Partiel – 26 octobre 2015

N. Sabouret, R. Bonaque, M. Gleize

Nom : **Prénom :** **Signature :**

— ***L'épreuve dure 1h30. Tous les documents sont autorisés.***
— ***Le sujet comprend 4 exercices indépendants.***
— ***Toutes vos réponses doivent être justifiées.***

Exercice 1 – Mémoire paginée (5 points)

On se place dans un système de mémoire de 4Go de mémoire géré de manière paginée avec des cadres de page de 4Ko. Chaque processus peut utiliser jusqu'à 64Mo de mémoire. Le système d'exploitation autorise jusqu'à 1024 processus.

1. Quelle est la taille (en bits) de l'adresse logique ? (0.5 point)

   Correction : $64Mo = 2^26o$ *donc l'adresse logique est sur 26 bits.*

2. Quelle est la taille (en bits) de l'adresse physique ? (0.5 point)

   Correction : $4Go = 2^{32}o$ *donc l'adresse physique est sur 32 bits.*

3. Combien y a-t-il de cadres de page dans la RAM ? (0.5 point)

   Correction : $4Ko = 2^{12}$. *Il y a donc* $2^{(32-12)} = 2^{20}$ *cadres de pages.*

4. Combien chaque processus peut-il contenir de pages ? (0.5 point)

   Correction : *Chaque processus peut contenir jusqu'à* $2^{(26-12)} = 2^{14}$ *pages.*

5. On suppose que la pagination se fait sur un seul niveau. Quelle quantité de mémoire est consommée par les tables des pages ? (1 point)

   Correction : *Chaque processus a une table de* 2^{14} *lignes de* 20 *bits chacune, soit environ 40Ko. La table des pages inversées (*2^{20} *lignes de 20 bits) est dans la MMU donc ne coûte rien. Il y a jusqu'à 1024 processus. Donc en tout, on peut consommer jusqu'à* $40Mo$ *de RAM pour la pagination.*

6. On suppose que la pagination se fait sur deux niveaux avec un répertoire sur 6 bits. Quelle quantité de mémoire est consommée par un processus qui utilise les plages d'adresses logiques suivantes ?

   Remarque : les adresses sont représentée en hexadécimal sur 32 bits (les bits en trop sont mis à 0).

de	`00 00 1B 07`	à	`00 00 2C 08`
de	`00 03 CA 00`	à	`00 0E 00 00`
de	`00 0D AB 10`	à	`00 0F B1 CC`
de	`00 2B 10 21`	à	`00 2B 3A 72`

(2 points)

Correction : *Pour commencer, il faut remarquer que dans chaque adresse, les 6 premier bits (les 3 premiers « chiffres » dans notre notation hexadécimale sur 32 bits car on ignore les 0 de tête) désignent le numéro de la table des pages concernée dans le répertoire, les 8 suivants (2 « chiffres ») la page et les 12 derniers bits (3 « chiffres ») constituent le décalage. Nous avons donc un processus qui utilise uniquement 2 tables des pages dans son répertoire. Ainsi, la quantité de mémoire utilisée est de* 2^6 *lignes de* 32 *bits chacune pour le répertoire (adresse de la table des pages), soit* 256*o, plus 2 tables de* 2^8 *lignes de* 20 *bits chacune, soit* 2×640*o, donc un total de moins de 1,5 Ko. Cela représente quand même 20 fois moins de mémoire que ce qu'on pourrait craindre...*

Exercice 2 – Ordonnancement (8 points)

On considère les processus suivants, définis par leur durée (réelle ou estimée) et leur date d'arrivée :

Processus	**P1**	**P2**	**P3**	**P4**	**P5**
Durée	6	3	1	5	2
Date d'arrivée	0	1	2	4	8

