Commencé le	lundi 12 décembre 2022, 09:30
État	Terminé
Terminé le	lundi 12 décembre 2022, 11:59
Temps mis	2 heures 28 min
Note	**12,25** sur 20,00 (**61,25**%)

Question **1**

Non répondue

Non noté

Directives :

1. Cet examen est composé de 20 (sous) questions au total pour une durée totale de 2 heures 30 minutes.
2. Pondération 45%.
3. Pour la documentation, vous avez accès aux sites Moodle et Moodle Quiz du cours INF2610. Aucune documentation papier n'est permise.
4. **Les ordinateurs personnels, tablettes, calculatrices et cellulaires ne sont pas permis**.
5. 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 page pour énoncer clairement vos suppositions. N'oubliez pas d'indiquer le numéro de la question. Nous tiendrons compte de toute supposition/interprétation sensée.
6. Pour les questions à développement, **vous devez répondre directement dans les cadres réservés aux réponses. Vous ne pouvez pas joindre de fichiers.**
7. Pour les questions à choix multiples, **vous devez sélectionner une seule réponse**.
8. Lisez au complet et attentivement chaque question avant de répondre.

Bonne fin de session à tous!

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.

KAWTAR REZZOUK 2000057

Question **3**

Terminé

Note de 1,00 sur 2,00

1- Donnez (sous forme textuelle) l'arborescence des processus créés par le code ci-dessous (supposez que les appels système ne retournent pas d'erreur).
2- Donnez aussi tous les ordres possibles d'affichages des chiffres.
3- Quelle(s) est (sont) la (les) valeur(s) de *x* affichée(s) à l'écran ? Justifiez votre réponse.

```
int x=0;
int main() {
  x=x+1;
  if (fork() == 0) {
          x=x+1;
          if (fork() == 0) { x=x+1; write(1,"8",1);}
          else {  wait(NULL); write(1,"9",1); }
          exit(0);
  } else {
          x=x+2;
         if (fork() == 0) { write(1,"6",1); exit(0); }
    }
  while(wait(NULL)>0);
  write(1,"7",1);
  printf("\n x=%d\n",x);
  exit(0);
}
```

1- le processus pere P0 creer u7n premier processus P1 il rentre dans le else et il creer un autre processus P2

P1 creer rentre dans la boucle if et creer un processus P11 et il se met en attente

P2 creer rentre dan sa boucle if ms ne creer pas d autres processus et de meme pour P11

donc en resumer P0 creer P1 et P2, et P2 creer P11

2- On aura 6-8-9-7 ou 8-9-6-7

car P1 attend la terminaison de P11 pour ecrire 9 donc on aura tous le temps 8 apres 9 et comme on sait pas l ordre d esxecution des processus P1 et P2 on aura soit 6-8-9 ou 8-9-6

et finalement P0 attend la terminaision des ses processus fils avant de write 7 donc 7 sera la derniere chose affichee.

3 les valeurs sont 2 et 3.

1- PP : incrémente x de 1, crée un processus fils F1, incrémente x de 2, crée un second fils F2, attend la fin de ses deux fils, affiche 7, affiche la valeur de x puis se termine. F1 : incrémente x de 1, crée un processus fils F11, attend la fin de son fils, affiche 9 puis se termine. F2 : affiche 6 puis se termine. F11 : incrémente x de 1, affiche 6 puis se termine. L'arborescence créé est donc PP crée F1 et F2 et F1 crée F11.

2- 8967 6897 et 8697

3 - x=3 (seul PP va afficher la valeur de sa variable x)

Commentaire :

Question **4**

Terminé

Note de 2,00 sur 2,00

Considérez les deux threads concurrents *A* et *B* suivants d'un même processus :

```
/*0*/ Sémaphore SA=0, SB=0;
A {                          B {
 while(1) {                   while(1) {
  /*1*/                        /*4*/
  a1();                       b1();
  /*2*/                        /*5*/
  a2();                       b2();
  /*3*/                        /*6*/
  }                           }
}                            }
```

Synchronisez, en utilisant les sémaphores *SA* et *SB*, les threads *A* et *B* pour forcer, à chaque cycle, (1) l'exécution de la fonction *a1* avant celle de *b2* et (2) l'exécution de *b2* avant celle de *a2*.
Attention : Aucune autre relation de précédence entre les fonctions ne doit être forcée. Vous devez utiliser les sémaphores *SA* et *SB* pour bloquer/ débloquer respectivement les threads *A* et *B*.
Pour répondre à cette question, sélectionnez les instructions à insérer aux endroits appropriés. Sélectionnez "*rien*", s'il n'y a aucune instruction à insérer.

