Programmation Système
Travaux Pratiques (1), Licence 2 Informatique
Commandes de base de gestion de processus

Les exercices suivants ont pour but de vous familiariser avec les commandes de base de manipulation des répertoires et fichiers.
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.

Les instructions des exercices se repèrent par des icônes, qui sont les suivantes :

	Information	Information concernant l'usage ou le rôle d'une commande, par exemple. Dans certains cas, il s'agit d'une information sur ce que vous êtes en train de faire ou sur ce qui se passe.
	Exemple	Exemple d'utilisation.
	Contrôle	Vérifier le résultat d'une (ou plusieurs) action(s).
	Action	Effectuer la ou les action(s) décrite(s).
	Question	Questions auxquelles vous devez répondre.

De plus, 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 times gras** correspond à ce que l'utilisateur doit introduire comme valeur de paramètre, ou encore, est utilisé pour attirer l'attention de l'utilisateur.

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

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

```
$ ps
PID TTY STAT TIME COMMAND
319 1 S  0:03 -bash
366 1 S  0:00 sh /usr/X11R6/bin/startx
367 1 S  0:00 xinit /usr/X11R6/lib/X11/xinit/xinitrc --
371 1 S  0:04 fvwm2 -cmd FvwmM4 -debug /etc/X11/AnotherLevel/fvwm2rc.m4
616 1 S  0:02 /usr/X11R6/lib/X11/fvwm2//FvwmTaskBar 9 4 /tmp/fvwmrcb00380 0 8
620 1 SN 0:00 xload -nolabel -geometry 32x20+0+0 -bg grey60 -update 5
```


Mohamed Taghelit , taghelit@univ-tours.fr

```
621 1 S  0:00 /usr/X11R6/lib/X11/fvwm2//FvwmPager 11 4 /tmp/fvwmrcb00380 0 8 0 1
622 p0 S 0:00 bash
656 p1 T 0:00 ./a.out
696 p0 R 0:00 ps
$
```

Utilisée sans option, la commande ne traite que les processus associés au même terminal que la commande `ps`.

Consulter les pages man de la commande `ps` pour prendre connaissance des différentes options offertes.

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`.

Tester la commande `top`.

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

Exécutions périodiques et différées de tâches : `cron`, `crontab`, `batch`

Le daemon `crond` est chargé d'exécuter toutes tâches définies et planifiées à l'avance. Ces tâches, qui peuvent êtres de simples commandes aux scripts complexes, sont décrites dans un fichier `crontab` et sont exécutées à heure fixe, éventuellement de façon périodique.

Note : Sauf demande explicite de redirection, la sortie standard et la sortie erreur sont redirigées sur la boîte à lettres de l'utilisateur (utiliser la commande `mail`).

La commande `crontab`

La commande `crontab` permet de créer (option `-e`), ou d'éditer pour modifications, votre fichier `crontab`. Consulter le manuel pour la description des entrées de `crontab`.

Note : le fichier `crontab` d'un utilisateur est sauvegardé dans `/var/spool/cron/$user`, où `$user` est le nom de login de l'utilisateur.

Éditer un fichier `crontab` où vous spécifierez que la tâche '`echo Bonjour !`' doit être exécutée toutes les 5 minutes.

Pour répéter toutes les 5 minutes (ou autre champs) écrire */5 dans le 1er champs (minutes).

Si le mail est installé (`chkconfig -list | grep mail`) vous devez avoir une ligne comme suit :

```
[student]$ chkconfig --list |grep mail
sendmail          0:arrêt     1:arrêt     2:marche    3:marche    4:marche
            5:arrêt     6:arrêt
[student]$
```

On peut aussi vérifier l'état de `sendmail` ainsi :

```
[student]$ sudo service sendmail status
sendmail est arrêté
sm-client est arrêté
[student]$
```

Si l'appel à mail répond « no mail for student », alors relancez le service avec

```
[student]$ sudo service sendmail restart
```

Ca relance le service `sendmail` et le service `sm_client` (peut prendre un certain temps).

Comme pour batch et at ci-dessous, pour lancer `firefox`, précédez de « `export DISPLAY=:0;` »

Idem si on utilise `xterm` à la place de `firefox`.

