POLYTECHNIQUE MONTRÉAL

Questionnaire examen différé

INF2610

Sigle du cours

Identification de l'étudiant(e)		
Nom :	**Prénom :**	
Signature :	**Matricule :**	**Groupe :**

Réservé

Sigle et titre du cours			
INF2610 Noyau d'un système d'exploitation			
Professeur		***Groupe***	***Trimestre***
Lévis Thériault		**Tous**	**2020.3**
Jour	***Date***	***Durée***	***Heures***
Mardi	**2 février 2021**	**2h30**	**9h30**

Documentation	***Calculatrice***	***Outils électroniques***
☐ Aucune ☐ Toute ☒ Voir directives particulières	☐ Aucune ☒ Toutes	**Les appareils électroniques personnels sont interdits.**

Directives particulières

- **Le professeur ne répondra à aucune question durant cet examen. Si vous estimez que vous ne pouvez pas répondre à une question pour diverses raisons, veuillez le justifier puis passer à la question suivante.**
- **Seule une feuille de notes personnelles 8,5x11 recto-verso est autorisée.**
- **Aucune autre documentation n'est permise.**
- **Utilisez des feuilles blanches pour répondre aux questions.**

Cet examen contient **11** questions sur un total de **13** pages **(incluant cette page).**

L'étudiant doit honorer l'engagement pris lors de la signature du code de conduite.

DIRECTIVES

L'examen se fait individuellement. Tout plagiat sera rapporté au Comité des fraudes.

Vous avez droit **SEULEMENT** à une feuille de notes personnelles au format 8,5x11 recto-verso.

La calculatrice est permise.

Vous devez **justifier toutes vos réponses**.

Répondez à chaque question sur des feuilles blanches.

SVP écrivez clairement afin que je puisse vous lire et vous corriger adéquatement.

QUESTION 1 [8 pts]. PROCESSUS ET FILS D'EXÉCUTION

1.1) [2 pts] Soit le code suivant :

```c
#include <stdio.h>
int value = 5;
int main()
{
    if (fork() == 0)
        value += 15;
    else
    {
        wait(NULL);
        printf("Value = %d\n", value);
    }
    return 0;
}
```

Quelle valeur va-t-il renvoyer à l'écran? Justifiez votre réponse.

1.2) [6 pts] Soit le code suivant :

```c
pid_t pid1 = fork();
if (pid1 == 0)
{
    pid_t pid2 = fork();
    thread_create(...);
    thread_join(...) if (pid2 > 0)
        waitpid(pid2);
}
pid_t pid3 = fork();
```

En comptant le fil d'exécution (et son processus) original :

A) [2 pts] Combien de processus sont créés? Justifiez votre réponse.

B) [2 pts] Combien de fils d'exécution sont créés? Justifiez votre réponse.

C) [2 pts] Parmi ces processus et fils d'exécution, combien de chaque peuvent être actifs en même temps? Justifiez votre réponse.

QUESTION 2 [8 pts]. SYNCHRONISATION

2.1) **[2 pts]** Une section critique utilisée par plusieurs processus est protégée de façon classique par un sémaphore d'exclusion mutuelle comme suit :

```
P(Mutex)
/* section critique */
V(Mutex)
```

Si un des processus provoque un déroutement durant la section critique et s'arrête. Quelle anomalie provoque-t-il ainsi sur la suite de l'exécution des autres processus?

2.2) **[6 pts]** On considère un système composé de processus accédant à des ressources critiques de différentes classes. On suppose qu'il existe P classes de ressources critiques et qu'un tableau `Libre[0..P-1]` précise initialement le nombre de ressources disponibles de chaque classe.

Le contrôle de l'allocation de ces ressources aux processus est réalisé à l'aide du moniteur suivant :

```
monitor Allocateur
{
    int Libre[P];    /* Compteurs de ressources par classe */
    condition Assez; /* Attente de ressources suffisantes  */

    /* fonction interne controlant si l'allocation est possible */
    boolean insuffisant(int dem[P])
    {
        int i = 0;
        while (i < P)
        {
            if (Libre[i] < dem[i])
                return true;
            i++;
        };
        return false;
    }

    void Allouer(int Dem[P])
    {
        while (insuffisant(Dem))
        {
            wait(Assez);
        }
        for (i = 0; i < P; i++)
            Libre[i] = Libre[i] - Dem[i];
        signal(Assez);
    }
```

```
    void Liberer(int Res[P])
    {
        for (i = 0; i < P; i++)
            Libre[i] = Libre[i] + Res[i];
        signal(Assez)
    }

    /* bloc d'initialisation */
    for (i = 0; i < P; i++)
        Libre[i] = V;
}
```

On suppose que les processus respectent les règles de comportement suivantes :

- ils ne demandent pas plus de ressources d'une classe donnée qu'il n'en existe dans cette classe;
- après avoir obtenu des ressources par une opération `Allouer`, ils libèrent TOUTES ces ressources par l'appel à l'opération `Liberer` adéquate avant de redemander éventuellement des ressources.

