INF3173 – Principes des systèmes d'exploitation

Automne 2023 – Examen mi-session

Enseignant : Francis Giraldeau
Département d'informatique, Université du Québec à Montréal

Instructions

1. Toute documentation papier permise.
2. Cet examen requiert une calculatrice. Les téléphones cellulaires sont interdits.
3. Répondez directement sur le questionnaire. Détaillez votre démarche pour aider à la correction.
4. Aucune question ne sera répondue pendant l'examen. Dans le doute, indiquez la supposition que vous faites pour répondre à la question.
5. Ne détachez pas les feuilles du questionnaire.

Identification

Nom et prénom : _____

Code permanent : _____

Signature : _____

Résultat

Q1		20
Q2		12
Q3		18
Q4		14
Q5		20
Q6		16
Total		100

Bonne chance!

Q1 (20 pts) Terminologie

Répondez aux énoncés suivants par un terme, un concept, un outil, une librairie ou une fonction vu en classe. Si plusieurs réponses sont indiquées pour le même item, une mauvaise réponse annule une bonne réponse pour cet item.

Description	**Réponse**
1. Appel système qui permet de définir la taille d'un fichier régulier.	
2. Nom de la structure de données en mémoire qui détermine la correspondance entre les adresses virtuelles et physique.	
3. Outil permettant d'inspecter les arguments et les valeurs de retour de tous les appels systèmes effectués par un programme.	
4. État d'une tâche lorsque celle-ci peut s'exécuter, mais qu'elle a épuisé son quantum et est remplacée par une autre tâche.	
5. Terme utilisé pour désigner la perte d'espace qui peut survenir avec l'allocation continue sur disque.	
6. Appel système qui, lorsqu'il fonctionne, ne retourne pas.	
7. Appel système utilisé pour changer le programme en cours d'exécution.	
8. Fonction utile pour afficher le détail de l'erreur survenu dans l'appel système précédent.	
9. Registre Intel x86 dans lequel on passe le numéro de l'appel système à exécuter	
10. Fonction qui retourne le numéro qui identifie la tâche courante.	

Q2 (12 pts) Mesure de performance

Cette question porte sur la mesure de performance, tel que réalisé dans le TP1. Le tableau ci-bas comprend les mesures de performances pour trois programmes mystères suivant :

- sleep: un certain délai d'attente
- cp: copie d'un fichier volumineux
- fib: calcul de la suite de Fibonacci

Complétez le tableau avec les éléments suivants pour chaque item. Utilisez l'espace réponse pour détailler votre démarche.

a) (3 pts) Calculez le taux d'utilisation du processeur.
b) (3 pts) Calculez le taux du temps processeur passé passé en mode utilisateur.
c) (6 pts) Identifiez le programme mystère qui correspond à chaque item.

Item	Temps écoulé (s)	Temps utilisateur (s)	Temps système (s)	Taux d'utilisation processeur (%)	Taux en mode utilisateur (%)	Programme mystère
1	0,829	0,012	0,808			
2	1,297	1,283	0,004			
3	1,523	0,002	0,001			

Q3 (18 pts) Ordonnancement

Considérez les données de simulation des processus dans le tableau ci-bas. Calculez le temps d'attente et le temps de séjour pour chaque tâche selon l'algorithme circulaire (aussi appelé tourniquet, ou Round-Robin RR) préemptif avec quantum de 1 unité, en faisant l'hypothèse que les processus ne bloquent pas. Lorsqu'une tâche devient prête, elle est ajoutée au début de la file de ce pas de temps.

a) (10 pts) Remplissez le tableau de Gantt pour détailler l'ordonnancement des processus. Utiliser la lettre **E** et **A** pour désigner l'exécution et l'attente, respectivement. Indiquez l'état de la file d'attente pour chaque pas de temps (la tête de la file en haut).
b) (6 pts) Calculez le temps de séjour moyen et maximum pour chaque processus.
c) (2 pts) Combien de changements de contexte (changement de tâche) se produit-il au total? _____

Données de simulation				
Tâche	Arrivée	Exécution	Attente	Séjour
P1	4	3		
P2	1	2		
P3	0	5		

