Examen - Systèmes et Réseaux 1

Licence 3 Informatique

Durée : 2h. Documents personnels autorisés. Le barème est indicatif.

Exercice 1: QCM (2 pts)

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

Q1 Les Pthreads Linux sont implanté dans le noyau.

Q2 La primitive `bind()` permet de connecter une socket à une machine distante.

Q3 L'accès à une base de donnée s'effectue selon le modèle producteur/consommateur.

Q4 Dans un système Linux, les droits d'accès d'un fichier sont stockés dans son inœud.

Exercice 2: Droits (4 pts)

1. Sur un système Unix, quels droits d'accès sont nécessaires à un utilisateur pour pouvoir connaître les droits attribués à un fichier ?
2. Écrire en shell/awk la commande `MemeDroits` telle que `MemeDroits fic` affiche la liste des fichiers du répertoire courant qui possèdent les mêmes droits d'accès que ceux du fichier `fic`.
3. Écrire un script shell qui affiche les 5 plus gros sous-répertoires d'un répertoire dont le nom est passé en paramètre.

Exercice 3: Sémaphores (4 pts)

1. Expliquer en quelques lignes le principe des sémaphores et leur implantation en C.
2. Soient trois programmes dont le code est le suivant :

Programme 1 :

```
while(1) {
  f1();
}
```

Programme 2 :

```
while(1) {
  f2();
}
```

Programme 3 :

```
while(1) {
  f3();
}
```

En utilisant les sémaphores,

(a) quelles sont les opérations à ajouter pour que les fonctions `fi` s'exécutent en s'excluant mutuellement (un seul processus exécute sa fonction `fi`, les autres étant en attente) ? Réécrire les trois programmes modifiés.

(b) quelles sont les opérations à ajouter pour que les fonctions `fi` s'exécutent toujours séquentiellement et toujours dans l'ordre `f1`, `f2`, `f3`, `f1`, ... ?

Exercice 4: Ordonnancement de commandes (3 pts)
Soit une application en C de communication de type client/serveur composée des fichiers `srv.c` contenant le code source du serveur, `cli.c` contenant celui du client, `common.c` contenant des fonctions communes au client et au serveur. Les fichiers `srv.h`, `cli.h` et `common.h` contiennent les déclarations des fonctions et types.

1. Dessiner le graphe de dépendences de ces programmes `srv` et `cli`.
2. Proposer un fichier Makefile le plus concis possible permettant
   — l'obtention des exécutables,
   — le nettoyage des fichiers objets,
   — l'impression des sources (commande `lp fic1 fic2...`)
   — l'archivage des sources (commande `tar cf MonArch fic1 fic2...`)
3. Que se passe-t-il si l'on modifie le fichier `common.c` et que l'on relance la commande `make` ?

Exercice 5: Communications (7 pts)
On considère des processus (tous identiques) organisés en boucle : chaque processus dispose de variables pidPRED et pidSUCC qui contiennent les pid de son prédécesseur et de son successeur dans la boucle.

1. Expliquer comment –en utilisant les signaux– on peut faire écrire dans un fichier la liste des pid des processus formant la boucle. Il convient en particulier d'expliquer comment vous garantissez que la boucle se termine bien. On supposera que c'est le premier processus de la boucle (P_1) qui initie la phase décriture vers le fichier et vérifie si la séquence d'écriture est bouclée.
2. Écrire la partie de programme correspondant à ce travail : la structure globale du programme (lancement, obtention des pid du prédécesseur et du successeur) est supposée donnée. Vous pouvez travailler au choix en C ou en shell.
3. Expliquer comment –en utilisant les signaux– on peut intervertir le sens de la boucle (chaque processus échange son prédécesseur et son successeur).
4. On suppose maintenant que chaque processus a besoin du résultat de calcul (un réel) du processus précédent pour effectuer son calcul (et ceci de façon itérative). Décrire la solution de communication que vous proposez en détaillant comment et où sont initialisés les mécanismes de communication et quand/comment/par qui sont utilisés ces mécanismes.