1. On suppose que l'ordonnancement est fait suivant l'algorithme du **plus court temps restant non-préemptif**. Indiquez dans chaque case du diagramme de Gantt ci-dessous quel processus est affecté à l'UC à chaque pas de temps de l'exécution, puis indiquez en dessous le temps d'attente moyen. (1 point)

   Correction :

Date	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*
UC	*P1*	*P1*	*P1*	*P1*	*P1*	*P1*	*P3*	*P2*	*P2*	*P2*	*P5*	*P5*	*P4*	*P4*	*P4*	*P4*	*P4*

   Temps d'attente moyen = (0 + 6 + 4 + 8 + 2) / 5 = 4

2. On suppose que l'ordonnancement est fait suivant l'algorithme du **plus court temps restant préemptif**. Indiquez dans chaque case du diagramme de Gantt ci-dessous quel processus est affecté à l'UC à chaque pas de temps de l'exécution, puis indiquez en dessous le temps d'attente moyen. (1,5 points)

   Correction :

Date	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*
UC	*P1*	*P2*	*P3*	*P2*	*P2*	*P1*	*P1*	*P1*	*P1*	*P1*	*P5*	*P5*	*P4*	*P4*	*P4*	*P4*	*P4*

   Temps d'attente moyen = (4 + 1 + 0 + 8 + 2) / 5 = 3

3. On suppose que l'ordonnancement est fait suivant l'algorithme « round robin » avec un quantum de temps fixé à 2. Indiquez dans chaque case du diagramme de Gantt ci-dessous quel processus est affecté à l'UC à chaque pas de temps de l'exécution, puis indiquez en dessous le temps d'attente moyen. (1,5 points)

   Correction :

Date	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*
UC	*P1*	*P1*	*P2*	*P2*	*P3*	*P1*	*P1*	*P4*	*P4*	*P2*	*P1*	*P1*	*P5*	*P5*	*P4*	*P4*	*P4*

Temps d'attente moyen = ((3+3) + (1+5) + 2 + (3+5) + 4) / 5 = 5,2

4. On définit ainsi un algorithme d'ordonnancement à deux niveaux de priorité :
   — Le niveau 0 obéit à un ordonnancement « round robin », quantum 2 : entre eux, les processus de priorité 0 suivent cet ordonnancement.
   — Le niveau 1 obéit à un ordonnancement « plus court d'abord » non préemptif (en cas d'égalité, le plus ancien dans la file obtient le processeur)
   — Le niveau 0 a priorité sur le niveau 1 : tous les processus de priorité 0 sont toujours prioritaires sur ceux de priorité 1, et ce de manière préemptive.

   Les priorités de nos processus sont définis de la manière suivante :

   P1 durée : 6, date 0, **priorité 1**

   P2 durée : 3, date 1, **priorité 0**

   P3 durée : 1, date 2, **priorité 1**

   P4 durée : 5, date 4, **priorité 1**

   P5 durée : 2, date 8, **priorité 0**

   Indiquez dans chaque case du diagramme de Gantt ci-dessous quel processus est affecté à l'UC à chaque pas de temps de l'exécution, puis indiquez en dessous le temps d'attente moyen. (1,5 points)

   Correction :

Date	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*
UC	*P1*	*P2*	*P2*	*P2*	*P3*	*P1*	*P1*	*P1*	*P5*	*P5*	*P1*	*P1*	*P4*	*P4*	*P4*	*P4*	*P4*

   Temps d'attente moyen = ((4+2) + 0 + 2 + 8 + 0) / 5 = 3,2

5. Quel est le meilleur algorithme suivant le critère du temps d'attente moyen ? (0,5 point)

   Correction : *En temps d'attente moyen, c'est le plus court préemptif (d'après le cours et confirmé par les valeurs obtenues sur l'instance).*

6. Quel est le meilleur algorithme suivant le critère du temps d'attente min-max ? (0,5 point)

   Correction : *Le temps d'attente max est le même pour les quatre algorithmes.*

7. Quel est le meilleur algorithme parmi les trois algorithmes préemptifs que nous avons proposés (plus court d'abord préemptif, round-robin et algorithme à deux niveaux de priorité) pour le critère du **temps de rotation** ? (1,5 points)

