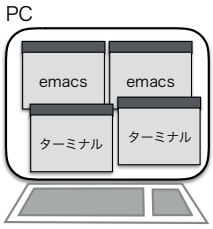
オペレーティングシステムの機能を使ってみよう
第6章 プロセスとジョブ

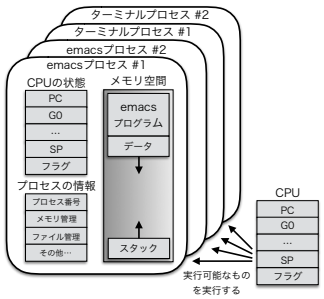
プロセス



- プログラムは機械語の羅列のこと．
- 同じプログラムが同時に複数実行されることもある．
- 実行中のプログラムのインスタンスをプロセスと呼ぶ．

プロセス ＝ 実行中のプログラム

プロセスの構造



- プロセスの情報，CPU の状態（仮想 CPU），メモリ空間（仮想メモリ）

プロセス ＝ 仮想コンピュータ

プロセス関連の UNIX コマンド

ps コマンド（*GUI* にも似たアプリがある）

- 実行中のプロセスの一覧表を表示するコマンドである．
- 一方のターミナルで emacs を起動し，もう一方のターミナルで ps コマンドを実行した例
- -bash（最近の macOS では-zsh）は入力されたコマンドを解釈して実行するシェルである．

```
$ ps
  PID TTY           TIME CMD
 2955 ttys000    0:00.02 -bash
 2964 ttys001    0:00.01 -bash
 2975 ttys001    0:00.34 emacs hello.c
$
```

- TTY は tty コマンドで確認できる．

```
$ tty
/dev/ttys001
$
```


プロセス関連の UNIX コマンド

ps コマンドの表示内容

欄	意味
PID	プロセス番号
TTY	制御端末
TT	TTY の簡易表示
TIME	プロセスがこれまでに CPU を使用した時間
CMD	プロセスを起動したコマンド
COMMAND	CMD と同じ
USER	誰の権限で実行しているか
%CPU	CPU の利用率
%MEM	メモリの利用率
VSZ	仮想記憶サイズ（KiB 単位）
RSS	常駐セット（KiB 単位）
STAT	プロセスの状態
STARTED	プロセスの開始時刻

プロセス関連の UNIX コマンド

ps コマンドのオプション

オプション	意味
u	ロングフォーマットで表示（詳しい表示）
a	他人のプロセスも表示
x	制御端末を持たないものも表示

ps コマンドの実行例（*u* オプション）

```
$ ps u
USER       PID  %CPU %MEM      VSZ    RSS   TT  STAT STARTED      TIME COMMAND
sigemura  2964   0.0  0.0  2452852   1556 s001  S    10:47AM   0:00.01 -bash
sigemura  2955   0.0  0.0  2461044   1592 s000  S+   10:46AM   0:00.02 -bash
sigemura  2975   0.0  0.2  2616528  14636 s001  S+   10:47AM   0:00.34 emacs
hello.c
```

- u オプションで詳しい表示がされた．

プロセス関連の UNIX コマンド

ps コマンドの実行例（*au* オプション）

```
$ ps au
USER       PID  %CPU %MEM      VSZ    RSS   TT  STAT STARTED      TIME COMMAND
sigemura  2975   0.0  0.2  2616528  14636 s001  S+   10:47AM   0:00.34 emacs
hello.c
sigemura  2964   0.0  0.0  2452852   1556 s001  S    10:47AM   0:00.01 -bash
root      2963   0.0  0.0  2460388   2664 s001  Ss   10:47AM   0:00.02 login -pf
 sigemura
sigemura  2955   0.0  0.0  2461044   1592 s000  S+   10:46AM   0:00.02 -bash
root      2954   0.0  0.0  2469604   2788 s000  Ss   10:46AM   0:00.02 login -pf
 sigemura
root      3790   0.0  0.0  2433188   1004 s000  R+   12:04PM   0:00.00 ps au
```

- a オプションで他人のプロセスまで表示された．

プロセス関連の UNIX コマンド

ps コマンドの実行例（*aux* オプション）

```
$ ps aux
USER               PID  %CPU %MEM      VSZ    RSS   TT  STAT STARTED      TIME
COMMAND
_windowserver      175   6.3  1.3  3693100 106960   ??  Ss   Wed10PM  17:10.51 /
System/Lib
_hidd              121   1.1  0.0  2473636   4052   ??  Ss   Wed10PM   6:43.03 /
usr/libexe
sigemura          1124   0.4  0.1  2542556   7888   ??  S    Thu09AM   0:32.58 /
Library/In
sigemura           861   0.3  0.0  2521980   3328   ??  S    Wed10PM   0:09.41 /
System/Lib
root               253   0.3  0.1  2472892   5548   ??  Ss   Wed10PM   0:44.44 /
usr/libexe
sigemura           891   0.2  0.0  2470772   2160   ??  S    Wed10PM   0:06.82 /
System/Lib
...300行程度続く...
$
```

- x オプションで制御端末を持たないプロセスまで表示された．

プロセス関連の UNIX コマンド

ps コマンド *STAT* 表示の意

一文字目	意味
I	20 秒以上 sleep している
S	20 秒未満の sleep
R	実行可能
T	一時停止状態（stop，Ctrl-Z）
Z	ゾンビ（Zombi）
...	

