Processus

I. Commandes Unix

1. Commandes `ps` et `top`

Nous allons utiliser les deux commandes `ps` et `top` qui permettent de lister les processus dans le terminal.

1. Dans un terminal : *à taper*

```
nsi@ordi$ gedit &
nsi@ordi$ ps
```

2. résultat : *à compléter*

```
PID TTY          TIME CMD
.
.
.
```

3. Dans le terminal : *à taper*

```
nsi@ordi$ ps -u
```

4. Que représente chacune des colonnes ? *à compléter*
   - `USER` indique
   - `PID` donne l'identifiant numérique du processus.
   - `%CPU` et `%MEM` indiquent respectivement
   - `STAT` indique l'état du processus, `S` pour *sleeping*, le processus est en attente et `R` pour *running*, le processus est dans l'état prêt ou élu.
   - `COMMAND` indique la commande utilisée pour lancer le programme.
   - `START` et `TIME` indiquent respectivement
5. Fermer l'application `gedit` et utiliser à nouveau la commande `ps` dans le terminal. Que constatez-vous ?
6. Dans le terminal : *à taper*

```
nsi@ordi$ top
```

   Quelles sont les principales différences avec la commande `ps` ?
   -
   -

2. Commandes `kill` et `killall`

La commande `cat` n'est utilisée ici qu'à titre d'exemple, sa fonctionnalité n'est pas importante ici.

1. Dans un premier terminal : *à taper*

   ```
   nsi@ordi$ cat
   ```

2. Nous allons envoyer un signal de terminaison au processus `cat` par le biais de la commande `killall` qui envoie un signal aux processus dont le nom est indiqué.
   Dans un deuxième terminal : *à taper*

   ```
   nsi@ordi$ killall cat
   ```

3. Que constatez-vous ?

4. Dans le premier terminal taper à nouveau `cat`, puis, dans le deuxième terminal, à l'aide de la commande `ps -u`, déterminer le PID du processus ainsi créé : *à compléter*

   ```
   USER     PID %CPU    %MEM    VSZ RSS TTY STAT    START   TIME    COMMAND
                0.0     0.0                 S+              0:00    cat
   ```

   Remarque On peut également utiliser la commande `pgrep` en tapant dans le terminal la commande `pgrep cat`.

5. dans le deuxième terminal, *à taper*
   en utilisant le PID de `cat` que vous avez obtenu :

   ```
   nsi@ordi$ kill 17369
   ```

6. Que constatez-vous ?

3. En résumé

ps : liste les processus actifs attachés au terminal, les processus sont identifiés par leur PID.

top : fournit une vue dynamique temps réel du système en cours d'exécution.

kill : interrompt le processus dont le PID est donné en paramètre.

killall : interrompt **les** processus dont le nom est donné en paramètre.

Tester les différentes options `-u`, `-a`, `-e` et `-f` de la commande `ps`.
Déterminer leur fonctionnement à l'aide de la commande `man ps`.

- `ps -u` :
- `ps -a` :
- `ps -e` :
- `ps -f` :

II. Les processus

1. Définition d'un processus

Définition Un **processus** est une instance d'exécution d'un programme.

- Un processus est décrit par :
  - la mémoire allouée par le système pour l'exécution du programme ;
  - les ressources utilisées par le programme ;
  - les valeurs stockées dans les registres du processeur.
- Un processus possède un numéro unique, le PID (*Process ID*).

Les notions de programmes et de processus sont différentes : le même programme exécuté plusieurs fois générera plusieurs processus.

2. Un exemple

1. Dans un terminal, exécuter la commande suivante :

   ```
   nsi@ordi$ gnome-system-monitor
   ```

2. Dans un deuxième terminal, faire en sorte d'obtenir le PID du processus comme nous l'avons vu précédemment.
3. Dans le deuxième terminal toujours : *à taper*
   En utilisant le PID de `gnome-system-monitor` que vous avez obtenu :

   ```
   nsi@ordi$ top -p 5963
   ```

4. Que constatez-vous comme changements dans l'affichage de la fonction `top` lorsque vous utilisez le programme `gnome-system-monitor` ?

3. Les différents états d'un processus

Définition Les principaux états d'un processus sont les suivants :

Prêt : le processus peut être le prochain a s'exécuter. Il est dans la file des processus qui attendent leur tour.

Élu (actif ou exécution) : les processus est entrain de s'exécuter.

Bloqué : le processus est interrompu et en attente d'un événement externe (entrée/sortie, allocation mémoire, etc.)

On peut rajouter à ces trois états deux états éphémères **nouveaux** et **terminé** qui correspondent respectivement à un processus en cours de création et au système d'exploitation qui désalloue les ressources attribués a un processus qui vient de se finir.

4. Le cycle de vie d'un processus

Après sa création un processus est mis dans l'état prêt. En temps normal un processus variera de entre *prêt*, *élu* et *bloqué*.

Plusieurs processus peuvent être dans l'état *prêt* mais un seul sera placé dans l'état *élu* ; c'est **l'ordonnanceur** (*scheduler*) du système d'exploitation qui est chargé de classer les processus dans une file. C'est lui qui décide quel processus est actif et a pour charge, de manière permanente et à une fréquence élevée, de désactiver le processus actif pour en activer un autre, faisant ainsi fonctionner alternativement les différents processus, donnant l'impression qu'ils fonctionnent en même temps.

Alors qu'il est élu, le processus peut avoir besoin d'attendre une ressource quelconque comme, par exemple, une ressource en mémoire. Il doit alors quitter momentanément le processeur, pour que ce dernier puisse être utilisé à d'autres tâches. Le processus passe donc dans l'état bloqué.

Une fois le processus terminé, il est placé dans l'état *terminé*, le système d'exploitation libère alors les ressources qui lui sont allouées. Notons que quel que soit l'état du processus, il peut se terminer de façon anormale (erreur dans un programme, problème matériel, interruption de l'utilisateur, etc.).

Le schéma ci-dessous résume le cycle de vie d'un processus. À chaque flèche du schéma, ajouter le numéro correspondant parmi ceux donnés ci-dessous (certains numéros peuvent apparaître plusieurs fois) :

1. *mise en exécution par l'ordonnanceur ;*
2. *interruption par l'ordonnanceur ;*
3. *terminaison normale ;*
4. *terminaison anormale ;*
5. *attente d'une ressource ;*
6. *obtention de la ressource.*