Université Toulouse III – Paul Sabatier
L2 Informatique

Système 2
6 avril 2013
Semestre 4

Projet — Système de Gestion de Fichiers

Antoine de ROQUEMAUREL (Groupe 2.2)

1 Complétion des fonctions de base du SGF (`mon_SGF.c`)

1.1 Lecture et écriture du SGF

`lire_superbloc` Afin de lire le superbloc, cette fonction positionne tout d'abord le curseur au premier bloc du fichier puis lit dans `mon_SGF->superbloc` les données du superbloc, sa taille est connue grâce à `sizeof(superbloc_t)`.

Si une erreur se produit, lors du `read`, la fonction retourne `EXIT_READ_PB`.

`lire_table_inoeuds` La lecture de la table des inoeuds est le même principe que pour le superbloc, nous savons que celle-ci est écrite à partir du bloc 1 [1], ainsi il faut se positionner au bloc 1, puis lire `sizeof(inoeud_t)` octets.

En cas d'erreur du `read`, la fonction retourne `EXIT_READ_PB`.

`lire_SGF` La lecture du SGF ne peut se faire que si le SGF est ouvert correctement, cette pré-condition est vérifiée à l'aide de `assert`. Afin de lire le contenu du SGF, il nous suffit d'utiliser les deux fonctions écrites précédemment, cependant avant d'appeler ses fonctions, un `malloc` doit être fait.

En cas d'erreur d'allocation mémoire, la fonction retourne `EXIT_MEM_ALLOC`.

`ouvrir_SGF` Tout d'abord, il faut allouer la mémoire pour la structure de type `SGF_t` contenant notre SGF, si ce `malloc` s'effectue correctement, il faut ouvrir le fichier `DEVICE_NAME` à l'aide de la fonction `open`.

Nous ne gérons pas le cas où le `open` retourne `NULL`, en effet ce cas est systématiquement géré après l'appel de notre fonction.

En cas d'erreur d'allocation mémoire, la fonction retourne `EXIT_MEM_ALLOC`.

1.2 Gestion des inoeuds et blocs

`bloc_libre_suivant` La liste des blocs libre est représentée par une liste chaînée. Sur le disque dur, à la position de chaque bloc libre est stocké le numéro du bloc libre suivant.

Ainsi, en se déplaçant de `num_bloc` blocs [2], nous pourrons ensuite lire le numéro du bloc libre suivant avec un simple `read`.

En cas de problème au niveau du `read`, la fonction retourne `EXIT_READ_PB`.

`allouer_n_blocs_dans_inoeud` Afin d'allouer le nombre de blocs à l'inoeud, il faut tout d'abord chercher le premier bloc libre de l'inoeud et vérifier que derrière celui-ci l'inoeud possède assez de place afin d'allouer le nombre de blocs demandés.

Ensuite, on alloue à la table des liens le premier numéro de bloc libre puis on met à jour le premier bloc libre. À la fin du programme on écrit nos modifications sur le disque dur à l'aide des fonctions habituels `ecrire_superbloc` et `ecrire_table_inoeud`

Si l'inoeud ne possède pas assez de blocs libre afin d'allouer `nb_blocs_a_allouer`, la fonction retourne `EXIT_FS_FULL`.

1. le bloc 0 étant réservé au superbloc
2. la taille d'un bloc étant connue dans le superbloc

lire_donnees_dans_inoeud Pour lire les données d'un inoeud, tant que la taille des données lues, n'est pas égale à `data_size`, on lit bloc par bloc, en faisant attention au dernier bloc, auquel on aura pas forcément besoin de lire le bloc entier, mais peut être uniquement le début. Afin de lire un bloc, j'utilise la fonction `lire_bloc`, pour le cas du dernier bloc, c'est la fonction `lire_donnees_dans_bloc` qui est utilisée, en effet, celle-ci permet de ne lire qu'une partie des données d'un bloc.

ecrire_donnees_dans_inoeud Le fonctionnement de cette fonction est similaire à la fonction `lire_donnees_dans_inoeud`, cependant cette fois-ci nous utilisons les fonctions `ecrire_bloc` et `ecrire_donnees_dans_bloc` pour le cas du dernier bloc. De plus avant d'appeler ces fonctions, il ne faut pas oublier d'allouer les bloc si c'est nécessaire à l'aide de `allouer_n_blocs_dans_inoe`

Une fois les données écrites, nous mettons à jour la taille de l'inoeud en lui ajoutant les données écrites.

creer_inoeud Cette fonction cherche tout d'abord le premier inoeud libre dans la table des inoeuds, une fois cette inoeud trouvé elle met la taille de l'inoeud à zéro et lui affecte le type d'inoeud passé en paramètre. Je suppose que lors de l'appel de cette fonction, tous les blocs de cet inoeud sont correctement libérés.

