Examen (1ère session) – 16 décembre 2014

N. Sabouret

L'épreuve dure 2h30. Tous les documents sont autorisés. Les exercices sont indépendants.

1 Exercice 1 – Question de cours (4 points)

1. Quel est l'intérêt de la pagination à deux niveaux ? Expliquez (1 point)
2. À quoi sert l'ouverture d'un fichier ? (1 point)
3. Quel est l'algorithme d'ordonnencement de processus le plus efficace ? Justifiez. (1 point)
4. Quel mécanisme permet d'éviter les attentes actives dans les méthodes de synchronisation de processus dans un OS ? (1 point)

2 Exercice 2 – Gestion de la mémoire (5 points)

On considère un système de gestion de mémoire de 16 Ko, gérée de manière segmentée et paginée avec un seul niveau de pagination. Un processus peut avoir au plus 4 segments. Chaque segment peut adresser au plus 1 ko de données ou de code. Enfin, la taille des cadres de page est fixée à 256 octets.

1. Quelle est la taille (en nombre de bits) et la composition de l'adresse physique? Expliquez. (0,5 point)
2. Quelle est la taille (en nombre de bits) de l'adresse logique? Expliquez. (0,5 point)
3. Quelle est la taille (en nombre de bits) de l'adresse linéaire? Expliquez. (0,5 point)
4. Quelle est la taille de chaque table des pages et de la table des segments? Expliquez. (1,5 point)
5. On considère l'adresse logique `9A1` (exprimée en hexadécimal, c'est-à-dire 1001 1010 0001). Expliquez comment est calculée l'adresse physique et donnez sa valeur en indiquant clairement tous les calculs intermédiaires (2 points).

   On suppose que le processus utilise 4 segments et que la table des segments contient les valeurs suivantes (toutes données en hexadécimal):

Segment:	Limite	Base
0	24A	D29
1	0B7	127
2	1C8	B32
3	037	086

On suppose aussi que la table des pages du processus est la suivante:

Page:	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Cadre:	0E	17	01	2A	20	18	30	23	00	00	12	3E	05	00	04	37
Valide:	1	1	1	1	0	1	0	1	1	0	0	1	1	0	0	1

3 Exercice 3 – Disques RAID (3 points)

Soit les données hexadécimales suivantes:

```
FF 16 12 39 A2 27 47 17 BD E3 89 62 38 C4 8C
```

que l'on souhaite écrire sur un disque RAID constitué de 4 disques et organisé en agrégat par bandes avec un disque de parité (RAID 5). On suppose que les bandes sont constituées d'un seul octet par disque, que la première bande utilise le quatrième disque pour stocker la parité des trois premiers octets, la deuxième bande verra sa parité enregistrée sur le premier disque, la troisième bande sur le deuxiéme disque, etc.

1. Quelle est la capacité totale de ce sous-système RAID5 sachant que chaque disque comporte 20Go d'espace disponible? Justifiez. (0,5 point)

2. Calculer la parité associée à chaque bande et compléter le tableau suivant (1,5 points). Vous pouvez rajouter des bandes si nécessaire.

	Disque 0	Disque 1	Disque 2	Disque 3
Bande 0				
Bande 1				
Bande 2				
...				

3. Supposons que le disque 1 est tombé en panne. Comment faire pour recalculer les quatre parties de bandes effacées de ce disque ? Expliquez. (1 point)

4 Exercice 4 – Ordonnencement de disques (5 points)

On considère un disque composé de 256 cylindres (numérotés de 0 à 255) recevant des requêtes d'accès à des blocs situés sur des cylindres différents. On s'intéresse ici uniquement aux cylindres. On suppose que le déplacement de la tête d'un cylindre à l'autre prend 1 unité de temps. On suppose que la vitesse de rotation du cylindre est de 10 unités de temps et, par conséquent, que le traitement d'une requête prend toujours 10 unités de temps une fois la tête positionnée. La table ci-dessous décrit les arrivées de requêtes sur le contrôleur de disque:

Date	Cylindre(s)
0	167, 21
53	213, 98
135	87
215	4, 8, 131
523	213, 81

Initialement, la tête est positionnée sur le cylindre 37 et il n'y a aucune requête dans la file d'attente.

Important: une requête de déplacement du bras ne peut pas être interrompue: si une nouvelle requête plus prioritaire arrive pendant que le bras se déplace pour atteindre la prochaine requête, elle sera traitée ultérieurement.

1. Décrivez l'exécution si le contrôleur utilise un algorithme FCFS (1 point) et donnez le temps d'exécution total (0,5 point).

2. Décrivez l'exécution si le contrôleur utilise un algorithme SSTF (1 point) et donnez le temps d'exécution total (0,5 point).

3. Décrivez l'exécution si le contrôleur utilise un algorithme C-LOOK (1,5 point) et donnez le temps d'exécution total (0,5 point). On suppose que la tête de lecture est montante dans l'état initial.

5 Exercice 5 – Processus parallèles (3 points)

On s'intéresse à des séquences d'opérations et on souhaite définir une méthode pour notre système d'exploitation qui parallélise les calculs dans des processus différents lorsque c'est possible. Par exemple, si on calcule successivement:

```
op1 : c = a+b
op2 : d = a-b
op3 : a = c-d
```

Les deux premières opérations sont parallélisables, mais la troisième a besoin du résultat des deux premières pour s'exécuter.

Considérons la séquence d'opérations suivante (on suppose que les variables x, y et z ont été définies avant):

```
op1 : a = x + y
op2 : b = x * y
op3 : c = a + z
op4 : d = a * b
op5 : e = c / d
```

1. Dessinez sous la forme d'un graphe orienté les dépendences entre les opérations. Les nœuds du graphes sont les opérations et les arêtes décrivent des dépendances. (1 point)

2. Cette séquence d'opérations est-elle parallélisable? Expliquez. (0,5 point)

3. On considère trois instructions suivantes:

   - `create(instructions)` permet de créer une thread à partir d'une liste d'instructions. Le résultat de cette fonction est un identifiant de thread;
   - `join(id)` permet d'attendre la fin de la thread *id* avant de continuer.
   - `print(var)` permet d'afficher la valeur de la variable *var*.

   En utilisant uniquement les 5 opérations ci-dessus et les deux instructions proposées pour les thread, écrivez un programme en pseudo-code qui effectue les même calculs tout en parallélisant tout ce qui peut l'être puis affiche la valeur de la variable e. (1,5 point)