Preview

- File System
  - File Name, File Structure, File Types, File Access, File Attributes, File Operation
- Directories
  - Directory Operations
- File System Layout
- Implementing File
  - Contiguous Allocation
  - Linked List Allocation
  - Linked List Allocation with File Allocation Table
  - Index-Node
- Implementing Directories
- Linux File Structure
- File Implementation in Linux

File System

Three essential requirements for long term information storage

1. To store a very large amount of information
2. To store information permanently
3. To share the information with multiple processes

File Name

<file_name>.<extension>

length: 255

UNIX: case sensitive

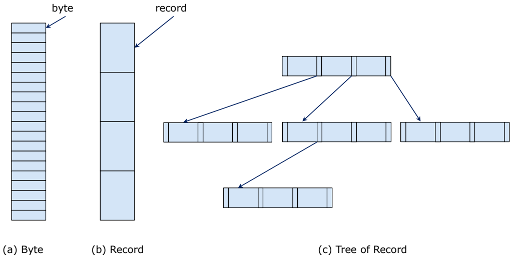
Dos/Window: not case sensitive

File Structure

Files can be structured

- File is an un-structured **sequence of bytes** – OS does not know what is in the file (UNIX, Window, Linux)
- File is **sequence of records** – used until 2nd Generation main frame computer – 80 column punched card – files consist of 80 character record.
- File consists of a **tree of records**

File Structure



(a) Byte (b) Record (c) Tree of Record

File Types

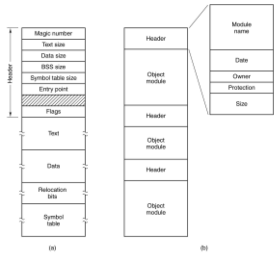
- **Regular Files** – for user's information
  - ASCII file – line of text. a line is terminated by carriage return or special character
  - Binary file
    - executable file - OS can execute a file if it has the proper format
    - archive file – consist of collection of library functions compiled but not linked (static & shared library), or data
- **Directories** – System files for maintaining the structure of the file system

Binary Files

Binary File

- Executable File
  - Header – all information regarding execution
    - Magic number – identifying the file as an executable
    - Size of files, the address where execution start
  - Text
  - Data
- Archive File – a collection of library modules compiled but not linked, or data file

File Type (Binary Files)



File Attributes

Attribute	Meaning
Protection	Who can access the file and in what way
Password	Password needed to access the file
Creator	ID of the person who created the file
Owner	Current owner
Read-only flag	0 for read/write; 1 for read only
Hidden flag	0 for normal; 1 for do not display in listings
System flag	0 for normal files; 1 for system file
Archive flag	0 for has been backed up; 1 for needs to be backed up
ASCII/binary flag	0 for ASCII file; 1 for binary file
Random access flag	0 for sequential access only; 1 for random access
Temporary flag	0 for normal; 1 for delete file on process exit
Lock flags	0 for unlocked; nonzero for locked
Record length	Number of bytes in a record
Key position	Offset of the key within each record
Key length	Number of bytes in the key field
Creation time	Date and time the file was created
Time of last access	Date and time the file was last accessed
Time of last change	Date and time the file has last changed
Current size	Number of bytes in the file
Maximum size	Number of bytes the file may grow to

File Operations

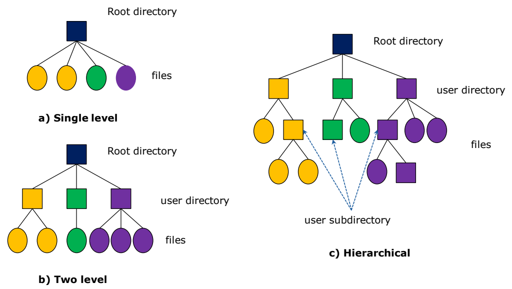
- Create
- Delete
- Open
- Close
- Read
- Write
- Append
- Seek
- Get attributes
- Set attributes
- Rename

Directories

To keep track of files, file systems have directories which themselves are files.