/*1*/	rien
/*2*/	V(SB); P(SA);
/*3*/	rien
/*4*/	rien
/*5*/	P(SB);
/*6*/	V(SA);

Votre réponse est correcte.

La réponse correcte est :
/*1*/ → rien,

/*2*/ → V(SB); P(SA);,

/*3*/ → rien,

/*4*/ → rien,

/*5*/ → P(SB);,

/*6*/ → V(SA);

Question **5**

Terminé

Note de 0,00 sur 2,00

Considérez la variante suivante du moniteur *Tour-a-Tour* de l'exercice *4* du chapitre *5 (Moniteurs)* :

```
Moniteur Tour-a-Tour {
     int tour=0;
     boolc wq;  // une seule variable de condition au lieu de 3
    void wtour (int i) {   while (tour!=i) wait(wq); }
   void stour () {   tour = (tour+1) % 3; signal(wq); }
 }

 Tour-a-Tour O;
 T0 {  while(1) {
           O.wtour(0);
           printf("Cycle de T0 \n”);
           O.stour();
         }
}
T1 {
   while(1) {
           O.wtour(1);
           printf("Cycle de T1 \n”);
           O.stour();
         }
}

T2 {
   while(1) {
           O.wtour(2);
           printf("Cycle de T2 \n”);
           O.stour();
         }
}
```

Les *3* threads peuvent se retrouver tous en attente dans la file d'attente de la variable de condition wq.
Utilisez la page suivante pour justifier votre réponse.

Veuillez choisir une réponse.

- Vrai
- Faux

La réponse correcte est « Vrai ».

Question **6**

Terminé

Non noté

Justifiez votre choix de réponse (si votre justification est absente ou erronée vous aurez **0 point** peu importe votre réponse dans le QCM).

On aura pas un interblocage car on verifie a chaque fois si c est mon tour ou pas dans la boucle ***while (tour!=i)*** donc necessairement on aura un thread qui roule a chaque fois donc le verrou va surement etre liberer une seul fois chaque fois qu un threads termine son execution. En utilisera 3 variables de condition dans le cas ou on veux forcer l ordre d execution des threads.

Vrai car, par exemple, le scénario suivant mène vers un état où les 3 threads sont en attente de wq :

T2 : O.wtour(2) -> met T2 en attente de wq.

T1 : O.wtour(1) -> met T1 en attente de wq.

T0 : O.wtour(0) -> T0 va réaliser son printf puis appeler O.stour().

O.stour() ->. débloque T2 qui est mis en attente du moniteur.

O.wtour(0) -> met T0 en attente de wq.

T2 : O.wtour(2) —> remet T2 en attente de wq.

Question **7**

Terminé

Note de 2,50 sur 2,50

On vous sollicite pour implémenter, sous forme d'un moniteur et de variables de condition, un gestionnaire de N imprimantes. Ce moniteur est composé d'attributs et de deux fonctions *Allouer* et *Liberer*. La fonction *Allouer* est appelée pour demander une imprimante. Si aucune imprimante n'est disponible, elle met en attente passive le demandeur. La fonction *Liberer* est appelée pour libérer une imprimante. Une imprimante libérée est aussitôt allouée à un demandeur, s'il y a des demandes en attente.

