IR209 - Développement

Rapport de Laboratoire 3: Parcours récursif(suite & fin) & allocation dynamique

Maljean Thimothé, Tasiaux Victor

05-10-2022

Table des Matières

1 Point 2: Gestion des arguments

1.1 Version 1

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


void dirContent(const char *rootpath);


int main(int argc, char* argv[]){

    dirContent(argv[1]);

    return 0;
}


void dirContent(const char *rootpath){
    puts(rootpath);
    DIR *d = opendir(rootpath);
    struct dirent *sd = readdir(d);

    if (d == NULL){
        perror("ERROR: ");
    }

    while (sd != NULL){

        if(sd -> d_type == DT_DIR){
        }

        else{
            if(sd -> d_type == DT_REG){
                printf("Nom du fichier: %s",sd->d_name);
            }
        }

    }

}
```

Dans cette version, on a vidé la partie lecture de directory pour se concentrer sur l'arv. On appelle la fonction `dirContent`, et on passe le string entré dans le terminal en paramètre de la fonction.

1.2 Version 2

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


void dirContent(const char *rootpath);
```

```c

int main(int argc, char* argv[]){
    if(argc == 2){
        dirContent(argv[1]);
    }
    else{

        if(argc > 2){
            puts("ERROR: too many arguments");
            return 0;
        }

        else {
            puts("ERROR: too few arguments");
            return 0;
        }
    }

    return 0;
}


void dirContent(const char *rootpath){
    DIR *d = opendir(rootpath);

    if (d == NULL){
        perror("ERROR: ");
    }
    struct dirent *sd = readdir(d);



    while (sd!= NULL){

       if(sd -> d_type == DT_DIR){

        }


        else{
            if( sd -> d_type == DT_REG){
                printf("Nom du fichier: %s\n",sd->d_name);
            }
        }
       sd = readdir(d);
    }
}
```

Dans cette version on ajoute un contrôle du nombre d'arguments grâce à argc.

2 Point 3: Allocation dynamique du Path

2.0.1 Version 1: tableau à taille définie

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

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


void dirContent(const char *rootpath);


int main(int argc, char* argv[]){
    if(argc == 2){
        dirContent(argv[1]);
    }
    else{

        if(argc > 2){
            puts("ERROR: too many arguments");
            return 0;
        }

        else {
            puts("ERROR: too few arguments");
            return 0;
        }
    }

    return 0;
}


void dirContent(const char *rootpath){
    DIR *d = opendir(rootpath);

    if (d == NULL){
        perror("ERROR: ");
    }
    struct dirent *sd = readdir(d);



    while (sd!= NULL){

        if(sd -> d_type == DT_DIR && strcmp(sd -> d_name,".") !=0 &&
           strcmp(sd -> d_name,"..") !=0 ){
            char new_path[4096] = "";
            strncat(new_path,rootpath,strlen(rootpath));
            strncat(new_path,"/",1);
            strncat(new_path,sd ->d_name,strlen(sd->d_name));
            printf("Path:%s\n",new_path);

            dirContent(new_path);
        }

        else{
            if( sd -> d_type == DT_REG){
                printf("Nom du fichier: %s\n",sd->d_name);
            }
        }
       sd = readdir(d);
    }
}
```

Dans cette version, on utilise un tableau à taille fixe et strncat pour créer le nouveau répertoire.

On vérifie que le fichier est un dossier et qu'il n'est pas le dossier courant(`.`) ou le dossier parent (`..`).

2.1 Version 2: Avec malloc

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


void dirContent(const char *rootpath);


int main(int argc, char* argv[]){
    if(argc == 2){
        dirContent(argv[1]);
    }
    else{

        if(argc > 2){
            puts("ERROR: too many arguments");
            return 0;
        }

        else {
            puts("ERROR: too few arguments");
            return 0;
        }
    }

    return 0;
}


void dirContent(const char *rootpath){
    DIR *d = opendir(rootpath);

    if (d == NULL){
        perror("ERROR: ");
    }
    struct dirent *sd = readdir(d);



    while (sd!= NULL){

        if(sd -> d_type == DT_DIR && strcmp(sd -> d_name,".") !=0 &&
           strcmp(sd -> d_name,"..") !=0 ){
            char* new_path = (char*)malloc(strlen(rootpath)+strlen(sd->
               d_name)+2);
            strcpy(new_path,"");
            strncat(new_path,rootpath,strlen(rootpath));
            strcat(new_path,"/");
            strncat(new_path,sd ->d_name,strlen(sd->d_name));
            printf("Path:%s\n",new_path);

```

```
51
52              dirContent(new_path);
53              free(new_path);
54          }
55
56          else{
57              if( sd -> d_type == DT_REG){
58                  printf("Nom du fichier: %s\n",sd->d_name);
59              }
60          }
61         sd = readdir(d);
62      }
63  }
```

On emploie ici la fonction `malloc` pour créer un tableau de taille variable.

On alloue de la mémoire pour le nouveau chemin → c'est la longueur du rootpath + du d_name + 2. 2 correspond au "/" + le \0 qui marque la fin du tableau. Comme malloc est à la base un void, on caste le malloc en tableau de char.