Manipulation de processus

Lorsque vous utilisez un interpréteur de commandes (shell) UNIX, chaque commande est exécutée dans un nouveau processus. Par défaut les processus démarrés par le langage de commande le sont en **avant-plan** : vous ne pouvez exécuter de nouvelles commandes (on dit que vous *n'avez pas la main sur le terminal*) tant que le processus démarré n'est pas terminé. Il est possible de démarrer une commande via un processus en **arrière-plan** (ou *tâche de fond*). Pour cela il suffit de faire suivre la commande souhaitée par le caractère «&».
De ce fait les langages de commande permettent de démarrer plusieurs commandes en **séquence** via l'utilisation du caractère de séparation «;» ou en **paralllèle** via l'utilisation du caractère «&».
De plus il est possible de grouper le démarrage des processus d'une suite de commande en les entourant de parenthèses. Par exemple, si un processus à créer en arrière plan doit comporter plusieurs commandes successives il suffit de grouper les commandes avec des parenthèses, de séparer les commandes souhaitées par des points virgule et de placer le caractère «&» après la dernière parenthèse fermante comme dans l'exemple suivant :

```
( commande1 ; commande2 ) &
```

Chaque processus est repéré par un identifiant système unique (`PID`). La commande `ps` permet de lister les processus en cours d'exécution sur le système. Cette commande comporte de très nombreuses options permettant, entre autre, de modifier les informations affichées ou la forme des affichages. La page du manuel vous renseignera en détail sur ces options.
Dans certains shells (dont celui que vous utilisez et qui se nomme `bash`) il existe une notion connexe aux identifiants de processus : les identifiants de *tâches* ou de *jobs*. Un job est un processus attaché à un terminal en cours d'utilisation. Dans les lignes de commandes on représentera les jobs par le caractère «%» suivi du numéro identifiant la tâche. La commande `jobs` permet de lister les tâches en cours et leur état. Pour chaque démarrage en arrière-plan le langage de commande signifie à l'utilisateur le numéro de job, suivi du numéro de processus (`PID`).
En `bash` si un processus est exécuté en avant plan, il est possible de le suspendre en frappant simultanément sur les touches Ctrl et Z . Un processus suspendu peut être redémarré en avant-plan via la commande `fg` (pour *foreground*), ou en arrière plan avec la commande `bg` (pour *background*).
La commande `kill` permet d'envoyer des signaux aux processus. Cette commande demande au moins deux paramètres. Le premier permet de préciser le signal à envoyer au processus. Le second correspond a l'identification du processus à signaler (via un `PID` ou numéro de job). Trois signaux sont particulièrement utile : `KILL` qui demande la destruction sans condition d'un processus, `STOP` qui en demande la suspension, et `CONT` qui demande le redémarrage d'un processus préalablement stoppé.

Exercice 1 : Manipulation de base

☞ **Avant de commencer les exercices lisez les pages du manuel de `kill` et de `ps`.**

☞ **De façon à ce que le shell vous signifie la fin d'un processus sans aucune attente exécutez la commande «`set -b`» dans votre terminal.**

1. Démarrez la commande `xeyes`.
2. Essayez maintenant de démarrer une nouvelle fois la commande `xeyes`. Est-ce possible ?
3. Stoppez la commande démarrée en frappant simultanément les touches Ctrl et Z .
4. Redémarrez la commande stoppée en arrière-plan.
5. Démarrez un nouveau processus en arrière plan exécutant la commande `xeyes`.
6. Exécutez la ligne de commande suivante :

   ```
   xeyes & xcalc & xlogo & xclock & xload & xterm &
   ```

7. En utilisant les messages que l'exécution de cette commande a produit ainsi que la commande `ps` associez à chaque numéro de *jobs* démarré le nom du programme qu'il exécute.
8. Après avoir lu la page du manuel de la commande `ps`, déterminez graphiquement la hiérarchie des processus en cours. Quelle commande avez-vous utilisée ?
9. Arrêtez maintenant complètement les processus exécutant les commandes `xlogo` et `xload` en utilisant uniquement la commande `kill` que vous appliquez à des numéros de jobs (numéros généralement inférieur à 10) c'est-à-dire sans utiliser de `PID`.
10. Stoppez maintenant les processus exécutant les commandes `xclock` et `xcalc` en utilisant leur numéro de processus (`PID`). Pour stopper les processus vous devez envoyer le signal `STOP` aux processus concernés via la commande `kill`.

11. Essayez de vous servir de la calculatrice. Que se passe-t-il ?
12. Utilisez la commande `jobs` pour faire le bilan des processus actuellement attachés à votre terminal (les *jobs*).
13. Combien y a-t-il de processus stoppé ? en cours d'exécution ? Lesquels ?
14. Faites redémarrer le processus exécutant `xclock` via l'envoi d'un signal (`CONT`) avec la commande `kill` à celui-ci.
15. Le processus redémarre-t-il en arrière plan ou en avant-plan ?
16. Faites redémarrer le processus exécutant `xcalc` en avant plan via l'appel de la commande `fg`. Comment avez-vous du spécifier le processus : via son `PID` ou son numéro de job ?
17. Arrêtez le processus exécutant la commande `xcalc` sans utiliser la souris. Comment avez-vous fait ?
18. Faites passer le processus exécutant la commande `xclock` en avant plan. Comment avez-vous fait ?
19. Stoppez-le, puis faites le redémarrer via la commande `bg`. Comment avez-vous du spécifier le processus : via son `PID` ou son numéro de job ?
20. Arrêtez tous les processus démarrés lors de cet exercice en une seule ligne de commande.

