1 Exercice : Questions Générales

Question 1 : Si plusieurs messages de différentes étiquettes sont présents dans une file de messages, Quelles sont les affirmations correctes ?

A) Le premier message qui sera extrait est le premier déposé ayant l'étiquette demandée.
B) Un processus peut extraire plusieurs messages en une fois.
C) Le premier message qui sera extrait est celui qui a l'étiquette la plus petite.
D) Un processus peut extraire un message de n'importe quelle étiquette.
E) Aucune de ces réponses n'est correcte.

Question 2 : A quoi sert l'étiquette dans la structure d'un message transmis via une file de messages ?

A) A identifier l'expéditeur du message
B) A identifier un message
C) A permettre la diffusion d'un message à plusieur processus
D) A identifier le type de messages à recevoir

Question 3 : Si un ensemble de processus réalisent une opération Z sur un sémaphore S au moment la valeur de S est 10 et sachant que ces processus ont les droits de lecture et d'écriture sur S,

A) Quand la valeur de S sera égale à 0, tous les processus de l'ensemble seront réveillés.
B) Quand la valeur de S sera mise à 0, un processus sera réveillé et il sera responsable de réveiller les autres processus.
C) Tous les processus de l'ensemble sont mis en attente, sauf un : il faut bien qu'un processus puisse poursuivre l'exécution pour éviter un interblocage.
D) Tous les processus de l'ensemble sont mis en attente.
E) Aucune de ces réponses n'est correcte

Question 4 : Dans une application, un ensemble de processus partage un tableau de sémaphores en lecture et écriture. Ce tableau contient au moins deux sémaphores. Parmi les affirmations suivantes, quelles sont celles qui sont correctes ?

A) Chaque processus peut demander à exécuter ensemble et de manière atomique plusieurs opérations sur différents sémaphores du tableau.
B) Chaque processus peut demander à exécuter de manière non atomique plusieurs opérations sur différents sémaphore du tableau. (….)
C) Chaque processus qui effectue une opération P sur un sémaphore doit obligatoirement effectuer une opération inverse V sur ce sémaphore ou sur un autre sémaphore du tableau.
D) Chaque processus doit obligatoirement utiliser tous les sémaphores du tableau.
E) Aucune de ces réponses n'est correcte.

Question 5 : Si on utilise des threads pour réaliser toute application réelle, il est indispensable que le nombre de threads pouvant s'éxécuter en parallèle ne dépasse pas le nombre de coeurs du processus utilisé pour l'exécution.

A) Vrai B) Faux

Question 6 : Dans le cadre du multi-threading avec des threads POSIX, si plusieurs variables partagées en lecture et écriture sont protégées par différents verrous, est-il possible d'utiliser une variable conditionnelle pour attendre l'occurrence d'un événement impliquant ces différentes variables partagées.

A) Oui, Si on reconsidère la façon de protéger les variables partagées.
B) Oui.
C) Non.

Question 7 : Les applications client/serveur ont pour objectif général :

A) d'accélérer les traitements.
B) de partager des ressources.
C) de construire des programmes portables.

Question 8 : Les applications client/serveur peuvent permettre :

A) d'accélérer les traitements.
B) de partager des ressources.
C) de construire des programmes portables.

Question 9 : Si un serveur en mode connecté (itératif ou concurrent) exécute listen(Br, 10), il peut dialoguer avec :

A) un nombre quelconque de clients.
B) au moins 10 clients.
C) au plus 10 clients.

Question 10 : En mode UDP, un processus doit connaître la taille exacte d'un message pour pouvoir recevoir :

A) Oui B) Non

Question 11 : En mode UDP, un processus qui reçoit plusieurs messages doit connaître la taille exacte de chaque message pour pouvoir le recevoir correctement.

A) Faux
B) Vrai

Question 12 : En mode UDP, un serveur peut traiter simultanément plusieurs clients dans un seul processus et sans avoir à créer des threads.

A) Vrai
B) Faux

Question 13 : En mode TCP, un serveur peut traiter simultanément plusieurs clients dans un seul processus et sans avoir à créer des threads.

A) Vrai .
B) Faux.

Question 14 : En mode TCP, si pour une raison quelconque, la couche transport décide de fermer une socket et si l'application locale est en attente d'un message sur cette socket (appel de la fonction recv(...)) :

A) La fonction renvoie -1.
B) L'application bloque indéfiniment.
C) La fonction recv renvoie 0.
D) La fonction recv peut renvoyer une valeur >0 correspondants à un nombre d'octets extraits du buffer de réception.

