File System Implementation

Lalintha Polpitiya | lalintha@umiami.edu

1 Getting Started

1.1 Block

A block is the fundamental unit of storage on a disc. It is a contiguous array of 512 bytes that are usable for data storage.

{Surface number, Track number, Sector number} $\Rightarrow$ Block number.

1 block = 512 bytes (2^9 bytes).

Typical disk size = 1 TB (2^{40} bytes), needs 2^{31} blocks.

1.2 Cluster

A cluster consists of a number of contiguous blocks. The number is selected when the disc is formatted.

When the size of a cluster increases:

- Advantage: With clusters, cluster numbers are used instead of block number. As there are fewer clusters, fewer bits will be required to represent them.
- Disadvantage: For a random-sized file, the expected amount of wasted space is half of the allocation unit, so increased cluster size means increased wasted space.

1.3 Access methods

Files store information. When it is used, this information must be accessed and read into computer memory.

- Sequential access: The simplest access method is sequential access. Information in the file is processed in order, one record after the other.

- Direct access: Allow programs to read and write records rapidly in no particular order. There are no restrictions on the order of reading or writing for a direct-access.

1.4 File system permissions (Unix)

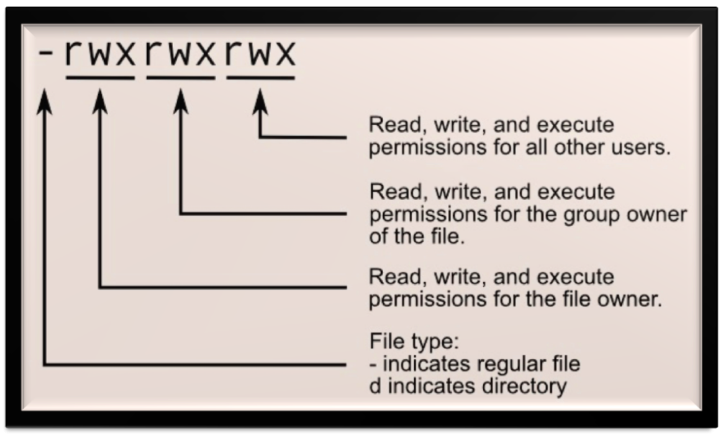

Figure 1: File system permissions (Unix)

1.5 Cyclic redundancy check (CRC)

CRC are error detecting codes stored in conjunction with disc blocks. As data is written to the disc, a computation is performed on all the bytes to produce a single number. This number is written at the end of the block. When data is read back, the same computation is performed and the result is compared with the stored value to check the integrity of the file system and data.

2 File Allocation Methods

2.1 Contiguous allocation

- Advantages: Simplicity and speed of access.

- Disadvantages: Condensing is slow and external fragmentation.

Consider a particular disc drive has a speed of 10800 RPM (revolutions per minute) and a seek time of 3 mS.

10800 RPM = 180 RPsec = 6 mS.
Expected rotational latency for a random block is half a revolution, 3ms.
Then the average random access time = seek + latency = 6 mS.
Time required to move a block = read + write = 12 mS.
Then the time required to more 2^{31} blocks = $2 * 10^9 * 12 * 10^{-3}$
$= 24 * 10^6$ S $\approx$ 10 months ($3 * 10^7$ S $\approx$ 1 year)

2.2 One level index format

First block of any file is the Header Block.

Consider a cluster which contains one block.
112 bytes are for meta data.
400 bytes are for an array of 100 block numbers (pointers).
Maximum file size = 51,200 bytes.
Opening a file ⇒ read header block and keep copy in memory.
Sequential read ⇒ one disc read per 512 bytes.
Random access ⇒ one disc read per access.

Consider a cluster which contains 4 blocks.
112 bytes are for meta data.
1936 bytes are for an array of 484 block numbers (pointers).
Maximum file size = 991,232 bytes.

2.3 Two level index format

100 * 128 * 512 bytes = 6,553,600 bytes.

Random access ⇒ 2 disc reads per access.
Sequential read ⇒ 3 reads to open a file, then one disc read per 512 bytes.
500 * 512 * 2048 = 500 MB. (Cluster - 4 blocks)

2.4 Three level index format

100 * 128 * 128 * 512 bytes = 800 MB.

4 level index ⇒ 100 GB max.

3 A Guide to UNIX File System Implementation

3.1 Block allocation

Block allocation of the file system looks like below.

