1516 L2 SEPS TP 3 - Systèmes d'Exploitation et Programmation Système

Multiprocessus Unix

Ce TP se fait sur les machines Unix. Vous devez rendre un rapport et vos programmes à la fin du cours sur Espadon. Assurez-vous que votre rapport porte un nom du type "1516_L2_SEPS_TDTP_3_MultiProcUnix_*votreNom*". Mettez dans le rapport des copies d'écran des résultats d'exécutions ("Applications" → "Accessoires" → "Capture d'écran") que vous ferez régulièrement le long du TP pour montrer la réussite ou l'échec de vos opérations. Rendez le tout dans un fichier Zip de même nom que précédemment. Pour tous les TP, ne validez votre dépôt sur Espadon qu'une fois le TP totalement terminé; à la fin de chaque séance, déposez un zip de vos travaux avec votre rapport en l'état, mettez-le à jour à chaque séance, et une fois le TP fini, validez.

I] Connexion

1) Connectez-vous à votre machine virtuelle Unix (Utilisez le mot de passe que vous avez défini au premier TP).

2) Dans le répertoire "/home/mathinfo2014/Documents/SE_and_PS", créez un répertoire "MultiTask" (commande *mkdir*; ce sera le répertoire de travail du TP).

II] Un bon coup de fourchette

1) Dans le répertoire "/home/mathinfo2014/Documents/SE_and_PS/MultiTask", créez un répertoire "FatherAndSon".

2) Créez un fichier "fatherandson.c" et un fichier "makefile".

Le makefile devra contenir le texte suivant:

```
fatherandson: fatherandson.c

	gcc -o fatherandson fatherandson.c -I.
```

3) Dans le fichier "fatherandson.c", écrivez un programme en C qui se scinde en un processus père, qui affiche son PID et celui de son fils et patiente quelques secondes, et un processus fils qui affiche son PID et celui de son père et patiente également quelques secondes.

II] Un jour c'est oui, un jour c'est non

1) Dans le répertoire "/home/mathinfo2014/Documents/SE_and_PS/MultiTask", créez un répertoire "Yesno".

2) Créez un fichier "yesno.c" et un fichier "makefile".

3) Corrigez le makefile en conséquence.

4) Dans le fichier " yesno.c", écrivez un programme en C qui se scinde en un processus père, qui affiche son PID et celui de son fils et patiente quelques secondes, le tout dans une boucle (de

quelques itérations aussi), et un processus fils qui affiche son PID et celui de son père et patiente un temps légèrement supérieur, le tout aussi dans dans un boucle.

5) Comment faire pour que le processus père attende la fin du processus fils pour terminer ?

III] Sweet Child O'Mine

1) Dans le répertoire "/home/mathinfo2014/Documents/SE_and_PS/MultiTask", créez un répertoire "GrandFather".

2) Créez un fichier "grandfather.c" et un fichier "makefile".

3) Corrigez le makefile en conséquence.

4) Dans le fichier " grandfather.c", écrivez un programme en C qui se scinde en:

- un processus père, qui affiche son PID et celui de son fils et effectue un deuxième 'fork()', chaque processus résultant déclinant également son identité.
- un processus fils qui affiche son PID et celui de son père et effectue aussi un deuxième 'fork()', chaque processus résultant déclinant également son identité.

Prenez soin de numéroter les affichages (printf) pour observer et noter dans quel ordre ils s'exécutent. Comment expliquez-vous les valeurs de PID observées ?

IV] O'brother

1) Dans le répertoire "/home/mathinfo2014/Documents/SE_and_PS/MultiTask", créez un répertoire "GrandFatherDelayed".

Répétez la section III, mais en rajoutant un délai de quelques secondes après chaque affichage. Prenez soin toujours de numéroter les affichages (printf) pour observer et noter dans quel ordre ils s'exécutent. Les valeurs de PID observées sont elles les même ? Comment expliquez vous la différence ?