Commencé le	vendredi 22 décembre 2023, 09:32
État	Terminé
Terminé le	vendredi 22 décembre 2023, 12:52
Temps mis	3 heures 19 min
En retard	49 min 29 s
Note	**13,41** sur 20,00 (**67,03**%)

Question 1

Terminé

Non noté

Directives :

1. Cet examen est composé de 9 questions au total (Q1 à Q9) pour une durée totale de 2 heures 30 minutes.
2. Pondération 45%.
3. En guise de documentation, vous aurez accès à votre compte Moodle et, par conséquent, à la totalité du site du cours INF2610, incluant les diapos, etc. Aucune autre documentation n'est permise. Les ordinateurs personnels, tablettes, calculatrices et cellulaires ne sont pas permis.
4. Aucune réponse aux questions durant l'examen. En cas de doute sur la compréhension de l'énoncé d'une question, énoncez clairement dans votre réponse toutes vos suppositions. Vous pouvez également utiliser cette l'espace ci-après pour énoncer clairement vos suppositions. N'oubliez pas d'indiquer le numéro de la question. Nous tiendrons compte de toute supposition/interprétation sensée.
5. Tous les appels système utilisés dans les questions sont supposés exempts d'erreurs et considèrent leurs options par défaut.
6. Il n'est pas demandé de traiter les cas d'erreurs, ni d'inclure les directives d'inclusion dans les codes à compléter.
7. Pour les questions à développement, vous devez répondre directement dans les cadres réservés aux réponses. Vous ne pouvez pas joindre de fichiers.
8. Pour les questions à choix multiples, **vous devez sélectionner une seule réponse**. Vous ne pouvez pas joindre de fichiers.
9. Lisez au complet et attentivement chaque question avant d'y répondre.
10. Vous pouvez inclure vos commentaires au responsable du cours dans l'espace ci-après.

Bon congé!

Question 2

Terminé

Non noté

Sur mon honneur, j'affirme que je compléterai cet examen en vertu du code de conduite de l'étudiant de Polytechnique Montréal et de sa politique sur le plagiat. J'affirme également que je compléterai cet examen par moi-même, sans communication avec personne, et selon les directives diffusées sur les canaux de communication.

Écrivez votre nom complet ainsi que votre matricule en guise d'approbation dans la zone de texte ci-dessous.

Thomas Rouleau
2221053

Question 3

Terminé

Note de 1,50 sur 1,50

[Q1] (1.5 pt)

Complétez le programme suivant de manière:

- À générer un signal SIGUSR1 destiné au processus courant, et ce, à chaque itération de la boucle *while*; et
- À faire en sorte qu'au bout d'exactement 10 signaux, le programme se termine avec les deux lignes suivantes:

```
Sortie après 10 requetes...
compteur = 10.
```

Vos modifications peuvent consister:

- À insérer un gestionnaire de signal en <<A>>;
- À insérer en <<C>> une instruction qui émet un signal SIGUSR1 à l'intention du processus courant;
- À insérer tout autre code requis en <<B>>.

Toute autre modification au programme fourni est interdite.

NOTES:

- Vous n'avez pas à réécrire le code fourni; indiquez simplement vos ajouts de la manière la plus claire possible.
- Il ne vous est pas demandé de créer de nouveaux processus ou threads.

```c
#include <stdio.h>
#include <unistd.h>
#include <signal.h>

int compteur;
int sortie;

// <<A>> INSÉREZ VOTRE GESTIONNAIRE DE SIGNAL ICI

int main()
{
  pid_t pid = getpid();
  compteur = 0;
  sortie = 0;
  // <<B>> INSÉREZ (LE CAS ÉCHÉANT) VOS MODIFICATIONS
  //       AU MAIN() ICI, AVANT LA BOUCLE WHILE
  while(!sortie)
    // <<C>> INSÉREZ DANS LA BOUCLE WHILE UNE INTRUCTION
    //       QUI EMET UN SIGNAL SIGUSR1
  printf("compteur = %d.\n", compteur);
  return 0;
}
```

<<A>> Gestionnaire de signal

```
void action(int sig) {
  compteur += 1;
  if (compteur >= 10) {              // + grand ou égal comme sécurité
     sortie = 1;
     printf("Sortie après %d requêtes...", compteur)
  }
}
```

<<B>>

```
signal(SIGUSR1, action);
```

<<C>>

```
kill(pid, SIGURS1);
```

Commentaire :
Good job !

Question 4

Terminé

Note de 0,00 sur 2,50

[Q2] (2,5 pts)

Considérez la variante du problème de synchronisation des *philosophes* illustrée par la fonction *dinerPerpetuel* ci-après.

```
void * dinerPerpetuel(void *arg)
{
  long i = (long) arg;
  while(1) {
    randomSleep(); // penser (duree aleatoire <= 1 microseconde)
    // DEBUT DE LA ZONE MODIFIABLE
    sem_wait (&f[i]);
    sem_wait (&f[(i+1)%N]);
    // FIN DE LA ZONE MODIFIABLE
    printf ("le philosophe %ld mange\n", i);
    randomSleep(); // manger (duree aleatoire <= 1 microseconde)
    printf ("le philosophe %ld a fini de manger\n", i);
    sem_post (&f[i]);
    sem_post (&f[(i+1)%N]);
  }
}
```

