QCM – Programmation répartie

(2022/2023)

1- Cocher une réponse si elle peut être utilisée(seule) pour mettre en place une exclusion mutuelle :
   - Un verrou au sens mutex et une variable conditionnelle
   - Un verrou au sens mutex
   - Une variable conditionnelle

2- Utilisons la fonction *send()* pour envoyer un message *m* avec le protocole TCP/IP parmi les affirmations suivantes quelles sont celles qui sont correctes :
   - Aucune des autres réponses n'est correct
   - Si la valeur de retour de cette fonction est strictement supérieure à 0 alors le buffer de destination n'était forcément pas plein lors de l'appel de *send*
   - Si la valeur de retour de cette fonction est strictement supérieure à 0 alors des octets du message m ont été copiés dans la couche transport

3- En mode TCP, un serveur peut traiter simultanément plusieurs clients dans un seul processus et sans avoir à créer des threads :
   - Vrai
   - Faux

4- Dans le cadre du multithreading avec POSIX, si plusieurs variables partagées en lecture et écriture sont partagées par différents verrous, est-il possible d'utiliser une variable conditionnelle pour attendre l'occurrence d'un évènement impliquant ces différentes variables partagées.
   - Non
   - Oui, si on considère la façon de protéger les variables partagées
   - Oui

5- Cocher une affirmation si elle est correcte :
   - Le multiplexage permet de scruter tout évènement sur toute entité manipulable via un descripteur de fichier
   - Le multiplexage des entrées/sorties permet d'éviter des situations d'interblocage
   - Le multiplexage ne doit pas être utilisé dans un programme si ce dernier est concurrent
   - Si on utilise le multiplexage des entrées/sorties sur toutes les entrées et sorties utilisées par un programme, aucune attente/blocage ne peut être observé sur ces entrées/sorties

6- La programmation repartie peut permettre :
   - de construire des programmes portables (pouvant s'exécuter sur différentes architectures, OS, …)
   - de partager des ressources
   - d'accélérer des traitements

7- Si à un instant de l'exécution le buffer de réception du socket TCP contient 240 octets et que la socket d'expédition est fermée, que se produira t'il lorsque le récepteur fera un appel à la fonction *recv()* avec une taille de message 320 octets ?

8- Si on utilise des threads pour réaliser une application réelle (non une simulation, il est indispensable que le nombre de threads pouvant s'exécuter en parallèle ne dépasse pas le nombre de cœurs du processeur utilisé pour l'exécution :
   - Vrai
   - Faux

9- L'implémentation d'un processus *Pi* nécessite l'utilisation de la programmation concurrente, nous avons en particulier utilisé la programmation multithread. Quel est le nombre de threads minimum nécessaire et suffisant pour l'implémentation d'un processus *Pi* ?
   - 2
   - 3
   - 4
   - 1
   - 5
   - Un autre nombre de threads

10- Justifier la réponse précédente

11- D'après l'algorithme de Riccart Agrawala, deux types de messages peuvent être reçus par un processus *Pi*, il convient … choisir au moins une réponse :
   - D'étiqueter chaque message pour différencier leur nature (demande ou autorisation)
   - De recevoir les autorisations dans la procédure d'acquisition
   - De séparer clairement la réception des demandes de celles des autorisations, par différents threads ou par des procédures différentes
   - D'utiliser le multiplexage des entrées/sorties pour la réception de n'importe quel message

12- Justifier la réponse précédente

13- L'implémentation d'un processus *Pi* étant multi-thread, on se pose la question du besoin de synchronisation entre threads, cocher une affirmation si elle correcte :
   - Il y a au moins un problème d'attente de l'occurrence d'un évènement (ce qui nécessite l'utilisation d'une ou de plusieurs variables conditionnelles)
   - Il n'y a aucun problème de synchronisation à gérer
   - Il y a au moins un problème d'exclusion mutuelle

14- Si vous avez coché la présence d'un problème d'attente de l'occurrence d'un évènement, décrire quel(s) évènement est/sont à attendre, où se situe l'attente, et qui est responsable du réveil (où se situe-t-il) et quel type de réveil est utilisé (signal ou broadcast)

15- Lors de l'exécution, quel scénario peut/peuvent se produire parmi les suivants :
   - *Pi* envoie une autorisation à un processus *Pj* alors que *Pj* est en section critique
   - *Pi* entre en section critique deux fois de suite sans qu'entre temps un autre processus n'entre en section critique
   - *Pi* reçoit une demande d'un processus *Pj* et la traite pendant que *Pi* est en section critique
   - Une attente/blocage temporaire de l'exécution de *Pi* avant d'entrer en section critique
   - L'envoie d'une autorisation par un processus *Pi* peut procéder immédiatement la sortie de *Pi* de la section critique
   - *Pi* reçoit une demande de la part d'un processus *Pj* en même temps que l'envoie d'une demande par *Pi*
   - *Pi* est hors section critique et reçoit en parallèle des demandes de la part d'autres processus