INF-2201: OS fundamentals
Project 6 File system
Lise Ekhorn Johansen
UiT id: ljo177@uit.no
Github username: Lise-johansen
GitHub classroom: project-6-Lise-johansen
31.05.2023

1. Introduction

This report describes the implementation and design of a simplified UNIX-like ext2 file system that resides on a USB stick. This simplified version has no notion of permissions for directory and file access, user management or anything “advanced”. Unit tests (not made by me) are provided for this project.

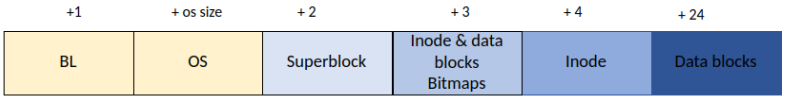
2. Technical background

Unix filesystem **ext2** is an extended filesystem based around inodes. [1] The **superblock** holds metadata about the file system (other metadata), such as the size, and a number of inodes and data blocks. [2] The **inode** is a structure that represents the blocks that contain data.[3][4] A **directory** is an inode but instead of holding information on data blocks, it holds all references to other files or directories. [5] The **data block** is what actually holds the file data.[6] A **bitmap** holds entries (0 or 1) that represent if an inode/data block is free or not. **Magic number** is a unique value residing in the superblock used to identify if a file system exists or not. A difference between the drive and run-time memory is made and represents how the data is stored on disk and how under run-time the system always retains a copy of the data.

3. Design

3.1 Layout of disk

The figure below shows the disk layout. The bitmaps are represented as arrays.



3.2 Design of Filesystem

The file system is composed of 3 main data structures, two structures that represent the superblock stored in disk and one in memory. And one structure represents the global inode table in memory (see figure 1). One root can contain other directors and files, see figure 2 for a representation of the file system.

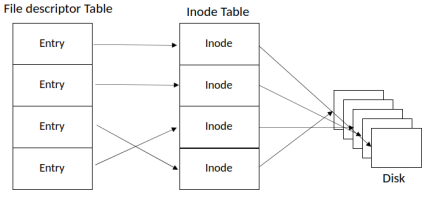


Figure 1: Layout in disk: Two first sectors holds the boot loader and the OS (size: OS size + 1), next are the superblock (position: OS +2). The next sector holds the bitmaps to the inodes and the data blocks (position: OS +3). Next are the inode (position: OS +4 * 20 inodes). One inode per block each inode are size 512 byte. Then lastly the data block (position: OS size + 24).

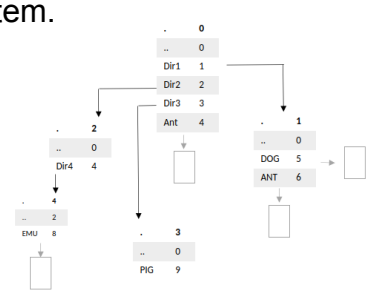


Figure 2: Overview of a multi-level file system (FS). Shows how the directories and file look like in the FS.

3.3 Initialize and Making the File System

Though is when the file system is initializing the magic number is checked to determine if it is needed to make a file system or just mount an existing file system.

3.4 System calls

3.4.1 Open and Close a File

Open a file is to make an entry in the file description table, the inode table and allocate a data block. When closing a file, appropriate values are incremented and set to its default values.

3.4.2 Making of Files, Directories and Links

How to make a file, link to a file and a directory is almost identical. A link is a mirror copy of the original file, same inode number and bata block. A directory only differs from a file with the additional entries of a parent entry (“.”) and a current entry (“..”).

3.4.3 Deleting of Files, Directories and Links

To delete files/links and directories: deallocate the data block, delete/ overwrite the table entries (inode and file descriptor), and decrement appropriate values (open count and links). A design chosen for files, when a file with links is deleted, the corresponding link is deleted instead of the file since only files with no links can be deleted.

4. Implementation

4.1 Initialize and Making the File System

Initializing the storage device and checking superblock for the magic number. If the file system does not exist, a new one is created. The inodes and data block bitmaps are connected to the storage device. A flag is set to indicate that the file system is unmodified. Creating a file system: sets superblocks default values, initializing all entries in bitmaps to zero. Set all inodes default values by iterating through it and writing it to disk. The root directory is created. The file descriptor table is initialized to its default values by iterating through all open files. Finally, the values in the superblock that exist in run-time memory are assigned to the superblock residing in the disk, mounting the file system.

4.2 System calls

3.4.1 Open and Close a File

If the file exists, check if the file is already open; if not assign the mode and inode, update the open count and position, and return the file descriptor. If the file does not exist, create a new file by finding the first free entry in the inode bitmap and assigning it to the file descriptor table, gets a free data block from the data block bitmap, sets values for the memory inode and disk inode, inserts the filename in the directory table, updates the size of the current working directory, updates the bitmaps, writes the disk inode to disk, and returns the file descriptor index.

3.4.2 Making Directories

The process of creating directories and files is similar. Only directions have a parent entry and a current entry that has to be inserted and the directory name has to be inserted in its parent directory table. Root directory is a normal directory only it has itself as its own parent.

