Système d'Exploitation Unix
TP
Commandes de base de gestion de processus

Les exercices suivants ont pour but de vous familiariser avec les commandes de base de gestion des processus.
Il vous est recommandé de consulter les pages man pour de plus amples informations sur leur syntaxe, leur sémantique et les éventuelles options qu'elles offrent.

Un `texte en police courier` correspond soit à une sortie écran soit à des noms spécifiques (menus, fenêtre, icône, processus, commandes…).
Un **texte en police gras** correspond à ce que l'utilisateur doit introduire comme valeur de paramètre, ou encore, est utilisé pour attirer l'attention de l'utilisateur.

1) Visualisation des caractéristiques des processus : `ps`, `top`

La commande `ps` permet de visualiser des informations sur les processus.

```
[ev00000@saphyr ~]$ ps
  PID TTY          TIME CMD
31903 pts/4    00:00:00 bash
31929 pts/4    00:00:00 ps
```

Utilisée sans option, la commande ne traite que les processus associés au même terminal que la commande `ps`.
1.a) Consulter les pages man de la commande `ps` pour prendre connaissance des différentes options offertes.
1.b) Afficher les informations suivantes concernant le processus associé à votre commande `ps` :

- `PID` : numéro du processus,
- `PPID`: numéro du processus parent,
- `TTY` : identification de son terminal de contrôle,
- `UID` : identité du propriétaire réel du processus,
- `PRI` : priorité courante du processus,
- `ADDR`: adresse du processus en mémoire s'il est en mémoire et sur disque sinon,
- `SZ` : taille du processus exprimée en nombre de blocs.

La commande `ps` présente un cliché instantané des processus en cours. Pour obtenir un affichage remis à jour régulièrement, utilisez la commande `top`.

```
0 R 60933 32023 31903  0  77   0 -  1035 -      pts/4    00:00:00 ps
```

1.c) Tester la commande `top`.

1.d) Préciser quand il est préférable d'utiliser la commande `top` plutôt que la commande `ps`.

Michel.Soto@parisdescartes.fr

2) Exécution différée : at, batch

La commande at

La commande at permet de différer, à une date spécifiée, l'exécution d'une suite de commandes. Sauf demande de redirection explicite, la sortie standard et la sortie erreur sont redirigées sur la boîte à lettres de l'utilisateur.

```
[ev00000@saphyr ~]$ at 17:24
at> ls >fls
at> <EOT>← frappe de ctrl d

job 13 at 2007-02-05 17:24
```

a) Tester la commande dans le cas où les commandes à exécuter sont lues sur l'entréestandard puis le cas où elles sont lues dans un fichier.

b) Comment êtes-vous avertis de la terminaison de la commande ?

L'option –l (respectivement –r) permet de lister (respectivement de supprimer) les commandes enregistrées. On peut également utiliser à la place les deux commandes atq et atrm.

c) Tester les options précédentes.

Le fichier /etc/at.allow (respectivement /etc/at.deny) contient la liste des utilisateurs autorisés (respectivement non autorisés) à utiliser la commande at.

La commande batch

La commande batch permet de différer, à une date déterminée par le système en fonction de la charge du système, l'exécution de commandes lues sur l'entrée standard. Les mécanismes de redirection sont les mêmes que ceux de la commande at.

```
[ev00000@saphyr ~]$ batch
at> ls >fls2
at> <EOT>← frappe de ctrl d
```

d) Tester la commande batch.
e) Donner des exemples où il est préférable d'utiliser l'une des deux commandes (at et batch) plutôt que l'autre ?

3) Emission d'un signal : kill

La commande kill permet d'envoyer au processus (ou groupe de processus) d'identification donnée le signal désigné.

```
[ev00000@saphyr ~]$ ps
  PID TTY          TIME CMD
 1059 pts/3    00:00:00 bash
```

```
 1173 pts/3    00:00:02 a.out
 1174 pts/3    00:00:00 ps
[ev00000@saphyr ~]$ kill -9 1173
```

Les signaux sont identifiés par des nombres entiers (numéros absolus dans le système tels que fournis par la commande `ps`) ou par des noms symboliques tels qu'ils apparaissent dans le fichier `signal.h` (privés du préfixe `SIG`). Les noms des signaux reconnus sont affichés par la commande `kill –l`.

a) Lancer un processus puis lui émettre certains signaux. Que se passe-t-il ?