Si aucun inoeud n'est libre, alors la fonction retourne `NO_INOEUD`

inoeud_libre Le premier inoeud libre est trouvé en parcourant la table des inoeuds jusqu'à trouver un inoeud ayant pour type `INOEUD_LIBRE`, si nous arrivons à la fin de la table sans avoir trouvé d'inoeud libre, alors il faut renvoyer `NO_INOEUD`.

liberer_blocs_du_inoeud Tout d'abord, on cherche la position du premier bloc dans l'inoeud, une fois celui-ci trouvé, on parcours `liens_directs_bloc` jusqu'au premier bloc nul, ou la fin de l'inoeud.

À chaque itération de la boucle, on ajoute un bloc libre dans la liste chaînée, pour cela on écrit dans le fichier le numéro du premier bloc libre à l'emplacement du bloc courant, une fois celui-ci écrit, on met à jour le premier bloc libre qui est égal au bloc courant.
Ainsi chaque nouveau bloc libre est ajouté en début de liste. Une fois tous les blocs libérés, on met à jour la table des inoeuds et le superbloc.

La fonction retourn `EXIT_PARAM` si `premier_bloc` n'appartient pas à l'inoeud, et `EXIT_WRITE_PB` si un problème apparaît lors du `write`.

liberer_inoeud Pour libérer un inoeud, il faut appeler la fonction développée précédemment, `liberer_blocs_du_inoeud`, celle-ci libérera tous les blocs de l'inoeud. Une fois les blocs libérés correctement, la taille et le type de l'inoeud sont modifiés. Ensuite il ne faut pas oublier de mettre à jour la table des inoeuds dans le fichier à l'aide de la fonction `ecrire_table_inoeuds`.

2 Commandes d'accès au SGF

2.1 `ls -l` (`mon_ls.c`)

Pour la commande `ls -l`, peu de modifications ont étés nécessaires. En effet, j'ai créer une nouvelle fonction devant s'occuper de l'affichage d'un inoeud ayant le prototype le suivant :

```
void afficherInoeud(int inoeud, char* chemin, int taille);
```

Listing 1 – Prototype de `afficherInoeud`

`inoeud` est le numéro de l'inoeud, `chemin` le chemin à afficher pour l'inoeud et `taille` la taille de l'inoeud, si celle-ci est égale à -1, alors elle ne sera pas affichée, cela correspond à un `ls -l`.

Ainsi, grâce à cette fonction, la gestion du `ls -l` devient facile, on vérifie les paramètre grâce à `argc` et `argv`, ajouter un paramètre modifie les test de vérifications que la fonction est appelée

correctement. Si le `-l` est omis, alors on passe en paramètre de `afficherInoeud -1` sinon on lui passe la taille de l'inoeud obtenue dans la table des inoeuds.

2.2 `du (mon_du.c)`

La commande `mon_du` à été basée sur la commande `mon_ls`, ainsi le début des deux fonctions sont très similaires.

Afin de calculer la taille des répertoires, deux sous-programmes ont été créés, un premier retournant la taille d'un inoeud passé en paramètre, et une deuxième fonction qui calcul la somme du répertoire passé en paramètre et de tous ses fils.

```
int tailleSousRep(int inoeud, SGF_t* mon_SGF);
int tailleTotalRepertoire(char* chemin, SGF_t* mon_SGF);
```

Listing 2 – Prototype de `afficherInoeud`

`tailleSousRep` Afin de calculer la taille d'un inoeud, il suffit d'utiliser l'attribut `taille` présent pour chaque inoeud dans la table des inoeuds. Cette fonction permet d'alléger l'écriture du code et de faciliter la compréhension du programme.

