An Introductory Glance at the Unix File System

CM10195 - Systems Architecture II
Coursework 1
University of Bath

March 21, 2019

Introduction

In the early days, filing systems referred to the literal storage of paper in an ordered way. A good filing system would enable files to be placed and retrieved in a logical manner (McGill 1922). However, an alternative meaning for the word came into general use when the need for digital based filing systems were invented (*Disc File Applications* 1964). A "File System" is now more commonly refereed to the digital storage of files. More specifically, a system that is capable of creating, manipulating, storing and retrieving data in an arranged way. This would entail creating a system which keeps track of each individual blocks written to the volume or partition, so that when data needs to be retrieved, the file system knows exactly which block it lies (Giampaolo 1999).

Structure of Storage Volumes

There are many ways in which digital data can be physically stored, such as Floppy Disks, Tapes, Solid-State drives, researchers are even developing ways to store data in strands of DNA, yielding a higher storage density (Goldman et al. 2013). These devices are often considered to be 'linear spaces', because the reference to each individual block that can be addressed increases by one (*Understanding File Systems* 2018). So in practice data can be thought a 1-dimensional array of blocks, each block has a fixed number of bytes and is the smallest addressable unit of memory, this is only true in the context of Files Systems.
The use of terms 'block' and 'sector' are often confused because their definitions rely on the context they are used in. For example, Figure 1 shows the structure of a hard drive. The drive is broken up into multiple tracks (Label A), these tracks are divided into multiple geometric sectors. The intersection of each geometric sector (Label B) and its track defines a Track Sector (Label C) (*Disk Sector* 2019). From a hardware view, this is the smallest addressable element of the disk. However, a file system can artificially modify the block size of a sector by clustering two together. Therefore addressing both sectors via the same address, this is called a block. Therefore, in the file systems view, a block is the smallest addressable element. Similarly with the term block, in a file systems view, a block is simply an abstraction over multiple disk sectors, where as in a purely hardware context it is referred to as a 'chunk of data' (Sectors Vs Blocks 2018).

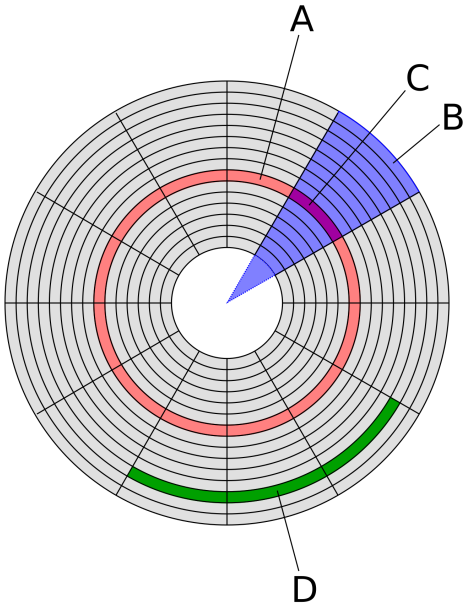

Figure 1 (Disk Diagram 2008)

UNIX File System

We now need a way of organising these blocks, hence why file systems were invented. One way of achieving this is the Unix Files system (UFS), first released in 1983 (RobboNuggie 2018) and first adopted by Berkley Software Distribution (BSD 4.2). It grew in popularity due to its advanced features such as Soft Updates, Snapshots and its ability to support fragments. Fragments allow abstract blocks to be broken down into smaller addressable spaces, therefore occupying less space and causing less fragmentation (RobboNuggie (2018), Kinicki (2006)).
The UNIX file system can be classified as an hierarchal structure, a tree like organization which links files together. Each directory can contain sub-directories, and the those sub-directories can contain further directories and so on. It is common practice to place directories of most importance in the root of this tree, allowing for quicker access (Liu et al. 2011).
The UNIX file system can occupy one of many partitions on a particular disk. In OSX implementation, a disk is referred to by the switching table by its 'major' number, and the partition of the disk is called the 'minor' number (Kinicki 2006). See figure 2, the OSX operating system is made up of many partitions that are referenced individually. For example, the 'Preboot' partition would be referenced via the path: "/dev/disk1s2". Where one is the major volume, and two is the minor partition. Although implementations do differ among operating systems (Peek et al. 1998).

```
/dev/disk1 (synthesized):
   #:                       TYPE NAME                    SIZE       IDENTIFIER
   0:      APFS Container Scheme -                      +1.0 TB     disk1
                                Physical Store disk0s2
   1:                APFS Volume Macintosh HD            508.0 GB   disk1s1
   2:                APFS Volume Preboot                 45.1 MB    disk1s2
   3:                APFS Volume Recovery                512.4 MB   disk1s3
   4:                APFS Volume VM                      2.1 GB     disk1s4
```