Exercice 2 : Manipulation des processus et de leur état

Q 1. Dans cet exercice vous allez utiliser une commande particulière nommée `sleep`. À l'aide du manuel informez-vous du comportement et de l'utilisation de cette commande.
Q 2. Essayez et comparez les commandes suivantes :

```
sleep 5 ; echo A
echo A ; sleep 5
sleep 5 & echo A
echo A & sleep 5
(echo A ; sleep 5) &
(sleep 5 ; echo A) &
```

Q 3. Remplissez le tableau suivant, de façon à représenter *graphiquement* le comportement de chaque commande (par convention le caractère $ repésentera l'apparition du prompt) :

Commandes	$\mathbf{t = 0s}$	$\mathbf{t = 5s}$	$\mathbf{t = 10s}$
`echo A ; sleep 5 ; echo B`	A	B$	
`echo A ; sleep 5 & echo B`			
`(echo A ; sleep 5 ) & echo B`			
`echo A ; (sleep 5 & echo B)`			
`echo A ; (sleep 5 ; echo B) &`			
`sleep 5 & echo A ; ( sleep 5 ; echo B)`			
`sleep 5 & echo A & ( sleep 5 ; echo B)`			
`sleep 5 & echo A & ( sleep 5 & echo B) &`			
`sleep 5 & echo A & ( sleep 5 ; echo B) &`			

Q 4. Exécutez la commande suivante, puis uniquement à l'aide des facilités de manipulations des jobs faites en sorte que le message `fini2` apparaissent avant le message `fini1` :

```
(sleep 60 ; echo fini1) & (sleep 120 ; echo fini2)
```

Q 5. Quelle suite de commandes ou actions avez-vous utilisée ?

Redirections

Les processus possèdent au moins trois descripteurs de fichiers qui peuvent être redirigés :
- **0** : l'**entrée standard**, par défaut c'est le terminal en mode lecture
- **1** : la **sortie standard**, par défaut c'est le terminal en mode écriture
- **2** : la **sortie d'erreur**, par défaut c'est le terminal en mode écriture

Pour rediriger ces fichiers il suffit de compléter les commandes par une syntaxe particulière. Pour mémoire le tableau 1 rappelle ces différentes syntaxes de redirection. Une même ligne de commande peut contenir plusieurs redirections qui seront évaluées en séquence.
Les terminaux correspondent, comme quasiment tous les éléments sous UNIX, à des fichiers. La commande `tty` permet d'obtenir le chemin absolu du fichier correspondant au terminal dans lequel elle est exécutée.

n<*fichier*	redirige en lecture le descripteur n sur *fichier*.
n>*fichier*	redirige en écriture le descripteur n sur *fichier*.
n<<*marque*	redirige en lecture les lignes suivantes sur le descripteur n jusqu'à ce que la *marque* soit lue.
n>>*fichier*	redirige le descripteur n à la fin de *fichier* sans détruire les données préalablement contenues dans ce fichier.
n<&m	copie le nom du fichier du descripteur m sur le descripteur n en lecture, ainsi n et m seront dirigés vers le même fichier.
n>&m	copie le nom du fichier du descripteur m sur le descripteur n en écriture.

TABLE 1 – La syntaxe des redirections sous `bash`

Exercice 3 : Les redirections d'entrées/sorties

Q 1. Sans utiliser la commande `vi`, ni un autre éditeur de texte, mais en utilisant la commande `cat` et l'opérateur > créez les fichiers `fich1` et `fich2` suivant :

— `fich1`

— `fich2`

Q 2. Lisez la page du manuel de `xterm` de façon à trouver comment vous pouvez lors du lancement de la commande spécifier un titre au terminal que vous démarrez. Ce titre devra être présent dans la barre de titre de la fenêtre démarrée. Essayez par exemple de démarrer un terminal X ayant pour titre le mot `toto`.

Q 3. Démarrez 3 terminaux graphiques en utilisant la commande `xterm` et les possibilités de démarrage en arrière-plan. Dans la suite de l'énoncé on désignera ces 3 terminaux par les alias `A`, `B` et `C`. Le terminal `A` correspondra toujours à celui dans lequel vous entrez vos commandes. Pour vous simplifier le travail donner comme titre à vos terminaux les alias spécifiés.

Q 4.

1. Identifiez pour chacun des terminaux le fichier associé.
2. Afficher le contenu du fichier `fich1` dans le terminal `B`
3. Toujours en utilisant la commande `cat`, mais cette fois après avoir lu le manuel, créez le fichier `fich3` constitué de la concaténation du contenu des fichiers `fich1` et `fich2`.
4. Lancez la commande `cat fich1 fich-inexistant`.
5. Lancez la commande précédente en redirigeant la sortie standard dans le terminal `B` et la sortie d'erreur dans `C`.
6. Comment peut-on rediriger la sortie standard sur la sortie d'erreur ?
7. En utilisant la réponse à la question précédente refaites la question 5 en redirigeant le tout vers le terminal `B`.

Q 5. Faites en sorte que tout ce que ce qui sera saisi dans le terminal `A` s'affiche dans le terminal `C`.

Q 6. Faites en sorte que tout ce que ce qui sera saisi dans le terminal terminal `B` s'affiche dans le terminal `C` mais après avoir été mis en majuscule. Une étude attentive de la page du manuel de la commande `tr` devrait vous aider.

Q 7. Faites en sorte que toutes les lignes qui seront saisies dans le terminal `B` s'affiche dans le terminal `C` uniquement si elles comportent le mot `toto` au moins une fois. Une étude attentive de la page du manuel de la commande `grep` devrait vous aider.