**TP0 : Mise à niveau Shell UNIX**

**Les processus**

Commandes utilisées: ps, &, jobs, fg, bg, kill, trap

Pages du support de cours : p16 et 17

**Objectif : approfondir la notion de processus, les différents modes de lancement ainsi que les commandes qui permettent d’envoyer des signaux aux processus.**

## 1. Notions théoriques autour des processus

### 1.1. Définition

Un processus est l'image dynamique d'un programme en cours d'exécution. Un processus représente donc le programme pendant qu'il est exécuté sur la machine.

### 1.2. Les processus Unix

Sous UNIX tous les travaux utilisateurs sont réalisés par des processus. A chaque processus est associé un certain nombre de caractéristiques : le numéro d’identification de son utilisateur (UID), son numéro d’identification (PID), le numéro de son processus père qui l’a crée (PPID),…

Trois fichiers sont automatiquement ouverts par chaque processus lors de son démarrage : l'entrée standard (stdin) dans laquelle le processus va lire les données qu’il doit utiliser ; la sortie standard(stdout) dans laquelle il produit ses résultats et la sortie d’erreur où il envoie ses messages d’erreur.Ces fichiers sont respectivement rattachés aux descripteurs 0, 1 et 2.

Dans le cas des processus interactifs, stdin (descripteur 0) est généralement associé au clavier, stdout(descripteur 1) et stderr (descripteur 2) sont associés à l’écran (ou à la fenêtre dans les environnements graphiques).

A sa terminaison, le shell retourne un code (entier) qui indique la manière dont il s’est déroulé. Parconvention, la valeur 0 indique une terminaison normale (sans erreur). Toute valeur différente de 0 est assimilée à un code d’erreur.

### 1.3. Quelques commandes

#### ▪ Visualiser

La commande 'ps' affiche des informations sur les processus en cours d’exécution pour le compte de l'utilisateur, ou dont la terminaison n’a pas encore été prise en compte par le système.  
ps -ef ou ps -auwx : visualise tous les processus du système.

ps [options] [pid…]

Quelques options : Attention les options du ps varient beaucoup suivant les shell et les systèmes.

-a : présente également les informations des processus des autres utilisateurs.

-u : présente le nom de l'utilisateur et l'heure de lancement.

-x : affiche également les processus qui n'ont pas de terminal de contrôle (démons).

-e : présente l'environnement à la suite de la ligne de commande exécutée.

-h : n’affiche pas l’entête.

-r : n’affiche que les processus en cours d’exécution.

La commande top : affiche une liste des processus actualisée et classée par ordre de consommation CPU décroissante.

#### ▪ Envoyer un signal

Envoyer un signal à un processus, c’est l’avertir que « quelque chose » s’est produit. La « nature » de ce qui s’est produit est déterminée par le signal envoyé.

Sous UNIX, la « nature » de l’événement à signaler est matérialisée par un entier. Pour préserver la compatibilité des « différents » UNIX, des constantes symboliques sont définies.

kill [–num] <pid> [<pid>…]

La commande kill permet d’envoyer un signal aux processus. Si on ne précise pas le numéro du signal avec -num, kill envoie le signal 15 qui entraîne souvent la terminaison du processus.

On détruit toujours un processus en lui envoyant le signal 9. Il est possible d’obtenir la liste des signaux disponibles sur le système en utilisant

'kill –l'.

### 1.4. Manipulation de processus

#### ▪ Arrêt / pause

Lorsque l’utilisateur veut arrêter (irrévocablement) l’exécution d’un processus interactif, il lui suffit d'envoyer le caractère 'CRTL-C' sur l’entrée standard du processus.

Pour suspendre temporairement l’exécution d’un processus interactif, il suffit de lui « envoyer » le caractère 'CRTL-Z' sur son entrée standard. La reprise s’effectuera en utilisant les commandes bg ou fg (pour une reprise en arrière plan ou au premier plan).

#### ▪ Exécution en arrière plan (background)

commande & //*(& à la fin de la commande)*

'&' ne s’applique qu’à une seule commande. Lorsque l’utilisateur lance une commande en arrière-plan, le Shell rend la main à l’utilisateur qui peut lancer d’autres commandes. L’utilisateur n’est pas informé lorsque la commande lancée en background est terminée. Il peut néanmoins s’en assurer par un ps.

Dans ce cas de figure, le shell et le processus généré continuent de partager la même sortie standard. En clair, si l’utilisateur continue d’exécuter des lignes de commandes, leurs affichages pourront se «mélanger » avec ceux de la commande lancée en arrière plan. La tâche de fond ne peut plus recevoir de données par l’intermédiaire de l’entrée standard.