`tailletotalrepertoire` Ce programme additionne la taille des répertoires de tous ses fils, pour cela, il est appelé récursivement jusqu'à qu'il ne reste que `.` et `..` (fond de l'arbre). Il utilise le sous-programme `tailleSousRep` à chaque appel de fonction.

Le sous-programme peut retourner `EXIT_PARAM` si le chemin est inexistant et `EXIT_DEVICE` si le SGF est inaccessible

3 Exemples d'exécution

```
#(ssh) aroquemaurel@Garp : ~/cours/L2/systeme4/sgf
#[0] % mon_formater
mon_formater: le périphérique ".device_name" contient peut-être des données qui vont être supprimées.
mon_formater: êtes-vous sur de vouloir le formater ? [oN] :o
mon_formater: nb inoeuds: 10, nb blocs 64, taille d'un bloc 64 o (taille min: 16 o)
mon_formater: terminé

#(ssh) aroquemaurel@Garp : ~/cours/L2/systeme4/sgf
#[0] % mon_affiche_SGF
*********************************
* Systeme de Gestion de Fichiers *
*********************************
Superbloc
nb_max_inoeuds    : 10
taille_du_SF      : 64
taille_bloc       : 64
premier_bloc_libre: 10
Table des inoeuds
0  type  :  2  taille: 48 o,  blocs:  9, -1, -1, -1, -1, -1, -1, -1, -1, -1,
1  type  :  0  taille:  0 o,  blocs: -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
2  type  :  0  taille:  0 o,  blocs: -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
3  type  :  0  taille:  0 o,  blocs: -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
4  type  :  0  taille:  0 o,  blocs: -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
5  type  :  0  taille:  0 o,  blocs: -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
6  type  :  0  taille:  0 o,  blocs: -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
7  type  :  0  taille:  0 o,  blocs: -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
8  type  :  0  taille:  0 o,  blocs: -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
9  type  :  0  taille:  0 o,  blocs: -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
Liste des blocs libres :
10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29
30  31  32  33  34  35  36  37  38  39  40  41  42  43 44  45  46  47  48  49  50
51  52  53  54  55  56  57  58  59  60  61  62  63
```

Listing 3 – Exemple d'utilisation de `mon_formater` et `mon_afficher`

```
(ssh) aroquemaurel@Garp : ~/cours/L2/systeme4/sgf
[0] % mon_mkdir /dir1 ; mon_mkdir /dir2 ; mon_mkdir /dir3 ; mon_mkdir /dir4 ; mon_mkdir /dir5

(ssh) aroquemaurel@Garp : ~/cours/L2/systeme4/sgf
[0] % mon_ls /
   0 '/' :
   0 '.' :
   1 'dir1' :
   2 'dir2' :
   3 'dir3' :
   4 'dir4' :
   5 'dir5' :

(ssh) aroquemaurel@Garp : ~/cours/L2/systeme4/sgf
[0] % mon_affiche_SGF
*********************************
* Systeme de Gestion de Fichiers *
*********************************
Superbloc
nb_max_inoeuds    : 10
taille_du_SF      : 64
taille_bloc       : 64
premier_bloc_libre: 22
Table des inoeuds
0  type  :  2  taille:168 o,  blocs:  9, 12, 19, -1, -1, -1, -1, -1, -1, -1,
1  type  :  2  taille: 72 o,  blocs: 10, 11, -1, -1, -1, -1, -1, -1, -1, -1,
2  type  :  2  taille: 72 o,  blocs: 13, 14, -1, -1, -1, -1, -1, -1, -1, -1,
3  type  :  2  taille: 72 o,  blocs: 15, 16, -1, -1, -1, -1, -1, -1, -1, -1,
4  type  :  2  taille: 72 o,  blocs: 17, 18, -1, -1, -1, -1, -1, -1, -1, -1,
5  type  :  2  taille: 72 o,  blocs: 20, 21, -1, -1, -1, -1, -1, -1, -1, -1,
6  type  :  0  taille:  0 o,  blocs: -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
7  type  :  0  taille:  0 o,  blocs: -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
8  type  :  0  taille:  0 o,  blocs: -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
9  type  :  0  taille:  0 o,  blocs: -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
Liste des blocs libres :
22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
41  42  43  44  45  46  47  48  49  50 51  52  53  54  55  56  57  58  59
60  61  62  63

#(ssh) aroquemaurel@Garp : ~/cours/L2/systeme4/sgf
#[0] % mon_mkdir /dir1/sdir1 && mon_mkdir /dir1/sdir1/ssdir1 && mon_mkdir /dir1/sdir1/ssdir1/sssdir1

#(ssh) aroquemaurel@Garp : ~/cours/L2/systeme4/sgf
#[0] % mon_ls /dir1/sdir1/ssdir1
   7 '/dir1/sdir1/ssdir1' :
   7 '.' :
   6 '..' :
   8 'sssdir1' :

#(ssh) aroquemaurel@Garp : ~/cours/L2/systeme4/sgf
#[0] % mon_ls -l /
   0    168 o   '/' :
   0    168 o   '.' :
   1     96 o   'dir1' :
   2     72 o   'dir2' :
   3     72 o   'dir3' :
   4     72 o   'dir4' :
   5     72 o   'dir5' :
```

