Ecole **I**nternationale des **S**ciences du **T**raitement de l'**I**nformation
2ième année ingénieurs

Examen de rattrapage de programmation système
Date17 juin 2009,
Durée 2 heures
Tous les documents sont interdits

1. Préambule

Pour cet examen, vous devez respecter les règles de programmation en vigueur à l'école.

Par ailleurs, pour chaque exercice, il vous est demandé de créer un répertoire contenant la réponse aux questions posées. Vous allez donc devoir créer deux répertoires qui s'appelleront ***Exercice1*** et ***Exercice2***. Dans chacun de ces répertoires, vous fournirez un **Makefile** qui permet la compilation, la génération de la documentation, ainsi qu'une fonction de nettoyage du répertoire. La présence d'un **Makefile** à la racine du projet est un plus, et sera apprécié. Dans chaque répertoire ***Exercice***, vous veillerez à avoir une arborescence correcte.

2. Signaux (4 pts)

Question 1 : Un signal n'est délivré à un processus qu'à l'occasion d'un changement d'état bien précis. Indiquez lequel ? *de l'état actif noyau à l'état actif utilisateur*
Est-ce que à dire qu'un processus exécutant une boucle infinie (while(1) ;) ne se verra jamais délivrer un signal ? expliquez ? *Oui il pourra comme même recevoir des signaux, car il effectue régulièrement des passages à l'état noyau grâce aux interruptions horloge*.

Question 2 : Réaliser un programme qui déroute le signal ALARM. Au bout d'un certain temps (choisit de manière aléatoire), le programme pose la question de savoir s'il doit s'interrompre ou pas. Si le choix de l'utilisateur est de continuer, alors on choisit un nouveau temps, puis on repose la question. Ce procédé est répété jusqu'à ce que l'utilisateur choisisse de terminer le programme. La terminaison du programme sera effectuée par l'envoi d'un signal.

3. Fichier et segment de mémoire partagée (8 pts)

Les entrées/sorties proposées par la bibliothèque standard C introduisent :
- un type FILE
- des primitives associées (fopen, fprintf, etc…).

1. Donner le descriptif de ce type FILE

A une variable de type FILE correspond une zone mémoire destinée à inscrire tout ce qui est nécessaire à la gestion d'un fichier utilisé par un programme pour des lectures ou (et) des écritures. Entre autres, seront notées dans cette zone :

- *l'adresse du fichier physique associé*
- *la position courante en lecture ou en écriture dans le fichier*
- *l'adresse du tampon associé au fichier (les accès à un fichier sont groupés pour être moins fréquents : lorsque le programme demande à écrire dans un fichier, les données à écrire sont mises en mémoire centrale dans un tampon jusqu'à ce que le tampon soit plein ou bien que le fichier soit fermé à la demande de l'utilisateur ; les données sont alors recopiées dans le fichier. De même pour des accès en lecture).*
- *...*

2. Quel est le nom de la structure des fichiers de bas niveau qui contient presque les mêmes informations que la structure référencée par FILE

```
struct stat {
   dev_t     st_dev;     /* Périphérique            */
   ino_t     st_ino;     /* Numéro i-noeud          */
   mode_t    st_mode;    /* Protection              */
   nlink_t   st_nlink;   /* Nb liens matériels      */
   uid_t     st_uid;     /* UID propriétaire        */
   gid_t     st_gid;     /* GID propriétaire        */
   dev_t     st_rdev;    /* Type périphérique       */
   off_t     st_size;    /* Taille totale en octets */
   blksize_t st_blksize; /* Taille de bloc pour E/S */
   blkcnt_t  st_blocks;  /* Nombre de blocs alloués */
   time_t    st_atime;   /* Heure dernier accès     */
   time_t    st_mtime;   /* Heure dernière modification */
   time_t    st_ctime;   /* Heure dernier changement état */
};
```

3. Expliquez les avantages d'utiliser ces primitives plutôt que d'utiliser directement les appels systèmes manipulant des descripteurs de fichiers. *Le type File sert à utiliser des entrées/sorties tamponnées, c'est-à-dire que des tampons sont utilisés en espace utilisateur pour limiter le nombre d'appels systèmes effectues pour lire ou écrire des octets. Il est donc généralement plus performant d'utiliser les primitives associées au type FILE que appels système open, read, write, etc*
4. Donner une architecture pour simuler le fonctionnement d'un fichier de type FILE en utilisant des fichiers de bas niveau (descripteur de fichier) et un segment de mémoire partagée
5. Implémentez la partie écriture dans un fichier uniquement (pas de lecture à partir d'un fichier) de votre architecture

4. Threads (8 pts)

On désire réaliser une multiplication de matrice : C=A*B. Chacune des matrices est une matrice carrée de taille N avec N supérieur à 1024. Pour réaliser le calcul, chaque thread calculera une partie des calculs par un élément. La somme de tous les calculs des threads donnera le résultat final de l'élément considéré. Par exemple le thread 1 calcule les 200 premières multiplications, réalise la somme, et stocke le résultat temporaire dans la matrice C. Parallèlement le thread 2 traite les 200 opérations suivantes, réalise la somme avec la somme mise à jour par le thread 1, ainsi de suite.