ACT1 Jouons avec les processus !

1. Visualiser les processus

1.1. Lancement de processus : avant-plan et arrière-plan

1) Lancer quelques programmes depuis l'interface graphique XFCE (le bureau de XUBUNTU)

2) lancer ensuite le terminal, et exécuter les utilitaires *xclock* ou *xcalc*.
Noter que la fenêtre de l'utilitaire s'ouvre, mais que l'on n'a plus la main dans le terminal : l'utilitaire s'exécute ***en avant-plan***; le retour au terminal ne se fait que lors de l'arrêt du processus, c'est à dire ici la fermeture de la fenêtre de l'utilitaire (à ce moment, le PID du processus fermé s'affiche alors dans le Terminal).
3) fermer la fenêtre de l'utilitaire, puis lancer les deux programmes successivement en faisant maintenant suivre leur nom du caractère *&* :

```
$ xclock &
$ xcalc &
```

A présent, les deux utilitaires « tournent » toujours, mais on a également toujours la main dans le terminal : les processus s'exécutent maintenant ***en arrière-plan***, et le terminal n'attend pas la fin de leur exécution pour rendre la main à l'utilisateur.

→ ***Laisser tourner les utilitaires et ne pas fermer le terminal pour la suite.***

1.2. Arborescence des processus : commande *pstree*

4) Dans le terminal, tester la commande pstree qui affiche l'arborescence des processus.
5) Quel est le processus à la « racine » de cette arborescence ?
6) Repérer dans l'arborescence les processus correspondant aux programmes que vous venez de lancer.
7) Identifier tous les ancêtres des processus correspondant aux utilitaires *xclock* et *xcalc*.
8) en consultant l'aide de la commande (*pstree -help*), déterminer l'option qui permet d'afficher le propriétaire de chaque branche de processus, et celle qui affiche le PID de chaque processus.

1.3. Informations sur les processus : commandes *ps* et *top*

Des informations plus précises sur les processus peuvent être obtenues à l'aide de la commande *ps.*

9) Sans fermer les programmes lancés précédemment, lancer *ps* sans argument depuis le même terminal. Lancer un second terminal et recommencer.

10) Que constate-t-on ? Fermer le second terminal.
11) L'option *-f* de *ps* permet d'afficher plus de détails sur les processus. Tester cette option et interpréter les informations données. Consulter au besoin le manuel de la commande pour connaître la signification de chaque colonne.
12) Utiliser l'aide pour trouver comment afficher la liste de tous les processus du système.

Pour un affichage dynamique et interactif de l'ensemble des processus, un troisième outil intéressant est la commande *top*. C'est en quelque sorte l'équivalent Unix du « gestionnaire de

tâches » de Windows. C'est cette commande qui est la plus utile pour gérer les processus et intervenir sur eux, notamment en cas de problème.

13) Lancer la commande *top*. Dans quel ordre les processus affichés sont-ils classés ?
14) Quand *top* est actif, un appui sur la touche « ***u*** » permet de sélectionner uniquement les processus appartenant à un utilisateur donné. Affichez uniquement vos processus.
15) Cet utilitaire offre un grand nombre de fonctionnalités, comme celle de modifier la colonne de tri, sélectionner les colonnes à afficher, etc.
    Explorer la page de manuel et tester certaines de ces options.

2. Contrôler l'exécution d'un processus

Les programmes ne fonctionnent 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.

2.1. Signaux

La commande *kill* permet d'envoyer différents types de signaux à un processus dont on connaît l'identifiant (PID). Malgré son nom, et même si c'est son usage principal, elle ne sert pas seulement à « tuer » un processus. Les signaux les plus courants sont *SIGTERM* et *SIGKILL*, qui servent à terminer un processus. D'autres signaux fréquents sont *SIGSTOP* (suspension) et *SIGCONT* (reprise).

La syntaxe d'envoi d'un signal à un processus est : *kill -signal PID*, où signal est un numéro ou un nom de signal (le signal par défaut est *SIGTERM*).

