Examen 1ère session – 23 avril 2013

N. Sabouret

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

1 Exercice 1 – Question de cours (3 points)

1. Quelle est la différence entre un système RAID4 et un système RAID5? Quelles sont les conséquences? (1 point)

   Le disque de parité change à chaque bande. Par conséquent, nous n'avons pas un disque (le disque de parité) plus sollicité que les autres.

2. Quel est l'algorithme d'ordonnancement de processus ou de threads le plus efficace? Justifiez votre réponse. (1 point)

   Il n'y a pas de meilleur algorithme. Un algorithme meilleur sur une instance donnée peut être moins bon sur une autre. Les algorithmes "plus court d'abord" et "Tourniquet" sont généralement utilisé car ils ont de bons résutlats empiriques.

3. Quelle la différence entre le cache et le tampon? (1 point)

   Le cache est une zone d'accès plus rapide contenant une copie des donnée. Le tampon est juste une zone de stockage pour le transfert de données dans les E/S.

2 Exercice 2 – Mémoire (5 points)

On considère un système de gestion de mémoire de 64 Mo, gérée de manière segmentée et paginée avec double niveau de pagination. Un processus peut avoir au plus 256 segments. Chaque segment peut adresser au plus 64 Ko de données ou de code. Enfin, la taille des cadres de page est fixée à 4Ko.
Rappels: 1M=2^{20}, 1k=2^{10}, 256=2^8 et 64=2^6.

1. Quelle est la taille (en nombre de bits) de l'adresse logique? Expliquez. (1 point)

   Pour adresser 256 segments par processus, il faut 8 bits. Donc le sélecteur dans l'adresse logique est sur 8 bits. Si chaque segment peut adresser au plus 64Ko de données, le décalage est sur 16 bits. L'adresse logique est donc sur 24 bits.

   Remarques: Chaque processus adresse donc au plus 16 Mo.

2. Quelle est la taille (en nombre de bits) de l'adresse physique? Expliquez. (1 point)

   Si on a 64Mo de RAM, l'adresse physique est forcément sur 26 bits ($64M = 2^{26}$). Si on veut décomposer: avec des cadres de pages de 4ko, il faut 12 bits pour le décallage. Comme on a 64Mo de mémoire totale, on a 64M/4k = 16k cadres de pages possibles, ce qui nécessite 14 bits.

Remarque: L'adresse linéaire doit permettre à chaque processus d'adresser 16Mo, donc 24bits suffisent. Si on avait un espace d'adressage virtuel plus grand, on serait limité à une adresse linéaire de 26 bits.

3. Soit un processus muni de la table des segments suivante (toutes les adresses sont données en notation hexadécimale):

Segment	Base	Limite
00	BE 0A 75	05 00
01	BE 23 D1	00 BF
02	BE 00 DA	03 61
03	BE 1A 26	05 D2
04	BE 0F F0	00 CF

du répertoire de pages suivant:

Répertoire	Table	Active
0	0	1
1	3	0
2	1	1
3	2	0

et de deux tables de pages (on ne fait figurer ici que les pages utilisées, les autres ont leur bit libre à 1):

Table 0

Page	Cadre	Active
2A0	23 40	1
2A1	05 BB	1
2A2	00 00	0
2A3	14 E0	1

Table 1

Page	Cadre	Active
3E0	00 00	0
3E1	27 FD	1
3E2	00 00	0
3E3	3A F6	1

Quelle est l'adresse physique correspondant à l'adresse logique 03 00 F0 ? Expliquez en détail les opérations de convertion. Vous devrez en particulier donner l'adresse linéaire, expliquer comment elle est construite et comment elle se décompose. (2 points)

Adresse logique: 03 00 F0

Table des descripteurs: 03 est associé au répertoire BE1A26 et à la limite 05D2. 00F0 est bien inférieur à 05D2 donc l'adresse ne provoque pas d'erreur: on additionne base + décalage pour obtenir l'adresse linéaire.

Adresse linéaire: BE 1B 16, soit en binaire 1011 1110 0001 1011 0001 0110

Puisque le répertoire des tables de pages contient 4 éléments, les deux premiers bits sont ceux du répertoire. Ici, il s'agit du répertoire 2, qui correspond à la table des pages 1.

