Université de Bretagne Sud
UFR SSI, Dept MIS
SYSTEME D'EXPLOITATION
CONCURRENCE
Les processus Unix
Les appels système fork, wait, exit et exec...

Luc Courtrai

Email : Luc.Courtrai@univ-ubs.fr

1 Multif

Ecrire un programme C qui affiche en parallèle sur la sortie standard le contenu des fichiers dont les noms sont passés en arguments de la commande. Chaque fichier sera pris en charge par un processus distinct.

2 Serveur de jobs

On désire faire exécuter à un programme des tâches en parallèle. Pour simplifier l'exercice chaque tâche aura à imprimer un caractère sur la sortie standard à un intervalle de temps. Chaque tâche doit être prise en charge par un processus.

La liste des tâches est décrite dans un fichier d'ordres dont voici un exemple (*donnees.jobs*) :

```
a 10 2
b 20 1
c 15 3
```

Dans cet exemple, la première tâche va consister à imprimer 10 caractères *a* sur la sortie standard. Entre chaque impression de caractère, au moins 2 secondes doivent s'écouler.

Le programme *pr* lit les données sur l'entrée standard :

```
pr < donnee.jobs
```

Ecrire en C le programme *pr*.

3 La fonction system() de la librairie standard C

Ecire la fonction mySystem(char *) correspondante à la fontion C :

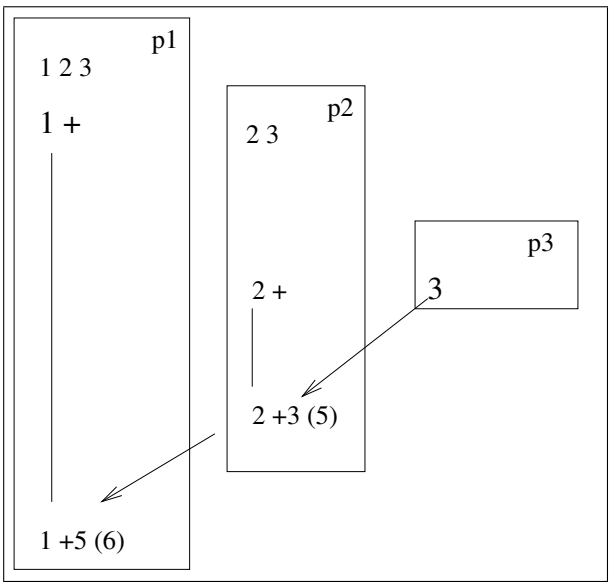
```
int system (const char * string);

  La  fonction  system()  exécute  la commande indiquée dans
       string en appelant /bin/sh -c  string,  et  revient  après
       l'exécution   complete   de   la   commande.
```

4 Répartition d'un calcul

On désire répartir le calcul d'une addition d'une liste d'entiers entre plusieurs processus. Chaque processus effectue la somme du premier de la liste avec le reste de la liste ; la somme du reste (sous liste) étant calculé par d'autres processus de la même façon.

exemple

```
somme 1 2 3
```



La transmission de la réponse se fera grâce au code de sortie (*exit*).

5 priorité

Ecrire un programme qui lance N processus de calcul (exemple :)

```
for (i = 1; i <= 100; i++)
   {
       for (j = 0; j < 1000000; j++) x=1./i;
   }
```

Un fils sur deux sera lancé avec une priorité minimale.

En utilisant la fonction clock()/CLOCKS_PER_SEC déterminez le temps de calcul des processus (include <time.h>).

En utilisant la fonction gettimeofday déterminez le temps réel de calcul des processus..

6 glue

Écrire un programme C qui "glue" une machine en créant un ensemble de processus de calcul et en remplissant la table des processus. La machine sera gluée lorsque le système ne pourra plus créer de nouveau processus (échec de l'appel système fork).

Le programme affiche le nombre de processus crées, attend une saisie clavier et détruit tous les processus de calcul. (les processus de calculs doivent appartenir à un même groupe de processus pour facilité leurs destructions)

7 charge

Écrire un programme C qui charge la CPU d'une machine en créant un ensemble de processus de calcul.

```
usage : charge N%
```

Charge la CPU à N % à plus ou moins 5%.

La charge de la machine sera prise en consultant le fichier /proc/stat qui donne le temps depuis le démarrage de la machine

```
> man proc
   cpu  3357 0 4313 1362393
The number of jiffies (1/100ths of  a  second)  that  the  system
spent  in  user  mode, user mode with low priority (nice), system
mode, and the idle task, respectively.  The last value should  be
100 times the second entry in the uptime pseudo-file.
```

Le processus de calcul doivent avoir une vie limitée pour permettre à la commande de maintenir le % de charge demandée.