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. La durée d'un tick horloge est obtenue 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 ?

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 REPONSE 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" "cd /usr/include ; grep time *.h > /dev/null"
total 32
-rw-r--r--    1 sensl    lip6          145 oct 11 14:15 Makefile
-rwxr-xr-x    1 sensl    lip6        14959 oct 11 14:15 mytimes
-rw-r--r--    1 sensl    lip6         1283 oct 11 14:11 mytimes.c
-rw-r--r--    1 sensl    lip6         1281 oct 11 14:10 mytimes.c~
-rw-r--r--    1 sensl    lip6         2072 oct 11 14:14 mytimes.o
-rw-r--r--    1 sensl    lip6            0 oct 11 15:16 t.txt

Statistiques de "ls -l"
Temps total  :   0.0300
Temps utilisateur : 0.0000
Temps systeme : 0.0000
Temps util. fils : 0.0300
Temps sys. fils : 0.0000

Statistiques de "cd /usr/include ; grep time */*.h > /dev/null"
Temps total  :   5.6200
Temps utilisateur : 0.0000
Temps systeme : 0.0000
Temps util. fils : 0.1800
Temps sys. fils : 0.1600
```

4.2

Testez votre nouvelle version avec l'exemple suivant :

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

Comparez vos résultats avec ceux de la question 1.1.

5. CHANGEMENT DE PRIORITE

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

5.3

Ecrivez un programme C dans lequel s'exécute une boucle effectuant 10^8 itérations.

5.4

Lancez deux exécutions en parallèle de votre programme de la question précédente, en mesurant leur temps d'exécution avec `mytimes`. L'une des exécutions sera lancée en priorité normale, l'autre en abaissant au maximum la priorité. Que constatez vous ?