4) Ignorer certains signaux : `nohup`

La commande `nohup` permet de lancer une commande de telle sorte que le processus l'exécutant ignore les signaux `SIGINT`, `SIGQUIT` et `SIGHUP`.

```
[ev00000@saphyr ~]$ nohup ./a.out &
[1] 1197
[ev00000@saphyr ~]$ nohup: ajout Ã  la sortie de `nohup.out'
```

Lancer un processus puis lui émettre chacun des signaux précédents.

a) Que se passe-t-il ? Les processus réagissent-ils de la même façon que précédemment ?

5) Modification de priorité par défaut d'une commande : `nice`

La commande `nice` lance l'exécution d'une commande en ajoutant la valeur d'un argument numérique au paramètre d'ordonnancement de son processus. La valeur de l'argument numérique doit être comprise entre 1 et 19 (valeur par défaut, 10). La commande, ainsi lancée, verra sa priorité abaissée.

```
[ev00000@saphyr ~]$ nice ./a.out &
[1] 1327
[ev00000@saphyr ~]$ ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S 60933  1301  1300  0  75   0 -  1134 wait   pts/1    00:00:00 bash
0 R 60933  1327  1301 99  93  10 -   361 -      pts/1    00:00:03 a.out
0 R 60933  1328  1301  0  77   0 -  1034 -      pts/1    00:00:00 ps
```

a) Lancer une commande avec une priorité moindre que celle par défaut.

b) Un argument négatif permet de lancer une commande avec une plus grande priorité.

c) Lancer une commande avec une priorité supérieure à celle par défaut.

   Quel est le résultat de votre action ? Pourquoi ?

6) Affichage du temps d'exécution d'un processus : `time`

La commande `time` permet de lancer une commande quelconque avec affichage, à la fin du processus l'exécutant, d'informations relatives à son temps d'exécution.

```
[ev00000@saphyr ~]$ time ps > fps

real    0m0.017s
user    0m0.004s
sys     0m0.007s
```

a) Tester la commande `time` avec certaines commandes.

7) Autre : Mécanisme `Job control`

Ce mécanisme permet une gestion des processus créés à partir d'un `shell` supportant cette facilité. Il permet aux utilisateurs de ne voir que les processus qu'ils ont lancés sous le `shell` interactif dans lequel ils travaillent et qui constitue le cadre d'une **session de travail**. Chaque commande analysée par le `shell` correspond localement à un `job` ou `tâche` : d'un point de vue interne, il s'agit d'un groupe de processus.
Les processus lancés depuis le `shell` interactif peuvent se trouver dans l'un des états suivants :

En premier plan

Dans une session, il existe au plus une tâche dont les processus peuvent lire et écrire sur le terminal de lancement et seront avisés lors de la frappe de caractères de contrôle `intr`, `quit` et `susp` sur le clavier de ce terminal.
Une commande sous la forme

[ev00000@saphyr ~]$ commande

correspond à une demande d'exécution en premier plan de la commande mentionnée.

En arrière plan

Les processus appartenant à une telle tâche ne peuvent pas lire au terminal et ne sont pas concernés par la frappe des caractères de contrôle précédents.
Une commande sous la forme

[ev00000@saphyr ~]$ commande&

correspond à une demande d'exécution en arrière plan de la commande mentionnée.

Suspendu

Les processus sont en sommeil et attendent que l'utilisateur demande explicitement leur passage à l'un des deux modes précédents.

La commande `jobs`

La commande `jobs` fournit la liste des tâches créées par le `shell`. La tâche repérée par le caractère + est la tâche courante.

```
[ev00000@saphyr ~]$ jobs
```

```
[1]-  Running                 time ./a.out &
[2]+  Stopped                 vim scriptdate
```

La commande `fg`

La commande `fg` permet l'exécution en premier plan de la tâche courante.

```
[ev00000@saphyr ~]$ nohup ./a.out &
[1] 1426
[ev00000@saphyr ~]$ jobs
[1]+  Running                 time ./a.out &

[ev00000@saphyr ~]$ fg %1
time ./a.out                  ← E/S standards
```

La commande `bg`

La commande `bg` permet l'exécution en arrière plan de la tâche courante.

```
[ev00000@saphyr ~]$ jobs
[2]-  Stopped (tty output)    vi script
[3]+  Stopped                 netscape
[ev00000@saphyr ~]$ bg
[3]+ netscape &
[ev00000@saphyr ~]$ jobs
[2]+  Stopped (tty output)    vi script
[3]-  Running                 netscape &
```

La commande `stop`

La commande `stop` permet l'interruption de l'exécution de la tâche courante. L'interruption de l'exécution d'un processus en premier plan peut être obtenue par la frappe de `ctrl z`.

```
[ev00000@saphyr ~]$ netscape
                              ← frappe de ctrl z
[3]+  Stopped                 netscape
```