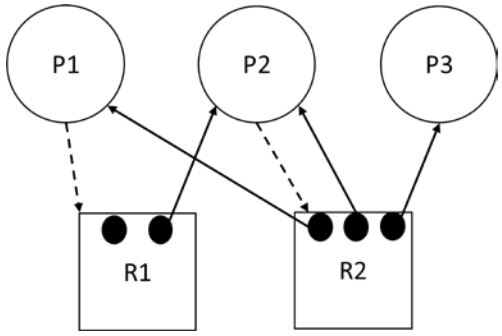
A) **[2 pts]** Justifiez la présence de l'appel de l'opération `signal` dans la procédure `Allouer`.

B) **[2 pts]** La solution proposée risque-t-elle de conduire à une situation d'interblocage? Justifiez votre réponse.

C) **[2 pts]** La solution proposée risque-t-elle de conduire à une situation de famine? Justifiez votre réponse.

QUESTION 3 [10 pts]. INTERBLOCAGE

3.1) [2 pts] Soit le graphe d'allocation des ressources suivant :



Réduisez le graphe d'allocation des ressources ci-dessus afin de déterminer s'il y a un interblocage. Si oui, quels sont les processus concernés? Sinon, pourquoi? Justifiez votre réponse.

3.2) [4 pts] Dans un système, il y a 5 processus (Pi, i=1..5) qui partagent 4 types de ressources (Ri, i=1..4). Le système possède 6 ressources de type R1, 3 ressources de type R2, 4 ressources de type R4 et 2 ressources de type R4. L'état courant du système est décrit par les matrices d'allocation et des besoins suivants :

La matrice d'allocation :

	R1	**R2**	**R3**	**R4**
P1	3	0	1	1
P2	0	1	0	0
P3	1	1	1	0
P4	1	1	0	1
P5	0	0	0	0

La matrice des besoins :

	R1	**R2**	**R3**	**R4**
P1	1	1	0	0
P2	0	1	1	2
P3	3	1	0	0
P4	0	0	1	0
P5	2	2	2	0

L'état courant est-il sûr? Détaillez votre raisonnement. Toutes les étapes sont requises pour l'obtention des points.

3.3) [4 pts] Comment est-il possible lors de la conception d'un système logiciel de prévenir les blocages? Proposez deux solutions permettant de les prévenir. Expliquez clairement en quoi vos solutions préviennent les blocages.

QUESTION 4 [22 pts]. GESTION DE LA MÉMOIRE

4.1) [4 pts] On considère un espace mémoire de 1000 blocs, utilisant une allocation contiguë.

On note par (+) une demande d'allocation en nombre de blocs et par (-) une demande de libération de nombre de blocs.

A) [2 pts] En utilisant l'algorithme d'allocation « First Fit », donnez les différents états de la mémoire centrale **après chacune des étapes** suivantes (les étapes sont successives, initialement la mémoire est vide) :
Étape 1 : A(+300), B(+200), C(+260)
Étape 2 : B(-200), D(+100), A(-300), E(+250), C(-260)
Étape 3 : G(+150), H(+120), D(-100), H(-120), I(+200)
Étape 4 : G(-150), E(-250), J(+100), J(-100), I(-200)

B) [2 pts] Quel est l'avantage d'une stratégie d'allocation « First Fit » par rapport à une stratégie « Best Fit »? Expliquez votre réponse.

4.2) [12 pts] On considère un système à mémoire paginé et ayant les caractéristiques suivantes :
- Une table de pages ayant 2^{16} entrées;
- Chaque entrée de la table de pages est codée sur 16 bits;
- Une entrée contient un numéro de cadre et un bit de présence/absence;
- Le déplacement est codé sur 16 bits;
- Une adresse virtuelle indexe 1 octet.

A) [2 pts] Quelle est la taille d'une page? Détaillez votre raisonnement. Calculs **obligatoires** pour l'obtention des points.

B) [2 pts] Quelle est la taille de la mémoire physique? Détaillez votre raisonnement. Calculs **obligatoires** pour l'obtention des points.

C) [2 pts] Quelle est la taille de la mémoire virtuelle? Détaillez votre raisonnement. Calculs **obligatoires** pour l'obtention des points.

D) [2 pts] Quelle est la taille (en bit) du bus d'adressage de ce système? Détaillez votre raisonnement. Calculs **obligatoires** pour l'obtention des points.

