標準 Linux とコンピュータハードウェアの間に付加されています．

標準 Linux から，この新しいレイアは実際のハードウェアのように見えます．この新しいレイアはそれ自体が所有するプライオリティスケジューラを取り入れます．このスケジューラは標準 Linux カーネルに対して最も低いプライオリティを割り当てます．

RTLinux によるアブストラクションレイアは全てのハードウェア割り込みをインターセプトすることで動作します．RTLinux カーネルがアイドルで，なおかつ標準 Linux カーネルが動作するとき，リアルタイム動作に関係のないハードウェアインタラプトが行われた場合，ソフトウェアインタラプトとして Linux カーネルに渡されます．

そうでないとすれば，適切なリアルタイムインタラプトサービスルーチン（ISR）が動きます．RTLinux のカーネルは，それ自体はノンプリエンプティブルです．RTLinux カーネル内での予測不能な遅延は RTLinux の小さいサイズおよびオペレーションの制限により取り除かれています．リアルタイムタスクは二つの特別な特性を持っています．それは“ 優先権を持っている ”ということ（ハードウェアへ直接アクセスするということ），そしてバーチャルメモリを使わないということです．リアルタイムタスクはメモリへダイナミ

ックにロードできる特別な Linux モジュールとして書かれています．それは Linux システムコールの実行を予測していません．リアルタイムタスクの初期化コードはリアルタイムタスク構造を初期化し，デッドラインや周期，そして解除時間制限を知らせます．非周期タスクは割り込みの利用を通じてサポートされます．