Partiel – 18 février 2014

R. Bonaque, H. Ding, M. Gleize, N. Sabouret

L'épreuve dure 2h00. Tous les documents sont autorisés. Les exercices sont indépendants.

Exercice 1 – Question de cours (2,5 points)

1. Dessinez sur un schéma les composants matériels principaux d'un ordinateur. Les périphériques externes seront représentés par un bloc unique. (1,5 points)

   Correction : *Conformément à ce qui a été vu en cours, le schéma doit comporter :*
   - *bus*
   - *processeur (avec UC, horloge, cache)*
   - *RAM*
   - *contrôleurs matériel (DMA ou autres)*
   - *périphérique externes*

2. Rajoutez un bloc pour les processus et un pour l'OS sur votre schéma puis indiquez à l'aide de flèches nommées et facilement différentiables la provenance et la destination d'une exception logicielle, d'une exception matérielle et d'un appel système. (1 point)

   Correction :
   - *L'exception matérielle va d'un périphérique externe à l'OS (on peut admettre qu'elle va au processeur dans la mesure ou c'est d'un point de vue strictement matériel vrai).*
   - *L'exception logicielle vient d'un processus (à la limite du processeur) et va à l'OS (à la limite, au processeur).*
   - *L'appel système vient du processus et va à l'OS (on n'admettra pas d'approximation).*

Exercice 2 – Ordonnancement (4 points)

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

P1 durée : 8, date 0

P2 durée : 3, date 2

P3 durée : 4, date 4

P4 durée : 1, date 4

P5 durée : 4, date 12

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. (1,5 points)

   Correction :

   $$^0P1^2P2^5P4^6P3^{10}P1^{16}P5^{20}$$

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

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 :

   $$^0P1^2P2^4P1^6P3^8P4^9P2^{10}P1^{12}P3^{14}P5^{16}P1^{18}P5^{20}$$

   À t=2, P2 est arrivé, prend la main et P1 est placé en tête de file (reste 6)
   À t=4, P3 et P4 sont arrivés, P1 prend la main, puis c'est P3 puis P4.
   À t=9, P4 est terminé. La file est constituée de P2 (1), P1 (4) et P3 (2). P2 prend la main.
   À t=10, P2 est terminé. P1 prend la main.
   À t=12, P5 est arrivé. P1 est remis en fin de file. P3 prend la main (et la file est constituée de P5 (4) et P1 (2).
   À t=14, P3 est terminé. P5 prend la main, puis P1 et enfin P5 termine.
   Temps d'attente moyen = (10 + 5 + 6 + 4 + 4)/5 = 5,8

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

   Correction : *En temps d'attente moyen, c'est le plus court préemptif (3 contre 5,8). En min-max, c'est aussi le plus court préemptif (8 contre 10).*

Exercice 3 – Allocation mémoire contigüe (3 points)

On se place dans un système de mémoire de 1200 Ko de mémoire haute (c'est-à-dire au delà de la partie utilisée par l'OS) en partitionnement variable (swapping).

Des processus se présentent à la mémoire au temps 0 ms dans l'ordre suivant :

— P1 : 600Ko de mémoire, durée 2ms
— P2 : 150Ko de mémoire, durée 10ms
— P3 : 400Ko de mémoire, durée 2ms
— P4 : 300Ko de mémoire, durée 3ms
— P5 : 200Ko de mémoire, durée 4ms
— P6 : 400Ko de mémoire, durée 2ms

1. Déroulez l'algorithme d'allocation First-fit (prochain bloc libre) en précisant à chaque instant les allocations et désallocations qui sont effectuées, et donnez le taux de fragmentation au temps 3 ms (1 point)

   Correction : *L'algorithme commence par placer P1, P2 et P3 en mémoire, ce qui ne laisse plus que 50Ko de libre. On s'occupe donc de placer P4, P5 et P6 après que P1 et P3 aient été discardés de la mémoire après 2 ms.*
   — *A 2 ms : 600 P2 :150 450*
   — *Placement de P4 : P4 :300 300 P2 :150 450*
   — *Placement de P5 : P4 :300 P5 :200 100 P2 :150 450*