2 Exercice : Files de messages

Question 16 : Si on souhaite réduire le coup de communications, quel moyen choisir si le protocole d'échange implique uniquement des échanges de type requête-réponse ?

A) Segment de mémoire partagée.
B) Files de messages.

Question 17 : Si on souhaite réduire le coup de communications, quel moyen choisir pour permettre à un serveur de transmettre un message à plusieurs clients ?

A) Files de messages.
B) Segment de mémoire partagée.

Question 18 : S'il n'y a pas d'autres processus en plus de S1, S2 et des N clients, quels sont les méthodes que vous estimez bonnes pour créer la file des messages ?

A tous les processus exécutent
`int id_file= msgget(cle , IPC_CREAT | 0600);`

B les processus $S1$ et $S2$ exécutent
`int id_file= msgget(cle , IPC_CREAT | IPC_EXCL | 0600);`
et agiront en fonction du résultat. Les clients exécutent
`int id_file= msgget(cle , 0600);`

C les processus $S1$ et $S2$ exécutent
`int id_file= msgget(cle , 0600);`
les clients exécutent
`int id_file= msgget(cle , IPC_CREAT | IPC_EXCL | 0600);`
et agiront en fonction du résultat.

D tous les processus exécutent
`int id_file= msgget(cle , IPC_CREAT | IPC_EXCL | 0600);`
et agiront en fonction du résultat.

E les processus $S1$ et $S2$ exécutent
`int id_file= msgget(cle , IPC_CREAT | 0600);`
les clients exécutent
`int id_file= msgget(cle , IPC_CREAT | IPC_EXCL | 0600);`

F les processus $S1$ et $S2$ exécutent
`int id_file= msgget(cle , IPC_CREAT | IPC_EXCL | 0600);`
et agiront en fonction du résultat. Les clients exécutent
`int id_file= msgget(cle , IPC_CREAT | 0600);`

G les processus $S1$ et $S2$ exécutent
`int id_file= msgget(cle , IPC_CREAT | 0600);`
les clients exécutent
`int id_file= msgget(cle , 0600);`
et tous agiront en fonction du résultat.

H les processus $S1$ et $S2$ exécutent
`int id_file= msgget(cle , 0600);`
les clients exécutent
`int id_file= msgget(cle , IPC_CREAT | 0600);`
et tous agiront en fonction du résultat.

I *Aucune de ces réponses n'est correcte.*

Question 19 : Rappelons qu'un message à envoyer via les files de messages doit être défini dans une structure. Supposons que S1 réalise des opérations à deux opérandes et que S2 réalise des opérations à x opérantes (2 <= x <= 100). Les opérandes sont tous de même type (exemple des entiers) et sont envoyés par les clients. Quelles solutions pensez vous être convenables ?

A) Représenter les requêtes à envoyer vers S1 et S2 en définissant une structure par requête.
B) Représenter les requêtes à envoyer vers S1 et S2 en définissant une structure commune.
C) Représenter les requêtes à envoyer vers S1 et S2 en définissant deux structures, une pour les requêtes destinées à S1 et l'autre pour les requêtes destinées à S2.
D) Aucune de ces réponses n'est correcte.

Question 20 : Chaque messages porte une étiquette. Pour les requêtes, les étiquettes auraient :

A des valeurs > 0.
B une valeur différente pour chaque requête. Les valeurs utilisées pour les requêtes ne seront pas utilisées pour les réponses.
C une valeur différente pour chaque requête. Les valeurs utilisées pour les requêtes peuvent être réutilisées pour les réponses.
D une seule valeur. Elle sera la même pour tous les messages (requêtes et réponses)
E une valeur pour toutes les requêtes destinées à $S1$ et une autre valeur pour celles destinées à $S2$. Ces deux valeurs ne seront pas utilisées pour les réponses.
F une valeur pour toutes les requêtes destinées à $S1$ et une autre valeur pour celles destinées à $S2$. Ces deux valeurs peuvent être utilisées pour les réponses.
G une seule valeur pour toutes les requêtes. Elle sera différente de celle(s) des réponses.
H des valeurs ≥ 0.
I *Aucune de ces réponses n'est correcte.*

Question 21 : Pour recevoir une requête, S1 et S2 peuvent demander un message en passant à msgrcv(...) la valeur 0 pour l'étiquette :

A) Faux
B) Vrai

Question 22 : Pour qu'un client puisse recevoir une réponse à une requête :

