UNIVERSITÉ DE ROUEN	ANNÉE 2008-2009
FACULTÉ DES SCIENCES	SYSTÈME D'EXPLOITATION
LICENCE INFORMATIQUE 3ème ANNÉE	CONTRÔLE CONTINU
17 NOVEMBRE 2008	2H00

Aucun document n'est autorisé.

Exercice 1 Soit $T_1, T_2, T_3, T_4, T_5, T_6$ six processus dont la durée relative (t_i) et le temps d'arrivée (τ_i) sont donnés par le tableau suivant :

	τ_i	t_i
T_1	3	0
T_2	6	$1-\varepsilon$
T_3	2	$1+\varepsilon$
T_4	1	$2+\varepsilon$
T_5	3	$3-\varepsilon$
T_6	6	$3+\varepsilon$

Vous donnerez un diagramme de Gantt pour chacun des algorithmes suivants. Vous donnerez également le temps de réponse (temps qui s'écoule entre le moment où le processus arrive et le moment où il obtient le processeur pour la première fois) ainsi que le temps de traitement (temps qui s'écoule en le moment où le processus arrive et le moment où son exécution se termine) de chacun des processus pour chacune des politiques d'ordonnancement

1. FCFS
2. SJF
3. tourniquet avec un quantum $q = 1$
4. tourniquet avec un quantum $q = 1$ et priorité dynamique $P_0(T_j) = j$ et $P_{i+1}(T_j) = P_i(T_j) + TTU$ où TTU est le temps total utilisé par le processus.

Exercice 2 L'inscription pédagogique des étudiants d'une université se passe de la manière suivante. Chaque étudiant doit s'inscrire dans exactement quatre cours. De plus, le nombre de place dans chaque cours est limité.

La scolarité de cet établissement met en place une inscription électronique à ces cours. Elle propose également un mécanisme de permutation des cours permettant à un étudiant ayant déjà ses quatre cours de changer de cours sans perdre le bénéfice du cours auquel il est inscrit si la modification n'est pas possible faute de place. voici un morceau du code mis en place par la scolarité :

```
coursEchange (utilisateur, cours1, cours2) {
verrouille (cours1);
desinscrit (cours1,utilisateur);
if (estPlein(cours2) == false) {
verrouille (cours2);
inscrit (cours2,utilisateur);
deverrouille (cours2);
}
deverrouille (cours1);
}
```

Vérifiez que cette implantation n'est pas correcte. Listez les différents problèmes qui peuvent se poser. Proposez une solution qui fonctionne avec les fonction existantes.

Exercice 3 Ecrire une fonction `int nFork(int nbProcs);` qui crée nbProcs processus fils en largeur et retourne l'indice du processus concerné (de 1 à nbProcs) pour les fils et 0 pour le père. Vous trouverez une description de la fonction `fork()` à la page suivante.

Exercice 4

1) Pour chacune des transitions suivantes entre les états des processus, indiquez si la transition est possible. Si c'est le cas, donnez un exemple.

- Prêt - En exécution
- En exécution - bloqué
- Bloqué - En exécution
- Prêt - Terminé

Exercice 5

1) Soit un tableau de n cases en mémoire partagée. La $i^{\text{ème}}$ somme partielle est la somme des i premiers termes du tableau. Le but de cet exercice est d'écrire un algorithme parallèle qui calcule toutes les sommes partielles d'un tableau. Le principe est le suivant. Un processus P_0 va créer $n-1$ processus. Le processus P_i va attendre que le processus $i-1$ ait terminé puis va faire la somme de la $i-1^{\text{ème}}$ case avec la $i^{\text{ème}}$ case puis écrire le résultat dans le tableau à la place i. Le processus P_0 se contentera d'afficher toutes les sommes partielles. Ecrivez l'algorithme en utilisant l'attente passive (mort d'un fils) pour la synchronisation.

NAME

fork -- create a new process

SYNOPSIS

```
#include <sys/types.h>
#include <unistd.h>

pid_t
fork(void);
```

DESCRIPTION

Fork() causes creation of a new process. The new process (child process) is an exact copy of the calling process (parent process) except for the following:

- The child process has a unique process ID.
- The child process has a different parent process ID (i.e., the process ID of the parent process).
- The child process has its own copy of the parent's descriptors. These descriptors reference the same underlying objects, so that, for instance, file pointers in file objects are shared between the child and the parent, so that an lseek(2) on a descriptor in the child process can affect a subsequent read or write by the parent. This descriptor copying is also used by the shell to establish standard input and output for newly created processes as well as to set up pipes.
- The child processes resource utilizations are set to 0; see setrlimit(2).

RETURN VALUES

Upon successful completion, fork() returns a value of 0 to the child process and returns the process ID of the child process to the parent process. Otherwise, a value of -1 is returned to the parent process, no child process is created, and the global variable errno is set to indicate the error.

ERRORS

Fork() will fail and no child process will be created if:

[EAGAIN] The system-imposed limit on the total number of processes under execution would be exceeded. This limit is configuration-dependent.

[EAGAIN] The system-imposed limit MAXUPRC (<sys/param.h>) on the total number of processes under execution by a single user would be exceeded.

[ENOMEM] There is insufficient swap space for the new process.