POLYTECH ANGERS

TP 3A - GNU/Linux
Processus

Nicolas Delanoue

Exercice 1 (une courte vie)

1. Depuis une console, lancez l'application `mate-calc`. (dans le reste de cet exercice et dans le cas où `mate-calc` n'est pas présente, vous pourrez utiliser `gnome-mines`.)
   (a) Assurez-vous que cette application fonctionne normalement.
   (b) Retournez dans la console : avez-vous la main (essayez de taper la commande `ls` par ex.) ?
   (c) Saisissez la combinaison de touches `Ctrl C`. Que s'est-il passé ?
2. Lancez à nouveau la commande `mate-calc`.
   (a) Lancez un autre terminal puis trouvez, avec la commande `ps aux`, le `pid` du processus correspondant à `mate-calc`.
   (b) Tuez ce processus avec la commande `kill`.
3. Lancez à nouveau la commande `mate-calc`. Lancez un autre terminal puis exécutez la commande `pstree`, quel est le père du processus `mate-calc` ? Fermez la calculatrice "normalement" (en utilisant le bouton de fermeture de la fenêtre). Vérifiez que vous récupérez bien la main dans la console.
4. Cette fois, lancez la commande `mate-calc &`. Vérifiez que l'application est bien active et qu'en plus, vous récupérez immédiatement la main dans la console : on dit que le processus est en *arrière-plan*. Mais que se passe-t-il si vous fermez la console ? Qui était le père de ce processus ?

Exercice 2 (calculatrice $\times$ 2 = ?)

1. Lancez la calculatrice (`mate-calc`) en arrière-plan ; vérifiez que vous avez toujours la main dans la console.
2. Notez le `pid` du processsus que vous venez de créer (il a été donné par le shell quand vous l'avez lancé).
3. Lancez une deuxième instance de la calculatrice et notez son `pid`.
4. Utilisez la commande `ps` pour visualiser les processus que vous avez lancés : vérifiez que vous y retrouvez bien les deux `pid` que vous avez notés.
5. Vous avez donc en mémoire deux *processus* `mate-calc` mais il n'y a qu'un seul *programme* `mate-calc`. Utilisez la commande `which` pour trouver où il est situé sur le disque.
6. Utilisez `kill` pour tuer le 1er processus `mate-calc`.
7. Terminez l'autre processus `mate-calc` de façon normale (*i.e.* depuis son interface graphique).

8. Tapez sur la touche `Enter` dans la console : que voyez-vous apparaître ?

Exercice 3 (Premier-plan et arrière-plan)

1. Minimisez toutes les fenêtres, lancez, depuis la console, `xeyes`.
2. Déplacez votre souris et observez l'observateur,
3. Suspendez-le en utilisant la combinaison de touches `Ctrl Z` : vérifiez que son interface graphique est complètement gelée.
4. Lancez la commande `ps aux | grep xeyes`, dans quel état est le processus `xeyes` ?
5. Faites-le passer en arrière-plan avec la commande `bg` : vérifiez que son interface graphique est à nouveau fonctionnelle.
6. Quelle différence aurions-nous pu observé si nous avions exécuté la commande `fg` ?

Exercice 4 (htop)

1. Installez `htop`, en exéctant la commande suivante dans un terminal :
   `sudo apt install htop`
2. Lancez `firefox` et un terminal,
3. Partagez votre écran de sorte à pouvoir visualiser les deux fenêtres en même temps.
   Indication : la combinaison des touches : touches windows et flèche gauche peut vous aider,
4. tapez la commande `top` dans terminal,
5. Dans `firefox`, regardez une vidéo sur la toile,
6. Est ce que ce `firefox` consomme des ressources ? réitérez si besoin.
7. Depuis `top`, identifiez ce processus trop gourmand !, quittez `top` avec un appui sur la touche "q" et tuez ce processus avec son identifiant.

Exercice 5 (Redirections)

1. Etablissez une commande qui permette de créer un fichier `/home/user/liste_fichiers.txt` contenant les noms des fichiers contenus dans le répertoire `/bin`.
2. Etablissez une commande qui permette d'ajouter les noms des fichiers du répertoire `/lib` au fichier créé à la question précédente.
3. Etablissez une commande qui permette de créer un deuxième fichier `/home/user/liste_fichiers_trie.txt` résultat du tri du premier fichier
   Indication : la commande `sort` permet de trier les lignes d'un fichier.
4. Regardez le contenu du fichier `/home/user/liste_fichiers.txt` avec la commande `cat`.
5. Etablissez une commande qui permette d'ajouter, à l'aide de la commande `cat`, une ligne saisie au clavier à la fin d'un fichier (on créera au préalable un fichier contenant du texte).
6. Créez trois fichiers texte `f1.txt`, `f2.txt` et `f3.txt`, insérez du contenu dans ces fichiers. Etablissez une commande qui permette de concaténer l'ensemble du contenu des fichiers textes en un fichier nommé `tout.txt`.

Exercice 6 (Redirections)

1. Etablissez une commande qui permette de compter le nombre de processus lancés sur la même machine (`ps aux`). On utilisera notamment la commande `wc -l`.

2. Etablissez une commande qui permette de compter le nombre de fichiers contenus dans `/bin` (et non dans les sous-répertoires). On utilisera notamment la commande `wc -l`.
3. Etablissez une commande qui permette d'afficher le contenu de `/bin` en ordre inversé (pensez à la commande `sort` et à l'option `reverse`).
4. Reprendre la question précédente avec un affichage page à page (commande `more`).
5. Quelle est l'objectif de cette commande :
   `find -name "a*"| xargs ls -l`
6. Etablissez une commande qui permette d'effacer les fichiers dont les noms sont contenus dans un fichier texte `fic.txt`. On utilisera notamment la commande `xargs` lit l'entrée standard et renvoie ce qu'elle a lu comme argument de la commande.

Dans les deux prochains exercices, on s'appuie sur la commande `date` pour tester d'autres mécanisme de création de processus. Cette commande `date` permet de générer la date et l'heure. Elle est employée ici car sa sortie change en fonction de l'instant où elle est exécutée ...

Exercice 7 (at)

1. Créez un fichier `testat.sh` contenant la ligne de commande suivante

```
1 date > /tmp/datestamp
```

2. Rendez-le exécutable et procédez à sa future exécution avec `at` :

```
1 $ chmod +x testat.sh
2 $ at now + 1 minute -f testat.sh
```

3. Cette tache va-t-elle s'exécuter ? On pourra vérifier avec `atq`.
4. Vérifiez que cette tache s'est produite avec un `cat /tmp/datestamp`.
5. Quel sera le comportement cette session shell :

```
1 $ at now + 1 minute
2 at> pkill mousepad
3 CTRL-D
4 $ atq
```

Exercice 8 (watch)

1. Lancez deux terminaux.
2. Dans le premier, exécutez la commande `watch "date >> la_date_actuelle.txt"` et attendez.
3. Dans le second terminal, lancez manuellement et successivement la commande `cat la_date_actuelle.txt`.
4. Quel est l'intérêt de la commande `watch` ?
5. Fermez le premier terminal brutalement, le fichier `la_date_actuelle.txt` continue-t-il de se remplir ? pourquoi ?