Processus (1)

1 La notion de processus Unix

Dans cette partie vous vous familiariserez avec la notion de **processus** sous Unix. Celle-ci est essentielle, pour comprendre le mécanisme fondamental de fonctionnement de ce système d'exploitation. Notamment vous verrez que sous Unix toute commande est rattachée à un processus. Vous vous familiariserez aussi avec les **mécanismes de redirection** et vous verrez que ces mécanismes sont étroitement liés aux processus.

1.1 Généralités

Tout d'abord, vous ne devez pas oublier que le système Unix est **multi-utilisateurs**[1] et **multi-tâches** (*i.e.* vous pouvez lancer sous une même session de travail plusieurs tâches qui s'exécuteront *simultanément*[2]). Cela signifie que, sous le même système, et à un instant donné plusieurs utilisateurs peuvent lancer des programmes exécutables différents ou même identiques. Dans ce cas, il faut bien que le système puisse faire la différence entre tous ces programmes. C'est pour cela qu'a été introduite la notion de processus.

1.1 Exercice Sur quel processeur (i.e., de quelle machine) s'exécutent les commandes que vous invoquez dans votre `xterm` (utilisez la commande `uname`) ?.

1.2 Définition d'un processus

Désormais plutôt que de parler de *tâche* qui est un terme un peu flou, nous parlerons de *processus* car ce terme est mieux défini.

Définition : Un programme est un fichier exécutable, et un **processus** est une occurrence d'un programme en exécution.

Les processus s'exécutent indépendamment les uns des autres. Pour parvenir à cette caractéristique, le noyau[3] du système attribue un identificateur à chaque processus. Cet identificateur de processus, un entier, est appelé **pid** pour « process identifier ». Les processus ont en quelque sorte une vie ; ils naissent (lorsque vous lancez un programme ou une commande), ils vivent (lorsqu'ils s'exécutent) et meurent (lorsqu'ils ont terminé leur exécution).

À leur naissance, chaque processus se voit attribuer par le système un **pid** distinct. Ainsi, à un instant donné, tous les processus vivants ont un **pid** différent, et il n'y a pas de conflit pour le système qui est alors capable de différencier tous les processus *via* leur identificateur[4]. Enfin, lorsqu'un processus meurt, son **pid** redevient disponible et pourra être réattribué par le système à un nouveau processus.

!!! Dans la suite du TD, n'hésitez pas à utiliser l'aide en ligne : `man` !!!

1. C'est pour cela que vous devez vous connecter *via* un **login** et un **password**.
2. En fait, le terme *simultanément* est un abus de langage, car très souvent une machine Unix ne possède qu'un seul processeur de calcul, ce qui implique que les différentes tâches qu'il doit exécuter s'effectuent en réalité en temps partagé.
3. « Kernel » en anglais.
4. Même si deux processus sont deux occurrences d'un même programme, ils ont des **pid** différents.

2 Première vision des processus

Dans cette partie, vous allez commencer à manipuler les processus à travers divers exemples. Les commandes à utiliser pour répondre aux questions suivantes sont : **ps, kill, jobs, fg, bg**.

2.1 Exercice Lancez la commande `xload`. Avez-vous toujours le contrôle dans la fenêtre `xterm` depuis laquelle vous avez lancé `xload` ?

2.2 Exercice Testez la commande `Ctrl-z` dans votre fenêtre xterm. Que se passe-t-il ?

2.3 Exercice Testez la commande `jobs`. Qu'indique-t-elle ?

2.4 Exercice Testez la commande `fg`. Interrompez à nouveau le processus.

2.5 Exercice Testez la commande `bg`. Que se passe-t-il ? Avez-vous toujours le contrôle dans la fenêtre xterm ?

2.6 Exercice Lancez à nouveau la commande `xload`. Interrompez-la.

2.7 Exercice Qu'indique maintenant la commande `jobs` ?

2.8 Exercice Que provoque la commande `kill -9 %1` ?

2.9 Exercice Testez la commande `ps`. Avez-vous trouvé les **pid** de vos processus ?

2.10 Exercice Testez la commande `ps -f`. Quelles sont les différences par rapport à la commande précédente ?

2.11 Exercice Quel est le numéro du processus `xload` restant ?

2.12 Exercice Testez la commande `ps -af`.

2.13 Exercice Essayez de tuer (par la commande `kill -9 pid`) un processus d'un autre utilisateur. Que se passe-t-il ?

2.14 Exercice Tuez votre processus `xload` restant en utilisant cette fois-ci son numéro de processus (**pid**).

2.15 Exercice Lancez la commande `xload &`. Avez-vous toujours le contrôle dans la fenêtre xterm ?

2.16 Exercice Faites passer le processus en **avant-plan** par la commande `fg`.

2.17 Exercice Faites à nouveau passer le processus en **arrière-plan** avec la commande `bg`.