16.2. SCHED_OTHER

SCHED_OTHER が指定されたプロセス／スレッドは、それぞれ公平に CPU の実行が割り当てられるようにスケジューリングされます。

SCHED_OTHER の特徴として、以下のものが挙げられます。

- UNIX の伝統的なタイムシェアリングに基づくスケジューリング
- Linux のデフォルトのポリシー
- 静的優先度は 100 ～ 139
- 動的優先度を持つ

SCHED_OTHER プロセスのスケジューリングが行われるタイミングは以下の通りです。

- システムコール終了時
- 割り込み処理終了時
- アイドル時

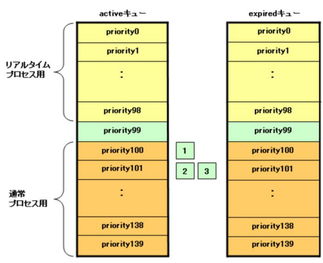
割り込み処理終了時にシステムコール実行中であった場合、Linux 2.4 以前の一般カーネルではシステムコール終了時までスケジューリングされませんでした。
Linux 2.6 からは、カーネルのプリエンプタブル機能の追加により、システムコール実行中であってもスケジューラを起動することが可能となりました。

通常プロセスの動作イメージ

通常プロセスの動作イメージを示します。

1. active キューに繋がっているプロセスを実行

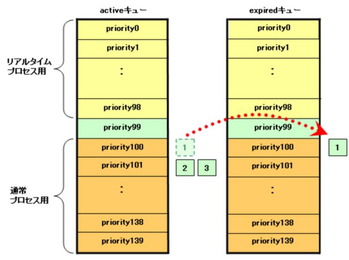
以下の図はプロセス1 ～ 3 が RUN キューに繋がっている様子です。　(図中の 1 ～ 3 の通常プロセスのみが動作しているとします)



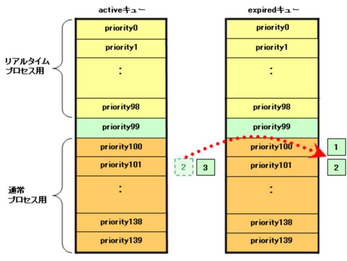
※ 実行可能なプロセスが繋がっているキューを active キュー、実行時間を使い切ったプロセスが繋がっているキューを expired キューと呼びます。

プロセス1 ～ 3 は次のようにスケジューリングされます。

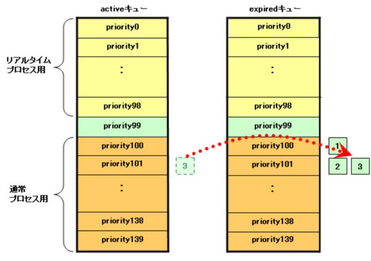
1. 優先度（静的優先度＋動的優先度）が最も高いプロセス1 が実行される
2. プロセス1 がタイムスライスを使い果たした後、expired キューに繋がれ、実行可能状態になる
   この時、プロセス1 の動的優先度が再計算される



3. プロセス2 が実行される
4. プロセス2 がタイムスライスを使い果たした後、expired キューに繋がれ、実行可能状態になる
   この時、プロセス2 の動的優先度が再計算される

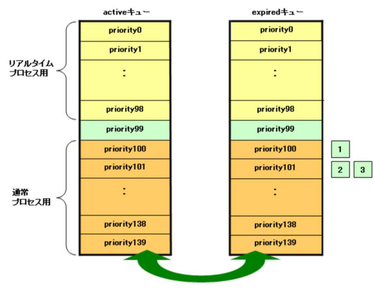


5. プロセス3 が実行される
6. プロセス3 がタイムスライスを使い果たした後、expired キューに繋がれ、実行可能状態になる
   この時、プロセス3 の動的優先度が再計算される



2. active キューと expired キューを入れ替え

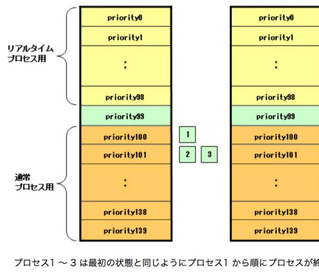
以下の図はプロセス1 ～ 3 がタイムスライスを使い果たした後の様子です。



active キューに実行可能状態のプロセスがなくなると、active キューと expired キューを入れ替えます。

3. active キューに繋がっているプロセスを実行

以下は active キューと expired キューを入れ替えた後の様子です。



プロセス1 ～ 3 は最初の状態と同じようにプロセス1 から順にプロセスが終了するまで繰り返し実行を行っていきます。