Vous avez créé un programme qui utilise cette variante et en confie l'exécution concurrente à 7 threads numérotés de 0 à 6. La fonction *dinerPerpetuel* est appelée par la boucle suivante, où N est égal à 7:

```
for (i = 0; i < N; ++i) {
  pthread_create(&tids[i], NULL, dinerPerpetuel, (void*)i);
}
```

a) (1 pt) À l'exécution de votre programme, vous constatez la présence d'une situation d'interblocage. Illustrez clairement, au moyen d'un *diagramme* ou d'un *tableau*, par quel autre thread chacun des 7 threads est bloqué (par exemple: le thread X est bloqué par le thread Y, etc). Complétez par une courte explication.

b) (0,5 pt) Indiquez quelles modifications devraient être apportées à la ZONE MODIFIABLE de manière à éliminer le risque d'interblocage et à prévenir les famines.

c) (1 pt) Expliquez de manière brève mais claire comment le code résultant de votre réponse à b) permet de prévenir les famines.

a)

Commentaire :
Aucune réponse.

Question 5

Terminé

Note de 1,00 sur 1,50

[Q3] (3 pts)

Dans un système, *4* processus *(P1, P2, P3 et P4)* partagent, en exclusion mutuelle, *3* types de ressources *(R1,R2 et R3)* en quantités respectives *(9, 3, 6)* avec :

E : nombre de ressources maximum disponibles (pas chargé pas devant l'exécution)

E : nombre de ressources maximum disponibles (ne change pas durant l'exécution) :

R1	R2	R3
9	3	6

A : nombre de ressources disponibles à l'instant *S* de l'exécution :

R1	R2	R3
1	1	2

Alloc : tableau de l'allocation de chaque processus en fonction des ressources disponibles à l'instant *S* de l'exécution:

	R1	R2	R3
P1	1	0	0
P2	5	1	1
P3	2	1	1
P4	0	0	2

Req : tableau des ressources qui seront demandées par chaque processus dans le futur i.e. après *S*:

	R1	R2	R3
P1	2	2	2
P2	1	0	2
P3	1	0	3
P4	4	2	0

Supposez que l'état courant (i.e. à l'instant *S*) du système est sûr.

a) (1.5 pt) Supposez que le système utilise l'algorithme du banquier pour éviter les interblocages. À l'instant *S*, le système reçoit de la part du processus *P2* une demande de 1 ressource de type *R1* et 1 ressource de type *R3*. Expliquez pourquoi le système devrait accepter cette demande. Justifiez votre réponse en déroulant pas-à-pas l'algorithme du banquier.

Conseil: nous vous suggérons de copier les matrices pertinentes dans Excel, d'y résoudre le problème, et de recopier l'ensemble du résultat dans l'espace-réponse ci-dessous.

a)

Le système devrait accepter la demande puisque suite à l'allocation des ressources à P2 de cette demande le système se trouve dans un état sûr selon l'algorithme du banquier (un chemin sans interblacage demandant l'allocation de la totalité des requis de chaque processus existe) :

				Dispo				
			R1	R2	R3			
			1	1	2			
Alloc					Req			
	R1	R2	R3			R1	R2	R3
P1	1	0	0		P1	2	2	2
P2	5	1	1		P2	1	0	2
P3	2	1	1		P3	1	0	3
P4	0	0	2		P4	4	2	0
				Dispo				
			R1	R2	R3			
			0	1	1			
	R1	R2	R3			R1	R2	R3
P1	1	0	0		P1	2	2	2
P2	6	1	2		P2	0	0	1
P3	2	1	1		P3	1	0	3
P4	0	0	2		P4	4	2	0

	R1	R2	R3
	0	1	0

	R1	R2	R3		R1	R2	R3
P1	1	0	0	P1	2	2	2
P2	6	1	3	P2	0	0	0
P3	2	1	1	P3	1	0	3
P4	0	0	2	P4	4	2	0

	R1	R2	R3
	6	3	3

	R1	R2	R3		R1	R2	R3
P1	1	0	0	P1	2	2	2
P2	0	0	0	P2	0	0	0
P3	2	1	1	P3	1	0	3
P4	0	0	2	P4	4	2	0

	R1	R2	R3
	4	1	1

	R1	R2	R3		R1	R2	R3
P1	3	2	2	P1	0	0	0
P2	0	0	0	P2	0	0	0
P3	2	1	1	P3	1	0	3
P4	0	0	2	P4	4	2	0

	R1	R2	R3
	7	3	3

	R1	R2	R3		R1	R2	R3
P1	0	0	0	P1	0	0	0
P2	0	0	0	P2	0	0	0
P3	2	1	1	P3	1	0	3
P4	0	0	2	P4	4	2	0

	R1	R2	R3
	6	3	0

	R1	R2	R3		R1	R2	R3
P1	0	0	0	P1	0	0	0
P2	0	0	0	P2	0	0	0
P3	3	1	4	P3	0	0	0
P4	0	0	2	P4	4	2	0

	R1	R2	R3
	9	3	4

	R1	R2	R3		R1	R2	R3
P1	0	0	0	P1	0	0	0
P2	0	0	0	P2	0	0	0
P3	0	0	0	P3	0	0	0
P4	0	0	2	P4	4	2	0

	R1	R2	R3
	5	1	4

	R1	R2	R3		R1	R2	R3
P1	0	0	0	P1	0	0	0
P2	0	0	0	P2	0	0	0
P3	0	0	0	P3	0	0	0

P4	4	2	2	P4	0	0	0

	R1	R2	R3
	9	3	6

	R1	R2	R3		R1	R2	R3
P1	0	0	0	P1	0	0	0
P2	0	0	0	P2	0	0	0
P3	0	0	0	P3	0	0	0
P4	0	0	0	P4	0	0	0

Commentaire :
Le nombre de Ressources disponible (A) et de Req au départ est incorrect.

