Université Mohammed Premier Oujda
École Nationale des Sciences Appliquées
Département : Électronique, Informatique et Télécoms
Module : Administration des systèmes
Filières : Génie Informatique
Niveau : GI4

TP10 Administration des Systèmes : Gestion des processus sous un système Linux

Enseignant : Mohammed SABER

Année Universitaire : 2018/2019

Ressources requises

Ressources nécessaires :

1. Un ordinateurs Windows 7 avec un logiciel de virtualisation ;

Consignes pour le TP

1. Suivez les instructions pour chaque énoncé.
2. A la fin de TP, SVP réorganiser votre table :
   - Éteindre toutes les machines.
   - Réorganiser les chaises à ces places avant de sortir.
   - MERCI d'avance.
3. Un rapport de TP individuel est rendu sur la plateforme Moodle à la fin de TP (en format PDF ou DOC).
4. **Chaque étudiant ne respect pas les consignes de TP sera sanctionné.**

Énoncé 1 : Modes d'exécution des processus

1. Se connecter en tant que «`root`» sur une console texte.
2. Récupérer le programme «`memoire.c`» à partir le site.
3. Compiler le programme. (ne pas tenir compte des messages de warning s'il y en avait) «`gcc memoire.c -o memoire.exe`». L'exécutable généré s'appellera «`memoire.exe`».
4. Attribuer le droit d'éxecution pour le «`memoire.exe`». (**Utilisation** : la commande `chmod`).
5. Lancer le sous le nom «`./memoire.exe`» avec un paramètre entier inférieur à **10**. Observez ce que fait le programme.
6. Relancer le programme maintenant en l'interrompant avant sa terminaison par «**Ctrl-C**». Qu'observez-vous ?
7. Relancer le programme maintenant en lui donnant **200** comme paramètre et interrompez-le avant sa terminaison par «**Ctrl-Z**». Qu'observez-vous ?
8. Refaites cette opération une, deux, trois, quatre, etc. fois de plus jusqu'à... ce que l'on ne puisse plus.
9. Comprenez-vous maintenant la différence fondamentale entre «**Ctrl-C**» et «**Ctrl-Z**» ?
10. Pour se sortir de tous ses programmes qui ont saturé la machine, faites «`jobs`». Qu'observez-vous ?
11. Tuez tous les **jobs** qui sont suspendus. Comment procédez-vous ?
12. Dupliquez le programme compilé précédemment en lui donnant un autre nom. Par exemple «`memory.exe`».
13. Se connecter en tant que «`root`» sur deux consoles texte.
14. Dans la première fenêtre, lancez «`memoire.exe` 1» et suspendez-le par «**Ctrl-Z**».
15. Dans la deuxième fenêtre, lancez «`./memory.exe` 1» et suspendez-le par «**Ctrl-Z**».
16. Lancez encore un autre «`./memory.exe` 1» et suspendez-le aussi par «**Ctrl-Z**».
17. Faites «`jobs`» dans chacune des deux consoles. Que observez-vous. Qu'en déduisez-vous sur ce que renvoi «`jobs`» ?
18. Dans la console 1, donnez la commande pour tuer le job suspendu.
19. Dans la console 2, donnez la commande pour tuer le job **1** suspendu. Donnez la commande pour remettre en premier plan, le job **2**.

Énoncé 2 : Exécution des processus en avant/arrière plan

1. Se connecter en tant que «`root`» sur une console texte.
2. La commande «`sleep`» sert à attendre pendant un nombre de secondes spécifié. Par exemple, « `sleep 5`» attend 5 secondes. Cette commande va servir de base pour ces manipulations car c'est une commande qui permet de simuler l'exécution d'une longue tâche telle qu'une grosse compilation par exemple.
3. Lancez la commande «`sleep 5`». Que se passe-t-il ?
4. Lancez la commande «`sleep 500`» en arrière-plan.
5. Vérifiez avec «`jobs`» que votre commande est toujours là.
6. Lancez la commande «`sleep 5`» en arrière-plan. Que se passe-t-il lorsqu'elle se termine ?

7. Votre commande «`sleep 500`» est toujours active. Mettez-la en avant-plan. (**Utilisation** : la commande `fg`).
8. Suspendez-la. Faites «`jobs`». Quel est son état ? Relancez-la en arrière-plan. (**Utilisation** : la commande `bg`).
9. Lancez une deuxième commande «`sleep 100`» en arrière-plan. Passez la première en avant-plan. Suspendez-la. Suspendez la deuxième.
10. Reprenez l'exécution de la première en avant-plan. Repassez la première en arrière-plan et reprenez la deuxième en arrière-plan. (**Utilisation** : les commandes `bg` et `fg`).
11. Faites «`ps`» pour contrôler les processus actifs.
12. Quelles sont les différences avec «`jobs`» ? Comment faire pour obtenir la liste de tous vos processus ?

Énoncé 3 : Utilisation de la mémoire

Rappel : `/dev/null 2>&1`

Chaque processus a 3 flux :
- un flux d'entré : **STDIN(0)** (typiquement le clavier) ;
- un flux de sortie standard : **STDOUT(1)** ;
- un flux de sortie d'erreur : **STDERR(2)** ;

`2>&1` redirige la sortie d'erreur(**STDERR**) vers la sortie standard (**STDOUT**) et la sortie **STDOUT** dans `/dev/null`.

1. Se connecter en tant que «`root`» sur une console texte.
2. Observer l'utilisation de la mémoire. Quelle est la quantité de mémoire utilisée pour les tampons du noyau (buffers) et le cache disque ? Quelle est l'utilisation de l'espace de pagination ? (**Utilisation** : la commande `free`).
3. Lancer la commande suivante qui parcourt tous les fichiers sur le système : `ls -lR / > /dev/null 2>&1`
4. Une fois la commande terminée, observer de nouveau la quantité de mémoire allouée aux tampons et au cache disque. Quelle est l'utilisation de l'espace de pagination. (**Utilisation** : la commande `free`).
5. Relancer la commande `ls` comme précédemment. L'exécution est-elle plus rapide ? est ce que l'utilisation de la mémoire est différente ? (**Utilisation** : la commande `free`).
6. Récupérer le programme «`memoire.c`» à partir le site.
7. Compiler le programme. (ne pas tenir compte des messages de warning s'il y en avait) «`gcc memoire.c -o memoire.exe`». L'exécutable généré s'appellera «`memoire.exe`».
8. Afficher les statistiques d'utilisation de la mémoire en Mo ainsi que le total de mémoire disponible (mémoire vive + swap). (**Utilisation** : la commande `free` avec les options adéquates dans le manuel `man`).
9. Attribuer le droit d'éxecution pour le «`memoire.exe`». (**Utilisation** : la commande `chmod`).
10. Lancer le script «`./memoire.exe`» de sorte que qu'il alloue **200 Mo** de mémoire et observer de nouveau l'utilisation mémoire sur une autre console texte. (**Utilisation** : la commande `free`).

11. Afficher les statistiques d'utilisation de la mémoire en Mo ainsi que le total de mémoire disponible (mémoire vive + swap). (**Utilisation** : la commande `free`). Que remarquez-vous ?
12. Lancer de nouveau le script «`./memoire.exe`» de sorte qu'il alloue cette fois **1Go** puis **2Go** de mémoire. Que se passe-t-il ?