

# TP1 - Structure d'un système de fichier et layout mémoire

Matthieu Lemerre      Julien Simonnet      Dorian Lesbre

IN201 - Cours de système d'exploitation

## Contents

<b>1</b>	<b>Exploration d'un système de fichiers</b>	<b>1</b>
1.1	Préparation . . . . .	1
1.2	Format du fichier . . . . .	2
1.3	Format des fichiers . . . . .	3

## 1 Exploration d'un système de fichiers

Le but de cette section est de savoir décoder (ou *parser*) un fichier binaire, contenant l'image d'un système de fichier au format ROMFS.

### 1.1 Préparation

**Question 1.** *Creez un nouveau fichier C avec le contenu suivant:*

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <assert.h>
#include <unistd.h>
#include <sys/mman.h>

void decode(struct fs_header *p, size_t size){

}

int main(void){
```

```

int fd = open("fs.romfs",O_RDONLY);
assert(fd != -1);
off_t fsize;
fsize = lseek(fd,0,SEEK_END);

// printf("size is %d", fsize);

char *addr = mmap(addr, fsize, PROT_READ, MAP_SHARED, fd, 0);
assert(addr != MAP_FAILED);
decode(addr, fsize);
return 0;
}

```

Ce programme ouvre le fichier fs.romfs, récupère sa taille, puis le rend accessible au processus Linux en le mettant dans son espace d'adressage. Ainsi, le *i*ème octet du fichier est disponible à l'adresse `&addr[i]`.

## 1.2 Format du fichier

Le format du système de fichier est le suivant (extrait de la documentation officielle):

offset	content	
	+---+---+---+---+	
0	-   r   o   m   \	
	+---+---+---+---+	The ASCII representation of those bytes
4	1   f   s   -   /	(i.e. "-rom1fs-")
	+---+---+---+---+	
8	full size	The number of accessible bytes in this fs.
	+---+---+---+---+	
12	checksum	The checksum of the FIRST 512 BYTES.
	+---+---+---+---+	
16	volume name	The zero terminated name of the volume,
	: :	padded to 16 byte boundary.
	+---+---+---+---+	
xx	file	
	: headers :	

Quelques précisions supplémentaires:

- “Padded to 16 byte boundary” signifie que le nom du volume est suivi d’autant de zéro que nécessaire pour que l’offset du premier “file header” soit un multiple de 16.
- Tous les nombres sur plusieurs octets sont dans l’ordre *big-endian* (les adresses les plus petites contiennent les octets les plus significatifs).

**Question 2.** Définissez une structure `fs_header` tel que les offsets de ses champs correspondent aux offsets dans le fichier (ainsi, le champs `checksum` doit être de taille 4 et correspondre à l’offset 12).

Pour cela, utilisez le header `stdint.h`, qui définit des types `uint8_t`, `uint16_t`, et `uint32_t`, dont la taille est garantie.

**Question 3.** À l’aide de la structure, vérifiez que les 8 premiers octets du fichier correspondent à ceux énoncés dans la spécification.

**Question 4.** Écrivez une fonction `uint32_t read32(char ptr[4])` qui lit un mot de 32 bit écrit en mémoire dans le format *bit*-endian, et vérifiez que la taille du fichier écrite dans le champs `size` est inférieure à la taille de fichier passée en paramètre à `decode`.

**Question 5.** Écrivez une fonction qui arrondit un nombre au multiple de 16 supérieur. À noter: si ce nombre est déjà un multiple de 16, la fonction ne doit rien faire.

**Question 6.** Utilisez cette fonction pour récupérer l’offset correspondant au premier “file header”.

À noter: l’expression “padded to 16 byte boundary” signifie qu’on rajoute du “padding” (des octets inutiles) afin que l’offset du file header suivant soit un multiple de 16.

### 1.3 Format des fichiers

**Question 7.** Écrivez une structure de donnée correspondant au format des en-tête de fichiers (file headers), donné ci-après.

offset	content
--------	---------

	+---+---+---+---+
--	-------------------

0	next filehdr X  The offset of the next file header
---	--

```

+---+---+---+---+ (zero if no more files)
4 | spec.info | Info for directories/hard links/devices
+---+---+---+---+
8 | size | The size of this file in bytes
+---+---+---+---+
12 | checksum | Covering the meta data, including the file
+---+---+---+---+ name, and padding
16 | file name | The zero terminated name of the file,
: : padded to 16 byte boundary
+---+---+---+---+
xx | file data |
: :
0

```

Since the file headers begin always at a 16 byte boundary, the lowest 4 bits would be always zero in the next filehdr pointer. These four bits are used for the mode information. Bits 0..2 specify the type of the file; while bit 4 shows if the file is executable or not.

Précisions que le “offset of the next file header” est l’offset relatif au début du fichier.

**Question 8.** *Le contenu d’une répertoire est décrit par une liste chaînée de “file headers”. Notez que des informations supplémentaires sont contenues dans les 4 derniers bits du champs “next”, qu’il faut supprimer avant de pouvoir suivre le pointeur.*

*Écrivez une fonction `ls` qui affiche le nom de tous les fichiers d’un répertoire en partant du premier fichier (la tête de la liste).*

*Appliquez `ls` au premier “file header”. Cela vous permet d’afficher le contenu du dossier racine.*

*Note: n’oubliez pas que le format du fichier est big-endian!*

*Note: tous les offsets sont relatifs au début du file system.*



**Question 9.** *Écrivez une fonction `find` traversant récursivement le système de fichier à la recherche d’un fichier dont le nom est passé en paramètre. La fonction doit retourner le “file header” du premier fichier trouvé.*

*N’oubliez pas que les fichiers `.` et `..` sont spéciaux, pointant respectivement vers le répertoire courant et le répertoire parent.*

*Pour cela, vous aurez besoin de décoder les 3 derniers bits du champs “next filehdr”, dont la correspondance entre leur valeur et les types de fichiers possible est donnée ci-après.*



value	file type	spec.info means
0	hard link	link destination [file header]
1	directory	first file's header
2	regular file	unused, must be zero [MBZ]
3	symbolic link	unused, MBZ (file data is the link content)
4	block device	16/16 bits major/minor number
5	char device	16/16 bits major/minor number
6	socket	unused, MBZ
7	fifo	unused, MBZ

**Question 10.** *Cherchez le fichier “message.txt”, affichez son contenu, et venez donner la réponse à votre encadrant.*