UNIVERSITÉ BOURGOGNE EUROPE

Examen - Systèmes et Réseaux 1 - Licence 3 Informatique

Durée : 1h30. Documents personnels autorisés. Le barème est indicatif.

Exercice 1: QCM (3,5 pts)

Répondre aux affirmations suivantes par Vrai ou Faux. Toute réponse fausse vous pénalise car elle annule une réponse juste.

Q1 La suppression d'un fichier avec `rm` nécessite les droits d'écriture sur le fichier.

Q2 `chmod 751 monscript.sh` rend le fichier exécutable uniquement par le propriétaire et le groupe.

Q3 Après un `fork()`, le processus fils hérite des descripteurs de fichiers de son père et partage le même espace mémoire.

Q4 Le lien symbolique d'un fichier a son propre inode indépendant de l'inode du fichier.

Q5 Un sémaphore permet la synchronisation entre processus distincts mais pas entre threads d'un même processus.

Q6 Deux sockets ne peuvent pas être créées sur le même port si elles ont la même adresse IP.

Q7 Deux clients peuvent se connecter en même temps sur le même port serveur TCP.

Exercice 2: Scripts (6 pts)

1. Écrire un script `droits.sh` qui prend en argument le nom d'un répertoire, et compte le nombre de fichiers accessibles en lecture et en exécution par l'utilisateur courant dans ce répertoire (directement, pas dans les sous-répertoires).
2. Modifier le script pour qu'il affiche tous les fichiers dans le répertoire donné ayant les mêmes droits qu'un fichier `Reference.txt` passé aussi en argument.
3. On dispose d'un fichier `utilisateurs.txt` contenant, par ligne, les informations suivantes séparées par le caractère " :" :
   `nom_utilisateur : UID : shell : fichier_associe`
   Ecrire un script awk qui :
   (a) Affiche uniquement les utilisateurs dont l'UID est supérieur à 1000, et dont le shell de connexion n'est pas `/usr/sbin/nologin`
   (b) Affiche la liste des utilisateurs ayant plus de 3 fichiers associés (une seule fois le nom de chaque utilisateur concerné).

Exercice 3: Système de fichiers (3 pts)

Dans un système de gestion de fichiers Unix, le bloc de données est de T Ko, chaque pointeur occupe P octets, et chaque inode comprend L liens directs, I liens indirects simples, et D liens indirects doubles.

1. Quel est le rôle de l'inode dans ce système ?
2. Quelle est la taille minimale que doit avoir un fichier pour qu'on soit obligé d'utiliser le lien indirect double ?

Exercice 4: Synchronisation de processus (7,5 pts)

Un pont étroit permet à des voitures de le traverser, mais uniquement dans une seule direction à la fois. Plusieurs voitures peuvent se trouver sur le pont en même temps, à condition qu'elles aillent dans la même direction (Nord → Sud ou Sud → Nord). Lorsqu'une voiture est sur le pont, toutes les voitures qui viennent de l'autre sens doivent attendre que le pont soit complètement libéré avant d'y accéder.

On souhaite synchroniser l'accès des voitures à ce pont à l'aide de sémaphores. Le système est simulé par deux types de processus correspondants aux voitures venant du Nord et à celles venant du Sud. Pour cela, les processus ont accès aux variables partagées `nbNord` et `nbSud` qui comptent le nombre de voitures sur le pont dans le sens Nord→Sud et Sud→Nord, respectivement. Les processus peuvent également utiliser les deux sémaphores `mutex` (pour gérer l'accès aux variables) et `pont` (pour contrôler l'accès au pont).

Questions :

1. Expliquer pourquoi l'utilisation de deux sémaphores (mutex et pont) est nécessaire dans ce problème et indiquer avec quelles valeurs les variables partagées et sémaphores doivent être initialisés au début.
2. Compléter le code ci-dessous pour les voitures venant du Nord, de manière à garantir l'exclusion mutuelle et le respect des contraintes.
3. Donner le code pour les voitures venant du Sud sur le même modèle.
4. Expliquer brièvement une situation possible de famine dans cette solution et proposer une idée (sans la coder) permettant d'éviter ce problème.
5. Expliquer comment les signaux UNIX (par exemple SIGUSR1, SIGUSR2) pourraient être utilisés en remplacement des sémaphores pour avertir un processus "gestionnaire de pont" de l'arrivée d'une voiture du Nord ou du Sud et donner l'accès à chaque voiture.

```
// Voiture venant du Nord
P(mutex);
    if (nbNord == 0) {
        ????
    }
    nbNord = nbNord + 1;
V(mutex);

// Traversée du pont depuis le Nord
TraverserPontDepuisNord();

P(mutex);
    nbNord = ???? ;
    if (nbNord == 0) {
        ????
    }
V(mutex);
```