E) [4 pts] En considérant les huit premières entrées de la table de pages représentée par le tableau 1, donnez les adresses virtuelles correspondants aux adresses physiques **33792** et **66048**. Détaillez votre raisonnement. Calculs **obligatoires** pour l'obtention des points.

Tableau 1. Table de pages

No de cadre	No 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

4.3) [4 pts] Supposez que ce système réserve quatre cadres physiques libres à chaque processus créé. Aucun autre cadre n'est alloué au processus. Durant son exécution, un processus référence dans l'ordre les pages : 7 **0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1**. Donnez pour chacun des algorithmes de remplacement de pages suivantes, l'évolution de l'état des cadres 0, 1, 2 et 3 réservés au processus.

A) [2 pts] LRU.

B) [2 pts] FIFO.

4.4) [2 pts] Soit un processus accédant à la suite des pages suivantes : **2 3 2 1 5 2 4 5 3 2 5 2** et une mémoire physique de 3 cadres. Faites l'analyse des défauts de page de la situation suivante :

Cadre	**2**	**3**	**2**	**1**	**5**	**2**	**4**	**5**	**3**	**2**	**5**	**2**
0	2	2	2	2	5	5	5	5	3	3	3	3
1		3	3	3	3	2	2	2	2	2	2	2
2				1	1	1	4	4	4	4	5	5

Quel est le nombre total de défauts de page? Détaillez votre raisonnement. Calculs **obligatoires** pour l'obtention des points.

QUESTION 5 [12 pts]. ORDONNANCEMENT DES PROCESSUS

5.1) [2 pts] Quel est l'effet de la diminution du quantum sur les performances de l'algorithme du tourniquet (*Round Robin*)? Expliquez votre réponse.

5.2) [2 pts] Les algorithmes d'ordonnancement basés sur des priorités peuvent engendrer la famine des processus à faible priorité. Comment peut-on éviter ce problème? Expliquez votre réponse et dites en quoi cela règle ce problème.

5.3) [4 pts] On considère une architecture monoprocesseur dans laquelle on désire exécuter l'ensemble des processus suivants : A, B, C, et D. Les caractéristiques de ces processus sont données au tableau 2. Toutes les durées sont en seconde.

Tableau 2. Caractéristiques des processus

Processus	Date d'arrivée	Temps d'exécution
A	0	10
B	0	6
C	1	8
D	5	4

A) [2 pts] On suppose que l'algorithme utilisé pour ordonnancer ces processus est l'algorithme du tourniquet avec un quantum de 2. Donnez les diagrammes de Gantt pour l'exécution de ces différents processus.

B) [2 pts] On suppose maintenant que l'algorithme utilisé pour ordonnancer ces processus est un algorithme d'ordonnancement à priorité préemptif, en supposant que la priorité d'un processus est inversement proportionnelle à son temps d'exécution total restant. C'est-à-dire que le processus ayant le temps d'exécution total restant le plus court est le plus prioritaire. Donnez les diagrammes de Gantt pour l'exécution de ces différents processus.

5.4) [4 pts] On considère les cinq processus : A, B, C, D et E. Les caractéristiques de ces processus sont données au tableau 3. Toutes les durées sont en seconde.

Tableau 3. Caractéristiques des processus

Processus	Temps d'exécution	Priorité
A	10	2
B	1	4
C	2	2
D	1	1
E	5	3

Le diagramme de Gantt de la figure 1 représente l'exécution des cinq processus : A, B, C, D et E en utilisant un algorithme d'ordonnancement inconnu. Les temps de changement de contexte sont supposés nuls. La date d'arrivée de tous les processus est 0.

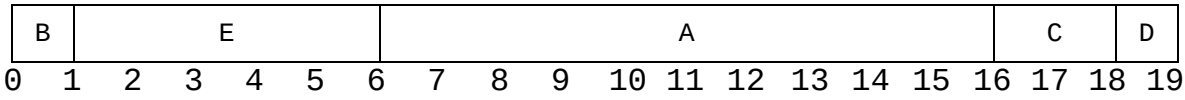


Figure 1. Diagramme de Gantt de l'algorithme d'ordonnancement inconnu

A) [1 pt] Déterminez quel est cet algorithme d'ordonnancement. Votre réponse doit être précise et contenir tous les éléments de réponse.

B) [1.5 pt] Quel est le temps moyen de séjour (TMS) pour cet algorithme? Détaillez votre raisonnement. Calculs **obligatoires** pour l'obtention des points.

C) [1.5 pt] Quel est le temps moyen d'attente (TMA) pour cet algorithme? Détaillez votre raisonnement. Calculs **obligatoires** pour l'obtention des points.