Question 6

Terminé

Note de 0,75 sur 0,75

b) (0,75 pt) Supposez toujours que le système utilise l'algorithme du banquier pour éviter les interblocages. Toujours à l'instant *S*, le système reçoit de la part du processus *P1* une demande de 1 ressource de type *R1* et 1 ressource de type *R3*. Expliquez pourquoi le système devrait refuser cette demande. Justifiez votre réponse en déroulant pas-à-pas l'algorithme du banquier.

Conseil: nous vous suggérons de copier les matrices pertinentes dans Excel, d'y résoudre le problème, et de recopier l'ensemble du résultat dans l'espace-réponse ci-dessous.

b)

Le système ne devrait pas accepter la demande puisque suite à l'allocation des ressources à P1 de cette demande le système se trouve dans un état non sûr selon l'algorithme du banquier (il n,existe pas de chemin d'allocation totale des requis à chaque processus permettant à tous de se compléter) : Cette allocation mène directement à un interblocage, il n'y a pas assez de ressources en fct des types permettant aux processus de se terminer.

	R1	R2	R3
	1	1	2

Alloc

	R1	R2	R3		R1	R2	R3
P1	1	0	0	P1	2	2	2
P2	5	1	1	P2	1	0	2
P3	2	1	1	P3	1	0	3
P4	0	0	2	P4	4	2	0

Dispo

	R1	R2	R3
	0	1	1

Alloc

	R1	R2	R3		R1	R2	R3
P1	2	0	1	P1	1	2	1
P2	5	1	1	P2	1	0	2
P3	2	1	1	P3	1	0	3
P4	0	0	2	P4	4	2	0

Rangée P1 req > Dispo => non
Rangée P2 req > Dispo => non
Rangée P3 req > Dispo => non
Rangée P4 req > Dispo => non

Commentaire :
Bravo

Question 7

Terminé

Note de 0,75 sur 0,75

c) (.75 pt) La question 7 du QCM sur l'interblocage nous a appris que la proposition suivante est fausse : *Un état non sûr mène forcément vers un interblocage*… Illustrez par un contre-exemple basé sur vos réponses précédentes.

Selon les notes de cours au chapitre 6 : "Un état est dit sûr si tous les processus peuvent terminer leurs exécutions dans le pire cas (c-à-d tous les processus demandent en même temps toutes les ressources manquantes à leurs exécutions complètes)" sur cette définition la question a) mène à un état non-sûr puisque la quantité de ressources disponibles ne peut pas permettre à tout les processus de les demander en même temps, ils doivent attendre qu'un autre processus se termine et libère ses ressources avant que leurs requêtes de ressources soit accepté par l'algorithme du banquier.

Ex tiré de a):

	R1	R2	R3
	6	3	3

	R1	R2	R3		R1	R2	R3
P1	1	0	0	P1	2	2	2
P2	0	0	0	P2	0	0	0
P3	2	1	1	P3	1	0	3
P4	0	0	2	P4	4	2	0

Le nombre de ressources R1 R2 et R3 ne peuvent pas permettrent `tout les processus d'avoir les ressources nécessaires, l'état est par définition non sûr, mais l'algorithme du banquier permet d'allouer les ressources à des processus dont cette allocation permet leurs complétion et la libération de ressources supplémentaire

Commentaire :
OK.

Question 8

Terminé

Note de 0,75 sur 2,00

[Q4] (2 pts)

Soit le code suivant :

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <pthread.h>
4   #include <string.h>
5   #include <errno.h>
6
7   int tour = 0;
8   int nb_threads = 3;
9
10  pthread_cond_t wq[3] ;
11  pthread_mutex_t acces = PTHREAD_MUTEX_INITIALIZER;
12
13  void *thread_code (void *param);
14
15  int main(int argc, char *argv[]) {
16
17      pthread_t thread_tcb[nb_threads];
18
19      /* Creer les threads */
20      for (int i = 0; i <= nb_threads-1; i++) {
21          wq[i] = PTHREAD_COND_INITIALIZER;
22          pthread_create (&thread_tcb[i], NULL, thread_code, (void *) i);
23      }
24      /* Attendre que les threads se terminent */
25      for (int i = 0; i <= nb_threads-1; i++)
26          pthread_join(thread_tcb[i], NULL);
27
28      printf ( "The threads finished\n");
29      return 0;
30  }
31
32  void *thread_code(void * param) {
33
34      int thread_no = reinterpret_cast<int> (param) ;
35      int mon_tour1, mon_tour2, mon_tour3;
36
37      switch (thread_no) {
```