Moyenne		
Maximum		

Charte de Gantt (RR, Quantum=1)											
Temps	0	1	2	3	4	5	6	7	8	9	10
P1											
P2											
P3											
File attente											

Q4 (14 pts) Gestion de processus

Le programme suivant exécute plusieurs fois une commande passée en argument. Pour simplifier le code, on suppose que les appels systèmes n'échouent pas. Cette version exécute les commandes l'une à la suite de l'autre. Créez une version du code dans laquelle les processus sont en exécution simultanément (en même temps). Par exemple, en passant la commande « sleep 1 », votre version devrait prendre environ 1 seconde à se compléter plutôt que 4 secondes.

```c
int main(int argc, char** argv) {
  int n = 4;
  int tasks[4];
  for (long i = 0; i < n; i++) {
    tasks[i] = fork();
    if (tasks[i] == 0) {
      execvp(argv[1], argv + 1);
    }
    waitpid(tasks[i], NULL, 0);
  }
  return 0;
}
```

Réponse :

Q5 (20 pts) Gestion de fichiers

Le code suivant montre des opérations sur le fichier régulier « data ». Le contenu de départ du fichier est représenté ci-bas (en hexadécimal, de l'octet 0 à 7). La taille du fichier de départ est de 4 octets.

a) (12 pts) Pour chaque opération de la séquence, indiquez le contenu du fichier et la position du curseur (colonne pos, en octet) du descripteur de fichier **après** l'opération.

b) (3 pts) Quelle est la valeur buf[2] affichée ? _____

c) (3 pts) Quelle est la taille finale du fichier ? _____

d) (2 pts) Nommez deux raisons pour lesquelles l'ouverture du fichier pourrait échouer :

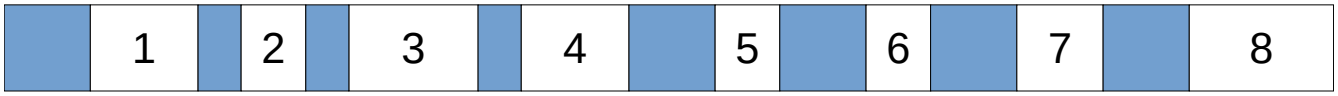
```c
void mystere() {
  char val[4] = {0xCA, 0xFE, 0xBA, 0xBE};
  char buf[2];
  int fd = open("data", O_RDWR);  // Operation 1
  ftruncate(fd, 8);               // Operation 2
  lseek(fd, 2, SEEK_SET);         // Operation 3
  write(fd, val, sizeof(val));    // Operation 4
  lseek(fd, 4, SEEK_SET);         // Operation 5
  read(fd, buf, sizeof(buf));     // Operation 6
  printf("%x %x\n", 0xFF & buf[0], 0xFF & buf[1]);
}
```

Contenu initial	46	4F	4F	0a					pos
Opération 1									
Opération 2									
Opération 3									
Opération 4									
Opération 5									
Opération 6									

Q6 (16 pts) Allocation continue

Le schéma ci-bas représente un espace mémoire, contenant des zone allouées et des zones libres (numérotées de 1 à 8). L'algorithme d'allocation de mémoire du « premier correspondant » (First-Fit Strategy en anglais) consiste à parcourir les espaces libres dans l'ordre, puis d'utiliser le premier espace libre assez grand pour contenir la demande d'allocation.

a) (8 pts) Selon cet algorithme, identifiez quelles zones seront utilisées pour répondre aux 4 requêtes suivantes dans l'ordre : A = malloc(6), B = malloc(10), C = malloc(4), et D = malloc(20). Indiquez à la ligne « Contenu » dans quelle zone se trouve chaque allocation (A, B, C, et D). Si l'allocation n'est pas possible, alors indiquez-le à la ligne « Échec ».
b) (8 pts) Calculez la taille libre de chaque zone à la fin des allocations.



Zones libres :

Numéro de zone	1	2	3	4	5	6	7	8
Taille libre début	10	4	2	2	20	9	12	15
Contenu								
Taille libre fin								

Échec(s) :_____