Figure 2 (Command: diskutil list)

Boot Block and Super Block

For a file system to operate, the root directory of the file system must be loaded. In Figure 3 you will see a diagram of the structure of a UFS file system. The Boot block is reserved for the boot program, it is responsible loading and mounting the root directory. The boot block is executed by the Firmware boot, which mounts the Boot directory and starts executing code inside it. Note, that the Firmware boot is "file system independent", meaning that it has know knowledge of the actual file system (Frank Batschulat & Rochford 2006).
The super block stores information about the file-system and is vital to its operation. It stores attributes such as: Size of the File System, Volume Name, Last Date and time of write, Group Size, Number of block in each group among other attributes. This informations is so important, it is copied across multiple sections of the storage medium to protect it against data loss ((2012),Frank Batschulat & Rochford (2006)).

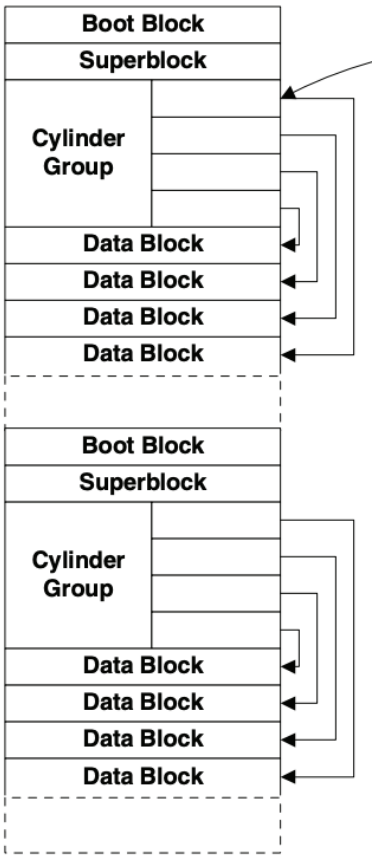

Figure 3 (Frank Batschulat & Rochford 2006)

Files and Directories (INodes) + Meta Data

Files and file directories are structured using INodes. Each INode contains information about a file or directory. Each file appears as a stream of bytes to the interface reading the file, the INode points to the data bytes of that file in the stream.
Each INode contains meta data, this is often refereed to as 'data about data', it is stored in the top section of each INode. It provides useful information such as the name of the file, its owners,

statistics, read-write permission etc (Organization 2001). For File system operation, it also stores information such as create and modify time, the size of the file, and the owner associated with the file. (Giampaolo 1999).
There are two different types of INodes in the UFS (Frank Batschulat & Rochford 2006):

- **In-Core:** An INode create when a file is opened for reading and writing.
- **On-Disk:** This is the normal link to a file which resides on the disk itself.

Each specific INode has a node number which is used for mapping individual nodes to another, aswell as a list of 'pointers' which point to addresses in memory containing files (plus length of file). This works well for small files. However, an INode list is of fixed length, therefore files that require a larger number of pointers need more space. In-direct pointers are used for pointing to other lists of pointers, therefore increasing the amount of possible locations for that file (Giampaolo (1999), Frank Batschulat & Rochford (2006)) (See Figure 4). However, a greater number of in-direct pointers increase read-time. For example, in the case of a Disk Drive, the head would have to hop to an increased amount of pointers. One solution is the de-fragmentation of the disk. This process attempts to put all of the blocks in the location, therefore decreasing the amount of references in the INode table (2017).

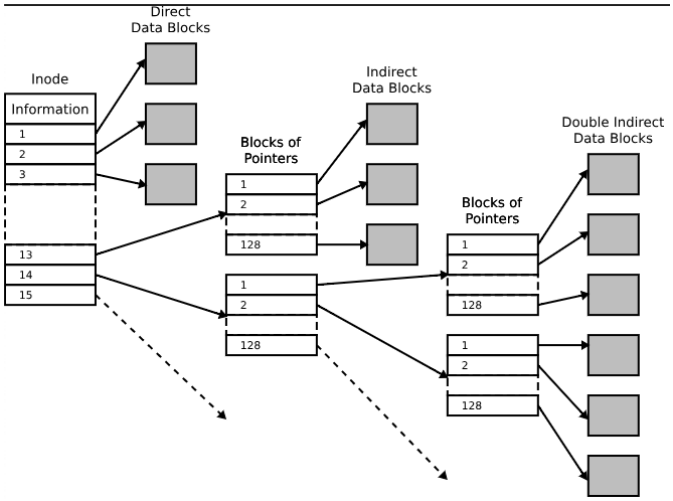

Figure 4 (WikiMedia 2019)

