Examen - Systèmes et Réseaux 1

Licence 3 Informatique

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

Exercice 1: Commandes et Système de fichiers (6 pts)

1. Vers combien de blocs de données de 32ko pointe une indirection triple dans un système à i-nœuds où les numéros de blocs sont codés sur 4 octets ? Expliquer quel est l'intérêt de ces indirections.
2. Donner les commandes à lancer sur le terminal pour :
   (a) afficher la liste des images au format PNG dont le poids est compris entre 200ko et 300ko.
   (b) recopier dans un fichier nommé `listeVar` la liste des variable d'environnement dont le nom contient au moins une fois le caractère
   (c) afficher l'ensemble des fichiers textes codés en UTF-8
   (d) affiche le nombre de fichiers dont le nom contient au moins deux chiffres à partir du répertoire `Documents`
   (e) écrire à la suite du fichier `listeVar` la liste des fichiers dont le chemin relatif contient les répertoires `cours` et `doc`
   (f) afficher l'ensemble des fichiers dans le répertoire `REP` que le propriétaire peut lire, modifier mais pas exécuter mais qu'un membre du groupe propriétaire peut juste lire.
   (g) afficher l'ensemble des liens symboliques du répertoire courant qui pointe vers un fichier PDF.

Exercice 2: Programmation système (3 pts)

Écrire un script shell et/ou awk `renomme préfixe nouveaupréfixe` qui permet de renommer tous les fichiers du répertoire courant dont le préfixe du nom est `préfixe` (et d'extension quelconque) en remplaçant `préfixe` par `nouveaupréfixe`.

Par exemple, si le répertoire courant contient les fichiers `exam.tex`, `exam.toc`, `exam.aux`, `exam.dvi`, `exam.pdf`, ceux-ci seront renommés en `examenSR.tex`, `examenSR.toc`, `examenSR.aux`, `examenSR.dvi`, `examenSR.pdf` par exécution de la commande `renomme exam examenSR`.

Exercice 3: Ordonnancement de commandes (4 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.h` et que l'on relance la commande `make` ?

1/2

Exercice 4: Communications (7 pts)
On souhaite simuler un système bancaire simplifié. Ce système comporte deux types de processus :

les **clients** qui envoient à la banque à intervalles réguliers des ordres de virement sous la forme de trois entiers *cd*, *cc* et *s* pris au hasard (dans des bornes raisonnables) représentant respectivement le compte à débiter, le compte à créditer et la somme à faire transiter de l'un à l'autre des comptes.
la **banque** qui exécute les ordres de virement et doit permettre, sur demande, d'afficher l'état cohérent de tous les comptes et de remettre à zéro l'ensemble des comptes.

1. Décrire de façon détaillée les problèmes à résoudre ainsi que les structures de données et les mécanismes système les mieux adaptés pour programmer ce système. On détaillera notamment le type de communication, les messages échangés entre les processus, comment le processus banque est au courant d'un nouvel ordre de virement, ...
2. Écrire, dans le langage de votre choix, le programme `banque n` qui lance n processus client et un processus banque, en accord avec les mécanismes choisis à la question 1.
3. Sans écrire le code, décrire les mécanismes à mettre place pour ajouter au système un archivage journalier des ordres de virement.
4. Expliquer les limitations de votre système dans le cas d'une utilisation à plus grande échelle en réseau (sur l'Internet, par exemple).