Université Joseph Fourier, Grenoble
Licence d'Informatique L3

Introduction aux Systèmes et Réseaux

TP n°1 : Introduction aux processus Unix

Indications générales et rappels (valables pour tous les TP). Travailler sous Unix [1], et écrire les programmes en C. La commande `gcc -o prog prog1.c ...progn.c`, ou `prog1, ...progn` sont des programmes en C, compile et lie ces programmes pour produire un exécutable `prog`, qu'on peut lancer par la commande `./prog`.

Pour utiliser certaines primitives (par exemple `Fork()`, etc.), vous aurez besoin de lier les programmes avec le programme `csapp.c` et d'inclure `csapp.h`. Ces programmes sont dans le placard de l'UE Systèmes et Réseaux (SR), dans la rubrique TP. Si vous les utilisez, il faudra ajouter dans la commande de compilation (`gcc`) l'option `-lpthread` pour inclure les bibliothèques correspondantes.

1 Identification des processus

1. Testez les commandes `top` et `ps` pour afficher les processus s'exécutant sur la machine que vous utilisez.
2. Écrire un programme qui affiche le numéro (`PID`) du processus qui l'exécute. Que remarquez-vous en le lançant plusieurs fois ?
3. Modifiez le programme pour qu'il affiche son `PID` et son `PPID`. Que remarquez-vous ?

2 Environnement des processus

1. Testez la commande `env` pour afficher les valeurs des variables d'environnement courantes (cf. TD n° 1, section 3).
2. Examinez en particulier la valeur de la variable d'environnement `PATH`. Cette variable permet de localiser automatiquement un fichier exécutable (binaire) en définissant une suite de répertoires dans lesquels on le recherche, dans l'ordre indiqué. C'est ce qui permet, par exemple, à l'utilisateur de taper la commande `gcc` au lieu (par exemple) de `/usr/local/bin/gcc`. Pour connaître la localisation exacte d'une commande, on utilise la commande `which` (essayez par exemple `which ls` puis `which gcc` et `which which`).
3. Pour cette question, commençons par un rappel de la syntaxe pour manipuler une variable d'environnement via un shell.
   - Définir ou redéfinir une variable d'environnement :
     - shell bash : `export MAVARIABLE=valeur`
     - shell tcsh : `setenv MAVARIABLE valeur`

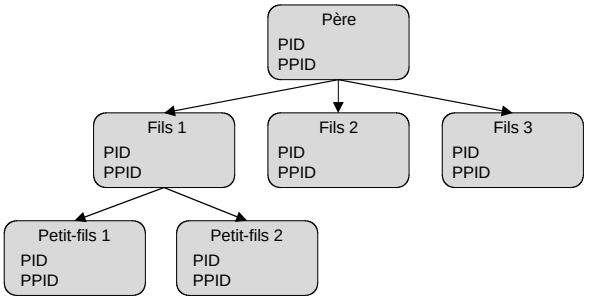
1. Bien que les TP puissent se faire sur tout système Unix, il est recommandé de travailler sur un serveur Linux.

– Détruire une variable d'environnement :
  – shell bash : `unset MAVARIABLE`
  – shell tcsh : `unsetenv MAVARIABLE`
– Afficher la valeur d'une variable d'environnement : `echo $MAVARIABLE`

Expériences à effectuer :
– Sauvegardez la valeur de la variable `PATH` dans une nouvelle variable `SAVEPATH`.
– Ensuite, modifiez affectez la valeur suivante à la variable `PATH` : `/:.`
– Tapez la commande `ls`. Que constatez-vous et comment l'expliquez vous ?
– Restaurez la valeur initiale de la variable `PATH` en recopiant la valeur de la variable `SAVEPATH` (exemple avec bash : `export PATH=$SAVEPATH`).
– Détruire la variable `SAVEPATH`.
– Comment se fait-il que les commandes de gestion des variables d'environnement continuent de fonctionner malgré la modification du `PATH` ?

3 Création de processus

1. Écrire un programme qui reproduit l'arbre généalogique représenté ci-après. Chaque processus doit afficher son `PID`, son `PPID`, et afficher les fils qu'il engendre.



2. Observer l'ordre d'apparition des messages à l'écran et commentez. Que se passe-t-il si on lance plusieurs fois le programme ?
3. Écrire les programmes (vus en TD, question 4) qui réalisent respectivement une chaîne de `n` processus, et un arbre de `n` processus (`n` est passé en paramètre à l'exécution du programme).

4 Synchronisation élémentaire de processus

1. Modifier le programme de la question précédente (3.1) pour que le fils 2 affiche son message avant les fils 1 et 3. La solution proposée est-elle déterministe ?
2. Modifier le programme pour que les petits-fils 1 et 2 affichent leur message avant les fils 2 et 3. La solution proposée est-elle déterministe ?
3. Reprendre et exécuter le programme vu à la question 5 du TD (ce programme est dans le placard du TP1 sous le nom `waitpid1.c`).

5 Exécution de processus

1. Écrire un programme qui affiche son PID ainsi que l'ensemble des variables définies dans son environnement.
2. Reprendre la question 8 du TD : écrire un programme qui exécute une commande Unix qu'on lui passe en paramètre.
3. Reprendre la question 9 du TD : écrire un programme qui exécute la commande `man` avec un paramétrage particulier.
4. Reprendre la question 10 de TD : écrire un programme qui exécute une commande Unix qu'on lui passe en paramètre en cherchant automatiquement l'emplacement du fichier exécutable correspondant.