+

二文字目	意味
+	フォアグラウンド
s	セッションリーダ
...	

- 前の実行例の STAT の意味

演習 5-1

CLI でプロセス一覧を見てみよう

- システム内の全プロセスを ps コマンドで表示してみる．
- less コマンドとパイプで接続して表示してみる．
- どんなプロセスが存在するかゆっくり眺める．

（コマンド一覧は「0510_UNIX コマンド（ps など）.pdf」参照）

プロセス関連の UNIX コマンド

kill コマンド

kill コマンドの書式

書式

```
kill [-シグナル] PID ...
```

シグナル（省略時は *TERM* と同じ）

番号	名前	意味
2	INT	終了（Ctrl-C と同じ）
9	KILL	強制終了
15	TERM	終了（オプション無しと同じ）
18	TSTP	一時停止（Ctrl-Z と同じ）
19	CONT	一時停止後の再開

プロセス関連の UNIX コマンド

kill コマンドの使用例

```
$ sleep 10000 &                          <--- サンプル用プロセスを起動
[1] 59882
$ ps
  PID TTY           TIME CMD
58863 ttys003    0:00.23 bash
59882 ttys003    0:00.00 sleep 10000     <--- PIDが分かる
$ kill 59882                             <--- プロセスを終了させる
$
[1]+  Terminated: 15          sleep 10000
$ sleep 10000 &                          <--- 新しいサンプル用プロセスを起動
[1] 59888                                <--- 実はPIDはここでも分かる
$ kill -TSTP 59888                       <--- プロセスを一時停止
[1]+  Stopped                 sleep 10000
$ ps
  PID TTY           TIME CMD
58863 ttys003    0:00.24 bash
59888 ttys003    0:00.00 sleep 10000     <--- プロセスは存在している
$ kill -CONT 59888                       <--- プロセスを再開させる
$ kill 59888                             <--- プロセスを終了させる
$
[1]+  Terminated: 15          sleep 10000
```

演習 5-2

*CLI*でプロセスを操作してみよう

- 前のページの操作を自分のコンピュータで試してみる．

（コマンド一覧は「0510_UNIX コマンド（ps など）.pdf」参照）

ジョブ

```
# 通常のコマンド実行
$ emacs hello.c                                  <-- 1プロセスが1ジョブ

# パイプを使用しファイルサイズ順にソートして表示
$ ls -l | sort -n --key=5                        <-- 2プロセスが1ジョブ

# 二つのコマンド（ジョブ）を順次実行
$ touch a.txt; chmod 777 a.txt                   <-- 2ジョブ

# 二つのコマンド（ジョブ）を並列実行
$ touch a.txt & touch b.txt                      <-- 2ジョブ
```

フォアグラウンド・ジョブ シェルがジョブの終了を待つ．ジョブが終了したらプロンプトが表示される．

バックグラウンド・ジョブ コマンドの最後に&を付けて実行する．シェルがジョブの終了を待たない．ジョブが終了していなくてもプロンプトが表示される．次のジョブと並列実行ができる．

ジョブ制御

```
$ sleep 2000                                     <-- フォアグラウンドで起動
^Z
[1]+  Stopped                 sleep 2000         <-- 一時停止した
$ bg                                             <-- バックグラウンドで再開
[1]+ sleep 2000 &
$ sleep 1000 &                                   <-- 新しくバックグラウンドで起動
[2] 59961
$ jobs                                           <-- 実行中のジョブを確認
[1]-  Running                 sleep 2000 &
[2]+  Running                 sleep 1000 &
$ fg 1                                           <-- 1番をフォアグラウンドに変更
sleep 2000                                           (zshの人は「fg %1」と入力)
^C                                               <-- Ctrl-C で終了
$ jobs
[2]+  Running                 sleep 1000 &       <-- 2番だけになった
```

Ctrl-C フォアグラウンド・ジョブに INT シグナルを送る．

Ctrl-Z フォアグラウンド・ジョブに TSTP シグナルを送る．

jobs そのシェルが管理しているジョブの一覧を表示する．

fg,bf バックグラウンド・フォアグラウンドの切替え．

課題 No.5

1. プロセス数を調べる．
   - システム内の全プロセスの個数
   - システム内の全ての自分のプロセスの個数
2. 配布された３分タイマープログラムを用いて以下を行う．
   - 「第１のターミナル」でタイマープログラムを起動する．
   - 「第２のターミナル」を操作し以下のことをする．

   (1) タイマープログラムの PID と STAT を確認する．
   (2) タイマープログラムを一時停止させる．
   (3) タイマープログラムの STAT を確認する．
   (4) タイマープログラムを再開させる．
   (5) タイマープログラムを終了させる．
   タイマープログラムは画面をクリアして終了したか？
3. タイマープログラムを起動したターミナルだけで行う．

   (1) タイマープログラムを一時停止
   (2) 一時停止したタイマープログラムを再開
   (3) タイマープログラムを終了