Université de Technologie de Compiègne Examen Médian SR02, Printemps 2018

Introduction

Seuls les documents de cours sous format papier sont autorisés.
L'examen comporte deux parties à rédiger sur des copies séparées.
Les réponses doivent être **claires et concises** : cela sera pris en compte lors de l'évaluation des réponses.
La durée de l'examen est 120 minutes.

Partie I (9 pts) —— Copie I

1) En s'appuyant sur des fonctions d'un système d'exploitation (SE) et des exemples précis, donner trois problèmes qui se poseraient en absence du SE. (**0.75 pt**)

2) a) On suppose qu'un processus peut-être dans un des états : Nouveau, Prêt, Actif, Bloqué et Terminé. Pourquoi un processus dans l'état Bloqué ne peut pas passer directement à l'état Actif ? Donner deux raisons. (**0.5 pt**)

b) A quoi servent un CO (Compteur Ordinal) et le PSW (Processor Status Word) dans le PCB (Process Control Block). (**0.5 pt**)

3) Donner un avantage d'un système monoprogrammation par rapport à un système multiprogrammation. (**0.25 pt**)

4) Donner la différence entre une interruption, un déroutement et un appel au superviseur en appuyant votre réponse avec un exemple à chaque fois. (**0.75 pt**)

5) On s'intéresse à l'aspect logiciel des entrées/sorties (E/S) avec DMA.
   a) Pourquoi dit-on que le DMA « vole » des cycles d'accès au bus durant le transfert des données ? (**0.25 pt**)

b) Donner les rôles joués par le processeur et le DMA lors d'une E/S avec DMA en s'appuyant sur les notions de : pilote du périphérique, interruption, routine d'interruption, etc. (**Réponse sur 5 lignes au plus**, **1 pt**)

6) Quel est l'intérêt d'avoir un niveau physique et un niveau logique dans la gestion des entrées/sorties ? Répondre en donnant des exemples tirés du cours. (**1 pt**)

7) Donner une méthode de calcul de la capacité totale d'un disque. Donner la taille maximale d'un fichier dans un système avec i-nœud de 12 entrées (les 10 premières entrées pointent sur les 10 premiers blocs de données. La 11 entrée est en simple indexation de 256 adresses et la 12 entrée est en double indexation), sachant que la taille d'un bloc de données est de 4 Ko. **(0,75 pt)**

8) Quels mécanismes permettent de limiter le ralentissement du processeur lors de réalisation des entrées/sorties ? (**0.5 pt**)

9) Un disque comporte 100 pistes numérotées de 0 à 99. La dernière requête traitée concernait la piste 28 et une requête pour la piste 30 est en cours. La liste des nouvelles requêtes dans l'ordre d'arrivée est la suivante : 60, 10, 80, 20, 70, 35, 0, 90

Calculer le déplacement total de la tête pour les algorithmes suivants **en illustrant votre réponse par un diagramme des déplacements effectués**: (**1.5 pts**)

a) PCTR f) LOOK

10) Répondre par Vrai ou Faux. **Attention** : bonne réponse (**0.25 pt**), pas de réponse (**0 pt**), mauvaise réponse (**-0.25 pt**)

a) Il n'est pas possible de développer des applications sur un matériel sans y avoir au préalable implanté un système d'exploitation
b) Le meilleur système de programmation est le système multiprogrammé
c) Une interruption ne peut interrompre une autre en cours de traitement
d) Dans un système de gestion de fichiers à indexation triple, où la taille d'un bloc est de 1 Ko, la capacité totale théorique d'un fichier est inférieure à 16 Go. (Rappel : $256=2^8$, $1\ Go=2^{20}$ Ko)

Partie II (11pts) —– Copie II

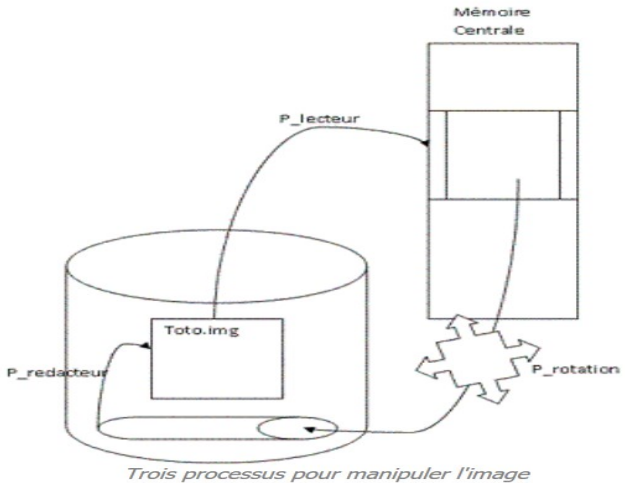
1) Les systèmes d'exploitation font la distinction entre les opérations au niveau de l'utilisateur et les opérations au niveau du noyau. Décidez si les éléments suivants doivent être utilisateur ou noyau, et justifiez brièvement votre réponse: (i) désactiver toutes les interruptions, (ii) lire l'horloge du jour, (iii) régler l'horloge du jour, (iv) tuer un processus (**1.5 pts**).
2) Quelle est la différence entre un signal et une interruption (**0.5 pt**)
3) L'attente active (polling) pendant l'exécution d'une opération est généralement une mauvaise idée car pendant l'interrogation, le processeur ne fait pas de travail utile. Néanmoins, ce n'est pas toujours vrai. Décrivez une circonstance où l'attente active pourrait être meilleure que les interruptions (**1 pt**).
4) La séquence de Fibonacci est la série de nombres 0, 1, 1, 2, 3, 5, 8, ... Formellement, elle peut être exprimée comme suit:

$$fib(0) = 0$$
$$fib(1) = 1$$
$$fib(n) = fib(n-1) + fib(n-2),\ n>1$$

   a) Écrivez un programme C en utilisant l'appel système *fork()* pour générer la séquence de Fibonacci dans le processus fils. Le numéro de la séquence sera fourni dans la ligne de commande. Par exemple, si 5 est fourni, les cinq premiers nombres de la séquence de Fibonacci seront sortis par le processus fils. Le père doit attendre la terminaison de son fils (**2 pts**).
   b) Que se passe-t-il si le père n'attend pas la terminaison de son fils (**0.5 pt**) ?

c) En réalité on veut que la séquence soit calculée par le processus fils et affichée par le processus père. Proposez cinq solutions qui permettent de réaliser cet objectif (sans écrire de code) (**0.5 pt**)

5) Dans le programme ***transposee.c*** ci-dessous trois processus coopèrent pour réaliser la transposée d'une image. L'image est codée en bitmap : chaque point de l'image est codé sur 256 couleurs. L'image est composée de 1024 lignes ; chacune composée de 1024 points. On choisira la structure d'une matrice carrée pour stocker et manipuler l'image. Initialement un processus *p_lecteur* lit l'image à partir d'un fichier "toto.img" dans un segment de mémoire, partagé avec les deux autres processus *p_rotation* et *p_redacteur*, en utilisant la primitive *mmap*, et l'affiche à l'écran. Le processus *p_rotation* effectue la rotation dans un pipe anonyme en utilisant la primitive *writev*. Lorsque la rotation est terminée, le processus *p_redacteur* remet l'image transposée dans le fichier "toto.img" en utilisant un seul *readv* à partir du pipe. La synchronisation entre les trois processus doit être réalisée avec des signaux. La figure suivante illustre le rôle de chaque processus et leur coopération.



Trois processus pour manipuler l'image

a) Expliquer pourquoi size=1024 ? (**1 pt**)
b) Quel est le rôle de *struct shared_data* ? Est-il possible de se contenter de mettre les champs de cette structure comme variables globales ? Justifiez par un contre-exemple. (**1 pt**)
c) Donner les appels systèmes (4) et (5) pour que le programme puisse continuer l'exécution correctement. (**1 pt**)
d) Donner les bouts de code nécessaire correspondant à (6), (7) et (8) pour que les trois processus puissent synchroniser leur intervention. (**1 pt**)
e) Donner les bouts de code nécessaire correspondant à (1), (2) et (3) pour résoudre le problème énoncé. (**1 pt**)

```
#include <median.h>
#define size 1024
char* image;
struct shared_data{
    pid_t p_lecteur, p_rotation, p_redacteur;
    int fd, piped[2];
} *shared_d;
void handler_lecteur(int x){
/**** (1) ****/
}
void handler_rotation(int x){
close(shared_d->piped[0]);
/**** (2) ****/
}
void handler_redacteur(int x){
close(shared_d->piped[1]);
/**** (3) ****/
}
int main(){
struct sigaction sigact_lecteur, sigact_rotation, sigact_redacteur;
int status, shmid;
/***** (4) *****/
/***** (5) *****/
shared_d->fd=open("toto.img",O_RDWR, 0666);
pipe(shared_d->piped);
shared_d->p_lecteur=fork();
if(!shared_d->p_lecteur){
    /***** (6) *****/
    pause();
}else{
    shared_d->p_rotation=fork();
    if(!shared_d->p_rotation){
        /***** (7) *****/
        pause();
    }else{
        shared_d->p_redacteur=fork();
        if(!shared_d->p_redacteur){
            /***** (8) *****/
            pause();
        }else{
            kill(shared_d->p_lecteur, SIGUSR1);
            while(wait(&status)!=-1);
            close(shared_d->fd);  shmdt(shared_d);  shmctl(shmid, IPC_RMID,0);
        }
    }
  }
}
```

Programme *transposee.c*