QUESTION 6 [7 pts]. ORDONNANCEMENT TEMPS RÉEL

On considère un système embarqué muni d'un microprocesseur séquentiel gérant trois tâches (A, B et C) périodiques, préemptibles et indépendantes. Les caractéristiques de ces tâches sont données au tableau 4. Activation en début de période pour chaque tâche. Échéance en fin de période.

Tableau 4. Caractéristiques des tâches

Tâche	Période	Calcul
A	36	12
B	15	4
C	10	2

A) [2 pts] Ces tâches sont-elles ordonnançables à coup sûr par RMA? Expliquez votre raisonnement.

B) [3 pts] Tracez le chronogramme des tâches. Expliquez votre raisonnement.

C) [2 pts] Si le temps de calcul de la tâche C passait à 4, les tâches sont-elles toujours ordonnançables à coup sûr par RMA? Justifiez votre réponse.

QUESTION 7 [3 pts]. SYSTÈMES DE FICHIER

Dans un système de fichier, quelle est la différence entre une allocation de type FAT et une allocation indexée?

QUESTION 8 [6 pts]. RETOUR SUR LE TP1

Donnez l'arbre de processus correspondant au code suivant. Considérez que le processus `P0` est celui qui exécute la fonction `arbre()`.

```
void arbre()
{
    for (int i = 0; i < 5; ++i)
    {
        if (i == 3)
        {
            Break;
        }
        else
        {
            fork();
        }
    }
    while(wait(NULL) > 0);
    exit(0);
}
```

QUESTION 9 [7.5 pts]. RETOUR SUR LE TP2

Donnez la ligne de commande bash correspondant au code suivant :

```
void commande()
{
    int fd[2];
    pipe(fd);

    if (fork() == 0)
    {
        dup2(fd[1], 2);
        dup2(fd[1], 1);
        execl("./progA", "progA", NULL);
        exit(1);
    }
    if (fork() == 0)
    {
        dup2(fd[0], 0);
        execl("./progB", "progB", NULL);
        exit(1);
    }

    while(wait(NULL) > 0);
}
```

QUESTION 10 [6 pts]. RETOUR SUR LE TP3

Écrivez un code créant 10 threads noyaux exécutant la fonction ayant la signature suivante :

```
void* action(int numero, int signalType);
```

où le numéro correspond au numéro de 0 à 9 de l'itération courante (lors de la création des 10 threads). Le second paramètre correspond à une entrée d'un tableau `SIGNAL_TYPES` auquel vous supposez avoir accès.

Par exemple, le premier thread créé devra appeler :

```
action(0, SIGNAL_TYPES[0]);
```

Vous devez **impérativement** passer les deux paramètres au moment d'appeler la fonction permettant de créer le thread.

Vous devez également attendre la fin de tous les threads avant de terminer le programme.

Vous devez ajouter des commentaires dans votre code pour expliquer votre logique, **sans quoi vous perdrez 50% des points**.

QUESTION 11 [10.5 pts]. RETOUR SUR LE TP4

Considérez un système de mémoire par pagination pure dont la taille des pages est de 1024 octets. Votre système de mémoire a également un **TLB de 32 entrées**. Rédigez la fonction permettant de traduire une adresse virtuelle (req->adresseVirtuelle) à partir d'une recherche dans le TLB. Si jamais aucune traduction n'est possible, vous devez assigner la valeur 0 à l'adresse physique (numéro et offset).

Lors de la rédaction de la logique de recherche dans le TLB, vous devez ajouter des commentaires dans votre code pour expliquer la logique d'accès, la façon de parcourir le TLB jusqu'au moment de construire la traduction, **sans quoi vous perdrez 50% des points**.

Voici la signature de la fonction à rédiger :

```
void rechercherTLB(struct SystemeMemoire* ms, struct RequeteMemoire*
req)
```

La définition des structures est à la page suivante.

```
struct RequeteMemoire{
    u_int8_t estDansTLB;
    u_int8_t estDansTablePages;
    unsigned long adressePhysique;
    unsigned long adresseVirtuelle;
    unsigned long date;
};

struct SystemeMemoire{
    struct TLB* tlb;
    struct TablePages* tp;
    struct Memoire* mem;
};

struct TLB{
    unsigned int* numeroPage;
    unsigned int* numeroCadre;
    u_int8_t* entreeValide;
    unsigned long* dernierAcces;
    unsigned long* dateCreation;
};

struct TablePages {
    unsigned int* numeroCadre;
    u_int8_t* entreeValide;
};

struct Memoire{
    unsigned int* numeroPage;
    unsigned long* dateCreation;
    unsigned long* dernierAcces;
    u_int8_t* utilisee;
};
```