— *Placement de P6 : P4 :300 P5 :200 100 P2 :150 P6 :400 50*
Taux de fragmentation à 3 ms = (100 + 50)/1200 ≃ 0.13

2. Déroulez l'algorithme d'allocation Best-fit (plus petit bloc libre) en précisant à chaque instant les allocations et désallocations qui sont effectuées, et donnez le taux de fragmentation au temps 3 ms (1 point)

   Correction : *L'algorithme commence par placer P1, P2 et P3 en mémoire, ce qui ne laisse plus que 50Ko de libre. On s'occupe donc de placer P4, P5 et P6 après que P1 et P3 aient été discardés de la mémoire après 2 ms.*
   — *A 2 ms : 600 P2 :150 450*
   — *Placement de P4 : 600 P2 :150 P4 :300 150*
   — *Placement de P5 : P5 :200 400 P2 :150 P4 :300 150*
   — *Placement de P6 : P5 :200 P6 :400 P2 :150 P4 :300 150*
   Taux de fragmentation à 3 ms = (150)/1200 ≃ 0.13

3. Déroulez l'algorithme d'allocation Worst-fit (plus grand bloc libre) en précisant à chaque instant les allocations et désallocations qui sont effectuées, et donnez le taux de fragmentation au temps 3 ms (1 point)

   Correction : *L'algorithme commence par placer P1, P2 et P3 en mémoire, ce qui ne laisse plus que 50Ko de libre. On s'occupe donc de placer P4, P5 et P6 après que P1 et P3 aient été discardés de la mémoire après 2 ms. chacun des algorithmes suivants, et précisez le taux de fragmentation au temps 3 ms :*
   — *A 2 ms : 600 P2 :150 450*
   — *Placement de P4 : P4 :300 300 P2 :150 450*
   — *Placement de P5 : P4 :300 300 P2 :150 P5 :200 250*
   — *On est forcé d'attendre la fin de P4 (durée 3 ms) pour placé P6*
   — *A 5 ms : 600 P2 :150 P5 :200 250*
   — *Placement de P6 : P6 :400 200 P2 :150 P5 :200 250*
   Taux de fragmentation à 3 ms = (300 + 250)/1200 ≃ 0.46

Exercice 4 – Pagination (7,5 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 des bouts de mémoire qui ne seraient pas référençables sans « sortir » de la page, donc de la fragmentation inutile.*

2. On suppose un espace d'adresses logiques de 64 pages de 256 octets chacune, représenté dans une mémoire physique de 128 cadres de pages. Combien de bits comporte l'adresse logique ? L'adresse physique ? Expliquez. (1 point)

   Correction : $256 = 2^8$ *donc 8 bits pour l'offset dans la page,* $64 = 2^6$ *donc 6 bits pour le numéro de page et* $128 = 2^7$ *donc 7 bits pour les cadres de pages. Adresse logique = 14 bits, adresse physique = 15 bits.*

3. On suppose maintenant un système de 4096 Ko de mémoire haute organisée avec des pages de 32Ko (un seul niveau de pagination). Décrivez le système d'adressage logique. Quelle est la taille maximum de la table des pages ? Expliquez. (1 point)

   Correction : $4096Ko/32Ko = 2^{12}/2^5 = 2^7$ *pages. Pour adresser* 2^7 *pages, il faut 7 bits. On a donc une table des pages qui peut faire* $2^7 * 7$ *octets = 112o (par processus). Si on rajoute le bit de validité, il faut* $2^7 * 8$ *bits, soit 128 octets.*

4. On suppose que, dans le système de la question précédente, on a trois processus qui s'exécutent sur le système : P1 nécessitant 1250Ko (code, données et pile), P2 nécessitant 100 Ko et P3 nécessitant 200 Ko. Quelle est la quantité de mémoire réellement utilisée par l'exécution de ces trois processus ? Quel est le taux de fragmentation ? Expliquez. (2 points)

   Correction : *P1 nécessite 40 cadres de pages, plus sa table de pages qui fait 40 octets (8 bits par page à adresser). P2 nécessite 4 cadres, plus 32 bits. P3 nécessite 7 pages, plus 7 octets. Donc le total est de 40+4+7 = 51 pages de 32Ko, plus les 51 octets des tables, soit* $51 * 32 * 1024 * 8 + 51 = 1671219$ *octets (environ 1.6Mo).*
   Fragmentation = 30Ko perdus pour P1 + 28Ko perdus pour P2 + 24Ko perdus pour P3 (on néglige les octets des tables de pages) soit 82Ko.