Les 10 bits suivants correspondent nécessairement au numéro de page dans la table des pages, puisque le décallage doit être sur 12 bits (pour les cadres de page de 4 Ko) et qu'il reste 22 bits. Notons au passage (en prévision de la question suivante) qu'il y a au plus $2^{10} = 1024$ *pages dans chaque table.*

Les 10 bits en questions sont 11 1110 0001, soit 3E1 en hexadécimale. D'après la table de pages 1, la page 3E1 est active (pas de déroutement) et correspond au cadre 07 FD.

Le reste (B16) est le décallage. L'adresse physique est donc: 2 7F DB 16

Note: sur 26 bits, la première lettre ne peut pas dépasser 3, l'adresse physique maximum étant 3 FF FF FF.

4. Quelle est la quantité de mémoire physique utilisée par le processus? Justifiez. (1 point)

   Commençons par la mémoire "perdue" en table de pages, répertoire et table des segments. Nous avons deux tables qui contiennent chacune 1024 entrées (10 bits) composées du numéro de cadre (12 bits) et des deux bits de validité et de liberté, soit 14 bits par ligne, soit un total de 1792 octets. Nous avons aussi un répertoire de 4 lignes de 2 bits, soit 1 octet (négligeable) et la table des segments (4 lignes de 5 octets = 20 octets, relativement négligeables aussi). Nous avons donc 1813 octets perdus pour la gestion de la mémoire.

   Ensuite, il y a plusieurs manières de voir le problème. Soit on regarde les segments, et on constate qu'on a 5 segments dont les tailles sont données par leurs limites respectives. Tous ces segments font moins d'une page, donc on a besoin de 5 pages de 4Ko, ce qui nous fait un total de 20Ko+1813o = 22293o. Soit on regarde les tables de pages et on constate qu'on a 5 cadres actifs (de 4ko chacun)... ce qui amène au même résultat.

3 Exercice 3 – NFS (3 points)

On souhaite mettre en place un serveur NFS sur la machine `popeye.universite.fr` qui exporte le répertoire `/dirs/public` en lecture seule sur toutes les machines du domaine `universite.fr`, et le répertoire `/dirs/private` en lecture-écriture sur la machine `olive.universite.fr`

1. Indiquez le contenu du fichier `/etc/exports` sur la machine `popeye` (0,5 point)

   ```
   /dirs/public *.universite.fr(ro)
   /dirs/private olive.universite.fr(rw)
   ```

2. Indiquez le contenu du fichier `/etc/exports` sur la machine `olive` (0,5 point) *Il n'y a pas besoin de modifier le fichier `/etc/exports` sur le client.*

3. On redémarre le serveur NFS à l'aide de la commande `service nfs restart`. Indiquez quels doivent être les permissions sur les répertoires `/dirs/public` et `/dirs/private`. (0,5 point)

   ```
   chmod 755 /dirs/public
   chmod 777 /dirs/private
   ```

4. Indiquez la commande à effectuer sur la machine `brutus.universite.fr` pour monter le répertoire partagé `public` dans `/mnt/popeye`. (0,5 point)

   ```
   mount -t nfs popeye.universite.fr:/dirs/public /mnt/popeye
   ```

5. Expliquez précisément ce qu'il se passe lors de la commande suivante exécutée sur la machine `brutus` après avoir monté le répertoire (1 point):

   ```
   echo "bonjour" > /mnt/popeye/test.txt
   ```

   Le client NFS traduit la commande en RPC pour ouvrir le fichier test sur le serveur. Le serveur traduit la RPC en une demande locale, qui sera refusée parce que les droits ne sont pas bons, et donc la RPC recevra en retour une erreur.

4 Exercice 4 – Ordonnancement (4 points)

On considère un disque composé de 256 cylindres (numérotés de 0 à 255) et 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 de lecture d'un cylindre au suivant ou au précédent se fait en 1 pas de temps et on néglige le temps de rotation et les accès aux secteurs.

Les requêtes sur les cylindres arrivent en 2 paquets. Dans chaque paquet, les requêtes sont ordonnées. Le premier paquet, arrivé au temps 0, comprend des requêtes pour les cylindres:

```
100, 63, 112, 98, 237, 160
```

Le deuxième paquet, arrivé au temps 100, comprend des requêtes pour les cylindres:

```
220, 255, 145, 12, 78, 50
```

Initialement, la tête de lecture est positionnée sur le cylindre 75.

