

FAT File System

Progetto per il corso di Sistemi Operativi
2022

Alessandro Rocchi

Capitolo 1

Scopo e funzionalità

1.1 Scopo

Lo scopo del programma è fornire alcune tra le funzionalità offerte da un file system attraverso un'interfaccia a riga di comando.

I contenuti del file system sono salvati su un file `my_disk.img` che viene mappato in memoria ad ogni esecuzione del programma simulando un disco.

1.2 Istruzioni per l'esecuzione

Per compilare il programma è necessario spostarsi nella cartella `src` ed eseguire il comando `make`.

Per lanciare il programma usare il comando `./shell`.

Per formattare il disco eseguire il comando `make clean`.

1.3 Funzionalità

Una lista delle funzionalità può essere ottenuta digitando il comando `help` all'interno della shell.

Tramite la shell è possibile:

- creare ed eliminare file e directory;
- leggere e scrivere file;
- cambiare la posizione corrente in un file;
- visualizzare e cambiare la directory corrente;
- ottenere una lista dei contenuti della directory corrente.

Capitolo 2

Struttura

2.1 Disco

Il disco è diviso in tre sezioni:

- la prima sezione contiene delle variabili "globali" a tutto il disco;
- la seconda sezione contiene la struttura rappresentante la FAT (Fat);
- la terza sezione contiene il buffer dove vengono effettivamente memorizzati i dati (data).

```
typedef struct {  
    int size;           // size of the disk array  
    int cur_dir;        // pointer to current directory  
    int root_dir;       // pointer to the root directory  
    Fat fat;            // pointer to start of the FAT  
    char data[DISK_SIZE - 3*sizeof(int) - sizeof(Fat)];  
} Disk;
```

2.2 FAT

La struttura Fat contiene:

- una variabile che tiene il conto dei blocchi liberi nella FAT;
- un array contenente i blocchi della FAT.

```
typedef struct {  
    int free_blocks;    // number of free blocks  
    FatEntry array[FAT_BLOCKS_MAX]; // FAT  
} Fat;
```

Un blocco della FAT (FatEntry) contiene:

-
- l'indice al blocco del disco relativo;
 - l'indice al prossimo blocco della FAT;
 - un flag per indicare se il blocco è occupato o libero;
 - l'indice del blocco stesso.

```
typedef struct {  
    int file;           // index of the disk block  
    int data;           // index of next block  
    int busy;           // 0 if the block is free  
    int idx;            // index of this block  
} FatEntry;
```

2.3 File

Un file è costituito da un header (**FileHead**) e uno o più blocchi dati (**File**).

L'header contiene diverse informazioni sul file, come il nome, dimensione, posizione attuale, ecc...

```
typedef struct {  
    int idx;             // index of the disk block  
    char name[30];  
    int is_dir;          // 0  
    int size;  
    int pos;             // current position in the file  
    int parent_dir;      // directory that stores the file  
    int start;           // pointer to first block of file  
} FileHead;
```

Un blocco dati contiene:

- il suo indice nel disco;
- il numero di bytes liberi nel file;
- il buffer in cui memorizzare i dati.

```
typedef struct File {  
    int idx;  
    int free_in_block;  
    char data[BLOCK_SIZE - 2*sizeof(int)];  
} File;
```

2.4 Directory

Una directory è rappresentata da una struttura `Dir`, che contiene diverse informazioni sulla directory, tra cui un array di indici dei file e delle sottodirectory che contiene.

```
struct Dir {  
    int idx;                // index of the disk block  
    char name[30];  
    int is_dir;             // 1  
    int parent_dir;        // index of parent directory  
    int num_files;         // number of files  
    int num_dirs;          // number of subdirectories  
    int start;             // position in the FAT  
    int files[BLOCK_SIZE - 30*sizeof(char) - 5*sizeof(int)];  
    // list of files and subdirectories  
};
```