Licence d'informatique
Module de C/Unix

Examen – Partie C

Jean-Luc LEVAIRE

Septembre 2001

Les documents de cours et TD sont autorisés.

On rendra deux copies séparées pour la partie Unix et pour la partie langage C. La partie C est à rendre sur une *copie de couleur blanche.*

1 Questions de cours

Question 1
Un processus sous Unix peut-il ouvrir plusieurs fois simultanément un même fichier ? Pourquoi ? Quel(s) peu(ven)t en être l'(es) intêret(s) et inconvénient(s) ?

Question 2
Dans le système de fichiers Unix, quelle est la différence entre un lien physique et un lien symbolique ?

Question 3
Que signifient les déclarations suivantes :

- `typedef struct {char *l1; char *l2; char *(*f)(char *)} *ftranslatetab;`
- `char *(*getftranslate(char *, char *, ftranslatetab))(char *);`

2 Tri externe

Le but du problème est de réaliser un tri sur un fichier de données en utilisant un arbre binaire de recherche. Le résultat sera stocké dans un second fichier de données. Un fichier de données est constitué d'un ensemble de lignes terminées par un `\n`. Chaque ligne forme un enregistrement, c'est-à-dire une donnée. On supposera dans la suite que les lignes comportent au maximum 255 caractères. Enfin une ligne est elle-même divisée en différents champs séparés par un caractère laissé au choix de l'utilisateur. Un exemple d'un tel fichier est le fichier des mots de passe `/etc/passwd` :

```
user1:motdepasse:501:50:User One:/home/user1:/bin/csh
user2:motdepasse:502:50:User Two:/home/user2:/bin/csh
```

Pour réaliser un index sur un tel fichier, l'utilisateur doit spécifier quel champ représente la clé d'un enregistrement. Pour cela, il donne le caractère séparateur des champs, ainsi que le numéro du champ formant la clé. On supposera que la clé est un entier positif codé dans le fichier en décimal sous la forme d'une chaîne de caractères.

Question 4
Ecrire une fonction

```
unsigned int key(char *record, char sep, int pos)
```

Id : xamc01-sep.tex,v 1.1 2001/09/10 13 :24 :47 marquet Exp

qui retourne la valeur entière de la clé d'un enregistrement `record`. `sep` est le caractère séparateur des champs dans l'enregistrement, `pos` est la position de la clé en commencant en 1. On supposera que cette clé existe toujours dans un enregistrement.

L'index est constitué d'un arbre binaire de recherche, qui stocke l'ensemble des clés correspondant à un fichier de données. En plus de la clé, un nœud de l'arbre contiendra également le déplacement en octets par rapport au début du fichier de la ligne associée à la clé. Ceci permettra d'avoir un accès rapide à cette ligne grâce à la fonction `fseek` lors de la phase d'écriture du fichier trié.

Question 5
Donner le type `NABR` correspondant à un nœud de l'arbre binaire de recherche et écrire la fonction

```
NABR *makeindex(FILE *f, char sep, int pos)
```

qui retourne l'arbre binaire de recherche correspondant au fichier de données pointé par `f`. `sep` et `pos` servent toujours à déterminer la clé d'un enregistrement. Comme le nombre de clés peut être très grand, on s'efforcera de proposer une insertion itérative dans l'arbre binaire.

Pour trier le fichier de données, il ne reste plus qu'à réaliser un parcours infixe de l'arbre binaire de recherche. Pour chaque nœud considéré dans le parcours infixe, il suffira de recopier la ligne du fichier origine, dont on a stocké le déplacement, dans le fichier résultat.

Question 6
Ecrire la fonction

```
void outputresult(NABR *abr, FILE *src, FILE dst)
```

qui stocke dans le fichier pointé par `dest` le résultat du parcours infixe de l'arbre binaire pointé par `abr`, `abr` étant l'index associé au fichier `src`.

Fin du sujet