```
38          case 0:
39              mon_tour1 = 0;
40              mon_tour2 = 3;
41              mon_tour3 = 6;
42              break;
43          case 1:
44              mon_tour1 = 1;
45              mon_tour2 = 4;
46              mon_tour3 = 7;
47              break;
48          case 2:
49              mon_tour1 = 2;
50              mon_tour2 = 5;
51              mon_tour3 = 8;
52              break;
53      }
54      int next_tour = mon_tour1;
55
56      for ( ; ; ) {
57          pthread_mutex_lock(&acces);
58          if (tour != next_tour) {
59              pthread_cond_wait (&wq[thread_no], &acces);
60          }
61          pthread_mutex_unlock(&acces);
62
63          // Section ou on mettrait le code principal a executer,
64          // representee ici par la commande printf suivante :
65          printf ("thread no %d execute function no %d\n", thread_no, next_tour);
66
67          pthread_mutex_lock(&acces);
68          tour =(tour+1) % nb_threads;
69          pthread_cond_signal(&wq[tour]);
70          pthread_mutex_unlock(&acces);
71
72          if (next_tour == mon_tour1)
73              next_tour = mon_tour2 ;
74          else if (next_tour == mon_tour2)
75              next_tour = mon_tour3;
76          else
77              next_tour = mon_tour1;
78      }
79      return NULL;
80  }
```

a) (1 pt) Décrivez ce que fait ce code en indiquant bien le rôle de *cond_t wq[3]* et *pthread_cond_wait*.

b) (1 pt) Faites la trace des 10 premiers *print*.

a)

L'on crée 3 threads exécutant la fonction thread_code et étant numéroté 0, 1 et 2. Ce numéro de thread définit l'état du switch case utilisé.

On utilise la valeur de tout pour déderminer le thread pour lequel il s'agit du tout en débutant par 0 pour ensuite incrémenter de 1 et faire le booléen de 3 afin de faire la séquence (0, 1, 2, 0, 1, 2, etc.)

Ainsi, lorsque que ce n'est pas le tour du thread il va attendre de recevoir le signal *pthread_cond_wait(&wq[],access) attend le signal correspondant `son numéro de thread et le signal access indiquant la possibilité d'accéder `la propriété*

Commentaire :
a) Pas d'explication de wq.

b) Aucune réponse.

Question 9

Terminé

Note de 0,50 sur 0,50

[Q5] (1 pt) - Synchronisation par variables conditionnelles VF

Répondez par vrai ou faux à la question suivante, et fournissez une courte justification au bas de la page.

a. (0,5 pt) Vrai ou faux: *Java et POSIX offrent la variable de condition de type Signal-and-Continue.*

- (●) Vrai
- ( ) Faux

La réponse correcte est « Vrai ».

Question 10

Terminé

Note de 0,00 sur 0,50

b. (0,5 pt) Vrai ou faux: *Dans le problème du producteur/consommateur, il n'y a aucun avantage à utiliser la variable de condition POSIX par rapport à l'approche utilisant des sémaphores compteurs et des mutex.*

- ○ Vrai
- ◉ Faux

La réponse correcte est « Faux ».

Commentaire :
Justification incorrecte. Les variables de conditions permettent de réveiller les threads en attente en cas ou les producteurs disparaissent de manière non attendue.

Question 11

Terminé

Non noté

Utilisez cet espace pour justifier chacune de vos réponses aux questions 5a et 5b. Si votre justification est absente ou erronée, vous aurez **0 point** peu importe votre réponse dans le QCM.

a)

Java utilise la sémantique signal-and-continue et POSIX également