Complétez le moniteur *Imprimantes* suivant pour qu'il gère les imprimantes selon les spécifications précédentes.

```
#define N 100
 Moniteur Imprimantes {
      int dispo=N; // au départ toutes les imprimantes sont disponibles
     /*0*/
    void Allouer() {
       /*1*/
     }
    void Liberer() {
        /*2*/
      }
}
```

```
#define N 100
 Moniteur Imprimantes {
      int dispo=N; // au départ toutes les imprimantes sont disponibles
     /*0*/
      boolc wq; // variable de condition pour une attente passive d une imprimante
     int nbw=0; // nombre de processus/threads en attente d une imprimante
    void Allouer() {
       /*1*/
       if (dispo <1) {nbw++; wait(wq); }
       else dispo--;
     }
    void Liberer() {
        /*2*/
        if(nbw>0) { nbw--; signal(wq); }
       else dispo++;
      }
}
```

```
// similaire au moniteur sémaphore
#define N 100
 Moniteur Imprimantes {
       int dispo=N; // au départ toutes les imprimantes sont disponibles
      /*0*/ boolc wq;
     void Allouer() {
         /*1*/ dispo--; if(dispo <0) wait(wq);  // si dispo<0 alors sa valeur absolue est le nombre de demandes en attente
      }
     void Liberer() {
          /*2*/ dispo++; if(dispo>=0) signal(wq);
       }
 }
```

Commentaire :

Question **8**

Non répondue

Non noté

Utilisez cette page, si nécessaire, pour compléter votre réponse à la question précédente.

Question **9**

Terminé

Note de 2,00 sur 2,00

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 *(6, 3, 6)*. Supposez l'état courant du système :

Matrice Alloc des ressources allouées :

Alloc :

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

Matrice *Req* des ressources requises mais non encore acquises :

Req :

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

1- Est-ce que cet état est sûr ? Justifiez votre réponse en déroulant pas-à-pas l'algorithme du banquier.

Oui

2- Supposez que le système utilise l'algorithme du banquier pour éviter les interblocages. Le système reçoit de la part du processus P4, une demande de 2 ressources de type *R1*. Le système va-t-il accepter cette demande ? Justifiez votre réponse en expliquant clairement toutes les étapes de l'analyse de la demande.

Non

Utilisez la page suivante pour justifier vos réponses. Les réponses non justifiées ne seront pas considérées.

Les ressources disponibles :

A = (6 3 6) - (1+3 1+2 2+1+2) = (2 0 1)

L'agorithme du banquier

Req(P3) <= A ? Oui ==> A= A+Alloc(P3) = (5 2 2)

Req(P1) <= A ? Oui ==>. A= A+Alloc(P1) = (5 3 4)

Req(P2) <= A ? Oui ==> A = A+ Alloc(P2) = (6 3 4)

Req(P4) <= A ? Oui ==> A = A+ Alloc(P4) = (6 3 6)

Oui, l'état est sûr car l'ordre d'exécution *P3, P1, P2, P4* permet aux processus de compléter leurs exécutions dans le cas pire cas (tous les processus demandent en même temps les ressources nécessaires à l'accomplissement de leurs exécutions).

2). *P4* demande *2R1*. Pour accepter cette demande, il faudrait qu'elle mène vers un état sûr :

A' = (0 0 1)

Alloc'

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

Req'

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

Req'(P1) <= A' ? Non
Req'(P2) <= A' ? Non
Req'(P3) <= A' ? Non
Req'(P4) <= A' ? Non
Req'(P1) <= A' ? Non

Cet état n'est pas sûr. La demande va être donc rejetée par le système.

Question **10**

Terminé

Non noté

Utilisez cette page pour justifier chacune de vos réponses ((si votre justification est absente ou erronée vous aurez **0 point** pour cette question peu importe votre réponse dans le QCM).

1- initialement A=(6,3,6)-(4,3,5) = (2,0,1)

on a Req(P1) n est pas inferieur a A donc P1 ne peux pas s executer pour le moment car il n y a pas assez de ressources disponibles

on a Req(P2) n est pas inferieur a A donc P2 peux pas s executer pour le moment car il n y a pas assez de ressources disponibles

on a Req(P3) inferieur a A donc P3 peu s executer et il va libere 3R1, 2R2 et 1R3 qu on ajoute a A qui devient A=(5,2,2)

on a Req(P4) n est pas inferieur a A donc P4 peux pas s executer pour le moment car il n y a pas assez de ressources disponibles

on a Req(P1) inferieur a A donc P1 peu s executer et il va libere 1R2 et 2R3 qu on ajoute a A qui devient A=(5,3,4)

on a Req(P2) inferieur a A donc P2 peu s executer et il va libere 1R1 qu on ajoute a A qui devient A=(6,3,4)

on a Req(P4) inferieur a A donc P4 peux s executer et il va libere 2R3 qu on ajoute a A qui devient A=(5,3,6)
Donc on a un etat sur vu que j ai assez de ressources pour satisfaire le demandes de tout les processus.

