laSalle ENG
Universitat Ramon Llull

Operating Systems Advanced

Practice

Course2021-2022

Version 1.1
16/03/2022

Arnau Castellà (arnau.castella)

Index

Development of the practice

Requirements

The main requirement of this practice was to implement three different functionalities for both EXT2 and FAT16 filesystems. The functionalities are:

- **Info**: the program receives a filename, it checks the filesystem type and prints out some information corresponding to the filesystem.
- **Find**: the program receives the filename of the volume, and the filename of the file to search in said volume. If it exists, it needs to print the size of the file.
- **Delete**: similarly to the previous functionality, the program receives the filename of the volume and the filename of the file to search. If the file exists, it deletes it.

Design

The design of this practice was quite simple. I created a module for each filesystem and a file called "values.h" which contains the values of all the constants used. Since there were so many values, I decided to create this auxiliary file so the modules of each filesystem are clearer.

The functions declared in the EXT module are:

- int checkExt(int fd);
  Returns true if the given file's filesystem is EXT2.
- ExtInfo readExtFile(int fd);
  Reads the necessary information of the filesystem and returns a data structure containing it.
- char * getDateTime(time_t time);
  Transforms a unix timestamp to a string.
- void printExtInfo(ExtInfo extInfo);
  Prints the information of the EXT2 filesytem.
- InodeTable getInodeTable(int fd, ExtInfo extInfo, int num_inode);
  Given an inode number, reads and returns its information.
- int readDirEntry(int fd, DirEntry * dirEntry, int block_pos, int * size, int max_size);
  Reads one directory entry. Returns 1 if there are other entries to read.
- int recursiveSearch(int fd, char * filename, int inode, ExtInfo extInfo, int delete);
  Reads each directory entry to check its name. If the entry is a directory, the function calls itself again.
- void findFileExt(int fd, char * filename, ExtInfo extInfo);
  Calls the previous function and prints the final message.
- void deleteFileExt(int fd, char * filename, ExtInfo extInfo);
  Calls the recursiveSearch function and deletes the file if it is found.

The functions in the FAT module are:

- int checkFat(int fd)
  Checks if the given file is FAT16.

- FatInfo readFatFile(int fd)
  Reads the information of the filesystem.
- void printFatInfo(FatInfo fatInfo)
  Prints the filesystem information
- int isDirectory(int fd, unsigned int address)
  Checks if the file in the given address is a directory.
- int isFile(int fd, unsigned int address)
  Checks if the file in the given address is a file.
- char * cleanFilename(char * filename)
  Cleans empty spaces from the filename
- int findFromAddress(int fd, char * filename, FatInfo fatInfo, unsigned int address, int delete)
  Recursive function that looks for the file with the given filename. If the file is a directory it calls itself again.
- void findFileFat(int fd, char * filename, FatInfo fatInfo, int delete)
  Calls the previous function.
- void deleteFileFat(int fd, char * filename, FatInfo fatInfo)
  Searches the file with the given filename and deletes it if it is found.

Data structures

EXT2

```
typedef struct {
    unsigned short size; //2 bytes
    unsigned int num_inodes; //4 bytes
    unsigned int first_inode;
    unsigned int inodes_group;
    unsigned int free_inodes;
}InfoInode;

typedef struct {
    unsigned int size;
    unsigned int reserved_blocks;
    unsigned int free_blocks;
    unsigned int total_blocks;
    unsigned int first_block;
    unsigned int group_blocks;
}InfoBlock;

typedef struct {
    char name[16];
    unsigned int last_check;
    unsigned int last_mount;
    unsigned int last_write;
}InfoVolume;
```

```
typedef struct {
    InfoInode inode;
    InfoBlock block;
    InfoVolume volume;
}ExtInfo;
```

Data structures used to store the information of the EXT2 filesystem.

```
typedef struct {
    char nothing[4];
    unsigned int size;
    char nothing2[32];
    unsigned int blocks[15];
    char nothing3[28];
}InodeTable;
```

Data structure used to read an inode with a single read. The only information we are interested in is the size and the blocks.

```
typedef struct {
    int inode;
    short rec_len;
    char name_len;
    char file_type;
    char name[EXT_FILE_NAME_LEN];
}DirEntry;
```

Data structure used to read a directory entry.

