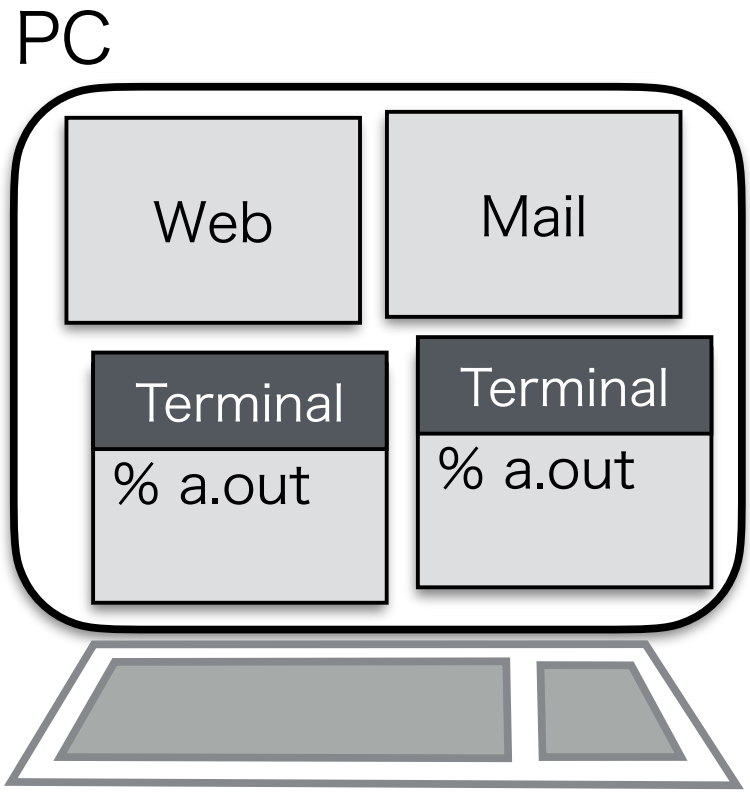
オペレーティングシステムの機能を使ってみよう
第６章 プロセスとジョブ

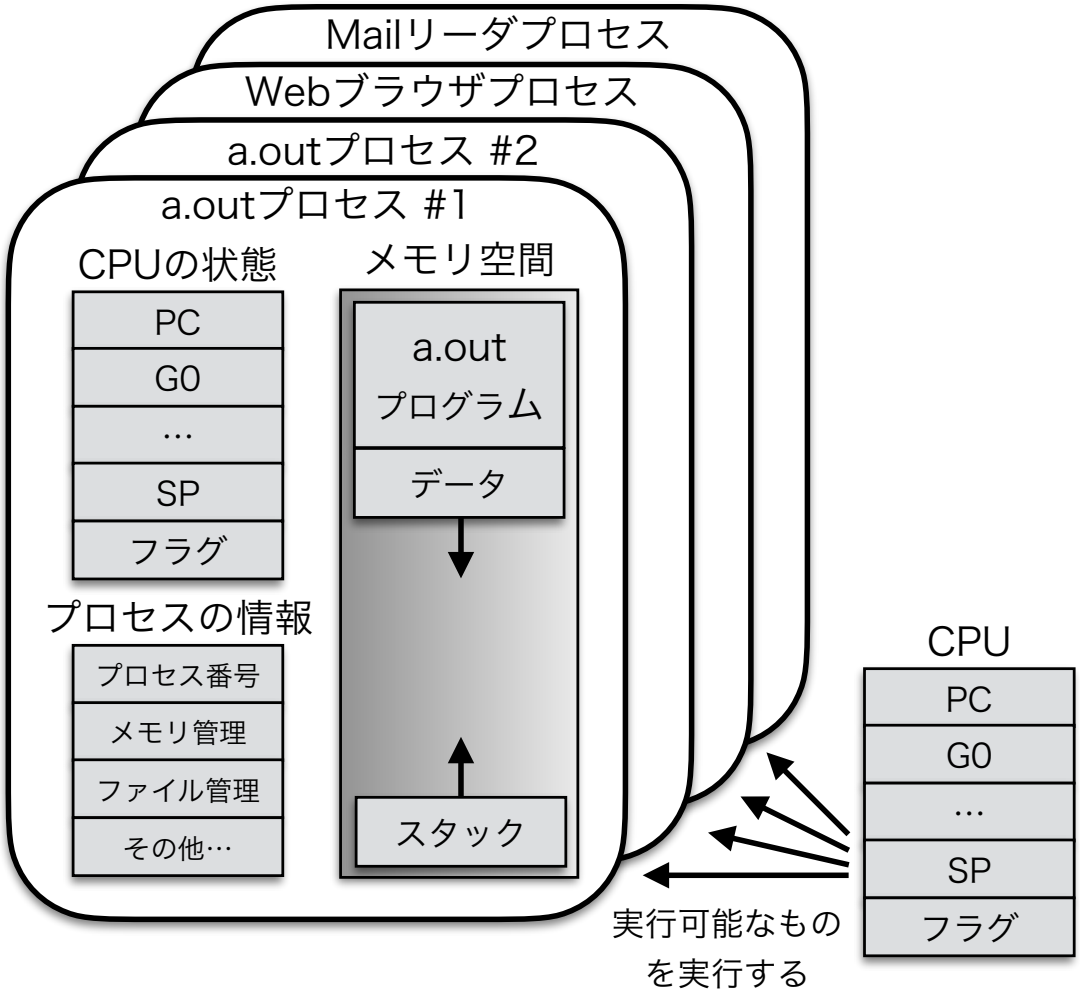
プロセス



- 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
27828 ttys000    0:00.51 -zsh
46471 ttys000    0:00.62 vi chap6s.tex
38060 ttys001    0:00.10 -zsh
% tty
/dev/ttys001
%
```

- 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 38060   0.5  0.0 408824272   5680 s001  S    11:03AM   0:00.15 -zsh
sigemura 46471   0.0  0.1 408816704   9264 s000  S+   11:33AM   0:00.74 vi
chap6s.tex
sigemura 27828   0.0  0.0 409086416   7472 s000  S    10:35AM   0:00.51 -zsh
```

- 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
root      60998   0.0  0.0 408766112   1808 s001  R+   12:08PM   0:00.01 ps au
sigemura 46471   0.0  0.1 408816704   9264 s000  S+   11:33AM   0:00.74 vi
chap6s.tex
sigemura 38060   0.0  0.0 408824272   5680 s001  S    11:03AM   0:00.15 -zsh
root      38059   0.0  0.0 408646464   4128 s001  Ss   11:03AM   0:00.02 login -
pfl sigemu
sigemura 27828   0.0  0.0 409086416   7472 s000  S    10:35AM   0:00.51 -zsh
root      27827   0.0  0.0 408646464   4464 s000  Ss   10:35AM   0:00.02 login -
pf sigemur
```

- 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      181  30.5  1.5 410618752 251024   ??  Rs   23Mar23  64:38.93
/System/L
sigemura         43374  10.5  0.6 410251920  97664   ??  R    11:16AM   5:13.83
/Applicat
sigemura           846   4.3  0.6 410068912 102544   ??  S    24Mar23  24:58.94
/System/A
sigemura         12009   3.1  1.5 410249072 245024   ??  S    24Mar23   1:35.29
/System/A