2- Si on alloue 2R1 a P4 on aura:

Alloc :

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

Req :

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

A=(0,0,1) donc t aura que une ressource R3 de disponible et peux importe le processeur qui demande les ressources on aura jamais Req(Pi) inferieur ou egal a A donc on est pas dans un etat sur, le systeme ne va pas accepter la demande.

Question **11**

Terminé

Note de 2,00 sur 6,00

Soit un système de gestion de la mémoire, à pagination pure, possédant les caractéristiques suivantes :

- une mémoire physique de *1 MiO (= 2^{20} octets)*,
- une adresse virtuelle codée sur *24 bits*,
- des pages de 2^8 octets chacune, et
- des entrées de tables des pages (TDPs) de 16 (=2^4) octets chacune.

La notation 2^*x*, dans les choix de réponses, signifie *2* puissance *x*.

1 - Quelle est la taille maximale en nombre de pages de l'espace virtuel ?

2^16

2 - Quel est le nombre de cadres dans l'espace physique ?

2^10

3 - Quel est le nombre de bits nécessaires pour stocker le déplacement dans un cadre ?

8

4 - Quel est, en hexadécimal, le numéro de page référencé par l'adresse virtuelle 0xA15BA3 ?

0xA15B

5 - Quelle est la taille en nombre de cadres de la TDP, dans le cas d'une TDP à un niveau ?

2^10

6 - Dans le cas d'une TDP à un niveau, quel est le nombre minimal de cadres requis (en mémoire physique) pour y stocker les informations nécessaires à la localisation des pages virtuelles référencées par un processus en cours d'exécution ? L'adresse de la table des pages est supposée chargée dans un registre, lorsque le processus est en cours d'exécution.

2^10

7 - Quelle est la taille en nombre de cadres de la TDP, dans le cas d'une TDP à 2 niveaux et la taille d'une table des pages du second niveau est égale à 3 fois celle de la table des pages du premier niveau ?

1+2^10

8 - Dans le cas d'une TDP à *2* niveaux de la question précédente, quel est, en hexadécimal, le numéro de l'entrée, dans la table des pages du 1er niveau, référencée par l'adresse virtuelle *0xA15BA3* ?

0xA1

9 - Toujours dans le cas d'une TDP à *2* niveaux de la question précédente, quel est, en hexadécimal, l'adresse physique de l'adresse virtuelle *0xA15BA3*, si la page référencée est en mémoire dans le cadre *9* ?

0x009A3

10 - Quelle est la taille en nombre de cadres de la TDP, dans le cas d'une TDP à 2 niveaux et la taille de la table des pages du premier niveau est égale à 3 fois celle d'une table des pages du deuxième niveau ?

2^8+2^10

1- La taille maximale en nombre de pages de l'espace virtuel est 2^24 / 2^8 pages = 2^16 pages.

2- Le nombre de cadres dans l'espace physique est 2^20 / 2^8 cadres = 2^12 cadres.

3- Le nombre de bits nécessaires pour stocker le déplacement dans un cadre est 8 bits (car on a un espace de 2^8 octets).

4- Le numéro de page, en hexadécimal, référencé par l'adresse virtuelle 0xA15BA3 est la valeur en hexadécimal des 16 premiers bits de poids le plus fort du code binaire de : 0xA15B.

5- La taille en nombre de cadres de la TDP à un niveau est : (2^16 entrées * 2^4 octets/entrée)/ 2^8 octets/cadre) = 2^12 cadres

6- Le nombre minimal de cadres requis est : 2^12 (espace nécessaire pour y stocker la TDP).

7- La taille en nombre de cadres de la TDP, dans le cas d'une TDP à 2 niveaux et la taille d'une table des pages du second niveau est égale à 3 fois celle de la table des pages du premier niveau est : (2^4 *2^4 + 2^4 *2^12 *2^4)/2^8 cadres, car les 16 bits premiers bits de poids le plus fort d'une adresse virtuelle donnent le numéro de page. Chaque table a donc 2^4 entrées de 2^4 octets chacune (= 2^8 octets = 1 cadre). Chaque entrée de cette table pointe vers une table de 2ième niveau. Il y a 2^4 tables de 2ième niveau de 2¨12 entrées chacune. Chaque entrée est sur 2^4 octets. Nous avons au total (1 + 2^4*2^12*2^4/2^8) cadres c-à-d 1+2^12.

