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

TITRE DU COURS: **Systèmes d'exploitation**
SIGLE DU COURS: **IFT2240**

NOM DU PROFESSEUR: Marc Feeley

DATE DE L'EXAMEN INTRA H02: 25 février 2002 HEURE: 15h30-17h20 SALLE: PAA 1355

DIRECTIVES PÉDAGOGIQUES: - Aucune documentation permise.
- Répondre sur le cahier réponse.

Numéro 1 – (25 points). Définissez et expliquez brièvement les termes suivants (5 à 10 lignes par terme):

1. Système d'exploitation "multitasking"
2. Coroutine
3. "Polling"
4. "Spinlock"
5. Mutex récursif

Numéro 2 – (15 points). Pour chacun des énoncés ci-dessous, indiquez s'il est vrai ou faux, et s'il est faux corrigez l'énoncé.

1. Sur le microprocesseur i386 de Intel, il est laborieux de manipuler des tableaux de plus de 64 Kbytes lorsqu'on utilise le mode protégé
2. Une entrée/sortie est de type synchrone lorsque le programme est bloqué pendant que l'opération d'entrée/sortie est en cours
3. Sous UNIX, la section "`text`" d'un processus contient les arguments de ligne de commande et les variables d'environnement
4. La commande "`setenv`" des shells `csh` et `tcsh` sont des commandes "builtin" car il est impossible d'écrire un programme (par exemple "`/bin/setenv`") qui aurait le même comportement
5. La présence d'un cycle dans le graphe d'allocation de ressources est une indication qu'il y a une étreinte fatale

Numéro 3 – (20 points). Quelles sont les 3 propriétés qu'un mécanisme d'exclusion mutuelle doit respecter? Donnez leur nom et une brève explication de chaque propriété. Analysez le mécanisme d'exclusion mutuelle pour deux processus donné ci-dessous en supposant que seulement les opérations de lecture et écriture sont atomiques et que la valeur initiale de `T` est 1. Lesquelles des 3 propriétés sont respectées. Utilisez les automates de processus dans votre analyse.

```
    processus P1                processus P2

while (T == 2) ;            while (T == 1) ;
<<<SECTION CRITIQUE>>>      <<<SECTION CRITIQUE>>>
T = 2;                      T = 1;
```

Numéro 4 – (20 points). Voici un programme utilisant la librairie "POSIX threads", qui crée `N` processus légers, chacun imprime les entiers de 0 à 3. On aimerait utiliser une synchronisation de type barrière pour s'assurer que tous les processus affichent tout d'abord 0, puis tous les processus affichent 1, etc. Vous devez compléter l'implantation de la fonction `barriere` pour obtenir ce comportement. Vous pouvez utiliser les variables globales `m`, `c` et `x`. Vous ne devez pas déclarer des variables ou fonctions supplémentaires. Indiquez uniquement ce qu'on doit trouver dans le corps de la fonction `barriere`.

```
1.  #define N 5  // nb. de processus
2.  pthread_mutex_t m;
3.  pthread_cond_t c;
4.  int x = 0;
5.
6.  void barriere ()
7.  { /*** à compléter ***/ }
8.
9.  void* p (void* param)
10. { int i;
11.   for (i=0; i<4; i++)
12.     { barriere ();
13.       printf ("%d\n", i);
14.     }
15.   return NULL;
16. }
17. int main ()
18. { int i;
19.   pthread_t tid[N];
20.   void* resultats[N];
21.
22.   pthread_mutex_init (&m, NULL);
23.   pthread_cond_init (&c, NULL);
24.
25.   for (i=0; i<N; i++)
26.     pthread_create (&tid[i], NULL, p, NULL);
27.
28.   for (i=0; i<N; i++)
29.     pthread_join (tid[i], &resultats[i]);
30.
31.   return 0;
32. }
```

Numéro 5 – (20 points). Écrivez un programme en C ayant le nom "`deuxfois`" qui accepte sur la ligne de commande un nombre quelconque d'arguments. Le premier argument est le nom d'un programme, qui doit être exécuté deux fois, et les autres arguments sont les arguments à passer à ce programme. La deuxième exécution du programme doit seulement commencer après la fin de la première exécution du programme. Par exemple, la commande de shell "`deuxfois ls -l /bin`" va lister en format "long" et à deux reprises le contenu du répertoire `/bin`. Faites une gestion raisonnable du statut de terminaison.