INFO 403 - TP3 - Annuaire

INFO 403 - TP3 - Annuaire

Objectif

L'objectif de ce TP est de développer un programme de gestion d'annuaire en langage C.

Vous devrez créer un éxecutable qui permettra de gérer un annuaire de contacts.

Consignes

- Vous pourrez travailler en binôme.
- Votre code sera structuré.
- Un Makefile permettra de compiler votre programme.
- Vous vérifierez que votre code ne présente pas de fuite mémoire avec Valgrind.

Détails de l'implémentation

Un contact sera représenté par une structure comme suit :

```
typedef struct {
    char name[NAME_MAX_LENGTH];
    char first_name[NAME_MAX_LENGTH];
    char phone[PHONE_LENGTH];
    char mail[MAIL_MAX_LENGTH];
} Contact ;
```

Fonctionnalités de base

On souhaite pouvoir ajouter, lister, supprimer, rechercher, extraire et fusionner des contacts dans un annuaire.

Le programme prendra en paramètre des options qui permettront de gérer l'annuaire.

Les options sont les suivantes :

```
./annuaire -a "nom" "prenom" "telephone" "mail" "nom du fichier"
./annuaire -l "nom du fichier"
./annuaire -r [ntm] "nom" "nom du fichier"
./annuaire -e [nptm] "nom du fichier"
./annuaire -h
./annuaire -f "nom du fichier1" "nom du fichier2"
./annuaire -s "nom" "nom du fichier"
```

Ces options permettent de :

- -a : ajouter un contact
- -l : lister les contacts. Les informations seront écrites sur la sortie standard.
- -h : afficher l'aide
- -r : rechercher un contact. Le contact sera recherché par nom (option n), téléphone (option t) ou mail (option m). Une seule lettre possible Le contact sera écrit sur la sortie standard.
- -e [nptm] : extraire du fichier les noms, prénoms, numéros de téléphone, adresses mail. Pluisieurs lettres possibles. Les informations seront écrites sur la sortie standard.
- -s : supprimer un contact
- -f : fusionner deux fichiers. Le résultat sera écrit dans le premier fichier.

Menu interactif et table de hashage

Ajouter une option -i qui permet d'ouvrir un menu interactif qui permet de gérer l'annuaire.

```
./annuaire -i "nom du fichier"
Que souhaitez-vous faire ?
1. Ajouter un contact
2. Afficher tous les contacts
3. ....
...
9. Quitter
```

Au lancement du programme, on chargera les contacts depuis le fichier. On les stockera dans une table de hashage. On sauvegardera les contacts à la fin du programme.

La table de hashage sera un tableau de liste chaînée de contacts. On étendra la structure Contact en y ajoutant un pointeur vers le contact suivant.

```
typedef struct contact_list {
    Contact contact;
    struct contact_list *next;
} *Contact_List;
```

On pourra utiliser la structure suivante :

```c
#define SIZE 100
typedef Contact_List HashTable[SIZE];
```

On pourra utiliser une fonction de hashage pour calculer l'index d'un contact dans la table de hashage.

```c
int hash(char *name, char *first_name) {
    int hash = 0;
    for (int i = 0; i < strlen(name); i++) {
        hash += name[i];
    }
    for (int i = 0; i < strlen(first_name); i++) {
        hash += first_name[i];
    }
    return hash % SIZE;
}
```

Tri des contacts, suppression des doublons et arbre binaire de recherche

Pour trier les contacts, on utilisera un arbre binaire de recherche, définit par la structure suivante:

```c
typedef struct node {
    Contact contact;
    struct node *left;
    struct node *right;
} Node;
```

On ajoutera les options suivantes :

```
./annuaire -t [nptm] "nom du fichier"
./annuaire -d [tm] "nom du fichier"
```

- -t [nptm]: trier les contacts. n: par nom, p : par prénom, t : par téléphone, m : par mail. Une seule lettre possible. On écrasera le fichier avec le résultat du tri.
- -d : supprimer les doublons. t : par téléphone, m : par mail. Si les deux lettres sont demandées, il faut que le téléphone ET le mail soient identiques. On écrasera le fichier avec le résultat.

Format du fichier

```
nom1 prenom1 telephone1 mail1
nom2 prenom2 telephone2 mail2
...
```

Bon travail !