

# Examen Intra

IFT 2245

20 février 2024

## Directives

- Vous avez droit à une page de notes (recto verso) écrite à la main (vous pouvez conserver votre feuille après l'examen).
- Calculatrice est autorisée.
- Répondez dans le cahier fourni à l'exception de **Question 1** que vous pouvez répondre sur cette feuille.
- Le pointage pour chaque question est entre parenthèses (total = 20).
- Les traductions en anglais sont en *italics*.
- Vous pouvez répondre aux questions en anglais ou en français.
- Notez clairement toutes les suppositions que vous faites.

### Question 0. Nom et matricule (1 point de bonus)

Ecrivez votre nom et votre matricule sur ce papier et sur votre cahier d'examen.

Liam Powell 1000000

### Question 1. Questions à choix multiple (3 points)

(a) (0.5 points) Parmi les trois choix ci-dessous, lequel est un exemple d'une abstraction, lequel est un exemple d'un mécanisme et lequel est un exemple d'une politique (écrivez les mots «abstraction», «mécanisme» et «politique» dans les espaces vides)

- A. Premier arrivé premier servi (*First come first served*) (Politique)  
B. Processus (Abstraction)  
C. Création de processus (Mécanisme)

(b) (0.5 point) Un fil (*thread*) parent et son fil (*thread*) enfant partagent (cochez tout ce qui s'applique) :

- Les registres (*registers*)
- La pile (*stack*)
- Le tas (*heap*)
- Les variables globales (*global variables*)

(c) (0.5 points) Parmi les trois systèmes d'exploitation ci-dessous, lequel a une structure monolithique, lequel a une structure modulaire et lequel a une structure micro-noyau (*microkernel*) (écrivez les mots «monolithique», «modulaire» et «microkernel» dans les espaces vides)

- A. Mach ( microkernel )
- B. MS-DOS ( monolithique )
- C. Linux ( modulaire )

(d) (0.5 points) En général, une fois qu'un système entre dans un état pas sûr (*unsafe state*), il est garanti qu'il finira par se retrouver dans une interblocage (*deadlock*).

- A. Vrai
- B. Faux

(e) (0.5 points) Lequel des énoncés suivants est vrai ?

- A. Nous pouvons implémenter la fonction `test_and_set` avec la fonction `compare_and_swap`
- B. Nous pouvons implémenter la fonction `compare_and_swap` avec la fonction `test_and_set`
- C. A. et B. sont vrai (les deux fonctions sont équivalentes)
- D. Rien de ce qui précède n'est vrai

(f) (0.5 points) Après qu'un processus ait été **interrompu** (*interrupted*) dans quel état est-il ?

- A. prêt (*ready*)
- B. nouveau (*new*)
- C. en exécution (*executing*)
- D. en attente (*waiting*)

## Question 2. Synchronization (4 points)

Dans le cas d'un problème de lecteur-écrivain (*reader-writer*), nous souhaitons autoriser plusieurs lecteurs simultanément. Voici une solution qui permet toujours à un nouveau processus de lire s'il existe déjà un processus en lecture :

La structure du lecteur :

```
do {
    wait(mutex);
    read_count++;
    if (read_count == 1)
        wait(rw_mutex);
    signal(mutex);
    ...
    /* reading is performed */
    ...
    wait(mutex);
    read_count--;
    if (read_count == 0)
        signal(rw_mutex);
    signal(mutex);
} while (true);
```

*mutex w\_mutex // initialisé à 1*

```
lecteur :
do {
    wait (w_mutex);
    if (write_count > 0)
        signal (w_mutex);
    wait (rw_mutex);
else
    wait (r_mutex)
    read_count++
    signal (r_mutex)
    // read
    wait (r_mutex)
    read_count--
    if (read_count == 0)
        signal (rw_mutex)
    signal (r_mutex)
}
```

écrivain :

```

do {
  wait (w-mutex);
  write-count ++ ;
  signal (w-mutex);
  wait (rw-mutex);
  // write
  wait (w-mutex);
  write-count -- ;
  signal (w-mutex);
  signal (rw-mutex);
}

```

La structure de l'écrivain :

```

do {
  wait(rw_mutex);
  ...
  /* writing is performed */
  ...
  signal(rw_mutex);
} while (true);

```

Mais cela peut causer une famine pour les processus d'écriture. Une autre option serait de donner la priorité aux processus d'écriture de sorte que s'il y a un processus d'écriture en attente, aucun nouveau processus de lecture ne soit autorisé à lire. Cela pourrait toutefois entraîner une famine pour les processus de lecture. Comment implémenteriez-vous cette deuxième option où les processus d'écriture ont la priorité (écrire le pseudo-code)?

