TME 2 - GESTION DU TEMPS

ECRITURE D'UNE COMMANDE MYTIMES

L'objectif de ce TME est d'écrire un programme C `mytimes` qui, comme la commande shell `time`, permet d'afficher les statistiques d'utilisation du processeur pour n'importe quelle commande shell. En particulier, `mytimes` devra afficher le temps passé en mode utilisateur et en mode système.

Fonctions utiles au TP :

Les commandes du shell :

`man` (indispensable sous unix… Pensez au `man -a`),

`time, nice, ps.`

Les fonctions en C (voir leurs descriptions détaillées en utilisant le manuel en ligne "`man`") :

`gettimeofday` permet de récupérer le temps écoulé depuis le 1er Janvier 70.

`times` permet de récupérer les statistiques d'utilisation de temps CPU d'un programme et de ses fils. Ces statistiques sont exprimées en nombre de "ticks" horloge. Le nombre de ticks horloge par seconde est obtenu en appelant la fonction `sysconf` avec le paramètre `_SC_CLK_TCK`.

`system` permet de lancer une commande shell depuis un programme C.

1. STATISTIQUES D'EXECUTION D'UNE COMMANDE SHELL

1.1

Exécutez la commande shell `time` pour afficher les statistiques d'utilisation du processeur pour la commande « `sleep 5` ». Que constatez-vous ?

1.2

Ecrivez un programme loopcpu.c qui s'exécute une boucle vide effectuant $5x10^9$ itérations. Lancez le programme avec la commande time. Que constatez vous ?

1.3

Ecrivez un programme loopsys.c qui s'exécute une boucle de $5x10^7$ itérations. A chaque itération, getpid(), un appel système, est appelé. Lancez le programme avec la commande time. Que constatez vous ?

2. LANCEMENT D'UNE COMMANDE SHELL DEPUIS UN PROGRAMME C

2.1

Ecrivez une fonction C

`void lance_commande(char *` commande`)`

qui utilise la fonction `system` pour lancer l'exécution de la commande shell passée en paramètre. `lance_commande` doit renvoyer un message d'erreur si la commande n'a pu être exécutée correctement.

2.2

Ecrivez une fonction `main` qui appelle `lance_commande` pour chaque argument passé sur la ligne de commande. Générez un exécutable `mytimes`.

3. CALCUL DU TEMPS DE RÉPONSE EN UTILISANT GETTIMEOFDAY

3.1

Modifiez votre fonction `lance_commande` pour afficher le temps mis par l'exécution de la commande. La mesure du temps pourra être effectuée à l'aide de la fonction `gettimeofday` .

3.2

Testez votre programme avec la commande :

```
$ ./mytimes "sleep 5" "sleep 10"
```

4. CALCUL DES STATISTIQUES

La version précédente de permet de connaître le temps qui sépare le début d'exécution de la commande de sa terminaison. On veut maintenant affiner les résultats obtenus en mesurant le temps passé mode utilisateur et le temps passé en mode système.

4.1

Créez une nouvelle version de la fonction `lance_commande` qui affiche toutes les statistiques fournies par la fonction `times`. On fournit ci-dessous un exemple d'exécution de mytimes. Votre programme devra fournir un affichage similaire.

```
$ ./mytimes "ls -l" "./loopsys"
total 52
-rwxr-xr-x  1 sens p6ipens  8496 janv. 16 13:08 loopcpu
-rwxr-xr-x+ 1 sens p6ipens    61 janv. 16 13:08 loopcpu.c
-rwxr-xr-x  1 sens p6ipens  8696 janv. 16 12:37 loopsys
-rw-r--r--  1 sens p6ipens   149 janv. 16 12:37 loopsys.c
-rwxr-xr-x  1 sens p6ipens 13072 janv. 16 12:32 mytimes
-rw-------+ 1 sens p6ipens  1604 janv. 16 12:32 times.c
Statistiques de "ls -l" :
```

```
Temps total : 0.010000
Temps utilisateur : 0.000000
Temps systeme : 0.000000
Temps utilisateur fils : 0.000000
Temps systeme fils : 0.000000

Statistiques de "./loopsys" :
Temps total : 8.680000
Temps utilisateur : 0.000000
Temps systeme : 0.000000
Temps utilisateur fils : 3.870000
Temps systeme fils : 4.800000
```

4.2

Testez votre nouvelle version avec les exemples suivants :

```
$ ./mytimes "sleep 5" ./loopcpu ./loopsys
```

5. CHANGEMENT DE PRIORITÉ

La commande `nice` permet de changer les priorités d'un programme. En tant qu'utilisateur non privilégié on ne peut que baisser la priorité de ses processus de façon à avantager les autres programmes (d'où le nom "`nice`" de la commande). Au maximum, on peut baisser la priorité d'un processus de 19.

5.1

Tapez la commande `ps -l`. Quelle est la priorité du processus `ps` ?

5.2

Tapez la commande `nice -19 ps -l`. Quelle est maintenant la priorité de la commande `ps` ?

5.3

Les machines ayant 8 cœurs, lancez 9 exécutions en parallèle de loopcpu, en mesurant leur temps d'exécution avec my`times`. L'une des exécutions sera lancée en abaissant sa priorité.