Examen - Systèmes et Réseaux 1

Licence 3 Informatique

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

Exercice 1: Commandes et Système de fichiers (7 pts)

Soit un répertoire `MonRapport`, situé dans dans le répartoire d'acceuil, qui contient les fichiers nécessaires à l'écriture d'un rapport en latex : un fichier `monRapport.tex` et un ensemble d'images.

1. Donner la commande qui permet de créer dans ce répertoire les répertoires PNG et JPG.
2. Donner la commande qui permet de déplacer les images png dans le répertoire PNG et les images jpg dans le répertoire JPG.
3. On va modifier le fichier pour prendre en compte ces modifications. Pour cela on utilisera la commande latex **includegraphics** qui prend en entrée le nom de l'image à afficher. On veut donner le chemin relatif pour les images png et le chemin absolu pour les images jpg.
   (a) Quels droits sont indispensables pour modifier le fichier ?
   (b) Quelle commande utiliser pour savoir si l'utilisateur possède ces droits ?
   (c) Compléter :
   \includegraphics{ image1.png}
   \includegraphics{ image2.jpg}
4. Donner la commande qui va compiler le document latex.
5. On déplace l'ensemble du répertoire `MonRapport` dans le répertoire père de `MonRapport`.
   (a) Quelle commande va-t-on utiliser ?
   (b) On compile le fichier tex. Que va-t-il se passer ? Justifier.
6. Donner la commande qui va afficher :
   (a) les lignes du fichier tex qui demandent l'affichage d'une image png.
   (b) le nombre de lignes du fichier tex qui commencent par *\begin{equation}*
   (c) le nombre de fichiers nommés `monRapport.tex` que vous avez créé sur votre session.

Exercice 2: Make (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 des programmes exécutables `srv` et `cli`.
2. Proposer un fichier Makefile le plus concis possible permettant l'obtention des exécutables.
3. Que se passe-t-il si, après avoir lancé la commande `make` une première fois, l'on modifie le fichier `common.c` et que l'on relance la commande `make` ?

1/2

Exercice 3: Programmation système (4 pts)
On souhaite effectuer un partage de dossiers à étudier (listés dans le fichier texte dossiers.txt) à un ensemble d'évaluateurs (listés dans le fichier utils.txt).

1. Écrire en shell/awk un script `part fic numpart nombrepart` qui va afficher la part numéro `numpart` du fichier texte `fic` dans un partage équitable en `nombrepart` parts (à une ligne près) sous la forme :
   `part numero numpart : de nom1 à nom2 inclu,`
   où nom1 est le nom se trouvant en début de la première ligne de la part et nom2 celui se trouvant en début de la dernière.
2. Écrire le script `assigne fic1 fic2`, qui va répartir la liste des dossiers de `fic2` en autant de parts qu'il y a d'évaluateurs dans le fichier `fic1` et afficher le récapitulatif de la répartition (voir exemple ci-dessous).

Par exemple, si les fichiers `dossiers.txt` et `utils.txt` contiennent les lignes suivantes :

```
$ cat dossiers.txt
tata 1 12 37
tete 5 5 62
titi 2 17 32
toto 9 2 77
tutu 6 11 32
```

```
$ cat utils.txt
Charles ch516 0618192021
Louis lo998 0622232425
```

la commande `assigne utils.txt dossiers.txt` va afficher

```
Charles -> part numero 1 : de tata à titi inclus
Louis -> part numero 2 : de toto à tutu inclus
```

Exercice 4: Communications (6 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 (dans l'Internet, par exemple).