$\frac{\quad}{15} + \frac{\quad}{10} + \frac{\quad}{10} + \frac{\quad}{10} + \frac{\quad}{10} + \frac{\quad}{20} + \frac{\quad}{25} = \frac{\quad}{100}$

Nom: _____ Code permanent: _____

FACULTÉ DES ARTS ET DES SCIENCES
DÉPARTEMENT D'INFORMATIQUE ET DE RECHERCHE OPÉRATIONNELLE

TITRE DU COURS: **Systèmes d'exploitation**
SIGLE DU COURS: **IFT2245**

NOM DU PROFESSEUR: Marc Feeley

DATE DE L'EXAMEN INTRA H07: 26 février 2007 HEURE: 15h30-17h20 SALLE: Y-115

DIRECTIVES PÉDAGOGIQUES:
- Aucune documentation permise.
- Répondre directement sur le questionnaire dans l'espace réservé.
- Soyez clair et concis.

Numéro 1 – (15 points). Définissez et expliquez brievement le rôle et fonctionnement des composantes suivantes de l'architecture IBM-PC:

1. PIC : _____

2. RTC : _____

3. "northbridge" : _____

Numéro 2 – (10 points).

1. Expliquez dans quelles circonstances un processus Unix devient *zombie*.

2. Pourquoi faut-il éviter autant que possible la création de processus *zombie*?

Numéro 3 – (10 points). Écrivez en assembleur i386 (syntaxe "GNU") pour le mode réel une fonction "`inverse`" qui prends un paramètre entier 16 bits sur la pile (x) et qui retourne dans le registre `%ax` la même séquence de bits que x mais dans l'ordre inverse. Par exemple le nombre binaire 1111111100000001 est transformé en 1000000011111111. Assurez-vous que votre fonction préserve le contenu de tous les registres (sauf `%ax` bien entendu). Notez que l'instruction "`shr` `$`n`,`R" décale le registre R de n bits vers la droite et l'instruction "`shl` `$`n`,`R" décale le registre R de n bits vers la gauche. Ajoutez des commentaires et diagrammes pour expliquer le code.

Numéro 4 – (10 points). Soit un programme qui contient quatre variables globales entières `a`, `b`, `c`, et `d` et trois fonctions `f`, `g` et `h` qui manipulent ces variables:

```
int a = 0, b = 0, c = 0, d = 0;

void f() { b += ++a; }
void g() { c += ++b; }
void h() { d += ++c; }
```

Expliquez quels changements il faut apporter au programme si on désire que ces fonctions s'effectuent correctement dans un contexte ou plus d'un processus peut appeler `f`, `g` et `h` (c'est-à-dire que les opérations dans le corps d'une fonction apparaissent indivisibles). Votre solution doit permettre à un processus d'exécuter les opérations dans `f` en même temps que l'exécution des opérations dans `h` par un autre processus.

Donnez le programme avec les changements. Vous pouvez utiliser du pseudocode pour vos ajouts.

Numéro 5 – (10 points). On cherche à synchroniser deux processus avec une synchronisation de type barrière. Le processus P1 fait l'appel "`barriere_P1()`" lorsqu'il arrive à la barrière et le processus P2 fait l'appel "`barriere_P2()`". Utilisez des sémaphores binaires pour définir ces deux fonctions. Vous pouvez utiliser du pseudocode.

Numéro 6 – (20 points).

1. Donnez un algorithme pour détecter si un groupe de processus est présentement en "deadlock". Supposez que les types de ressource peuvent avoir plus d'une instance.

2. Expliquez une approche simple et générale pour empêcher les deadlocks.

Numéro 7 – (25 points). Écrire un programme en C pour Unix ayant le nom "`multi`" dont le premier argument de ligne de commande est un entier N et le deuxième est le nom d'un programme P. Le programme "`multi`" tente d'exécuter N fois le programme P. Par exemple

```
multi 3 ls
```

listera trois fois le contenu du répertoire courant. Le programme "`multi`" arrête son travail aussitôt que la dernière exécution du programme P se termine anormalement (les autres exécutions du programme P ne doivent pas avoir lieu). Le programme "`multi`" termine normalement si et seulement si les N exécutions de P se terminent normalement.

Note : la macro "`WEXITSTATUS(`*statut*`)`" extrait le code de sortie d'un processus étant donné son statut de terminaison *statut*. Le programme doit être robuste (détecter les cas d'erreur).