A la valeur de l'étiquette passée en paramètre de *msgrcv(...)*, appelée pour recevoir une réponse coté client, doit être différente pour chaque réponse à chaque requête.
B les serveurs $S1$ et $S2$ peuvent utiliser une même valeur d'étiquette dans les réponses à plusieurs clients.
C le serveur (peu importe, $S1$ ou $S2$), doit donner à l'étiquette de la réponse la valeur de l'étiquette de la requête associée.
D les serveurs $S1$ et $S2$ peuvent utiliser une même valeur d'étiquette pour différentes réponses à un même client.
E *Aucune de ces réponses n'est correcte.*

3 Exercice : Tableaux de sémaphores

Dans une application, un ensemble de N processus $Proc_1$, $Proc_2$,..., $Proc_N$ doit effectuer un ensemble de traitements dans un ordre déterminé : $traitement_1$, $traitement_2$,..., $traitement_N$, où un processus $Proc_i$ effectuera le traitement $traitement_i$.

Le schéma algorithmique suivant tente de répondre à ce problème d'ordre :

```
// processus Proc_1
idSem= identifiant du tableau de sémaphores;
afficher(1, "début traitement");
traitement_1(...);
afficher(1, "fin traitement");
V_1(idSem);
afficher(1, "fin");
```

```
//processus Proc_i, avec 1 < i ≤ N
idSem= identifiant du tableau de sémaphores;
afficher(i, "avant traitement");
P_i(idSem);
afficher(i, "début traitement");
traitement_i(...);
afficher(i, "fin traitement");
V_{i+1}(idSem);
afficher(i, "fin");
```

Avec :

```
// opération P_k(idSem)
struct sembuf semoi;
semoi.sem_num = 0;
semoi.sem_op = -k;
semoi.sem_flg = 0;
semop(idSem, &semoi, 1);
```

```
// opération V_k(idSem)
struct sembuf semoi;
semoi.sem_num = 0;
semoi.sem_op = k;
semoi.sem_flg = 0;
semop(idSem, &semoi, 1);
```

Sachant que : **1)** le tableau de sémaphores est créé par un processus d'initialisation et ses éléments sont initialisés à N, **2)** le nombre total de processus (N) et l'identifiant du tableau de sémaphores est passé en paramètre de tous les processus, **3)** chaque processus peut accéder au tableau en lecture et écriture, **4)** k est un entier strictement positif, **5)** un appel "afficher(i, "exemple")" affichera sur la sortie standard : "$Proc_i$: exemple" et **6)** tous les processus sont lancés en début d'exécution de l'application.

Question 23 : Pour les 5 traitements, parmi les débuts de trace d'exécution suivants, quels sont ceux qui peuvent être obtenus ?

A

```
Proc2 : avant traitement
Proc3 : avant traitement
Proc2 : début traitement
Proc1 : début traitement
Proc3 : début traitement
Proc2 : fin traitement
Proc1 : fin traitement
Proc3 : fin traitement
Proc1 : fin
Proc2 : fin
Proc3 : fin
```

B

```
Proc4 : avant traitement
Proc4 : début traitement
Proc1 : début traitement
Proc4 : fin traitement
Proc1 : fin traitement
Proc1 : fin
Proc4 : fin
Proc3 : avant traitement
Proc3 : début traitement
Proc3 : fin traitement
Proc2 : avant traitement
Proc2 : début traitement
```

C

```
Proc2 : avant traitement
Proc3 : avant traitement
Proc1 : début traitement
Proc1 : fin traitement
Proc2 : début traitement
Proc5 : avant traitement
Proc2 : fin traitement
Proc3 : début traitement
Proc3 : fin traitement
Proc1 : fin
Proc2 : fin
Proc3 : fin
```

D

```
Proc1 : début traitement
Proc1 : fin traitement
Proc1 : fin
Proc2 : avant traitement
Proc2 : début traitement
Proc2 : fin traitement
Proc2 : fin
Proc3 : avant traitement
Proc3 : début traitement
Proc3 : fin traitement
Proc3 : fin
```

E

```
Proc2 : avant traitement
Proc4 : avant traitement
Proc3 : avant traitement
Proc5 : avant traitement
Proc1 : début traitement
Proc4 : début traitement
Proc2 : début traitement
Proc1 : fin traitement
Proc2 : fin traitement
Proc4 : fin traitement
```

F

```
Proc1 : début traitement
Proc1 : fin traitement
Proc1 : fin
Proc2 : avant traitement
Proc2 : début traitement
Proc2 : fin traitement
Proc3 : avant traitement
Proc2 : fin
Proc3 : début traitement
Proc3 : fin traitement
Proc3 : fin
```

G *Aucune de ces réponses n'est correcte.*

Question 24 : La solution proposée à ce problème d'ordre est-elle toutefois correcte ?

