Episode 4 - Structures de données

Programmation système unix

BOOTSTRAP

Les structures

Stocker des int et des char *, c'est bien.
Mais pour faire décoller des fusées (ou gérer le dossier des étudiants de Poudlard), nous allons avoir besoin d'un modèle de stockage de données un peu plus pratique et performant.

Les listes chaînées

Il n'y a pas qu'un seul étudiant à Poudlard. Nous allons voir comment gérer des listes d'éléments.

PREAMBULE

SPOILER ALERT
NE LISEZ PAS LA PAGE SUIVANTE

VOUS L’AUREZ VOULU

• Dans Star Wars, Dark Vador est le père de Luke Skywalker.

• Dans The Usual Suspects, Verbal est Keyser Soze.

• Dans Fight Club, Tyler Durden et le narrateur sont la même personne.

• Dans Sixième Sens, Bruce Willis est mort depuis le début.

• Dans Les Autres, les habitants de la maison sont les fantômes et vice-versa.

• Dans Bambi, la mère de Bambi meurt.

• Dans Le Village, les monstres sont les villageois et l’action se situe, en réalité, dans notre époque.

• Dans Harry Potter, Dumbledore meurt.

• Dans La Planète des Singes, l’action se situe sur Terre.

• Dans Le Trône de Fer, Robb Stark et Joffrey Baratheon meurent le soir de leurs noces.

• Dans Twilight, les vampires brillent au soleil.

• Dans Stargate SG-1, Saison 1, Episode 18, O’Neill et Carter sont en Antarctique.

• Dans The Dark Knight Rises, Miranda Tate est Talia Al’Gul.

• Dans Super Mario Bros, la princesse est dans un autre château.

EXERCICES

EXERCICE 1

Ecrire la **structure (et les autres déclarations)** nécessaire qui fera compiler le programme suivant :

main.c

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

// Structs and functions declarations
XXX ICI XXX

// Execution
int         main(void)
{
            t_point     point;

            set_point(&point);
            return (0);
}

void        set_point(t_point *point)
{
            point->x = 42;
            point->y = 21;
}
```

Tips :

https://youtu.be/EIvYuXLN7G8?si=pJESoD39HXwPg5CR

EXERCICE 2

Écrire la **structure** (et les autres déclarations) et la fonction **print_user** afin d'obtenir le résultat suivant :

Terminal

```
~/> ./a.out
Ron Weasley => rwisley@hogwarts.uk
Harry Potter => hpotter@hogwarts.uk
Albus Dumbledore (administrator) => albus@team.hogwarts.uk
```

main.c

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

// Structs and functions declarations
// XXX ICI XXX

// Execution
int         main(void)
{
            t_user    user1;
            t_user    user2;
            t_user    user3;

            user1.name = strdup("Ron Weasley");
            user1.mail = strdup("rwisley@hogwarts.uk");
            user1.status = 0;
            user2.name = strdup("Harry Potter");
            user2.mail = strdup("hpotter@hogwarts.uk");
            user2.status = 0;
            user3.name = strdup("Albus Dumbledore");
            user3.mail = strdup("albus@team.hogwarts.uk");
            user3.status = 2;
            print_user(&user1);
            print_user(&user2);
            print_user(&user3);
            return (0);
}

void        print_user(t_user *user)
{
            // XXX ICI XXX
}
```

EXERCICE 3 - ADD_USER

A partir de cet exercice, nous utiliserons la structure **t_user** que vous avez créé à l'exercice 2.

Écrire la fonction **add_user()** qui ajoute un élément t_user à la liste chaînée dont le premier élément est pointé par **begin**. Si des éléments sont déjà présents, le nouvel élément est ajouté en dernière position.

La fonction retourne un pointeur sur le premier élément de la liste chaînée.

```
t_user      *add_user(t_user *begin, char *name, char *email);

int         main(void)
{
            t_user    *begin = NULL;

            begin = add_user(begin, "Ron Weasley", "rwisley@hogwarts.uk");
            begin = add_user(begin, "Severus Snape", "ssnape@hogwarts.uk");
}
```

Tips :
Pour allouer l'espace des chaînes de caractères *name* et *email*, vous pouvez utiliser strdup() ou malloc()

Tips 2 :
https://www.youtube.com/watch?v=RUZ2Ti9wFx8&list=PLVQYiy6xNUxwmUOmyYSaI6gD1UyfF9MSj&index=1

EXERCICE 4 - COUNT_USER

Écrire la fonction **count_user()** qui retourne le nombre d'élément de la liste chaînée dont le premier élément est pointé par **begin** :

```
int         count_user(t_user *begin);
```

EXERCICE 5 - ADD_USER_FIRST

Écrire la fonction **add_user_first()** qui ajoute un élément t_user à la première position de la liste chaînée dont le premier élément est pointé par **begin.**
La fonction retourne un pointeur sur le (nouveau) premier élément de la liste chaînée.

```
t_user*     add_user_first(t_user *begin);
```

EXERCICE 6 - INSERT_USER

Écrire la fonction **insert_user()** qui ajoute un élément t_user à la position **n** de la liste chaînée dont le premier élément est pointé par **begin.**
La fonction retourne un pointeur sur le premier élément de la liste chaînée.

```
t_user*     insert_user(t_user *begin, int n);
```

EXERCICE 7 - REVERSE_USER

Écrire la fonction **reverse_user()** qui inverse l'ordre de la liste chaînée dont le premier élément est pointé par **begin**. Le premier élément devient le dernier, etc.
La fonction retourne un pointeur sur le (nouveau) premier élément de la liste chaînée.
Seuls les mouvements de pointeurs sont autorisés. Les éléments t_user ne doivent pas être copiés.

```
t_user*     reverse_user(t_user *begin);
```