1) La liste des signaux que l'on peut envoyer aux processus s'obtient grâce à l'option *-l* de *kill*. Repérer les numéros des signaux mentionnés ci-dessus.
2) Teste les signaux mentionnés ci-dessus sur le processus *xclock* préalablement lancé (utilisez l'une des commandes précédentes pour accéder à son PID).
3) Lancer un second terminal. Repérer son ide: intifiant de processus, puis testez les signaux *SIGTERM* et *SIGKILL* sur ce processus. Que constate-t-on ?

Ce comportement illustre le fait que dans certains cas, être « poli » ne suffit pas : *SIGTERM* demande au processus de s'arrêter (ce qu'il peut refuser), tandis que *SIGKILL* demande au système de le « tuer » (utile aussi si un programme ne répond pas...).

4) Repérer parmi les processus actifs un processus dont vous n'êtes pas propriétaire, et tenter de le stopper à l'aide du signal *SIGSTOP*. Que peut-on en déduire ?

2.2. Tuer un processus avec style.

Il existe au moins deux autres techniques pour terminer des processus (sans compter le fait d'appuyer sur le bouton « fermer » dans le cas d'un processus fenêtré).

5) Lancer deux ou trois processus quelconques depuis le terminal ou l'interface graphique.
6) Dans un terminal, lancer la commande *top*.

7) La touche « ***k*** » permet d'indiquer que l'on souhaite terminer un processus. Il est ensuite demandé de désigner un processus par son identificateur, puis de spécifier un signal à lui envoyer. C'est en quelque sorte un « kill interactif ».
   Terminer ainsi les programmes qui viennent d'être lancés.

2.3. Utilisation du clavier

Depuis un terminal, lancez un processus *xclock* et un processus **firefox** avec un *&* final : les fenêtres correspondantes s'affichent, et l'on ne perd pas la main sur le terminal car les processus s'exécutent en arrière-plan. Le PID du processus déclenché s'affiche.

8) Depuis un terminal, lancer un processus **libreoffice** sans le *&* final. **LibreOffice** fonctionne mais on perd la la main sur le terminal (processus en avant-plan).
9) Depuis le terminal, presser la combinaison de touche ***Ctrl-Z***, aussi appelée commande de suspension (*suspend*). Quel est le résultat ? Peut-on encore utiliser **LibreOffice** ? Le processus libreoffice est alors ***suspendu*** : il est toujours en mémoire, mais inactif.

La commande ***Ctrl-Z*** correspond à l'envoi d'un signal *SIGTSTP* aux éventuels processus d'avant-plan. Il est donc aussi possible d'envoyer explicitement ce signal grâce à la commande *kill*.

10) Suspendre le processus **firefox** de cette façon, et constater l'effet sur le programme.
11) Terminer chacun des processus en les ramenant à l'avant-plan avec la commande *fg*, puis en pressant la combinaison de touches ***Ctrl-C*** correspondant à l'envoi du signal *SIGINT*.

Vocabulaire :

- **Thread** ou **processus léger** : exécution d'une suite d'instructions démarrées par un processus.

Deux processus sont l'exécution de deux programmes (par exemple un navigateur web et un traitement de texte), deux threads sont l'exécution concurrente de deux suites d'instructions d'un même processus. Par exemple, pour un navigateur web, il peut y avoir un thread pour afficher la page web et un autre thread qui télécharge un fichier.
La différence fondamentale entre **processus** et **thread** est que les processus ne partagent pas leur mémoire, alors que les threads, issus d'un même processus, peuvent accéder aux variables globales du programme et occupent le même espace en mémoire.

- Exécution **concurrente** : deux processus ou threads s'exécutent de manière concurrente si leur exécution se fait en alternance dans le temps; si cette alternance est rapide, l'exécution donne l'illusion de la simultanéité.

Sur les systèmes mono-processeurs, c'est la seule manière dont les threads et les processus peuvent s'exécuter.

- Exécution **parallèle** : deux processus ou threads s'exécutent de manière parallèle s'ils s'exécutent *réellement* en même temps.

Pour que plusieurs threads ou processus s'exécutent en parallèle, il faut plusieurs processeurs sur la même machine.