A) Oui.
B) Non.

Question 25 : En supposant qu'aucune erreur se produit, un interblocage est il possible ?

A) Oui
B) Non

Tableau de sémaphores 2

Nous souhaitons réaliser une application dans laquelle deux processus doivent écrire simultanément sur la sortie standard, l'un, les 26 lettres de l'alphabet en minuscule, l'autre les 26 lettres en majuscule. Le résultat doit être : AaBbCcDd...

Nous utilisons un tableau de sémaphores pour résoudre le problème de synchronisation sous-

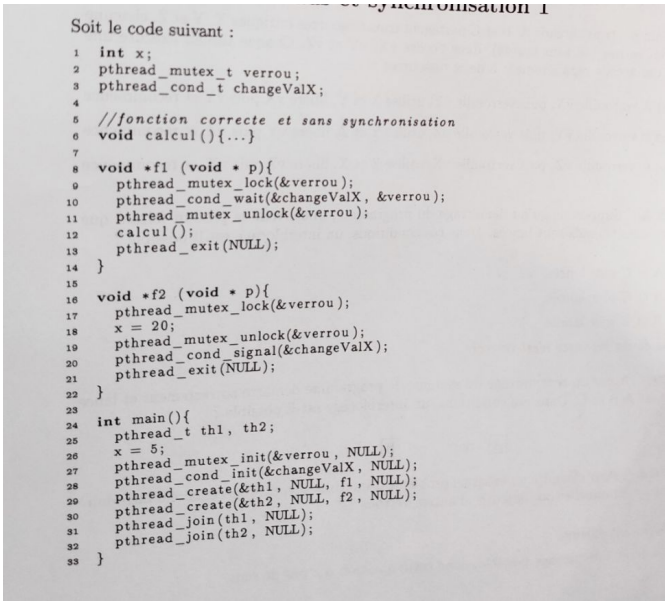
Question 26 : Le nombre de sémaphore minimum permettant de résoudre ce problème est :

A) 1.
B) 26.
C) 52.
D) un nombre strictement inférieur à 3.
E) 3.

F) 2.

Question 27 : Dans cette exercice, la sortie standard est-elle une ressource critique qui nécessite un accès en exclusion mutuelle ?

A) Oui
B) Non



Soit le code suivant :

```
1  int x;
2  pthread_mutex_t verrou;
3  pthread_cond_t changeValX;
4
5  //fonction correcte et sans synchronisation
6  void calcul(){...}
7
8  void *f1 (void * p){
9    pthread_mutex_lock(&verrou);
10   pthread_cond_wait(&changeValX, &verrou);
11   pthread_mutex_unlock(&verrou);
12   calcul();
13   pthread_exit(NULL);
14 }
15
16 void *f2 (void * p){
17   pthread_mutex_lock(&verrou);
18   x = 20;
19   pthread_mutex_unlock(&verrou);
20   pthread_cond_signal(&changeValX);
21   pthread_exit(NULL);
22 }
23
24 int main(){
25   pthread_t th1, th2;
26   x = 5;
27   pthread_mutex_init(&verrou, NULL);
28   pthread_cond_init(&changeValX, NULL);
29   pthread_create(&th1, NULL, f1, NULL);
30   pthread_create(&th2, NULL, f2, NULL);
31   pthread_join(th1, NULL);
32   pthread_join(th2, NULL);
33 }
```

Question 27 : Quelles sont les affirmations correctes ?

A) Les threads th1 et th2 peuvent s'exécutent en séquentiel.
B) L'instruction de la ligne 12 peut s'exécuter avant celle de la ligne 20.
C) L'instruction de la ligne 10 peut s'exécuter avant celle de la ligne 17. Dans ce cas, la ligne 17 ne sera jamais exécutée et un interblocage se produira.
D) Aucun problème n'aura lieu à l'exécution.
E) A l'exécution, le thread th1 peut attendre indéfiniment à la ligne 10.
F) L'instruction de la ligne 20 doit être déplacée pour être entre les lignes 18 et 19.
G) Aucune de ces réponses n'est correcte.

4 Exercice : Threads et synchronisation

Dans un programme, cinq threads T1 - T5 partagent quatre ressources critique R12, R23, R34, R45 et R51, où chaque ressource Rij est partagé entre les threads Ti et Tj et est verrouillé par un verrou (au sens mutex), Vij. Chaque thread exécute une boucle infinie qui accède régulièrement à 2 ressources :

- T1 verrouile V51, puis verrouille V12, utilise R51 et R12, libère V51, puis V12 et recommence.
- T2 verrouile V12, puis verrouille V23, utilise R12 et R23, libère V12, puis V23 et recommence.