Pour changer l'éditeur par défaut (`vi`) de `crontab`, faire :

```
[student]$ export EDITOR=/usr/bin/gedit
```
(ou un autre éditeur de son choix)

La commande `crontab` permet aussi de lister (option `-l`) le contenu du fichier `crontab`.

Vérifier que votre fichier `crontab` contient bien la tâche précédemment introduite et son heure d'exécution.

Comment êtes-vous avertis de la terminaison de l'exécution de la tâche ?

Modifier l'entrée de votre fichier `crontab` de façon que la sortie de chaque exécution de votre tâche soit concaténée dans un fichier (par exemple nommé `cron.log`).

Vérifier que ce qui précède est bien pris en compte.

La commande `crontab` permet également de supprimer (option `-r`) le fichier `crontab` (et donc tout son contenu).

Supprimer votre fichier `crontab`.

Vérifier que votre fichier `crontab` a bien été supprimé.

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 `crontab`.

```
$ batch
at> lpr fichier_salaires
at> <EOT>                    ← frappe de ctrl+d
warning: commands will be executed using /bin/sh
job 5 at 1999-10-11 19:01
$
```

Tester la commande `batch`.

Pour lancer comme commande `firefox`, précédez l'appel de "`export DISPLAY=:0`"

```
[student] $ batch
at> export DISPLAY=:0
at> firefox
at> <EOT>
job 45 at 2014-09-15 15:00
[student]
```

Donner des exemples où il est préférable d'utiliser l'une des deux commandes (`crontab` et `batch`) plutôt que l'autre ?

La commande `at` permet l'exécution de commandes, lues sur l'entrée standard ou à partir d'un fichier, à une heure précise. Les commandes `atq`, `atrm` permettent respectivement, d'afficher, respectivement de supprimer, la liste des commandes en attente.

```
$ at 14:50
```

```
at> export DISPLAY=:0
at> firefox
at> <EOT>                    ← frappe de ctrl+d
job 42 at 2014-09-15 14:50
$
```

Tester la commande `at`.

Pour lancer comme commande `firefox`, précédez l'appel de "`export DISPLAY=:0`"

[student] $ at

```
at> export DISPLAY=:0
at> firefox
at> <EOT>
job 46 at 2014-09-15 15:09
[student]
```

Émission d'un signal : `kill`

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

```
$ ps
…
656 p1 T 0:00 a.out
…
$ kill –9 656
$
```

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`.

Lancer un processus puis lui émettre certains signaux.

Que se passe-t-il ?

Ignorer certains signaux : `nohup`

La commande `nohup` permet de lancer une commande de telle sorte que le processus l'exécutant ignore le signal `SIGHUP`.

```
$ nohup firefox
nohup: appending output to `nohup.out'
$
```

Lancer un processus puis lui émettre le signal précédent.

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

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.

```
$ nice a.out &
[1] 662
$ ps cl
FLAGS   UID   PID  PPID PRI  NI   SIZE   RSS WCHAN   STA TTY TIME COMMAND
…
100000   500   662   622  13  10    716   216  11366b S N p0  0:00 nice …
100000   500   663   622  14   0    844   484  0       R   p0  0:00 ps
```

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

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

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

Quel est le résultat de votre action ? Pourquoi ?

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.

```
$ time ps > toto
real   0m0.017s
user   0m0.004s
sys    0m0.013s
```

```
$
```

Tester la commande `time` avec certaines commandes.

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

```
$ 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

```
$ 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.

```
$ jobs
[1]-  Stopped                 time ./a.out
[2]+  Running                 xedit ../script2 &
$
```

La commande `fg`

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

```
$ ./a.out &
[1] 667
$ jobs
[1]+  Running                 ./a.out &
$ fg %1
./a.out
```

← E/S standards

La commande `bg`

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

```
$ jobs
[2]-  Stopped (tty output)    vi script
[3]+  Stopped                 firefox
$ bg
[3]+ firefox &
$ jobs
[2]+  Stopped (tty output)    vi script
[3]-  Running                 firefox &
$
```

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`.

```
$ firefox
```

← frappe de `ctrl+z`

```
[3]+  Stopped                 firefox
$
```