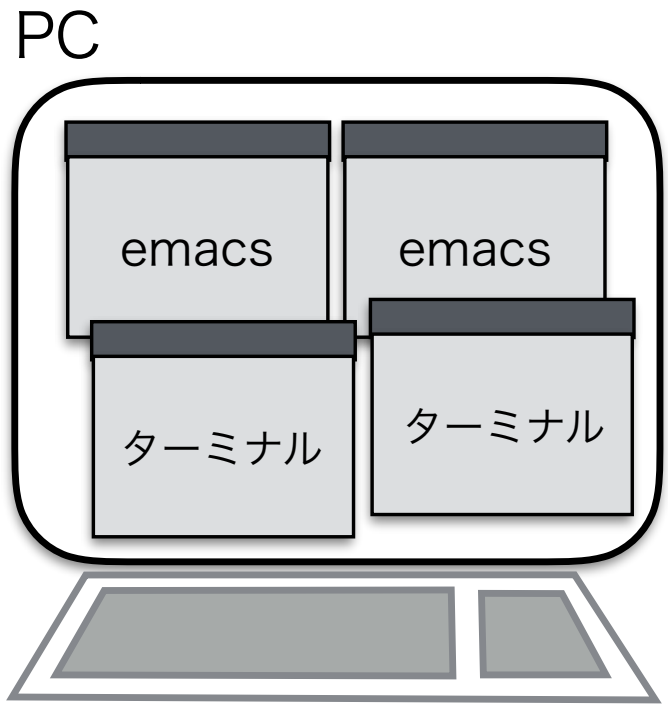
オペレーティングシステムの機能を使ってみよう
第６章 プロセスとジョブ

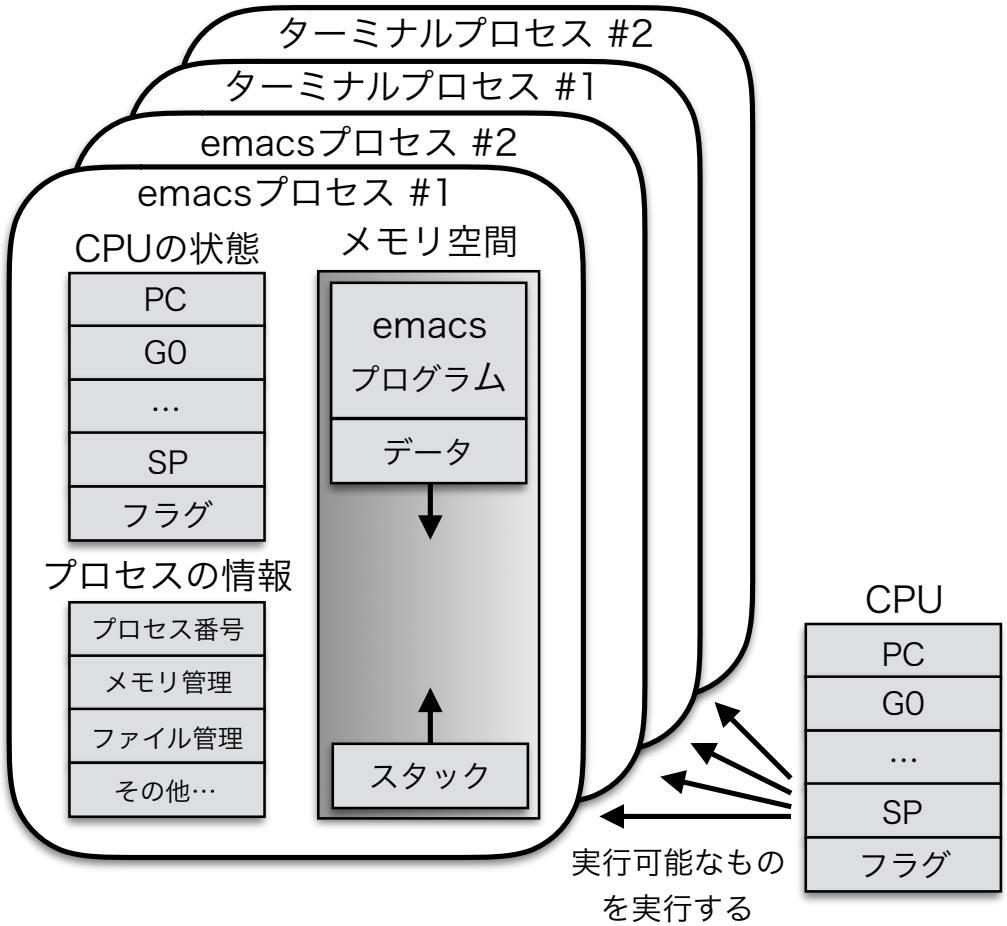
プロセス



- 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

- 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 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）
...	

+

ニ文字目	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

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

```
$ /Applications/Utilities/Grapher.app/Contents/MacOS/Grapher &
[1] 34339
$ ps
  PID TTY           TIME CMD
34306 ttys000    0:00.03 -bash
34339 ttys000    0:01.08 /Applications/Utilities/Grapher.app/Contents/MacOS/
Grapher
$ kill 34339
$
[1]+  Terminated: 15  /Applications/Utilities/Grapher.app/Contents/MacOS/Grapher
$ /Applications/Utilities/Grapher.app/Contents/MacOS/Grapher &
[1] 34346
$ kill -TSTP 34346
[1]+  Stopped         /Applications/Utilities/Grapher.app/Contents/MacOS/Grapher
$ kill -CONT 34346
$ kill -2 34346
$
[1]+  Interrupt: 2    /Applications/Utilities/Grapher.app/Contents/MacOS/Grapher
$
```

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

```
$ /Applications/Utilities/Grapher.app/Contents/MacOS/Grapher
^Z
[1]+  Stopped     /Applications/Utilities/Grapher.app/Contents/MacOS/Grapher
$ bg
[1]+ /Applications/Utilities/Grapher.app/Contents/MacOS/Grapher &
$ /Applications/Utilities/Grapher.app/Contents/MacOS/Grapher &
[2] 34655
$ jobs
[1]-  Running     /Applications/Utilities/Grapher.app/Contents/MacOS/Grapher &
[2]+  Running     /Applications/Utilities/Grapher.app/Contents/MacOS/Grapher &
$ fg 2
/Applications/Utilities/Grapher.app/Contents/MacOS/Grapher
^C
$ jobs
[1]+  Running     /Applications/Utilities/Grapher.app/Contents/MacOS/Grapher &
$
```

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. 本文と照らし合わせながらプロセスの実行例（リスト全部）を試し，内容をよく理解しなさい．一時停止状態の Grapher を操作するとどうなるか等，よく観察すること．
2. 以下の操作方法を考えて手順を説明しなさい．
   (1) フォアグラウンドで暴走してしまったジョブを強制終了する手順．（Ctrl-C ではなく，KILL シグナルを使用すること）
   (2) 間違ってフォアグラウンドで起動した emacs をバックグラウンドに変更する手順．
   (3) バックグラウンドで実行中のジョブをフォアグラウンドに変更して Ctrl-C で終了する手順．