Introduction aux systèmes d'exploitation (IS1)

TP 4 – Manipulation de processus

On appelle *processus* un objet dynamique correspondant à l'exécution d'un programme ou d'une commande Unix. Cet objet regroupe plusieurs informations, en particulier l'état d'avancement de chaque programme, l'ensemble des données qui lui sont propres, ainsi que d'autres informations sur son contexte d'exécution.

Connaître les processus en cours d'exécution

Tout comme un fichier, un processus possède des caractéristiques qui permettent (notamment) au système de l'identifier. Parmi ces caractéristiques, on trouve en particulier :

- son numéro d'identification (PID ou *process identifier*), qui est un nombre entier positif ;
- l'identifiant du processus qui lui a donné naissance, ou processus parent, appelé PPID (pour *parent PID*) ;
- son propriétaire, en général (mais pas toujours) l'utilisateur qui l'a lancé ;
- éventuellement le terminal dont il dépend (s'il existe) ;
- son répertoire courant,
- sa priorité de travail,
- son temps d'exécution,
- etc...

Un des outils permettant d'afficher la liste des processus est la commande `ps`. Sans argument, elle affiche le statut des processus en cours, et quelques-unes de leurs caractéristiques.

Exercice 1 – *Lister les processus :* `ps`.

1. Lancez quelques programmes depuis l'interface graphique XFCE, puis lancez les utilitaires `xcalc` et `emacs` depuis votre terminal :

```
xcalc &
emacs &
```

(on rappelle que le caractère `&` permet de reprendre la main immédiatement dans le terminal).

2. Sans fermer les programmes que vous venez de lancer, testez la commande `ps` depuis le terminal. Lancez un second terminal et recommencez. Décrivez les changements. Puis fermez le second terminal.

Correction. Les processus sont rattachés à des terminaux différents. ps n'affiche par défaut que les processus du terminal actuel ("pts/0...").

3. L'option `-l` de `ps` permet d'afficher plus de détails sur les processus. Testez cette option et repérez les caractéristiques précédemment évoquées. Détaillez, en vous aidant, au besoin du manuel, la signification des colonnes `UID, PRI, TT, VSZ, TIME`.

Correction. Les propriétés sont UID : user id ; PRI : priorité (une chiffre élevé signifie une priorité plus basse) ; TT : terminal auquel le processus est rattaché ; VSZ : mémoire virtuelle ; TIME : temps processeur cumulé. (Attention : le format change entre linux, bsd...)

4. Comment peut-on afficher la liste de tous les processus du système (notamment ceux dont vous n'êtes pas le propriétaire, et ceux qui ne dépendent pas d'un terminal) ?

   Correction. Pour voir tous les processus : "ps -ax" (bsd), "ps -e" ou "ps ax" (linux).

5. Comment peut-on obtenir la liste des processus dont le propriétaire est le *super utilisateur* ?

   Correction. "ps -u ou -v" "ps -U root" (bsd) ; idem sans tiret sous linux.

Contrôler l'exécution d'un processus

Les programmes ne fonctionnent malheureusement pas toujours comme prévu. Un garant important de la stabilité du système est donc de pouvoir mettre fin à l'exécution de processus devenus instables ou ne répondant plus. Il existe plusieurs méthodes pour mettre fin à des processus récalcitrants.

Exercice 2 – *Signaux.*

La commande `kill` permet d'envoyer différents types de signaux à un processus dont on connaît l'identifiant (PID). Malgré son nom, elle ne sert pas *seulement* à "tuer" un processus. Il existe de nombreux signaux différents dont vous trouverez la signification dans le `man`. Les signaux les plus courants sont `SIGTERM`, `SIGKILL` qui servent à terminer un processus ainsi que `SIGSTOP` et `SIGCONT` qui servent à arréter provisoirement l'execution d'un processus puis la reprendre là où il en était.

1. Comment peut-on obtenir la liste de tous les signaux signaux que l'on peut envoyer aux processus ? Quels sont les numéros correspondant aux signaux `SIGTERM`, `SIGKILL`, `SIGSTOP` et `SIGCONT` ?

