IR209 - Développement

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

Maljean Thimothé, Tasiaux Victor

Table des Matières

1 Point 2: Gestion des arguments																3										
	1.1	Versio	n 1																 							3
	1.2	Versio	n 2																						•	3
2 Point 3: Allocation dynamique du Path															4											
		2.0.1	Vers	sion	1:	tab	lea	u a	à ta	aill	e d	éfi	nie	e												4
	2 1	Vorcio	n 2. A		m	مالہ																				6

1 Point 2: Gestion des arguments

1.1 Version 1

```
#include <dirent.h>
2 #include <stdlib.h>
3 #include <stdio.h>
4 #include <string.h>
7
   void dirContent(const char *rootpath);
8
9
int main(int argc, char* argv[]){
11
       dirContent(argv[1]);
12
13
14
       return 0;
15
   }
16
17
18
  void dirContent(const char *rootpath){
19
       puts(rootpath);
20
       DIR *d = opendir(rootpath);
21
       struct dirent *sd = readdir(d);
23
       if (d == NULL){
24
           perror("ERROR: ");
25
       }
26
27
       while (sd != NULL){
28
29
           if(sd -> d_type == DT_DIR){
           }
31
           else{
32
                if(sd -> d_type == DT_REG){
34
                    printf("Nom du fichier: %s",sd->d_name);
                }
           }
38
       }
39
40 }
```

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

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

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

```
9
10
   int main(int argc, char* argv[]){
11
        if(argc == 2){
12
            dirContent(argv[1]);
13
        else{
14
15
            if(argc > 2){
16
17
                puts("ERROR: too many arguments");
18
                return 0;
            }
19
20
21
            else {
                puts("ERROR: too few arguments");
23
                return 0;
24
            }
25
        }
27
        return 0;
28 }
29
31
   void dirContent(const char *rootpath){
32
        DIR *d = opendir(rootpath);
34
        if (d == NULL) {
            perror("ERROR: ");
37
        struct dirent *sd = readdir(d);
39
40
        while (sd!= NULL){
41
42
43
           if(sd -> d_type == DT_DIR){
44
45
            }
46
47
48
            else{
49
                if( sd -> d_type == DT_REG) {
50
                     printf("Nom du fichier: %s\n",sd->d_name);
51
52
            }
53
           sd = readdir(d);
54
        }
55 }
```

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

```
1 #include <dirent.h>
2 #include <stdlib.h>
```

```
3 #include <stdio.h>
 4 #include <string.h>
 6
 7
    void dirContent(const char *rootpath);
 8
 9
10
    int main(int argc, char* argv[]){
        if(argc == 2){
11
12
            dirContent(argv[1]);
13
        }
14
        else{
15
16
            if(argc > 2){
                puts("ERROR: too many arguments");
17
18
                return 0;
19
            }
20
            else {
22
                puts("ERROR: too few arguments");
23
                return 0;
            }
24
25
        }
26
27
        return 0;
28
   }
29
31
    void dirContent(const char *rootpath){
32
        DIR *d = opendir(rootpath);
34
        if (d == NULL){
35
            perror("ERROR: ");
        struct dirent *sd = readdir(d);
39
40
        while (sd!= NULL){
41
42
43
            if(sd -> d_type == DT_DIR && strcmp(sd -> d_name,".") !=0 &&
                strcmp(sd -> d_name,"..") !=0 ){
                char new_path[4096] = "";
44
45
                strncat(new_path,rootpath,strlen(rootpath));
46
                strncat(new_path,"/",1);
                strncat(new_path,sd ->d_name,strlen(sd->d_name));
47
48
                printf("Path:%s\n", new_path);
49
50
                dirContent(new_path);
            }
51
            else{
53
                if( sd -> d_type == DT_REG){
54
                     printf("Nom du fichier: %s\n",sd->d_name);
                }
57
            }
58
           sd = readdir(d);
        }
60
    }
```

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

```
1 #include <dirent.h>
2 #include <stdlib.h>
3 #include <stdio.h>
4 #include <string.h>
5
6
7
   void dirContent(const char *rootpath);
8
9
int main(int argc, char* argv[]){
11
       if(argc == 2){
           dirContent(argv[1]);
12
13
       }
14
       else{
15
16
            if(argc > 2){
17
                puts("ERROR: too many arguments");
18
                return 0;
19
            }
20
21
            else {
22
                puts("ERROR: too few arguments");
23
                return 0;
            }
24
25
       }
26
27
       return 0;
28 }
29
30
31
   void dirContent(const char *rootpath){
32
       DIR *d = opendir(rootpath);
34
       if (d == NULL){
35
            perror("ERROR: ");
       }
       struct dirent *sd = readdir(d);
40
41
       while (sd!= NULL){
42
43
            if(sd -> d_type == DT_DIR && strcmp(sd -> d_name,".") !=0 &&
               strcmp(sd -> d_name,"..") !=0 ){
44
                char* new_path = (char*)malloc(strlen(rootpath)+strlen(sd->
                   d_name)+2);
                strcpy(new_path,"");
45
                strncat(new_path,rootpath,strlen(rootpath));
46
                strcat(new_path,"/");
47
48
                strncat(new_path,sd ->d_name,strlen(sd->d_name));
49
                printf("Path:%s\n",new_path);
50
```

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