IUP MIAGE
Système d'exploitation

TP — Création de processus

Ce TP a pour but de mettre en évidence l'exécution concurrente de deux processus à partir d'un programme simple. Pour chaque exécution, on notera soigneusement les résultats affichés afin de pouvoir comparer les différentes exécutions.

1 Exécution concurrente

Écrire un programme C s'exécutant sous forme de deux processus, le processus fils affichant les nombres de 1 à N, le processus père affichant les nombres pairs seulement parmis les nombres de 1 à N. Lors de l'affichage d'un nombre, chaque processus affiche aussi son identité (PID). Chaque processus affiche un message de fin avant de se terminer. Écrivez ce programme de façon à pouvoir facilement modifier la valeur de N.

Exercice 1 (Premier essai)
Essayez le programme pour N=10. Que constate-t-on ?

Exercice 2 (Changer)
Changez progressivement la valeur de N jusqu'à voir apparaître une vraie simultanéité dans l'exécution des deux processus.

Exercice 3 (Fin d'exécution)
Observez les fins d'exécution des deux processus. Que remarque-t-on ? Utilisez l'appel système `wait` pour changer éventuellement l'ordre des fins d'exécution.

2 Gestion de processus

Les deux commandes `ps` et `kill` sont utilisées pour la gestion des processus.

La commande `ps` affiche des informations sur les processus actifs. Sans options, elle affiche les processus du même terminal que le processus `ps`. La sortie contient uniquement le `pid` du processus, le nom du terminal, le temps cumulé d'exécution et le nom de la commande.

Les options les plus courantes sont :

`-a` liste tous les processus sauf ceux non associés à un terminal ;

`-e` liste tous les processus du système ;

`-f` donne plus d'information pour chaque processus ;

`-u uidlist` liste les processus des utilisateurs donnés dans `uidlist`.

La commande `kill` permet d'envoyer un signal à un processus. Deux syntaxe sont disponibles :

`kill -s nomdesignal pid ...` envoie le signal dont le nom est `nomdesignal` aux processus dont on donne le pid ;

`kill -numerodesignal pid ...` envoie le signal de numéro `numerodesignal` aux processus dont on donne le pid.

Id: tpprocessus.tex,v 1.7 2001/02/15 15:25:30 marquet Exp

Un signal peut donc être nommé symboliquement (`kill -l` donne la liste des noms) ou par son numéro. Nous utiliserons ici le signal `KILL` de numéro 9 qui permet de tuer un processus.

Exercice 4 (Illustration)
Écrivez un programme à plusieurs processus qui effectue la tâche suivante. Le processus père crée N processus fils. Chaque processus fils réalise dans une boucle infinie les instructions suivantes :

- afficher son `pid` pour préciser qu'il est toujours vivant ;
- s'endormir pour 5 secondes grâce à la fonction `sleep(int nombredesecondes);`

Le processus père, après avoir créé tous ses fils, affiche les processus actifs du terminal et ce à l'aide de la commande `ps`. Pour cela, il fera appel à la fonction `system(char *commande)` qui permet d'invoquer par programme une commande Unix, donc ici `system("ps");`. Puis, jusqu'à ce qu'il n'ai plus de fils, il exécutera le code suivant :

- attendre la terminaison d'un fils
- afficher le pid du fils terminé.

Exercice 5 (Première exécution)

- Utilisez la commande `kill` pour terminer les processus fils.
- Que se passe-t-il si vous tuez le processus père ?

Exercice 6 (État des processus)
Modifiez le programme de telle façon que le processus père attende, par exemple sur une saisie clavier, avant de commencer à attendre la terminaison de ses processus fils à l'aide de `wait`.

- Tuez un de ses processus fils et examinez l'état des processus à l'aide de la commande `ps -a`. Expliquez le résultat.
- Faites continuer le processus père afin qu'il attende la terminaison de son fils et examinez l'état des processus à l'aide de la commande `ps -a`.