Partiel – 26 février 2013

N. Sabouret & A. Methni

L'épreuve dure 1h30. Tous les documents sont autorisés. Les exercices sont indépendants.

1 Exercice 1 – Question de cours (3 points)

1. Qu'est-ce qu'un système d'exploitation? (1 point)

   Correction: *c'est un programme (ou ensemble de programmes) qui fait l'intermédiaire entre la partie matériel et les applications.*

2. Quelle est la différence entre une thread et un processus? (1 point)

   Correction: *un processus peut contenir plusieurs threads qui partagent le même code, le même tas et le même environnement, mais qui ont des piles séparées.*

3. Quel est l'intérêt de la pagination à double niveau?

   Correction: *elle permet de charger des tables de pages plus petites. Seules les tables de pages correspondant aux méta-pages utilisées seront chargées. Par exemple, si on a des tables de pages de 4Ko, si toutes les pages utilisées par le processus sont dans le même groupe de pages, alors on ne chargera qu'une table de pages, au lieu de charger toute la table complète en pagination simple niveau.*

2 Exercice 2 – Processus (5 points)

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

P1 durée: 9, date 0, priorité 3

P2 durée: 7, date 2, priorité 3

P3 durée: 4, date 2, priorité 1

P4 durée: 8, date 4, priorité 2

P5 durée: 2, date 6, priorité 4

1. Dessinez un diagramme de Gantt correspondant au résultat d'un ordonnancement préemptif plus court d'abord (avec remise en fin de file) et indiquez le temps d'attente moyen. (2 points)

   Correction:

   $$^{0}P1^{2}P3^{6}P5^{8}P2^{15}P1^{22}P4^{30}$$

   temps d'attente moyen = (13 + 6 + 0 + 18 + 0) = 7,4

Attention : la priorité n'est pas prise en compte (le plus court d'abord préemptif signifie qu'un processus court prend la main à un plus long, pas que les processus sont gérés sur la priorité).

2. Dessinez un diagramme de Gantt correspondant au résultat d'un ordonnancement « round robin » avec un quantum de temps fixé à 2 et indiquez le temps d'attente moyen. (2 points)

   Correction:

   $$^0P1^2P2^4P3^6P1^8P4^{10}P2^{12}P5^{14}P3^{16}P1^{18}P4^{20}P2^{22}P1^{24}P4^{26}P2^{27}P1^{28}P4^{30}$$

   À t=2, P2 et P3 sont arrivés, P2 prend la main et P1 est placé derrière P3.

   À t=4, P4 est arrivé, P3 prend la main et la file est constituée de P1, P4, P2.

   À t=6, P5 est arrivé, P1 prend la main et la file est P4 (reste 8), P2 (reste 5), P5 (reste 2), P3 (reste 2)

   À t=8, il reste 5 pour P1 (mis en fin de file) et on dépile le tourniquet.

   temps d'attente moyen = (19 + 18 + 10 + 18 + 6) = 14,6

   Notons que la priorité n'est pas prise en compte.

3. Quel est le meilleur algorithme suivant le critère du temps d'attente moyen? Du temps d'attente min-max? (1 point)

   Correction: *dans les deux cas, c'est le plus court préemptif.*

3 Exercice 3 – Allocation mémoire contigüe (2 points)

On se place dans un système de mémoire de 1700 Ko de mémoire haute (c'est-à-dire au delà de la partie utilisée par l'OS) répartie en cinq partitions de 100Ko, 500Ko, 200Ko, 300Ko et 600Ko (dans cet ordre).

1. On suppose que le système d'exploitation doit allouer des processus de taille 212Ko, 417Ko, 112Ko et 426Ko (dans cet ordre). Pour chacun des algorithmes suivants, donnez l'allocation obtenue et le taux de fragmentation:

   - First-fit (prochain bloc libre)
   - Best-fit (plus petit bloc libre)
   - Worst-fit (plus grand bloc libre)

   Quel algorithme utilise le plus efficacement la mémoire sur cet exemple?