b)

Les variables de conditions permettent d'assurer qu'un seul thread est actif à la fois dans un moniteurs. Il permet de placer les threads en attente passive plutôt qu'active

Question 12

Terminé

Note de 2,00 sur 3,00

[Q6] (3 pts) - Remplacement de pages pour mémoire virtuelle

a. (2 pts) Complétez les tableaux de remplacement de pages pour ces 3 politiques: FIFO, LRU et Horloge. Indiquez pour chacune des périodes et chacune des politiques les numéros de page pour les divers cadres ainsi que les fautes de page. (Veuillez vous référer aux instructions ci-après.)

INSTRUCTIONS:

i. Veuillez utiliser le gabarit en format Excel disponible **ICI**.
ii. Remplissez la feuille pour les trois politiques. Les cellules en couleur gris moyen peuvent prendre une valeur de 1 à 5, ou rester vides. Les cellules en gris foncé peuvent prendre la valeur "F" ou rester vides. Les autres cellules ne peuvent être modifiées.
iii. Pour remettre votre réponse, sélectionnez l'ensemble des trois tableaux en débutant dans le coin supérieur gauche, puis faites un copier-coller vers la zone de texte ci-après. *Vous n'avez pas à remettre le fichier Excel*.

b. (1 pt) Pourquoi affirme-t-on que la politique de l'horloge est un bon compromis entre celui du FIFO et celui du LRU? Fournissez une courte explication.

a)

2	3	2	1	5	2	4	5	3	2	5	2
2	2	2	2	5	5	5	5	3	3	3	3
	3	3	3	3	2	2	2	2	2	5	5
			1	1	1	4	4	4	4	4	2
F	F		F	F	F	F		F		F	F
					FIFO						

2	2	2	2	2	2	2	2	3	3	3	3
	3	3	3	5	5	5	5	5	5	5	5
			1	1	1	4	4	4	2	2	2
F	F		F	F		F		F	F		
					LRU						

2	2	2	2	5	5	5	5	3	3	3	3
	3	3	3	3	2	2	2	2	2	2	2
			1	1	1	4	4	4	4	5	5
F	F		F	F	F	F		F		F	
					Horloge						

b)

La politique de l'horloge est un bon compromis entre celui du FIFO et celui du LRU car il mixe un pointeur vers la page la plus anciennes (FIFO) pour débuter sa liste circulaire et utilise une référence prenant partiellement en compte si une information est utilisé récemment ou fréquemment (LRU).

Commentaire :
a) Bravo

b) Incorrect

Question 13

Terminé

Note de 1,41 sur 1,50

[Q7] (2,5 pt) - Mémoire virtuelle et pagination pure à 2 niveaux

Soit une mémoire virtuelle avec pagination pure similaire à l'exercice 3 fait en classe où:

- La mémoire physique est de 64 octets.
- La taille d'une page est de 4 octets.
- La table des pages est à deux niveaux. Les tables des pages du 1er et 2e niveau ont chacune 4 entrées. Chaque entrée est composée de:
  - 2 bits de contrôle (1 bit de présence + 1 bit de référence). Un bit de présence à 1 indique que la page est en mémoire physique. Si le bit de présence est à 0 et le bit de référence est à 1, cela signifie que la page est dans la zone de va-et-vient (swap).
  - 6 bits pour l'adresse en mémoire physique ou dans la zone de va-et-vient.

Deux processus P1 et P2 sont en mémoire. La table des pages du premier niveau de P1 commence à l'adresse 32 (en décimal). La table des pages du premier niveau de P2 commence à l'adresse 28. L'état courant de la mémoire physique ainsi que celui de la zone de swap sont présentés à la page suivante.

a. (1,5 pt) Pour chaque page de P2, indiquez laquelle des situations suivantes s'applique :
   i. Présente en mémoire physique (spécifiez dans quel cadre).
   ii. Absente de la mémoire ou invalide.
   iii. Dans la zone de swap (indiquez à quelle position).