5. En considérant les huit premières entrées de la table de page présentée par la figure suivante :

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

   et en supposant une taille de cadre de page de 32Ko, donner les adresses logiques correspondantes aux adresses physiques 31792 et 90348 ? Expliquez. (2,5 points)

   Correction : *@physique = (No cadre,offset)*
   @physique (divisée par) Taille du cadre = No du cadre, le reste donne le déplacement
   *@physique 31792/(32*1024) => Numéro du cadre = 0 et déplacement = 31792*
   *Numéro de page = 3 => @logique = 3*32768 + 31792 = 130096*
   @physique 90348/32768 => Numéro du cadre = 2 et déplacement = 24812
   *Numéro de page = 1 => @logique = 1*32768 + 24812 = 57580*

Exercice 5 – Estimation du temps UC (3 points)

Un certain nombre d'algorithmes d'ordonnancement on besoin d'une estimation du temps processeur que va prendre le prochain cycle d'un processus. On supposera, comme dans le cours, que cette estimation est la moyenne exponentielle des cycles précédents, avec un facteur de pondération α. On note d_n l'estimation du temps pris par le cycle n et t_n son temps réel. On a donc (comme vu en cours) :

$$d_n = \alpha d_{n-1} + (1 - \alpha) t_{n-1}$$

1. Exprimez d_n en fonction de d_0 de α et des $(t_i)_{i \in \mathbb{N}}$. (1 point)

   Correction : $d_n = \alpha^n d_0 + (1-\alpha) \sum_{k=0}^{n-1} alpha^{n-1-k} t_k$ *On peut accepter une expression avec des ... si elles contient au moins le premier et le dernier terme de la somme.*

2. On rappelle que pour $x \in \mathbb{R}$ $exp(x) = \sum_{k=0}^{+\infty} \frac{x^k}{k!}$. Vers quelle valeur l'expression $\frac{d_n}{\alpha^n}$ tend elle si $d_0 = 0$ et $t_i = \frac{1}{i!}$ ms pour tout $i \geqslant 0$? (1 point)

   Correction : *Dans ces conditions* $d_n = \alpha^n 0 + (1-\alpha) \sum_{k=0}^{n-1} alpha^{n-1-k} \frac{1}{k!}$ $ms = (1-\alpha)\alpha^{n-1} \sum_{k=0}^{n-1} \frac{1/alpha^k}{k!}$ ms.

   D'où $\frac{d_n}{\alpha^n} = (1-\alpha)\alpha \sum_{k=0}^{n-1} \frac{1/alpha^k}{k!}$ *qui tend vers* $(1-\alpha)\alpha exp(\frac{1}{\alpha})$ ms

3. Que se passerait-il avec ce processus si l'ordonnanceur avait comme stratégie « le plus court d'abord » ? Cela vous parait-il possible pour un processus réel, se passerait-il la même chose avec un « round-robin » ? (1 point)

   Correction : *Cela permet plusieurs observations :*
   - *comme le temps estimé diminue très vite l'ordonnanceur finit par toujours donner la priorité à ce processus : il y a famine pour les autres processus.*
   - *cela n'est pas possible tel quel dans le cas réel car une exécution d'un processus dure toujours au moins une instruction donc le temps réel ne peut pas descendre infiniment, néanmoins rien s'empêche un processus de ne faire qu'un petit nombre d'instructions par cycle et de mettre en famine les autres, c'est un des inconvénients du « plus court d'abord ».*
   - *avec un round-robin on n'aurait pas se problème car il est par nature équitable : après un cycle (très court à priori) de ce processus les autres processus peuvent exécuter*