INTRODUCTION AU SE

TP 3 : Les processus

P. Mely - IUT de Metz – R1.04

Exercice 1

Pour voir apparaître l'ensemble des processus sur la machine sur laquelle vous travaillez, il suffit par exemple de taper la commande top dans une fenêtre. Vous aurez alors l'ensemble des processus ainsi que leur PID et la quantité de temps processeur que chacun utilise. Pour quitter cette commande, tapez q, tapez h pour avoir l'aide. Combien de mémoire avez-vous ? De swap ? Quel pourcentage de cpu est utilisé ?

Exercice 2

- Lancez la commande ps : sur quels processus a-t-on des renseignements ?
- En vous servant de man, donnez les options permettant de voir l'état de tous les processus de la machine.
- A quelle heure votre machine a démarré ?
- Pouvez-vous établir le nombre approximatif de processus créés depuis le démarrage de votre machine ?
- Rechercher le processus de la console que vous avez lancé. Ensuite, rechercher le processus père jusqu'à trouver le processus initial.

Exercice 3

Sous UNIX, chaque processus (excepté le premier) est créé par un autre processus, son processus père. Le processus père d'un processus est identifié par son PPID (Parent PID).

- Trouver une option de la commande ps permettant d'afficher le PPID d'un processus.
- Donner la liste ordonnée de tous les processus ancêtres de la commande ps en cours d'exécution.
- Reprendre la question précédente avec la commande pstree et chercher comment afficher l'arbre d'un processus

Exercice 4

Lorsque vous lancez une commande à partir d'un terminal interactif, par défaut, elle est lancée en premier plan : le terminal "attend" la réponse de la commande pour vous redonnez la main. Pour une commande dont on attend une réponse immédiate (exemples : date, who), cela ne pose pas de problème. Mais pour une commande qui lance un logiciel, comme xclock, on peut vouloir récupérer la main sans avoir à fermer xclock. On lance alors le processus en arrière plan (ou tâche de fond) en ajoutant le caractère & à la fin de la commande.

- Dans un terminal, lancez la commande xclock en premier plan, puis en arrière plan. Observez la différence.

Exercice 5

La commande **nice** sans argument permet d'afficher la priorité des processus. La valeur de priorité par défaut est 10.

Exemple : $ nice
0 --> il indique que bash a une priorité 0
$ nice bash --> je donne une priorité de 10 à bash
$ nice --> bash a donc une priorité de 10

La priorité peut être ajustée avec nice dans l'intervalle -20 (le plus prioritaire) à 19 (le moins prioritaire).

Exemple : $ nice -n 19 top --> top est très gourmand et ceci permet de ne pas utiliser toutes les ressources pour top

Exercice 6

La commande **sleep** permet de faire une pause pendant une durée en seconde (par défaut), en minutes, en heures, en jours avec la possibilité d'additionner ces valeurs.

Exemple : $ date;sleep 10; date --> cette commande va lancer une commande date 10s après
$ date;sleep 10 1m;date --> attente d'une minute et 10 secondes

Exercice 7

Ecrire un shell compteur en suivant l'algorithme :

- i = 0
- Répéter infiniment :
  i = i + 1
  si i est multiple de 1000 afficher i et le numéro de processus. (% est l'opérateur modulo)

- Lancer l'exécution de ce programme et vérifier qu'il fonctionne. L'arrêter en tapant CTRL-C.
- En utilisant les fonctionnalités du shell (&, fg, bg), lancer deux instances du programme compteur en même temps. Mettre au premier plan la deuxième, l'arrêter (CTRL-Z) puis la relancer en arrière plan. Regarder ce qui se passe au niveau processus.
- A l'aide des commandes jobs et kill %n, arrêter tous les compteurs (aidez-vous de man).
- Même question en utilisant les commandes ps et kill (avec un PID).
- Lancer l'exécution d'une instance du programme compteur en lui donnant une priorité plus faible à l'aide de la commande nice. Lancer une autre instance : que remarquez-vous ?

Exercice 8

La commande **at** permet de lancer des commandes (ou depuis un fichier) qui s'exécuteront à une date ultérieure.
atq permet d'afficher ce qui est en cours pour son compte
atrm efface les travaux en cours
Exemple : at -f script.sh 18h05

at est bien pour programmer une action une fois mais pour des tâches à automatiser, il est mieux d'utiliser la table **cron**.
Chaque utilisateur peut définir avec cron les programmes qu'il veut lancer périodiquement. Il lui suffit d'éditer la table **cron** à l'aide de **crontab -e** et d'ajouter une ligne qui va correspondre à l'exécution de la commande (si le fichier correspondant n'existe pas, il sera créé). N'oubliez pas de sauvegarder (suivez l'aide en bas d'écran).
Exemple: 30 18 * * * /home/users/mely/script-nettoie.sh --> la commande sera exécutée tous les jours à 18h30

Exercice 9

Changer l'exercice 7 afin d'avoir un compteur qui s'arrête lorsque i atteint le nombre de 10000 et que les sorties soient redirigées vers un fichier. Vous ajouterez aussi une ligne affichant la date de début et de fin du programme.

- A l'aide de la commande at, programmez l'exécution de votre programme à une heure donnée. (Vous utiliserez atq + ps pour voir le déroulement de votre exécution)
- Ensuite à l'aide de la commande crontab, vous exécuterez votre programme toutes les 5 minutes.
- Dans les deux cas, vous pourrez utiliser tail pour visualiser les dernières lignes de votre fichier de log.

Lister les processus qui tournent en tache de fond