8 - Le numéro de l'entrée, en hexadécimal, dans la table des pages du 1er niveau, référencée par l'adresse virtuelle 0xA15BA3 est la valeur, en hexadécimal, des 4 premiers bits de poids le plus fort du code binaire de 0xA15BA3 : 0xA.

9- l'adresse physique de 0xA15BA3 est obtenue en remplaçant le numéro de page 0xA15B par le numéro de cadre : 0x009A3.

10 - La taille en nombre de cadres de la TDP, dans le cas d'une TDP à 2 niveaux et la taille de la table des pages du premier niveau est égale à 3 fois celle d'une table des pages du deuxième niveau est (2^12 *2^4 + 2^12 *2^4 *2^4) / 2^8 = 2^8 + 2^12.

Question **12**

Terminé

Note de 1,25 sur 2,00

Supposez un système monoprocesseur et les *5* processus *(P1, P2, P3, P4* et *P5)* décrits dans le tableau suivant :

Tableau

Processus	*Date d'arrivée*	*Priorité*	*Temps d'exécution*
P1	*0*	*3*	*5 (2) 2*
P2	*1*	*3*	*5*
P3	*2*	*2*	*2 (3) 1*
P4	*4*	*1*	*4*
P5	*5*	*2*	*2*

Ce système dispose d'un seul périphérique d'E/S qui gère les requêtes d'E/S selon la discipline FIFO. Le temps de commutation de contexte est nul. La priorité 1 est la plus basse. Le temps d'exécution ***X (Y) Z*** d'un processus Pi signifie que l'exécution de *Pi* nécessite, dans l'ordre, *X* unités de temps CPU, *Y* unités de temps en E/S et *Z* unités de temps CPU.

Donnez le diagramme de Gantt montrant l'ordre d'exécution des *5* processus, dans le cas d'un ordonnancement préemptif à files multiples et priorités fixes. L'ordonnancement des processus de même priorité est circulaire avec un quantum égal à *3*. Pour répondre à cette question, complétez le tableau suivant :

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

18	P4
19	P4
20	P4
21	rien
22	rien

Votre réponse est partiellement correcte.

Vous en avez sélectionné correctement 19.
La réponse correcte est :
0 → P1,

1 → P1,

2 → P1,

3 → P2,

4 → P2,

5 → P2,

6 → P1,

7 → P1,

8 → P2,

9 → P2,

10 → P1,

11 → P1,

12 → P3,

13 → P3,

14 → P5,

15 → P5,

16 → P4,

17 → P3,

18 → P4,

19 → P4,

20 → P4,

21 → rien,

22 → rien

Commentaire :

Question **13**

Terminé

Note de 1,50 sur 1,50

Un système temps réel gère *3* tâches indépendantes périodiques *(T1, T2* et *T3)* avec respectivement des périodes de *(10, 10* et *x ms*). Supposez que les tâches *(T1, T2* et *T3)* aient besoin, au maximum, respectivement de *(4, 3* et *6 ms)* de temps processeur pour réaliser leurs traitements périodiques.

Quelle est la plus petite valeur de *x* pour laquelle il est possible d'ordonnancer ces trois tâches, selon un ordonnancement préemptif à priorités ? Justifiez votre réponse à la page suivante.

- 10 ms
- 20 ms
- 30 ms
- Aucune de ces réponses
- 40 ms

Votre réponse est correcte.

La réponse correcte est :
20 ms

Question **14**

Terminé

Non noté

Justifiez votre choix de réponse (si votre justification est absente ou erronée vous aurez **0 point** pour cette question peu importe votre réponse dans le QCM).

On souhaite determiner la plus petite valeur de *x* pour laquelle il est possible d'ordonnancer les trois tâches, pour cela il faux respecter la condition suivante:

4/10 +3/10 + 6/x <= 1

donc 6/x <= 3/10

donc x/6 >= 10/3

donc x>=20

d ou la plus petite valeur de x sera alors 20ms