- Single-level directory System – one directory containing all files
- Two-level directory System – Giving each user a private directory
- Hierarchical directory System – User can create a directory to group their files in logical way

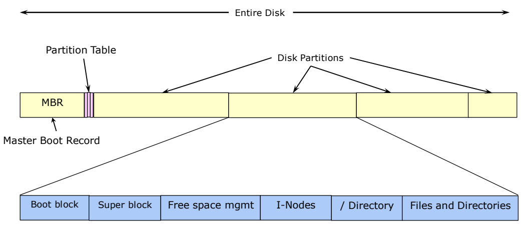
Directories



Directory Operations

- create
- delete
- open directory
- close directory
- read directory
- rename directory
- link directory

File System Layout



File System Layout

- When the Computer is Booted
  - The BIOS reads in and execute MBR.
  - MBR (Master Boot Record) locate the active partition
    - Read in the first block (boot block) and execute it
    - The program in the boot block loads the OS contain in that partition (active partition)

File System Layout

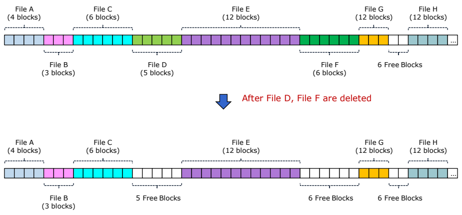
- The System Layout of a disk partition varies strongly from file system to file system.
  - Supper block – key parameters about the file system:
    1. magic number to identify the file system type,
    2. the number of blocks, size of a block…
  - Free space management
  - I –Node
  - Root Directory (/)
  - Directories and Files

Implementing File

- Disk spaces are divided into blocks and one or more block is used to save a file.
- Implementing a File – How a file is saved in the disk.
  - Contiguous Allocation- a file is saved in contiguous blocks.
  - Linked-List Allocation- a block is used for saving data and next block information (block number)
  - Linked-List Allocation with File Allocation Table – block information is saved in FAT.
  - Index-Node Allocation – block information is saved in I-node

Implementing File
(Contiguous Allocation)

Contiguous Allocation- a file is saved in contiguous blocks.



Implementing File
(Contiguous Allocation)

Two significant advantages with contiguous file allocation

- **Simple implementation** because keeping track of where a file's blocks are is reduced to remembering two numbers: disk address of the first block and the number of blocks in the file.
- **The read performance is excellent** – there is only one seek is needed to read entire file.

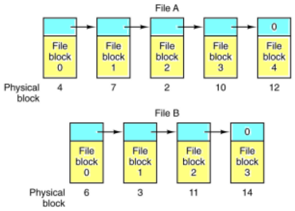
Implementing File
(Contiguous Allocation)

Drawback with the contiguous allocation

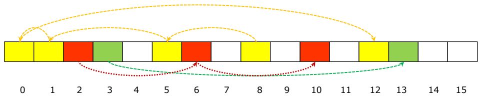
- Fragmentation
- If a newly created file size is fixed, system just finds out a big hole fit to the file. But if a file size is not fixed, there will be problem

Implementing File
(Linked List Allocation)

Linked-List Allocation- a block is used for saving data and next block information (block number)



Implementing File
(Linked List Allocation)



File name	First Block #
Yellow	8
Red	2
Green	3

Directory for files

Implementing File
(Linked List Allocation)

Disadvantage

- Reading a file sequentially is straightforward, but random access is extremely slow!!
- Since storage space per a block is not power of 2 anymore, need effort for reading and write in the block!!

Implementing File
(Linked List Allocation with File Allocation Table)

- All link information for each of file is written into the File Allocation Table (FAT) which is located in main memory.

Primary disadvantage

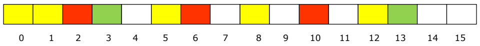
- Since entire table must be in the memory all the time

Ex)

20 GB Hard Drive with 1KB block size, need over 20 million (20×2^{30} / 1×2^{10} = 20×2^{20} = 20,971,520 blocks) entries for the FAT

if one entry take 3 bytes, we need 60MB for the FAT.

Implementing File
(Linked List Allocation with File Allocation Table)



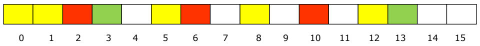
File name	First Block #
Yellow	8, 5, 1, 0
Red	2, 6, 10
Green	3, 13
…	

File Allocation Table

Implementing File
(Index-Node)

- Index-Node lists the attributes and disk addresses of the file's block
- To open a file, a system only need load corresponding i-node into the memory!!
- One problem with i-nodes is that if each one has room for a fixed number of disk addresses, what if when a file grows beyond this limit? - Solution: Reserve the last disk address not for block for the address of block containing more block addresses.

