Licence d'informatique
Module de C/Unix

Examen – Partie C

Jean-Luc LEVAIRE

Septembre 2003

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

Questions de cours

Question 1

1. Expliciter les indirections des blocs de données dans un inode.
2. En supposant que les blocs logiques d'un système de fichiers Unix font 4 Koctets, qu'une adresse de bloc logique occupe 32 bits, et que la table des blocs de données d'un inode contient :
   - 10 adresses de bloc de données en indirection simple ;
   - 3 adresses de bloc de données en indirection double ;
   - 1 adresse de bloc de données en indirection triple ;

   (a) Quelle est la taille maximale possible d'un fichier dans ce sytème de fichiers?

   (b) Combien de blocs logiques seront utilisés par le programme suivant dans ce système de fichiers : (On considèrera les blocs de données et les blocs utilisés pour l'adressage de ces blocs de données dans l'inode.)

```
#include <stdio.h>
#include <stdlib.h>

int main() {

    file *fd = fopen("test", "w");
    char *s = (char *)malloc(8192 * sizeof(char));

    fseek(fd, 12619776);
    fwrite(s, sizeof(char), 8192, fd);
    exit(0);
}
```

□

Question 2

Que signifient les déclarations suivantes ?

1. `int *f(int *, int *);`
2. `int (*d)(int *, int*);`

□

Id: xamc03-sep.tex,v 1.3 2003/09/24 15:08:40 marquet Exp

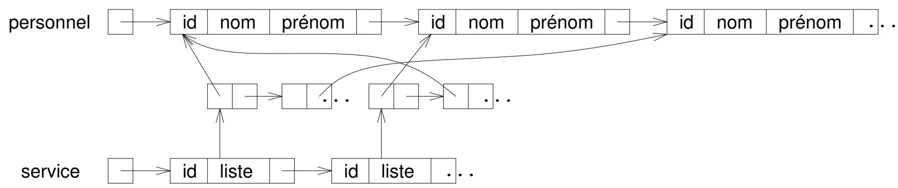

FIG. 1 – Gestion du personnel.

Gestion du personnel

Le but du problème est de gérer le personnel dans une entreprise, et plus précisèment leur appartenance aux services de cette entreprise. L'ensemble du personnel est stocké dans une liste non triée: chaque employé possède un identifiant (un entier positif), un nom et un prénom (deux chaînes de caractères de type `char *`). L'ensemble des services est aussi stocké dans une liste non triée, chaque service possédant un identifiant (un entier positif), et une liste des employés appartenant à ce service. Cette liste pointe directement dans la liste du personnel. La figure 1 vous présente un exemple. Notez qu'un employé peut appartenir à plusieurs services.

Question 3
Donner les types de données nécessaires à l'implémentation des deux listes `personnel` et `service`. □

Question 4
Écrire une fonction

```
unsigned int CreerEmploye(personnel *p, char *nom, char *prenom)
```

qui crée un nouvel employé de nom `nom` et de prénom `prénom`[1]. Son identifiant sera calculé à partir des identifiants existants: ce sera le plus petit entier positif non encore utilisé pour un autre employé. Cet identifiant sera la valeur de retour de la fonction. Pour simplifier, notez que l'on ne copiera pas caractère par caractère les chaînes `nom` et `prenom`, mais que l'on fera une simple copie de pointeurs (il n'y a donc pas d'allocation dynamique des chaînes de caractères). □

Question 5
Écrire une fonction

```
unsigned int CreerService(service *s)
```

qui crée un nouveau service[2]. Son identifiant sera calculé à partir des identifiants existants: ce sera le maximum des identifiants déjà utilisés plus 1. Cet identifiant sera retourné par la fonction. □

Pour simplifier, on supposera dans la suite que les employés et les services existent dans les listes utilisées en paramètre.

Question 6
Écrire une procédure

```
void InsereEmployeService(unsigned int pid, unsigned int sid,
                          personnel p, service s)
```

qui insère l'employé d'identifiant `pid` dans le service d'identifiant `sid`. □

Question 7
Écrire une procédure

[1]On passe un pointeur sur le début de la liste `personnel` au cas où celle-ci est vide.
[2]Voir la remarque ci-dessus pour la liste `service`.

```
void EmployesService(unsigned int sid, service s)
```

qui affiche tous les noms des employés du service d'identifiant `sid`. □

Question 8
Écrire une procédure

```
void ServicesEmploye(char *nom, personnel p, service s)
```

qui affiche tous les identifiants des services auxquels appartient l'employé de nom `nom`. □

Fin du sujet