cadre 0	0	1	1	1
	1	1	1	2
	2	0	0	0
	3	0	0	0
cadre 1	4	1	0	6
	5	0	1	9
	6	0	0	59
	7	1	0	62
cadre 2	8	1	0	7
	9	1	0	6
	10	1	0	5
	11	0	1	2
cadre 3	12	0	0	0
	13	0	0	30
	14	0	0	0
	15	0	0	0

cadre 4	16	0	0	0
	17	0	0	0
	18	0	0	0
	19	0	0	0
cadre 5	20	1	1	4
	21	1	1	44
	22	0	1	3
	23	1	1	12
cadre 6	24	0	0	0
	25	0	0	0
	26	0	1	0
	27	1	0	16
cadre 7	28	1	1	48
	29	1	1	24
	30	1	1	36
	31	1	1	60

cadre 8	32	0	0	4
	33	0	0	0
	34	1	1	20
	35	0	0	52
cadre 9	36	1	1	52
	37	0	0	0
	38	0	0	0
	39	0	0	0
cadre 10	40			
	41			
	42			
	43			
cadre 11	44	1	0	48
	45	1	0	52
	46	0	0	56
	47	0	0	60

cadre 12	48	1	0	8
	49	1	0	0
	50	0	0	0
	51	0	0	0
cadre 13	52	1	0	4
	53	1	1	7
	54	1	0	1
	55	1	0	5
cadre 14	56	1	0	8
	57	1	1	7
	58	1	0	6
	59	1	0	5
cadre 15	60	0	1	2
	61	0	0	0
	62	0	0	0
	63	0	0	0

Mémoire secondaire (suite au remplacement)

Page 0	0	0	0	40
	1	0	0	0
	2	0	0	0
	3	0	0	0

Page 1	0	0	1	10
	1	0	0	12
	2	0	0	14
	3	1	0	16

Page 2	0	0	1	1
	1	0	0	0
	2	0	0	0
	3	0	0	0

Page 3	0	1	0	63
	1	0	0	29
	2	0	1	0
	3	0	0	0

# de page de P2: **0**	En mémoire, cadre 2
# de page de P2: **1**	En mémoire, cadre 0
# de page de P2: **2**	Absente ou invalide
# de page de P2: **3**	Absente ou invalide
# de page de P2: **4**	Absente ou invalide
# de page de P2: **5**	Absente ou invalide
# de page de P2: **6**	Zone de swap, page 0
# de page de P2: **7**	En mémoire, cadre 4
# de page de P2: **8**	En mémoire, cadre 13
# de page de P2: **9**	Absente ou invalide
# de page de P2: **10**	Absente ou invalide
# de page de P2: **11**	Absente ou invalide
# de page de P2: **12**	Zone de swap, page 0
# de page de P2: **13**	Absente ou invalide
# de page de P2: **14**	Absente ou invalide
# de page de P2: **15**	Absente ou invalide

Votre réponse est partiellement correcte.

Vous en avez sélectionné correctement 15.

La réponse correcte est :

de page de P2: **0** → En mémoire, cadre 2,

de page de P2: **1** → En mémoire, cadre 0,

de page de P2: **2** → Absente ou invalide,

de page de P2: **3** → Absente ou invalide,

de page de P2: **4** → Absente ou invalide,

de page de P2: **5** → Absente ou invalide,

de page de P2: **6** → Zone de swap, page 0,

de page de P2: **7** → En mémoire, cadre 4,

de page de P2: **8** → En mémoire, cadre 13,

de page de P2: **9** → Absente ou invalide,

de page de P2: **10** → Absente ou invalide,

de page de P2: **11** → Absente ou invalide,

de page de P2: **12** → Zone de swap, page 2,

de page de P2: **13** → Absente ou invalide,

de page de P2: **14** → Absente ou invalide,

de page de P2: **15** → Absente ou invalide

Question 14

Terminé

Note de 1,00 sur 1,00

b. (1 pt) Pour chaque cadre en mémoire, indiquez laquelle des situations suivantes s'applique :
   i. Contient une page d'un processus (indiquez lequel).
   ii. Contient une table de pages d'un processus (indiquez lequel).
   iii. Libre.

NOTE: *Le cadre 14 a été volontairement omis puisqu'il contient une page de type "autre".*

# de cadre: **0**	Page 1 de P2
# de cadre: **1**	Page 8 de P1
# de cadre: **2**	Page 0 de P2
# de cadre: **3**	Page 11 de P1
# de cadre: **4**	Page 7 de P2
# de cadre: **5**	TDP de P1
# de cadre: **6**	TDP de P2
# de cadre: **7**	TDP de P2
# de cadre: **8**	TDP de P1
# de cadre: **9**	TDP de P2
# de cadre: **10**	Libre
# de cadre: **11**	Page 9 de P1
# de cadre: **12**	TDP de P2
# de cadre: **13**	Page 8 de P2
# de cadre: **15**	TDP de P2

Votre réponse est correcte.