   Correction : *Il faut regarder les dates de début et de fin pour en déduire la durée :*

algo	*P1*	*P2*	*P3*	*P4*	*P5*	*moyenne*	*max*
PCNP	*6*	*3*	*1*	*5*	*2*	*3,4*	*6 (P1)*
PCP	*10*	*4*	*1*	*5*	*2*	*4,4*	*10 (P1)*
RR	*12*	*8*	*1*	*10*	*2*	*6,6*	*12 (P1)*
mixte	*12*	*3*	*1*	*5*	*2*	*4,6*	*12 (P1)*

   Si on se base sur le temps de rotation moyen, c'est l'algorithme plus-court (préemptif). Si on se base sur le min des max, c'est aussi cet algorithme.

Exercice 3 – Synchronisation (4 points)

On propose les fonctions suivantes, écrites en C, pour implémenter un MUTEX sur 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. Donner un example d'exécution à deux processus pour lequel il se produit un inter blocage. (1 point)

   Correction : *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. Récrivez les fonctions `entrer_section_critique` et `sortir_section_critique` de telle sorte qu'il n'y ait plus d'interblocage possible. (1 point)

   Correction : *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.*

3. Quelle propriété parmi l'exclusion mutuelle, le déroulement et la vivacité n'est pas vérifiée par le mécanisme que vous avez proposé en question 2 ? (0.5 point)

   Correction : *La vivacité (les deux autres sont vérifiées) car rien ne garantit qu'un processus va acquérir la main (si l'autre passe son temps à re-rentrer en SC).*

4. Nous souhaitons écrire un nouveau mécanisme « équitable » qui garantit les trois propriétés pour un système à deux processus. En plus du MUTEX, de quoi avons-nous besoin ? (0.5 point)

   Correction : *L'idée est d'implémenter un sémaphore avec une file d'attente à un seul élément. Il nous faut donc un block(n) et un wakeup(n), en plus de entrerSC(n) et sortirSC(n).*

5. Écrivez en langage C les fonctions de votre mécanisme d'exclusion équitable. (1 point)

   Correction : *Voir le code ci-après.*

   ```
   int s = 0; // taille du sémaphore
   int prochain = -1; // "liste" d'attente

   void acquerir(int n) {
     /* n = numéro proc. appelant */
     entrerSC(n);
     while (s>0) {
       prochain = n;
       block(n);
   ```

```
  }
  s++;
  sortirSC(n);
}

void relacher(n) {
  entrerSC(n);
  s--;
  if (S>0) {
    wakeup(prochain);
    prochain = -1;
  }
  sortirSC(n);
}
```

Exercice 4 – Question de cours (3 points)

1. Quelle est la différence entre la multiprogrammation et le temps partagé (1 point)

   Correction : *La multiprogrammation consiste à permettre à plusieurs processus d'utiliser en parallèle des ressources différentes du système (par exemple, l'UC pour p_1 et le disque pour p_2). Le temps partagé permet à plusieurs processus de s'exécuter « en parallèle » (de manière entrelacée) sur l'UC.*

2. Qu'est-ce qui permet, dans un OS, de mettre en œuvre la multiprogrammation et le temps partagé ? (1 point)

   Correction : *Pour la multiprogrammation, le cycle de vie du processus prévoit un état E/A pour les processus en attente d'une E/S sur un périphérique (en complément de la file d'attente pour les processus prêts) ; pour le temps partagé, c'est l'utilisation d'un ordonnanceur.*

3. Quelle est la fonction de l'édition de liens ? (1 point)

   Correction : *L'édition de lien permet de passer d'une adresse symbolique à une adresse logique, par exemple lorsqu'on utilise des bibliothèques.*