FAT16

```
typedef struct {
    char name[8];
    unsigned short size;
    unsigned char sectors_per_cluster;
    unsigned short reserved_sectors;
    unsigned char fats_num;
    unsigned short max_root_entries;
    unsigned short sectors_per_fat;
    char label[11];
} FatInfo;
```

The only data structure used in the FAT module is this one, used to store the necessary information of the filesystem.

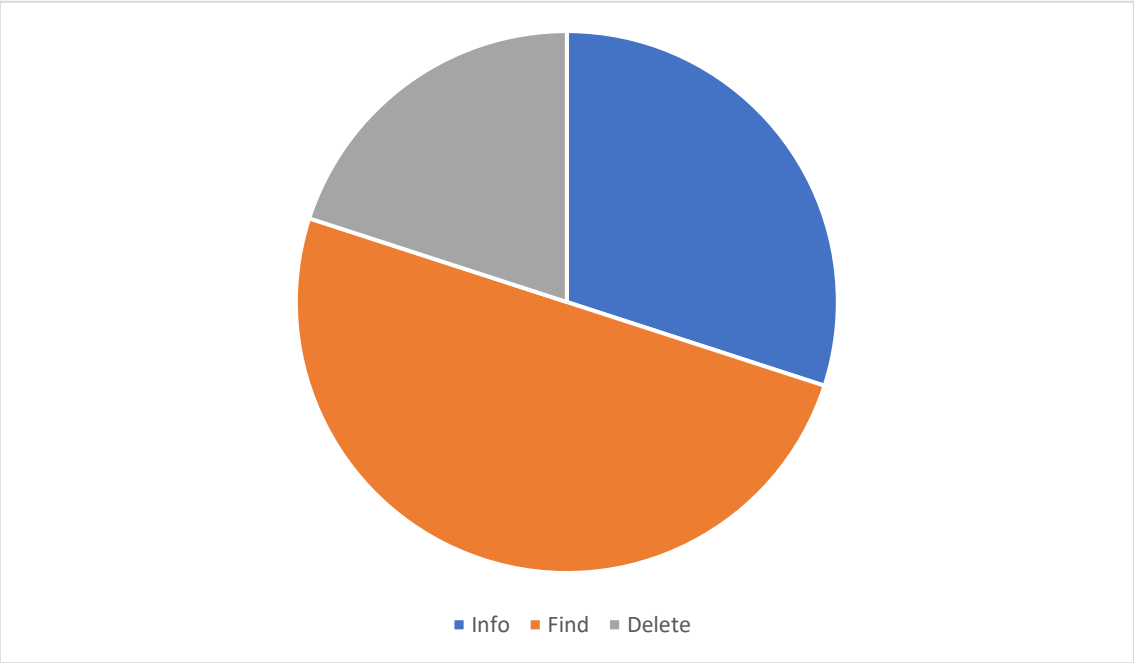
Tests performed

The testing of this practice has been done purely on execution. First of all, I executed the program while displaying the names of all files. Then, to test the find functionality, I tried to find one of those names. After successfully finding a file, I ran the delete functionality and tried to find the same file again. Since the program could no longer find said file, the delete and find functionalities have been successfully tested.

Observed problems

I did not encounter any specific problem during the development of this practice, however, it was a tedious job to find the necessary information in the filesystems documentation. There is a lot of information on the internet and in the documentation, and since the structure of both filesystems is so different, it made it a bit difficult sometimes.

Temporary estimation



Conclusions

After hours of research, development and testing, I can confidently say that I have gained a lot of knowledge about these filesystems.

It was very interesting to understand how files are managed and stored within an operating system, since I always thought it was an extremely complicated process that I could never understand.

Although there are many more filesystems out there, this practice has allowed me to have an idea of how they work, at least in general terms.

Rating

I found the methodology used for this practice to be very efficient. Since we had not seen any of the contents of this project in class, we had to do our own research using the available materials to find the information we were looking for. Although it might be a bit overwhelming at the beginning, you start learning at your own pace and only things that matter.