La réponse correcte est :
de cadre: **0** → Page 1 de P2,

de cadre: **1** → Page 8 de P1,

de cadre: **2** → Page 0 de P2,

de cadre: **3** → Page 11 de P1,

de cadre: **4** → Page 7 de P2,

de cadre: **5** → TDP de P1,

de cadre: **6** → TDP de P2,

de cadre: **7** → TDP de P2,

de cadre: **8** → TDP de P1,

de cadre: **9** → TDP de P2,

de cadre: **10** → Libre,

de cadre: **11** → Page 9 de P1,

de cadre: **12** → TDP de P2,

de cadre: **13** → Page 8 de P2,

de cadre: **15** → TDP de P2

Question 15

Terminé

Note de 0,25 sur 1,00

[Q8] (1,5 pt) - ICCP et héritage de priorité

Considérez les 6 tâches suivantes T1 à T6 sous POSIX (Figure 3.1). La séquence d'exécution peut être interprétée comme suit:

- E correspond à une période d'exécution sans mutex;
- A correspond à une période d'exécution à l'intérieur d'une section critique protégée par A (similaire pour B et C);
- (AC) correspond à une période d'exécution à l'intérieur des sections critiques protégées par A et C, et
- (AB) correspond à une période d'exécution à l'intérieur des sections critiques protégées par A et B.

Tâche	Priorité	Nombre de *ticks* en attente avant de démarrer.	Nombre de *ticks* à exécuter	Séquence d'exécution
T6	8	7	5	EA(AC)AE
T5	10	7	3	EAA
T4	13	5	4	ECCE
T3	15	3	6	EBBBEE
T2	18	2	3	EEE
T1	24	0	6	EAA(AB)AE

Figure 6.1

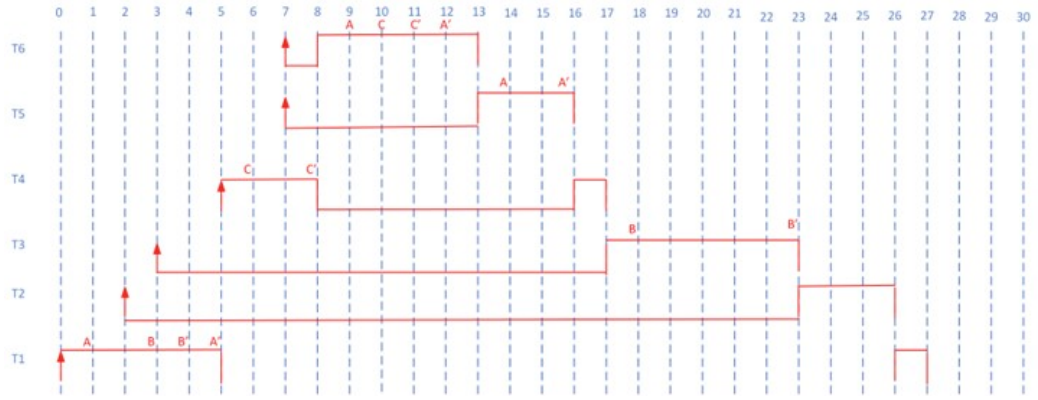


Schéma 1

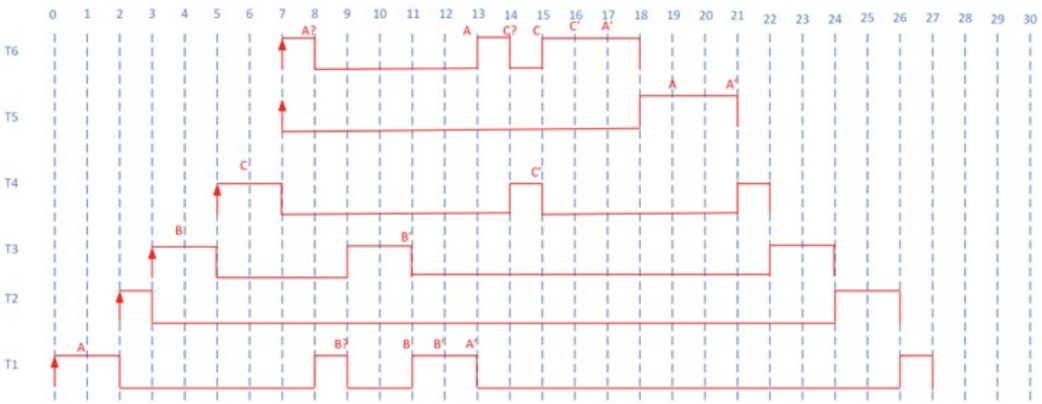


Schéma 2

a. (1 pt) À partir des spécifications de la Figure 6.1, lequel des 2 schémas suivants est le protocole héritage de priorité et lequel représente le protocole ICPP? Justifiez bien votre choix, avec concision, dans l'espace prévu au bas de la page.

Rappel sur les symboles utilisés dans les schémas 1 et 2 :

- *A? (ou B? ou C?) veut dire qu'une requête a été demandée (via OSMutexPend) pour accéder à la section critique, mais la section critique était occupée;*
- *A (ou B ou C) veut dire qu'une requête pour accéder à la section critique a été acceptée; et*
- *A' (ou B' ou C') indique que la tâche a terminé son exécution dans la section critique.*

(•) ICPP = Schéma 1
Héritage de priorité = Schéma 2

( ) ICPP = Schéma 2
Héritage de priorité = Schéma 1

Votre réponse est partiellement correcte.

La réponse correcte est : ICPP = Schéma 1
Héritage de priorité = Schéma 2

Commentaire :
Justification peu claire et précise. Il n'y a pas d'échange de priorité entre T5 et T1.

Question 16

Terminé

Non noté

Utilisez cet espace pour justifier votre réponse à la question 8a. Si votre justification est absente ou erronée, vous aurez **0 point** peu importe votre réponse dans le QCM.

Le schéma 1 correspond au protocole ICPP puisqu'un thread nécessitant une ressource possédé par un autre échange son niveau de priorité avec le thread possesseur avant même de débuter. Par exemple, c'est la raison pour laquelle le thread T5 nécessitant l'accès à la section critique A échange sa priorité avec le thread T1 qui poursuis alors son exécution en priorité par rapport à T2 et T3 jusqu'à la libération de l'accès à la zone A ou T1 reprend alors sa priorité initial.