Protection Mechanisms

In order to minimise risk to files, UNIX forces permission for files to be set. These permissions entail: write access (W) allows modify or deletion, Read (R) access allows contents to be viewed and execute access (X) allows the user to run the program. Each flag has permitted or denied option, therefore giving 9 possible combinations. (Dermeir (2018),Linfo (2006)) Permissions for a particular file or INode can reference one specific user, or a group of users. However, the root user always has read, write and Execute privileges (Linfo 2006). The main idea of this system is to give each user the least privilege possible, decreasing access. For example, a typical user without escalated privileges would have no need to modify root folder of the OS, so therefore, a typical user would not be given write access to the root folder.(Dermeir 2018).

UFS Special Features

The first version of UFS allowed snapshots. This entailed recording the state of a drive at a particular time, therefore allowing the user to revert back to previous versions. This proved to be a popular feature by consumers (RobboNuggie 2018). However, snapshots do impose limitations,impacting performance and read/ write times due constant copying (Oracle 2017).
UFS allows for 'Soft Updates'. This means in the event of a system failure data is not lost (RobboNuggie 2018). This is done by only allowing a-synchronous data writes.

Conclusion

Finally, the Unix File System is special in many ways. It unique implementation of block fragmentation allowed for less wasted space, therefore decreasing read and write time due to a smaller number of INode hops being required to access a file. Furthermore, it's ability to protect data is exceptional, adding options for backups and forcing the use of file protections.

References

(2012).
URL: *https://docs.oracle.com/cd/E19455-01/805-7228/6j6q7uf0r/index.html*

(2017).
URL: *https://www.youtube.com/watch?v=KN8YgJnShPM*

(Disk Diagram 2008).
URL: *https://commons.wikimedia.org/wiki/File:Disk-structure2.svg*

(Sectors Vs Blocks 2018).
URL: *https://unix.stackexchange.com/questions/14409/difference-between-block-size-and-cluster-size/1441114411*

Dermeir, T. V. (2018).
URL: *https://www.youtube.com/watch?v=CbFms89xx3Y*

Disc File Applications (1964), Technical report, American Data Processing. Accessed: 2019-03-20.

Disk Sector (2019).
URL: *https://en.wikipedia.org/wiki/Disk*$_s$*ector*

Frank Batschulat, Shawn Debnath, S. J. D. M. & Rochford, K. (2006), *The UFS File System*, pearsoncmg.
URL: *http://ptgmedia.pearsoncmg.com/images/0131482092/samplechapter/mcdougall*$_c$*h*15*.pdf*

Giampaolo, D. (1999), *Practical File System Design with the Be File System*, Morgan Kaufmann Publishers, Inc. Accessed: 2019-03-20.

Goldman, N., Bertone, P., Chen, S., Dessimoz, C., LeProust, E. M., Sipos, B. & Birney, E. (2013), 'Towards practical, high-capacity, low-maintenance information storage in synthesized dna', *Nature* **494**, 77 EP –.
URL: *https://doi.org/10.1038/nature11875*

Kinicki, B. (2006), 'Unix filesystem organization'.
URL: *https://web.cs.wpi.edu/ rek/DCS/D04/UnixFileSystems.html*

Linfo (2006), Permissions definition, Technical report, Linux Information Project.
URL: *http://linfo.org/permissions.html*

Liu, Y., Yue, Y. & Guo, L. (2011), *UNIX File System*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 149–185.
URL: *https://doi.org/10.1007/978-3-642-20432-6*$_6$

McGill, F. E. (1922), *Office Practice and Business Procedure*, Gregg Publishing Company. Accessed: 2019-03-20.

Oracle (2017), 'Ufs snapshots overview'.
URL: *https://docs.oracle.com/cd/E23823*$_0$*1/html/817 − 5093/bkupsnapshot − 11.html*

Organization, N. N. I. S. (2001), 'Understanding metadata'.
URL: *Archived at: https://web.archive.org/web/20141107022958/http://www.niso.org/publications/press/*

Peek, J., O'Reilly, T. & Loukides, M. (1998), 'Unix power tools'.

RobboNuggie, C. (2018), 'For the love of ufs'.
URL: *https://www.youtube.com/watch?v=oqbc7ICylcw*

Understanding File Systems (2018). `https://www.ufsexplorer.com/articles/file-systems-basics.php` Accessed: 2019-03-20.

WikiMedia (2019).
URL: *https://upload.wikimedia.org/wikipedia/commons/0/09/Ext2-inode.svg*