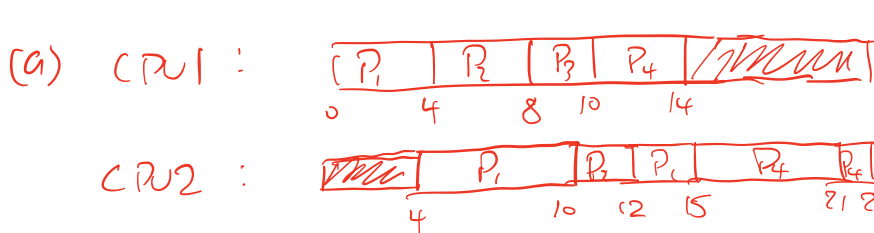
**Question 3. L'ordonnancement (5 points)**

Pour un ensemble de processus  $P_{1..4}$ , nous anticipons la situation suivante :

Processus	CPU burst time(ms)
$P_1$	13
$P_2$	6
$P_3$	2
$P_4$	11

Nous allons ordonnancer l'exécution de ces processus avec une queue multi-niveaux à rétroaction (*multi-level feedback queue*). Dans ce cas, il y aura deux niveaux dans la hiérarchie, et chaque niveau sera traité par un CPU distinct (ils pourront fonctionner en parallèle). Chaque niveau utilise une politique d'ordonnancement tourniquet (*round robin*), le premier niveau (sur CPU1) avec un quantum de 4 ms et le second (sur CPU2) avec un quantum de 6 ms. Autrement dit, chaque processus sera traité par le premier niveau (round robin avec  $q = 4$  ms), et s'il ne se termine pas après un quantum, il passera au deuxième niveau (round robin avec  $q = 6$  ms), où il restera jusqu'à son terminaison.

- (a) (2 points) Dessinez les diagrammes de Gantt pour les deux processeurs CPU1 et CPU2.
- (b) (1 point) Quelle est l'utilisation du CPU (*CPU utilization*) pour chacun des deux processeurs CPU1 et CPU2 (entre le moment où le premier processus commence à s'exécuter et le moment où le dernier processus termine son exécution) ?
- (c) (1 point) Quel est le temps de réponse moyen (*average response time*) pour les quatre processus ?
- (d) (1 point) Quel est le temps d'attente moyen (*average waiting time*) pour les quatre processus ?



(b) CPU1  $14/22$   
 CPU2  $18/22$

(c)  $\frac{1}{4} (0 + 4 + 8 + 10) = 22/4$

(d)  $\frac{1}{4} (2 + (4+2) + 8 + (10+1)) = 27/4$

**Question 4. Forks et Threads (5 points)**

(a) (3 points) Qu'est qui est imprimé quand ce code est exécuté :

```
int value = 0;
void *runner(void *param);
int main(int argc, char *argv[])
{
    pid_t pid;
    pthread_t tid;
    pthread_attr_t attr;
    pid = fork();
    if (pid == 0) {
        pthread_attr_init(&attr);
        pthread_create(&tid,&attr,runner,NULL);
        pthread_join(tid,NULL);
        printf("LINE A value = %d\n",value);
    }
    else if (pid > 0) {
        wait(NULL);
        printf("LINE C value = %d\n",value);
    }

    pthread_attr_init(&attr);
    pthread_create(&tid,&attr,runner,NULL);
    pthread_join(tid,NULL);
    printf("LINE D value = %d\n", value);
}

void *runner(void *param) {
    value += 5;
    pthread_exit(0);
}
```

LINE A value = 5  
LINE D value = 10  
LINE C value = 0  
LINE D value = 5

(b) (2 points) Écrivez une équation pour décrire le nombre de fois que le “bonjour” sera imprimé en fonction de la variable  $N$  pour le code suivant :

```
int main()
{
    pid_t pid;
    int i = 1;
    int n = N; // N est remplacé par un entier
    for (;i<=n;){
        pid = fork();
        printf("bonjour \n"); // combien de fois cela est-il exécuté?
        i++;
    }
    return 0;
}
```

$$\sum_{i=1}^N 2^i = 2^{N+1} - 2$$

**Question 5. Interblocage (3 points)**

Considérons un système avec trois types de ressources, chacune avec un nombre différent d'unités totales :

$$A - 5, B - 7, C - 9$$

À un instant donné, il existe trois processus dont les allocations de ressources et les requêtes actuelles sont indiquées dans le tableau ci-dessous :

Processus	Alloué	Requêtes
	$A, B, C$	$A, B, C$
$P_0$	1, 2, 0	0, 1, 1
$P_1$	2, 0, 3	3, 6, 5
$P_2$	0, 1, 1	0, 0, 1
$P_3$	2, 2, 2	1, 0, 0
$P_4$	0, 2, 2	2, 0, 1

(a) (1 point) Quel est l'ordre d'exécution des processus qui démontre que nous ne sommes pas dans une interblocage (*deadlock*)?

(b) (1 point)  $P_3$  fait une requête pour une deuxième instance de type  $A$ , sommes-nous dans une interblocage (*deadlock*)?

(c) (1 points) Quel est le nombre maximum de chaque type de ressources qui pourraient être demandées par  $P_4$  (au-delà de ce qui est déjà demandé) sans que le système ne se retrouve dans une interblocage (*deadlock*)?

$$(a) P_2, P_0, P_3, P_4, P_1$$

$$(b) \text{ oui}$$

$$(c) (3, 5, 4) - (2, 0, 1) = \begin{matrix} (1, 5, 3) \\ A \quad B \quad C \end{matrix}$$