Implementing File
(Index-Node)



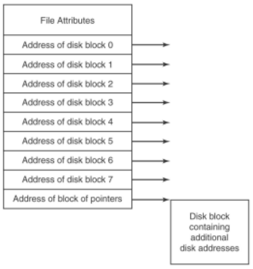
yellow
8
5
1
0

red
2
6
10

green
3
13

i-nodes for each file

Implementing File
(Index-Node)


File Attributes

Address of disk block 0

Address of disk block 1

Address of disk block 2

Address of disk block 3

Address of disk block 4

Address of disk block 5

Address of disk block 6

Address of disk block 7

Address of block of pointers

Disk block containing additional disk addresses

Implementing Directories

- The directory entry provide the information needed to find the file disk blocks. The directory entries are depends on the system
  - Contiguous allocation: Starting block number, number of block used.
  - Linked list allocation: the first block used for each file
  - Index-Node : i-node number

Implementing Directories

Where attributes for each file should be saved?

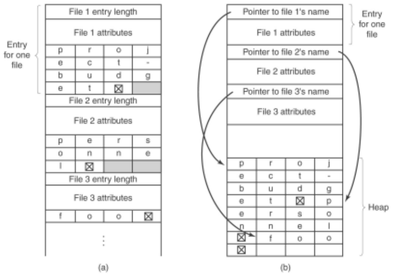
- MS-DOS – stored in a directory entry
- UNIX – stored in the i-node entry

Implementing Directories

How to handle long file name?

1. Set a limit on file name and stored in a directory entry or in the i-node entry – simple, but wasting space
2. All directory entries are start with length of the entry followed by data with fixed format – when a file is deleted, and created, there is fragmentation!!!
3. Make the directory entries all fixed length and keep the files names together in a heap at the end of the directory

Implementing Directories



Linux File Structure

- In Linux, almost everything is represented as a file, program can use disk files, serial ports, printers and other devices in the same way they would use a file.
- Mainly we need to user five basic function to use, **open**, **close**, **read**, **write** and **ioctl**.
- Directories are also special kind of file where file management information need to be saved.

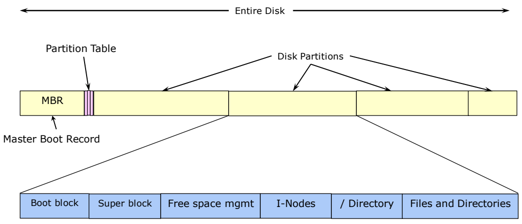
File Implementation in the Linux

- Disk space are divided into a small size of space called disk block.
- A block size varies in different system. Numbers of blocks are used for saving a file.
- To manage a file system, information which blocks are used for a file must be saved for file management.

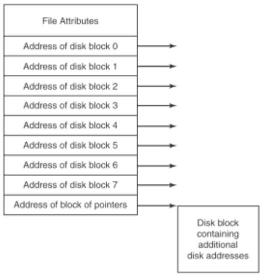
File Implementation in the Linux

- Linux system use I-Node file implementation method.
- Each file combined with an I-node. Information for a file is saved in its I-node: such as attributes, block numbers used by the file

File Implementation in the Linux



File Implementation in the Linux



File Attributes

Address of disk block 0

Address of disk block 1

Address of disk block 2

Address of disk block 3

Address of disk block 4

Address of disk block 5

Address of disk block 6

Address of disk block 7

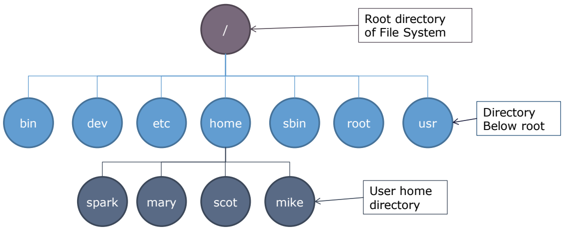
Address of block of pointers

Disk block containing additional disk addresses

Directory

- Linux use **Hierarchical directory System**
- A Linux directory is a special file that acts as a container for other files and even other directories.
- The directory entry provide the information needed to find the file disk blocks or subdirectory

Linux Hierarchical Directory



/

Root directory of File System

bin dev etc home sbin root usr

Directory Below root

spark mary scot mike

User home directory

Linux Hierarchical Directory

- **/bin** Contains the Linux system commands and programs (also called binaries).
- **/dev** Contains special device files that correspond to hardware components.
- **/etc** Contains configuration files for Linux and other installed software
- **/home** Contains the home directories (personal storage) for each user on the system.

Linux Hierarchical Directory

- **/sbin** Contains more Linux binaries (special utilities not for general users).
- **/root** The home directory for the root user; not to be confused with /. Some Linux systems use **/home/root** instead of **/root**.
- **/usr** Contains system programs and other files for general users such as games, online help, and documentation. By convention, a user should not put personal files in this directory.