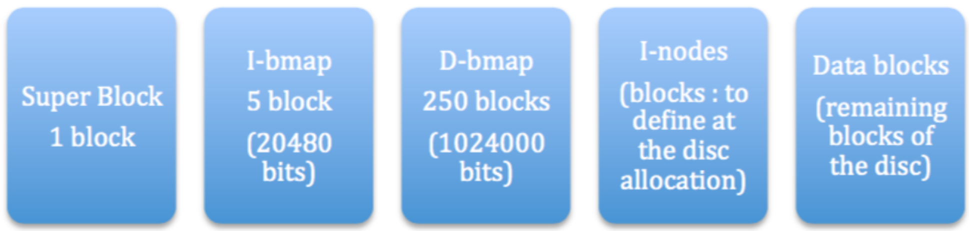

Figure 2: Block allocation of the file system

3.2 Super Block (1 block, 512 bytes)

Some information to include in the Super Block:

disc number
disc name (max 32 bytes)
disc name size
total blocks
created time
last accessed time
last modified time
I-bmap blocks
start I-bmap blocks
D-bmap blocks
start D-bmap blocks
I-nodes count
start I-nodes
Data blocks count
start data blocks
size root directory
root directory start

3.3 I-bmap (5 blocks, 20480 bits)

I-bmap is used to keep track of the free/used state of I-nodes.

This can be used to represent a maximum of 20480 I-nodes.
User has to define the I-nodes count at the disk allocation.
Free bitmap numbers use a stack structure to keep the efficiency of I-node allocations.

3.4 D-bmap (250 blocks, 1024000 bits)

D-bmap is to keep track of the free/used state of data blocks.

This can be used to represent a maximum of 1024000 data blocks.
Free bitmap numbers use a stack structure to keep the efficiency of data block allocations.

3.5 I-node representation with data blocks

User has to define the block count at the disk allocation.

Size of an I-node is 512 bytes.
Meta-data: 128 bytes.
100 pointers: 4 bytes for each.

Meta data fields look like:

file or folder
what level
free/used
owner of the I-node
I-node name
I-node name size (20 bytes)
byte length
total blocks
created time
last accessed time
last modified time
deleted time

3.6 I-node structure

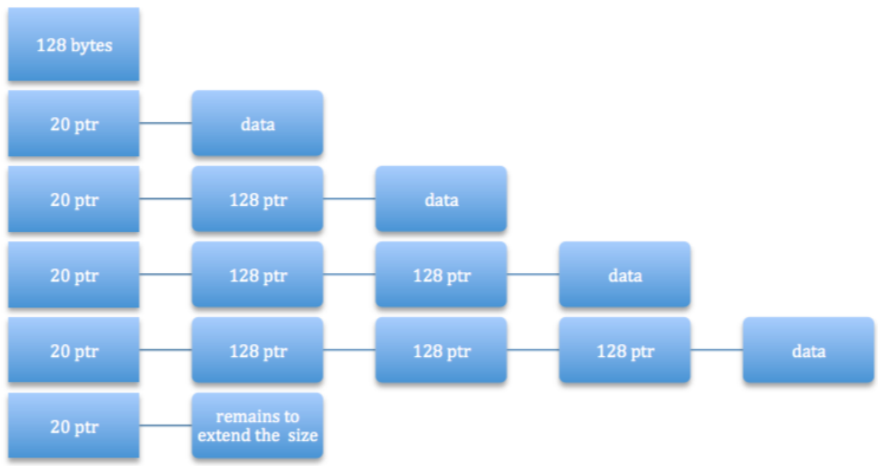

Figure 3: I-node structure

3.7 Maximum file/folder count and maximum file size

Allocated block count for the I-bmap segment is 5 blocks (10480 bits). Therefore the achievable maximum file and folder count is 10480.

Maximum file size according to D-bmap segment bit count.
250 blocks were allocated to D-bmap segment (1024000 bit)
It can be used to represent 1024000 blocks.
Maximum allowable file size $\approx$ 1024000 * 512 Bytes = 500MB.

Maximum file size according to I-node structure.
Direct (level 1) = 20 * 512 bytes.
Indirect 1(level 2) = 20 * 128 * 512 bytes.
Indirect 2(level 3) = 20 * 128 * 128 * 512 bytes.
Indirect 3(level 4) = 20 * 128 * 128 * 128 * 512 bytes.
Total = approximately 20 GB

According to D-bmap allocation, the system supports only 500 MB. The allowable file size can be increased by increasing the defined D-bmap blocks.

References

[1] Silberschatz, Abraham and Galvin, Peter Baer and Gagne, Greg. *Operating System Concepts*. Wiley Publishing, 2008.

[2] EEN521 T, Operating Systems, Spring 2014 - University of Miami, `http://rabbit.eng.miami.edu/class/een521/142index.html`

[3] EEN521 T, Operating Systems, Spring 2018 - University of Miami, `http://rabbit.eng.miami.edu/class/een521/182index.html`