3.4.3 Deleting of Files, Directories and Links

Deleting a directory: reading the inode and current working directory (cwd) up from disk, free inode entry and its bitmap entry, remove the directory entry for the parent directory, update size of cwd, update bitmap, and write down the update directory and inode to disk. Deleting a file: get the inode of the file using its name, check for links, read the inode up from disk, freeing the inode entry in the inode bitmap, removing the directory entry form the parent directory, update the bitmap, read up the cwd to update its size, write it down, and update the inode on disk.

4. Experiments and Result

Correctness is evaluated through 3 different tests:, 1. python unit test, 2. the unix shell simulator and 3. Not much focus on bosch as it is not mentioned as a requirement in the assignment information.

The Python unit test takes output and sees if it matches the expected output. “.” means passed and “F” means failure. The 22 original unit tests were made by Isak Kjerstad (Date: 24.05,22). Modifications and new unit tests were added in and clearly marked in code.

The Python unit test that was not passed all had to do with paths (absolute and relative) and the ability to read/write more than one data block, the reason for failure in 5.3. And failure to open the first directory in a multi-level directory.

```
Ran 25 tests in 72.124s

FAILED (failures=6)
```

Unix shell simulator and Bosch is also used to test shell system calls, here is the result: Pictures showcase an array of commands, not all but most.

```
/$ cat dog
cat>hello dog this is shell
cat>.
/$ ln link dog
/$ more dog
more> hello dog this is shell

/$ more link
more> hello dog this is shell

/$ stat dog
stat
filename: type, refs, size
dog: 1 1 24
/$ stat link
stat
filename: type, refs, size
link: 1 1 24
/$ ls
        . 0
        .. 0
        dog 1
        link 1
/$
```

```
/dir1/dir2/dir3$ ls
        . 3
        .. 2
        emu 4
/dir1/dir2/dir3$ cd ..
dirs:
        ..
/dir1/dir2$ cd /
/$ cd dir1
dirs:
        dir1
/dir1$ ls
        . 1
        .. 0
        dir2 2
/dir1$
```

Picture 1: Result of running the command: cat, link, more, stat and ls and making a multi-level directory system and demonstrating the ability to jump to / (root) and navigating path in Linux simulated shell

```
$ ls
. 0
.. 0
dog 1
link 1
$
```

```
$ rm link
$ ls
. 0
.. 0
dog 1
$
```

Picture 2: In Bosch can make a file and a link that share the same inode number and can delete the link without deleting the file.

5. Discussion

5.1 Implementation

To guarantee that it is the news update data is worked with, a read from disk is always performed before any operations. Additionally when disk value is modified, it is written back to the disk. Not the time effective way of doing it but it is the safest.

Each inode takes one block in memory, not space friendly but makes it easy to work with i.e. time complexity is better. Since the file system is small, i.e. not many inodes, this trade off is worth it. Something inconsistent is that the both bitmaps have a size of 256 byte, even though max inodes is 20 and max data blocks is 160.

5.2 Correctness

A multi-level file system was implemented, though it does not have a working absolute path. Despite this limitation, the making of files and multi-level directory parts or the files system works. Statement is supported by the python unit test [7].

5.2 Know errors

Deallocation of data blocks in the bitmap is not fully done. The program deallocates the block number representing the data block location on disk and not the data block number representing the entry in the data block bitmap. To address this, a restructuring of the data block bitmap design is necessary. However, since the file systems (since it is a small file system and only have to support a minimum of shell command) function without fixing the issue, it will just go out if data blocks after enough data is written to disk, the error was demanded as non-critical and left unfixed. For it to be possible to read/write over a single data block the program needs a dynamic method to obtain data blocks in the directory array instead of using fixed numbers. Last error prevents the completion of the relative and absolute path. When reading parent form up from disk it is always zero regardless of level in the file system. In addition, the system can only call “ls” on a directory if it contains another file or directory. Regrettably, it is not known what causes this error.

6. Conclusion

A functional file system was implemented, despite a few errors concerning the multi-level directory hierarchy of the file system the needed behavior was achieved as shown under testing.

Bibliografi

[1] (wiki.os.dev.com), “Ext2”, Last updated: 23.06.2022, Read: 04.04.2023
URL: Ext2 - OSDev Wiki
[2] (techopedia.com), “Suberblock”, Rouse. Margaret, Last updated:03.10.2018, Read: 04.04.2023
URL: https://www.techopedia.com/definition/13376/superblock
[3] (bluemataddor) “What is an inode and what are they used for”, Matador. Blue, Last update: 29.02.2020, Read: 04.04.2023
URL: https://www.bluematador.com/blog/what-is-an-inode-and-what-are-they-used-for
[4] (wikipedia.com), “inode” Last updated: 24.05.2023, Read: 04.05.2023
URL: https://en.wikipedia.org/wiki/Inode
[5] IBM.com), Последна актуализация, “Directories”, Last updated: 24.03.2023, Read: 05.04.2023
URL: https://www.ibm.com/docs/bg/aix/7.2?topic=systems-directories
[6] (oracle.com), “Data Blocks” Last updated: 2010, Read: 03.04.2023
URL: https://docs.oracle.com/cd/E19455-01/805-7228/fstroublefsck-13/index.html
[7] Experiment (4): Python unit test made by Isak Kjerstad, Date: 24.05.22, Published: 05.05.2023 9:18 PM