root               418   1.1  0.1 409363184  13936   ??  Rs   23Mar23   3:45.99
/Library/
root               153   1.1  0.1 408268064  16240   ??  Ss   23Mar23   1:28.69
/System/L
...600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「0610_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] 75868
% ps
  PID TTY           TIME CMD
38060 ttys001    0:00.22 -zsh
75868 ttys001    0:00.01 sleep 10000      <--- PIDが分かる
% kill 75868                                     <--- プロセスを終了させる
[1]  + terminated  sleep 10000
% sleep 10000 &                                  <--- 新しいサンプル用プロセスを起動
[1] 75871                                        <--- 実はPIDはここでも分かる
% kill -TSTP 75871                               <--- プロセスを一時停止
[1]  + suspended  sleep 10000
% ps
  PID TTY           TIME CMD
38060 ttys001    0:00.26 -zsh
75871 ttys001    0:00.00 sleep 10000      <--- プロセスは存在している
% kill -CONT 75871                               <--- プロセスを再開させる
% kill 75871                                     <--- プロセスを終了させる
[1]  + terminated  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「0610_UNIX コマンド（ps など）.pdf」参照)

ジョブ

```
# 通常のコマンド実行
% vi 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
zsh: suspended  sleep 2000                    <-- 一時停止した
% bg                                          <-- バックグラウンドで再開
[1]  + continued  sleep 2000
% sleep 1000 &                                <-- 新しくバックグラウンドで起動
[2] 80228
% 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
[1]  - running    sleep 2000                  <-- フォアグラウンドに変更された
^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,bg バックグラウンド・フォアグラウンドの切替え.