1. Décrivez précisément ce qu'il se passe lors de l'exécution de l'algorithme d'ordonnancement SSTF (plus proche d'abord). (1 point)

   ```
   75 (0) -> 63 (12) -> 98 (47) -> 100 (49) -> 112 (61) -> 160 (109)
   Arrivée du 2e paquet (reste 237)
   -> 145 (124) -> 78 (191) -> 50 (219) -> 12 (257) -> 220 (465)
     -> 237 (482) -> 255 (500)
   ```

2. Quel est le temps total de traitement des requêtes avec cet algorithme? (0,5 point)

   500 pas de temps, comme indiqué précedemment.

3. Décrivez précisément ce qu'il se passe lors de l'exécution de l'algorithme d'ordonnancement C-LOOK (circulaire sans aller jusqu'au bout). On suppose que la tête était montante. (1 point)

   ```
   75 (0) -> 98 (23) -> 100 (25) -> 112 (37) -> 160 (85) -> 237 (162)
   Arrivée du 2e paquet (reste 60)
   -> 255 (180) -> 12 (181) -> 50 -> 60 -> 78 -> 145 -> 220 (389)
   ```

   Note: il ne faut pas compter le retour de 255 à 12 comme 243 pas de temps. Si le disque est bien fichu, on a une tête de chaque côté ce qui permet de repartir dans l'autre sens sans avoir à bouger le bras.

4. Quel est le temps total de traitement des requêtes avec cet algorithme? (0,5 point)

   389 pas de temps, comme indiqué précedemment.

5. Quel est le meilleur algorithme sur cet exemple? (0,5 point)

   C-LOOK

6. Quel serait l'ordonnancement optimal? (0,5 point)

   On n'attend pas forcément une réponse exacte, mais pour commencer, dans la solution proposée par C-LOOK, si on revient traiter 220 avant de repartir sur 255, on gagne 58 par rapport à C-LOOK. Le reste ne peut pas forcément être amélioré puisque prendre le 60 ne donne rien (d'autre valeurs petites vont arrivée au 2e paquet).

5 Exercice 5 – Remplacements (3 points)

On s'intéresse aux algorithmes de remplacement de pages dans un cache capable de contenir 5 pages. Le gestionnaire de mémoire accède successivement aux pages suivantes:

```
1, 7, 8, 2, 3, 1, 6, 1, 2, 7, 3, 5, 6
```

Initialement, le cache est vide.

1. Déroulez l'algorithme clock-based sur cet exemple et indiquez le nombre de défauts de pages. (2 points)

   *Je note avec * lorsque le bit de seconde chance est mis et - lorsqu'il est retiré.*

```
Requête: 1 7 8 2 3 1 6      1 2 7 3 5      6
   1     1         - *    6           *    -
   2       7          *     1          *
   3         8          *        7       *
   4           2        *     -     *    5
   5             3       *        -  *
```

   Total de défaut de pages: 9

2. Quel est le nombre de défauts de page minimal sur cet exemple? Justifiez. (1 point)

   Il faut remplacer la page qui sera utilisée le plus tard dans le futur. C'est donc la page 8 qui saute en premier (elle n'est pas réutilisée).

```
Requête: 1 7 8 2 3 1 6 1 2 7 3 5 6
   1     1                     5
   2       7
   3         8       6
   4           2
   5             3
```

   Total de défaut de pages: 7

6 Exercice 6 – Synchronisation (2 points)

On propose les fonctions d'exclusion mutuelle suivantes, écrites en C, pour un système à deux processus:

```
int[] sc = {0,0};
void entrer_section_critique(int id) {
  sc[id]=1;
  while(sc[1-id])
    ;
}
void sortir_section_critique(int id) {
  sc[id]=0;
}
```

1. Expliquez quel est le problème de cette solution. Illustrez sur un exemple concrêt. (1 point)

   Rien n'empêche deux processus de se marquer tous les deux en SC et donc de rentrer dans une boucle d'attente infinie. P0 appelle SC et effectue sc[0]=1, puis est interrompu, P1 fait de même, donc attend P0, qui reprend la main et attend P1...

2. Proposez une solution simple pour éviter de problème. (1 point)

   Comme proposé dans le cours (transparent 11), on peut rajouter une variable de tour de priorité. Lors de l'entrée en SC, on donne le tour à l'autre puis on met son booléen SC à vrai. On attend alors tant que à la fois l'autre à la main et il est en SC. Si l'autre fait pareil, il nous donnera la main et donc on n'attendra plus.