A tout instant, à condition de « garder la main », l’utilisateur peut identifier les processus s’exécutant en fond au travers de la commande 'jobs'. Cette commande renvoie un numéro local identifiant chaque processus. Ce numéro n’est valable que dans le terminal de lancement du processus et est différent du PID. Les commandes 'bg' ou 'fg' lui permettent alors de mettre en fond(reps. au premier plan) la tâche de son choix.

#### 

## 2. Exercices

**I. Visualisation des processus :**

*Question 1 :*

Visualisez tous les processus du système. Quel est le PID du processus qui est le père de bon nombre de processus système (processus de l’utilisateur root) ?

Pour afficher tous les processus, la commande est : ps -ef.

On obtient donc la liste des processus et, pour ma machine, le PID du processus de l’utilisateur root est 2.

*Question 2 :*

Visualisez tous les processus vous appartenant à l’aide de la commande ps (pensez à faire un man ps) ; donnez leurs PID.

La commande est ps -u [nom utilisateur].

**II. Modes de lancement des processus**

Il existe deux modes de lancement des processus :

* • le mode background : le processus rend la main une fois lancé (processus &)
* • le mode foreground : le processus garde la main pendant son exécution, il utilise le clavier et l’écran (processus ou ./processus si la variable PATH n’inclut pas le répertoire courant).

*Question 3 :*

Créez un petit script qui effectue une boucle infinie.

Exécutez ce script en foreground ; pouvez vous taper des commandes dans le terminal ?

Le terminal ne recoit plus de commande une fois xeyes lancé.

*Question 4 :*

Exécutez ce script de façon à pouvoir taper des commandes dans le terminal. Terminez le processus en utilisant son identificateur local.

Pour pouvoir exécuter des commandes pendant l’exécution d’un processus, i faut le lancer avec un & : xeyes &.

Il est donc possible de lancer des commandes avec xeyes de lancer.

Pour le terminer, nous exécutons la commande kill %[idLocal].

*Question 5 :*

Exécutez ce script en foreground puis passez-le en background puis repassez le en foreground. Terminez-le.

Pour mettre un process en foreground, on utilise fg %[id]. Pour background c’est bg.

*Question 6 :*

Lancez deux exécutions du script en background et terminez-les en utilisant leurs identifiants locaux.

*Question 7 :*

Lancez deux exécutions en background et passez les alternativement en foreground. Terminez les ensuite.

Pour pouvoir passer les deux en foreground, il faut mettre en pause le premier processus.

**III. Envoi de signal à un processus**

En utilisant les combinaisons de touches ctrl^C, ctrl^Z, on envoie un signal à un processus. Il est également possible d’envoyer des signaux de façon explicite à l’aide de la commande kill.

kill -l affiche la liste des signaux qu’il est possible d’envoyer à un processus.

*Question 8 :*

Dans un terminal, exécutez le script faisant une boucle infinie en foreground. Tuez-le depuis un second terminal.

Pour le terminer a partir d’un second terminal, il faut utiliser ps -u pour obtenir tous nos processus et trouver le pid de xeyes. Ensuite utiliser kill [PID].

*Question 9 :*

Dans un terminal, exécutez le script faisant une boucle infinie en foreground. Depuis un second terminal, suspendez son exécution et passez-le en background (cela revient à faire l’équivalent d’un Ctrl^Z + bg avec les signaux explicites de la commande kill).

Pour mettre en pause : kill -s STOP [PID] et pour mettre en background : kill -s CONT [PID].

*Question 10 :*

Tuez un processus appartenant à un autre utilisateur. Quelle est la réponse du système et pourquoi ?

Le terminal répond :  “operation not permitted”.

*Question 11 :*

Quelle commande (et donc quel signal) devez vous utiliser pour tuer le shell s’exécutant dans votre terminal ?

Nous devons déjà obtenir le PID du terminal avec ps. Ensuite, on utilise : kill -s ABRT [PID].

**IV. Arborescence des processus**

*Question 12 :*

Nous supposons que vous avez un terminal A ouvert. Dans A, créer un nouveau terminal avec la commande xterm -fn 7\*14 (crée un terminal avec la police de caractères 7\*14). Nous appelons ce nouveau terminal B. Dans B, créer un nouveau terminal avec la commande xterm -fn 8\*16. Nous appelons ce nouveau terminal C. (voir schéma de création des terminaux) :

Terminal A Terminal B Terminal C

Quel est le nombre de processus créés ?

6

Donnez la filiation des différents processus.

zshC a comme père xtermC, xterm C a comme père zshB, zshB a comme père xtermB, xtermB a comme père zshA et zshA a comme père gone-terminal.

Terminez le shell s’exécutant dans le terminal B.

Combien de processus se terminent ?

2

Quelle modification de filiation est apportée au terminal C.

Le terminal C est adopté par un autre processus

En déduire la résolution du problème des orphelins sous Unix.