UNIX

Lecture 6 : Process Management

Filipe Vasconcelos[1]
[1]ESME, Lille, filipe.vasconcelo@esme.fr

Outline

Intended Learning Outcomes

ILO5

Analyze and explore the file system hierarchy of a Unix-type OS using basic commands, manage the access rights of directories and files within the hierarchy and manage processes

Outline

Processes
The ‘ps‘ Command

A process corresponds to the execution of any program, command, or script. If process 2 has been initiated by process 1, it is referred to as a **child process**, and process 1 is known as the **parent process**. Each process is identified by a unique number called **PID**.
To display all processes initiated by the current user, you can use the ‘ps‘ command.

```
ps
```

```
  PID TTY          TIME CMD
10038 pts/0    00:00:01 bash
10364 pts/0    00:00:00 ps
```

The ‘ps‘ Command
Options

To display all processes (even those initiated by root)

```
ps -A
```

```
  PID TTY          TIME CMD
    1 ?        00:00:31 init
10036 ?        00:00:00 init
10037 ?        00:00:02 init
10038 pts/0    00:00:02 bash
10666 pts/0    00:00:00 ps
```

To obtain a process hierarchy

```
bash    # to launch a bash process
ps -elH
```

```
$ ps -elH
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0     1     0  0  80   0 -   380 -      ?        00:00:31 init
5 S     0 10036     1  0  80   0 -   293 -      ?        00:00:00   init
1 S     0 10037 10036  0  80   0 -   293 -      ?        00:00:02     init
4 S  1000 10038 10037  0  80   0 -  2739 do_wai pts/0    00:00:04       bash
```

Outline

Running a Task

To run a task, simply execute the script in the terminal.

```
./simulation
```

```
iter : 0      value = 6438   Elapsed Time : 0     (s)
iter : 1      value = 7679   Elapsed Time : 3     (s)
iter : 2      value = 24387  Elapsed Time : 6     (s)
iter : 3      value = 25687  Elapsed Time : 7     (s)
iter : 4      value = 13268  Elapsed Time : 8     (s)
iter : 5      value = 20162  Elapsed Time : 10    (s)
```

Foreground or Background Task

If you start a task and want to keep control of the terminal, you can send the process to the **background**. To regain control, you can temporarily pause the process (`Ctrl+Z`).

```
./simulation > log
Ctrl+Z
```

```
^Z
[1]+  Stopped                 ./simulation > log
```

To move the process to the background, you can use the `bg` (background) command.

```
bg 1
```

```
[1]+ ./simulation > log &
```

Running in the Background Directly

To run a process directly in the background, you can add & to the command.

```
./simulation > log &
```

```
[1] 12264
```

If you launch multiple processes:

```
./simulation > log1 &
./simulation > log2 &
./simulation > log3 &
```

```
[1] 12725
[2] 12727
[3] 12730
```

Bringing a Process to the Foreground

To bring a process to the foreground, you can use the `fg` (foreground) command.

```
fg 1
```

```
./simulation > log1
```

`nohup` Command

The `nohup` command allows you to launch a process that will remain active even after the user logs out. When combined with `&`, which enables running in the background, `nohup` creates processes that run seamlessly without being tied to the user.
Simple Usage:

```
nohup <command> &
```

If you don't want to redirect standard output to the `nohup.out` file:

```
nohup <command> > /dev/null &
```

Outline

`kill` Command

The `kill` command sends a termination signal to a process. There are numerous signals (list of signals: `kill -l`).

```
kill -l
```

```
 1) SIGHUP       2) SIGINT       3) SIGQUIT      ...
 9) SIGKILL      ...            15) SIGTERM      ...
19) SIGSTOP      ...
```

`top` Command

The `top` command is an advanced tool for process management. The header provides an overview of the physical or virtual machine's health.

```
top
```

```
top - 13:16:38 up 1 day, 17:36,  0 users,  load average: 0.08, 0.03, 0.01
Tasks:  11 total,   1 running,  10 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  6.9 sy,  0.0 ni, 93.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   6252.0 total,   5732.9 free,     98.6 used,    420.6 buff/cache
MiB Swap:   2048.0 total,   2048.0 free,      0.0 used.   5932.4 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 4097 filipe    20   0    8752   3728   3264 S   0.3   0.1   0:00.06 simulation
    1 root      20   0    1060    576    468 S   0.0   0.0   0:26.82 init
  487 root      20   0     924    108     20 S   0.0   0.0   0:15.40 init
 2095 root      20   0    1252    436     20 S   0.0   0.0   0:10.16 init
 3444 filipe    20   0   64196  24408   7288 S   0.0   0.4   0:00.96 mupdf-x11
 3613 filipe    20   0   62620  23344   7672 S   0.0   0.4   0:04.40 mupdf-x11
 4027 root      20   0    1060    244     20 S   0.0   0.0   0:00.01 init
 4028 root      20   0    1060    244     20 S   0.0   0.0   0:00.12 init
 4029 filipe    20   0   11144   6328   3492 S   0.0   0.1   0:00.59 bash
```