Listing 4 – Exemple d'utilisation de `mon_mkdir` et `mon_ls`

```
#(ssh) aroquemaurel@Garp : ~/cours/L2/systeme4/sgf
#[0] % mon_rmdir /dir4 ; mon_rmdir /dir3

#(ssh) aroquemaurel@Garp : ~/cours/L2/systeme4/sgf
#[0] % mon_ls -l /
0   120 o   '/' :
0   120 o   '.' :
1    96 o   'dir1' :
2    72 o   'dir2' :
5    72 o   'dir5' :

#(ssh) aroquemaurel@Garp : ~/cours/L2/systeme4/sgf
#[0] % mon_affiche_SGF
*********************************
* Systeme de Gestion de Fichiers *
*********************************
Superbloc
```

```
nb_max_inoeuds    : 10
taille_du_SF      : 64
taille_bloc       : 64
premier_bloc_libre: 16
Table des inoeuds
0  type  :  2  taille:120 o,  blocs:  9, 12, -1, -1, -1, -1, -1, -1, -1, -1,
1  type  :  2  taille: 96 o,  blocs: 10, 11, -1, -1, -1, -1, -1, -1, -1, -1,
2  type  :  2  taille: 72 o,  blocs: 13, 14, -1, -1, -1, -1, -1, -1, -1, -1,
3  type  :  0  taille:  0 o,  blocs: -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
4  type  :  0  taille:  0 o,  blocs: -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
5  type  :  2  taille: 72 o,  blocs: 20, 21, -1, -1, -1, -1, -1, -1, -1, -1,
6  type  :  2  taille: 96 o,  blocs: 22, 23, -1, -1, -1, -1, -1, -1, -1, -1,
7  type  :  2  taille: 96 o,  blocs: 24, 25, -1, -1, -1, -1, -1, -1, -1, -1,
8  type  :  2  taille: 72 o,  blocs: 26, 27, -1, -1, -1, -1, -1, -1, -1, -1,
9  type  :  0  taille:  0 o,  blocs: -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,

Liste des blocs libres :
16  15  19  18  17  28  29  30  31  32  33  34  35  36  37 38  39  40  41  42
43  44  45  46  47  48  49  50  51  52 53  54  55  56  57  58  59  60  61  62  63
```

Listing 5 – Exemple d'utilisation de `mon_rmdir`

```
#(ssh) aroquemaurel@Garp : ~/cours/L2/systeme4/sgf
#[0] % mon_ls -l /dir1/sdir1
   6     96 o   '/dir1/sdir1' :
   6     96 o   '.' :
   1     96 o   '..' :
   7     96 o   'ssdir1' :

#(ssh) aroquemaurel@Garp : ~/cours/L2/systeme4/sgf
#[0] % mon_dur /
zsh: correct 'mon_dur' to 'mon_du' [nyae]? y
624 o
#(ssh) aroquemaurel@Garp : ~/cours/L2/systeme4/sgf
[0] % mon_ls -l /dir1/
   1     96 o   '/dir1/' :
   1     96 o   '.' :
   0    120 o   '..' :
   6     96 o   'sdir1' :

#(ssh) aroquemaurel@Garp : ~/cours/L2/systeme4/sgf
#[0] % mon_du /dir1/sdir1
264 o

#(ssh) aroquemaurel@Garp : ~/cours/L2/systeme4/sgf
#[0] % mon_du /dir5
72 o
```

Listing 6 – Exemple d'utilisation de `mon_du`