   Correction: *on note P1 (212), P2 (417), P3 (112), P4 (426) les processus.*

   - *First-fit:*
     - *Etat initial : 100 500 200 300 600*
     - *Placement de P1 : 100 P1+288 200 300 600*
     - *Placement de P2 : 100 P1+288 200 300 P2+183*
     - *Placement de P3 : 100 P1+288 P3+88 300 P2+183*
     - *Placement de P4 : impossible tant que P1 ou P2 n'a pas libéré sa zone*
   - *Best-fit:*
     - *Etat initial: 100 500 200 300 600*

- *État final: 100 P2+83 P3+88 P1+88 P4+174*

Taux de fragmentation = (100+83+88+88+174)/(100+500+200+300+600) = 533/1700 $\simeq$ 0.3

- *Worst-fit:*
  - *Etat initial : 100 500 200 300 600*
  - *Placement de P1 : 100 500 200 300 P1+388*
  - *Placement de P2 : 100 P2+83 200 300 P1+388*
  - *Placement de P3 : 100 P2+83 200 P3+188 P1+388*
  - *Placement de P4 : impossible*

La meilleure allocation ici est best-fit.

Remarque: le taux de fragmentation n'a pas été noté (car il y a le piège dans les cas first-fit et worst-fit). Les erreurs de calcul (par exemple $500 - 212 = 280$) n'ont pas été pénalisées. Enfin, nous avons accepté que P3 soit mis avec P1 dans les cas BF et WF (cela ne change pas le résutlat mais, en théorie, on alloue un seul processus par partition).

4 Exercice 4 – Pagination (6 points)

1. Expliquez pourquoi les tailles de pages sont toujours une puissance de 2. (1 point)

   Correction: *si ça n'était pas le cas, il y aurait de bouts de mémoire qui ne seraient pas référençables sans « sortir » de la page, puisque les adresses sont de la forme 2^{offset}, donc de la fragmentation inutile.*

2. On suppose un espace d'adresses logiques de huit pages de 1024 octets chacune, représenté dans une mémoire physique de 32 cadres de pages. Combien de bits comporte l'adresse logique? L'adresse physique? Expliquez. (1 point)

   Correction: $1024 = 2^{10}$ *donc 10 bits pour l'offset dans la page,* $8 = 2^3$ *donc 3 bits pour le numéro de page et* $32 = 2^5$ *donc 5 bits pour les cadres de pages. Adresse logique = 13 bits, adresse physique = 15 bits.*

3. On suppose maintenant un système de 2048 Ko de mémoire haute organisé avec des pages de 8Ko. Décrivez le système d'adressage logique. Quelle est la taille maximum de la table des pages? Expliquez. (1 point)

   Correction: $2048Ko/8Ko = 2^{11}/2^3 = 2^8$ *pages. Pour adresser* 2^8 *pages, il faut 1 octet. On a donc une table des pages qui peut faire* 2^8 *octets = 256o (par processus). Si on rajoute le bit de validité, il faut* $2^8 * 9$ *bits.*

   Il y avait une erreur dans l'énoncé (2096 au lieu de 2048). Nous avons donc donné tous les points aux étudiants qui ont fait le calcul: 2096/8 = 262 donc 9 bits par adresse donc $2^9 \times 9$ *bits = 4608 bits (576 octets). Nous avons aussi donné tous les points à tous ceux qui ont deviné qu'il s'agissait de 2048.*

4. On suppose que, dans le système de la question précédente, on a trois processus qui s'exécutent sur le système: P1 nécessistant 200Ko (code, données et pile), P2 de 545 Ko et P3 de 337 Ko. Quelle est la quantité de mémoire réellement utilisée par l'exécution de ces trois processus? Quel est le taux de fragmentation? Expliquez (1,5 point)