   Correction. "kill -l"

2. La syntaxe d'envoi d'un signal est : `kill` *-signal PID* où *signal* est un numéro ou un nom de signal (le signal par défaut est `SIGTERM`).
   Lorsque le processus est lancé avec `&` pour reprendre la main immédiatement, le shell donne le PID :

   ```
   $ xcalc &
   [3] 31926   $
   ```

   où 31926 est le PID. Le PID d'un processus peut aussi être obtenu avec la commande `ps`. Testez les signaux mentionnés plus haut sur des processus `xcalc` lancés de cette façon, et décrivez le résultat.

3. Lancez un second terminal, puis testez les signaux `SIGTERM` et `SIGKILL` sur le processus correspondant à ce nouveau terminal. Que constatez-vous ? Qu'en déduisez-vous sur le fonctionnement des signaux `SIGTERM` et `SIGKILL` ?

   Correction. SIGTERM ne ferme pas le terminal.

4. Repérez parmi les processus actifs un processus dont vous n'êtes pas propriétaire, et tentez de le stopper à l'aide du signal `SIGSTOP`. Que se passe-t-il et qu'en déduisez-vous ?

Gérer les tâches dans le shell : *jobs*

Lorsque des processus sont lancés depuis un terminal, certains shells modernes et en particulier celui que vous utilisez (`bash`), fournissent un ensemble de mécanismes pour gérer leur exécution. Dans ce contexte, on parle de tâches (*jobs*).

Exercice 3 – *Avant et arrière-plan.*

1. Depuis un terminal, lancez un processus `xcalc` et un processus `xeyes` **avec un `&` à la fin**. Les fenêtres correspondantes s'affichent, et vous ne perdez pas la main sur le terminal (l'invite réapparaît immédiatement) : on dit que le processus que vous venez de lancer est à l'*arrière-plan* du shell (*background job*).
2. Depuis un terminal, lancez un processus `emacs` **sans le `&` final**. Emacs fonctionne mais vous perdez la main sur le terminal (l'invite ne réapparaît pas). On dit que le processus que vous venez de lancer est à l'*avant-plan* du shell (*foreground job*).
3. Depuis le terminal, pressez la combinaison de touche `Ctrl-Z`, aussi appelée commande de suspension (*suspend*). Quel est le résultat ? Pouvez-vous encore utiliser Emacs ? On dit que le processus `emacs` est *suspendu*.
4. La commande `Ctrl-Z` correspond à l'envoi d'un signal via la commande `kill`. Pouvez-vous deviner quel est ce signal à partir de la liste des signaux ? Vérifiez votre réponse par l'expérience.
5. En utilisant `kill`, envoyez un signal à un processus suspendu comme dans la question précédente, qui lui fait reprendre son exécution (à vous de trouver le bon).

   Pour simplifier la manipulation de plusieurs processus dépendant d'un même shell, il leur est attribué un numéro de tâche (*job number*) propre au shell qui les contrôle (différent en particulier du PID).
   La commande `jobs` permet d'afficher une liste de ces processus triée par numéro de job, ainsi que leur état actuel (suspendu ou en cours). Un signe "+" marque le job courant, un "-" le job précédent.
6. Testez cette commande, et comparez sa sortie à celle de `ps` sans argument.
7. Exécutez `xeyes &` puis `glxgears &` puis `jobs`. Que constatez-vous sur les numéros de job associés aux programmes ?

   Nous sommes maintenant capables de lancer des processus à l'avant ou à l'arrière-plan et de suspendre des processus.
   Deux commandes supplémentaires permettent de ramener un processus à l'avant-plan (`fg`) ou de faire reprendre son exécution à l'arrière-plan à un processus interrompu (`bg`). Sans argument, ces commandes s'appliquent au processus courant (cf. question précédente), elles peuvent aussi être suivies d'un argument de la forme `%n`, où `n` est un numéro de job.
8. Testez les commandes `fg` et `bg` pour faire successivement passer à l'avant-plan et à l'arrière-plan les jobs que vous avez lancés. Contrôlez l'état de vos processus avec la commande `jobs`.
9. A quoi est équivalente la séquence : `xcalc` suivi de `Ctrl-Z` suivi de `bg` ?
10. Terminez chacun de vos jobs en les ramenant à l'avant-plan et en pressant la combinaison de touches `Ctrl-C`. A quel signal correspond ce raccourci ?