Le schéma 2 correspond au protocole héritage de priorité puisque l'inversion de priorité ne se fait qu'au moment ou un thread demande l'accès à une zone critique dans laquelle un thread moins prioritaires se trouve, alors ils échangent de priorité seulement à ce moment. C'est la raison pour laquelle on peut voir beaucoup plus de changement de contextes.

Question 17

Terminé

Note de 0,50 sur 0,50

b. (0,5 pt) Nous avons discuté en classe de 3 critères qui avantagent le protocole ICPP par rapport au protocole héritage de priorité. Donnez deux de ces critères qui apparaissent dans les schémas fournis.

b)

1. ICPP demande peu de changement de contexte
2. ICPP minimise le temps de blocage

Commentaire :
Bien

Question 18

Terminé

Note de 0,50 sur 0,50

[Q9] (3 pts) - Ordonnancement périodique

Soit l'ensemble de tâches à ordonnancer suivant:

Tâche	Temps d'exécution (C)	Période (T)	Deadline (D)
T1	2	10	10
T2	1	6	6
T3	9	15	15

Tableau 9.1 - Ensemble de tâches d'un système

a. (0,50 pt) Effectuez le test de *Liu et Layland*. Que peut-on conclure du résultat?

a)

La condition nécessairement suffisante (CNS) pour des tâches échéances sur requêtes :

$\sum Ci/Pi = \sum Ci/Di \leq 1 \Rightarrow 2/10 + 1/6 + 9/15 = 58/60 \leq 1$

$n = 3 \Rightarrow Ui = 78\% = 0,78$

$0,78 < 58/60$

=> Le test n'est pas concluant, la condition est suffisante, mais pas nécessaire. (On ne peut rien conclure pour l'ordonnancement RMA) (CS ordonnancement RMA)

=> Possible à ordonnancer avec ordonnancement EDF (CNS ordonnancement EDF

Commentaire :
Good job ! ( )

Question 19

Terminé

Note de 1,25 sur 1,25

b. (1,25 pt) Représentez graphiquement l'ordonnancement de ces tâches en utilisant l'algorithme d'ordonnancement RMA (Rate Monotonic Assignement). Négligez le temps requis pour les changements de contexte. Un ordonnancement est-il possible? Si l'ordonnancement est impossible, arrêtez-vous à l'endroit où les contraintes ne sont pas respectées.

INSTRUCTIONS:

i. Veuillez utiliser le gabarit en format Excel disponible **ICI**.
ii. Remplissez le tableau. Les cellules du tableau peuvent prendre la valeur X, ou rester vides.
iii. Pour remettre votre réponse, sélectionnez l'ensemble du tableau en débutant dans le coin supérieur gauche, puis faites un copier-coller vers la zone de texte ci-après. *Vous n'avez pas à remettre le fichier Excel*.

RMA => + petite période = + prioritaire

T2 => P1 (6 < 10 <15)

T1 => P2 (10 < 15)

T3 => P3 (15)

T3				x	x	x		X	X	X				X	X										
T2	X						X						X												
T1		X	X									X	X												
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	2
	T2->T1->T3						<-D2				<-D1		<-D2			<-D3 C3 = 8 non respect T3			<-D2		<-D1				<

Les lignes 6 à 8 peuvent servir à vos calculs et ne seront pas corrigées.

=> Ordonnancement RMA impossible (T3 temps d'exec 8 au moment de son deadline)

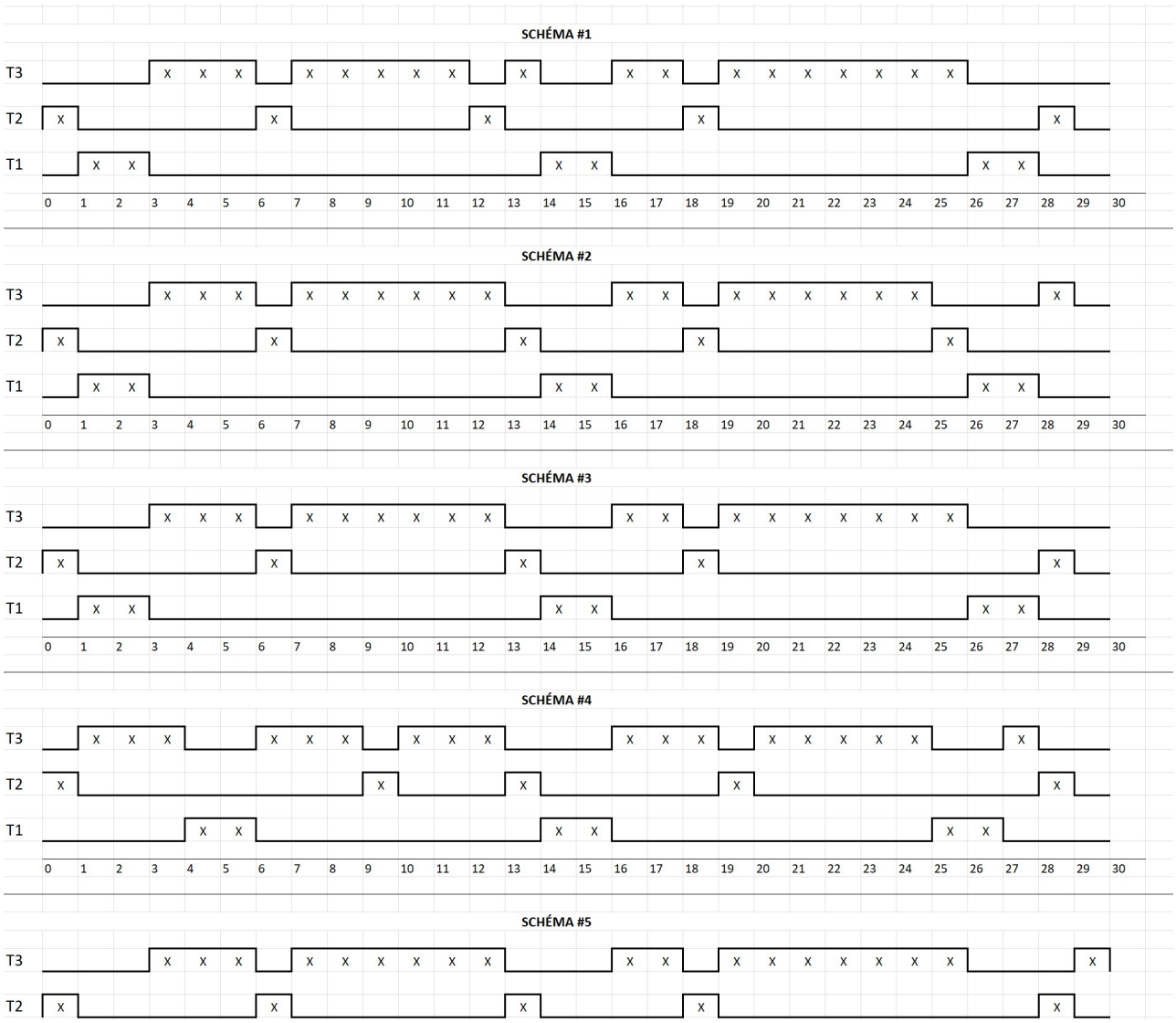
Commentaire :
Bravo

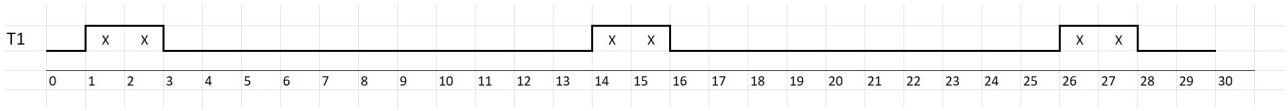
Question 20

Terminé

Note de 1,25 sur 1,25

c. (1,25 pt) Lequel des cinq schémas suivants représente un ordonnancement EDF valide? Si, à un instant donné, deux tâches ont le même deadline, la priorité d'ordonnancement est donnée à celle des deux qui occupe déjà le CPU. Si aucune des deux n'occupe le CPU, la priorité est tirée au hasard. Négligez le temps requis pour les changements de contexte.





- Schéma #5
- Schéma #3
- Schéma #4
- Schéma #2
- Schéma #1

Votre réponse est correcte.

La réponse correcte est :
Schéma #3