Operating Systems

第7回 小テストとプログラミング実習

ラウンドロビンスケジューリング

- Round-Robinスケジューリング
  - タイマー割り込みを利用し，全プロセスを一定時間のタイムスライス（クォンタムタイム，数十～数百ミリ秒）で切り替えて実行していく
  - 「持ち回り」のことを英語ではRound-robinという
- プリエンプション
  - あり ⇒ タイムスライスが経過したプロセスは横取りされ，次に譲る
- 特徴
  - 平等・均一で，常に状態表示が必要な人間相手のシステムに向く
  - 問題点？ ⇒ 時間管理や切り替え自体の処理が多い
- 優先度つきラウンドロビン
  - UNIX，Linux，Windows等ではRRに優先度を組み合わせている

タイマー割り込み

- ハードウェアタイマー
  - コンピュータに内蔵されている高精度な時計
  - 指定した時間後または指定した時間間隔ごとに，割り込みを発生
- タイマー割り込み
  - ハードウェアタイマーからの割り込み
  - デバイス制御，マルチタスク機能の実現に必須
  - 音楽・動画再生などにも用いられる
- HOSでの実装
  - Windows上のHOSでは，Windowsのタイマーでエミュレーション
  - Windows APIで，擬似的にHOSのハードウェアタイマーを発生
  - ostimer.c, wintimer.c 参照

ostimer.c

```
#include "kernel.h"
#include "ostimer.h"
#include "wintimer.h”


/* 初期化 */
void OsTimer_Initialize(VP_INT exinf)
{
    WinTimer_Initialize(0);
}
```

Windowsでタイマーをエミュレーション
するための初期設定をする

ostimer.c

```
/* タイマ割り込みハンドラ */
void OsTimer_Handler(VP_INT exinf)
{
    ID id;

    isig_tim();

    /* レディキューの回転(タスクの切り替え) */
    if (E_OK == iget_tid(&id)) {
        irot_rdq(3);
        irot_rdq(4);
        irot_rdq(5);
    }
}
```

ホストOS(Windows)によって，
一定時間ごとに駆動される

タイムティックの供給：カーネルが直接タイマーを
制御しない環境で，外部から時間経過を通知する

現在実行状態のタスクのIDを取得
(実行状態のタスクがあればE_OK)

タスクスイッチを行う rot_rdq の
割り込みの中でも使えるバージョン

wintimer.c（内部処理）

```
/* 初期化 */
void WinTimer_Initialize(INTNO intno)
{
    DWORD dwThreadId;

    /* 擬似割り込み番号を設定 */
    intnoTimer = intno;

    /* タイマ用イベントの作成 */
    hEventTimer = CreateEvent(NULL, FALSE, FALSE, NULL);

    /* マルチメディアタイマの開始 */
    timeSetEvent(10, 1, (LPTIMECALLBACK)hEventTimer, 0,
                 TIME_PERIODIC | TIME_CALLBACK_EVENT_PULSE);

    CreateThread(NULL, 0, WinTimer_Thread, 0, 0, &dwThreadId);
}
```

イベントハンドラを作る Windows API

10ms間隔のタイマーを生成

タイマーを処理するスレッドを生成

wintimer.c（内部処理）

```
/* タイマ割り込み用スレッド関数 */
DWORD WINAPI WinTimer_Thread(LPVOID param)
{
    for ( ; ; )
    {
        WaitForSingleObject(hEventTimer, INFINITE);

        hospac_win_int(intnoTimer);
    }
}
```

イベント（タイマー）を待つ

割り込み番号（intnoTimer）=0として，HOSに制御を渡す
⇒ その番号に対応するOsTimer_Handlerが駆動される

演習課題

- 課題 7a HOSにおけるラウンドロビンスケジューリング
  - この課題の狙いは，ラウンドロビンスケジューリングによるプリエンプティブな（真の）マルチタスクの仕組みを理解することである。
  - ラウンドロビンスケジューリングでは，OSが一定周期で実行プロセスを順番に（かつ強制的に）切り替えていく必要がある。
  - そのためにタイマー割り込みでスケジューラーを駆動し，プロセスの切り替えを行う。HOSでは割り込み対応版の irot_rdqを使う。

- 手順
  - sample¥win-scheduling を（元のまま）使う。
  - （擬似）タイマー割り込み処理である ostimer.c の中でコメントアウトされている if 文と irot_rdq の部分を有効にして実行する。
  - 実行結果を示し，なぜそのような動作するのか，スケジューリングの観点から考察（理由を説明）せよ。

発展課題

- 課題 7b OSのソースコードを読む（その1）
  - 今までOSのプロセス管理について学んできたので，実際にどのようなプログラムで実現されているのか，OSのソースコードを読んでみよう。
  - HOSはコメントがすべて日本語で詳しく書かれているので，内部の処理を調べることが比較的容易である。
  - この課題では，サービスコール rot_rdq の動作について調査する。
- 手順
  - 配布資料（OS2021-07op.pdf）を読んで解答せよ。
- 提出について
  - この課題（7b）は，発展的な内容なので提出は任意とする。
  - 提出期限は特に設けないので，期末試験期間まで提出を受け付ける。

HOS（μITRON）のタスク管理

サービスコール	意味	説明
cre_tsk	create task	タスクの生成
acre_tsk		同上（ID自動割り当て）
act_tsk	activate task	タスクの起動
iact_tsk		同上（割り込み用）
can_act	cancel activation	タスク起動予約の取り消し
ext_tsk	exit task	自タスクの終了
ter_tsk	terminate task	タスクの強制終了
chg_pri	change priority	タスク優先度の変更
get_pri	get priority	タスク優先度の参照
rot_rdq	rotate ready queue	実行可能キューの回転
irot_rdq		同上（割り込み用）

HOS（μITRON）のタスク管理

サービスコール	意味	説明
dly_tsk	delay task	自タスクの遅延
slp_tsk	sleep task	自タスクの休止（起床要求待ち）
tslp_tsk		同上（タイムアウトあり）
wup_tsk	wake up task	タスク起床要求
iwup_tsk		同上（割り込み用）
can_wup	cancel wakeup	タスク起床要求の取り消し
rel_wai	release wait	待ち状態の強制解除
irel_wai		同上（割り込み用）
sus_tsk	suspend task	強制待ち状態への以降
rsm_tsk	resume task	強制待ち状態からの再開
frsm_tsk	force resume task	強制待ち状態からの強制再開