- T3 verrouile V23, puis verrouille V34, utilise R23 et R34, libère V23, puis V34 et recommence.
- T4 verrouile V34, puis verrouille V45, utilise R34 et R45, libère V34, puis V45 et recommence.
- T5 verrouile V45, puis verrouille V51, utilise R45 et R51, libère V45, puis V51 et recommence.

Question 23 : Supposons qu'au démarrage du programme, une erreur se produit faisant seulement 3 des 5 threads sont lancés. Dans ces conditions, un interblocage est-il possible ?

A) Oui, si T1, T2 et T4 sont lancés.
B) Oui, si T1, T3 et T5 sont lancés.
C) Oui, si T1, T2 et T3 sont lancés.
D) Aucune.

Question 24 : Dans le cas où seuls 3 threads sont lancés, quel est le degré maximum qu'on obtiendrait à l'exécution ? En d'autres termes, quel est le nombre maximal (parmi les trois) de threads pouvant exécuter simultanément leur section critique ?

A) 3, si T1, T3 et T5 sont lancés.
B) 2, si T1, T3 et T5 sont lancés.
C) 1, si T1, T3 et T5 sont lancés.
D) 1, si T1, T2 et T3 sont lancés.
E) 2, si T1, T2 et T3 sont lancés.
F) 3, si T1, T2 et T3 sont lancés.
G) Aucune.

Question 25 : Après un démarrage du système, le programme démarre correctement et lance les threads. Sans erreur à l'exécution, quel est le degré de parallélisme maximum qu'on obtiendrait d'exécution ? En d'autres termes, quel est le nombre maximum de threads (parmi les cinqs) pouvant lancer simultanément leur section critique (utilisation des ressources).

A) 2 B) 5 C) 4. D) 1. E) 3.

Question 26 : Après un démarrage du système, le programme démarre correctement et lance les threads. Sans erreur à l'exécution, un interblocage est-il possible ?

A) Non
B) Oui

Question 27 : Pour résoudre un éventuel problème d'interblocage, une solution serait :

A) de rien modifier : il n'y a pas d'interblocage, donc cette question n'a pas de sens.
B) d'ajouter une variable conditionnelle par ressource pour qu'un thread soit réveillé uniquement lorsque cette ressource est libérée.
C) de reconsidérer entièrement la solution proposée.
D) Qu'un seul thread charge l'ordre des demandes des ressources (verrouillage).
E) d'utiliser un verrou supplémentaire (unique) que chaque thread verrouille avant de demander les deux ressources et libère avant d'utiliser les ressources.
F) Aucune de ces réponses n'est correcte.

Question 28 : En supposant qu'un interblocage soit possible et que vous avez choisi la solution d'utiliser un verrou supplémentaire pour éviter cet interblocage, quelle est la conséquence possible de ce choix ?

A) Le degré maximum de parallélisme pourrait ne jamais être atteint.
B) Une situation de famine peut apparaître. La famine est la situation dans laquelle un thread restera bloqué indéfiniment.
C) Je n'ai pas choisi cette solution car elle ne résout pas le problème d'interblocage.
D) Il n'y a pas d'interblocage, donc cette question n'a pas de sens.
E) Aucune de ces réponses n'est correcte.

Question 29 : En supposant qu'un interblocage soit possible et que vous avez choisi la solution d'ajouter une variable conditionnelle par ressource pour éviter cet interblocage, quelle est la conséquence possible de ce choix ?

A) Le degré maximum de parallélisme pourrait ne jamais être atteint.
B) Une situation de famine peut apparaître. La famine est la situation dans laquelle un thread restera bloqué indéfiniment.
C) Je n'ai pas choisi cette solution car elle ne résout pas le problème d'interblocage.
D) Il n'y a pas d'interblocage, donc cette question n'a pas de sens.
E) Aucune de ces réponses n'est correcte.

Question 30 : En supposant qu'un interblocage soit possible et que vous avez choisi qu'un seul thread change l'ordre des demandes de ressources (verrouillage) pour éviter cet interblocage, quelle est la conséquence possible de ce choix ?

A) Une situation de famine peut apparaître. La famine est la situation dans laquelle un thread restera bloqué indéfiniment.
B) Il n'y a pas d'interblocage, donc cette question n'a pas de sens.
C) Je n'ai pas choisi cette solution car elle ne résout pas le problème d'interblocage.
D) Le degré maximum de parallélisme pourrait ne jamais être atteint.
E) Aucune de ces réponses n'est correcte.