   Correction: *P1 nécessite 25 cadres de pages, plus sa table de pages qui fait donc 25 octets (1 par page à adresser). P2 nécessite 69 cadres, plus 69 octets. P3 nécessite 43*

pages, plus 43 octets. Donc le total est de 25+69+43 = 137 pages de 8Ko, plus les 137 octets, soit $137 * 2^{13} + 137 = 1122441 octets$ *(un peu plus de 1Mo).*

Fragmentation = 0Ko perdus pour P1 + 7Ko perdus pour P2 + 7Ko perdus pour P3 (on néglige les octets des tables de pages) soit 14Ko.

5. En considérant les huit premières entrés de la table de page présentée par la figure suivante, donner les adresses logiques correspondantes aux adresses physiques 33792 et 66048? Expliquez. (1,5 point)

N° cadre de page	N° de page	Bit de présence/absence
7	0	0
6	0	0
5	0	1
4	1	1
3	0	0
2	0	0
1	2	1
0	3	1

Correction: *Attention: il y a un oubli dans l'énoncé. Si les cadre de pages font 8Ko, on ne peut pas adresser au delà de 32Ko (l'exercice était repris du TD 5 où les cadres de pages font 64Ko). Aucun étudiant n'a signalé ce problème, ce qui n'est pas bon signe sur votre compréhension de la pagination...*

La correction ci-après utilise des cadres de pages de 64Ko (2^{16}*) mais nous avons compté bon les éventuels calculs faits avec des pages de 8Ko. Le barème a été revu pour tenir compte de cette erreur.*

@physique = (n° cadre, offset)

@physique (divisée par) Taille du cadre = N° du cadre, le reste donne le déplacement

@physique 33792/2^{16} *=> Numéro du cadre = 0 et déplacement = 33792*

Note: avec 2^{13} : numéro du cadre = 4 et déplacement = 1014, mais le cadre 4 n'existe pas dans l'énoncé...

Numéro de page = 3 => @logique $= 3 * 2^{16} + 33792 = 230400$

@physique 66048/2^{16} *=> Numéro du cadre = 1 et déplacement = 512*

Note: avec 2^{13} : numéro du cadre = 8 et déplacement = 512, mais le cadre 8 n'existe pas dans l'énoncé...

Numéro de page = 2 => @logique $= 2 * 2^{16} + 512 = 131584$

5 Exercice 5 – Processus en C (4 points)

1. Écrivez un programme C qui affiche "bonjour", crée deux processus P1 et P2 et dit "au revoir". P1 doit attendre deux secondes puis afficher les 10 premiers entiers avant de se terminer. P2 doit attendre la fin de P1 puis lister le contenu du répertoire courant.

Correction:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
int main() {
```

```
  pid_t fils;
  int i;
  printf("Bonjour\n");
  fils = fork();
  if (fils==-1) {
    fprintf(stderr, "fork failed");
    return 1;
  }
  if (fils==0) /* c'est le fils */ {
    fils = fork();
    if (fils==-1) {
      fprintf(stderr, "fork failed");
      return 2;
    }
    if (fils==0) /* c'est le fils du fils (le second fils) */ {
      sleep(2);
      for(i=0;i<10;i++)
printf("%d\n",i);
    } else /* c'est toujours le fils */ {
      int r;
      waitpid(fils, &r, 0);
      execlp("ls", "ls_fils", NULL);
    }
  } else /* c'est le père */ {
    printf("Au revoir\n");
  }
  return 0;
}
```

Nous n'avons pas retiré de point si les fonctions n'ont pas exactement le bon nom ou le bon appel (ex: execlp ou execv, erreurs dans des formats de printf, oublis de paramètres, erreurs de point-virgule). Ce qui compte, c'est 1) la structure avec un fils qui crée un fils, 2) les bons tests sur les valeurs de retour de fork, et 3) l'utilisation de waitpid.