CSIT 231: Systems Programming

Michelle Zhu

Computer Science Department
Montclair State University

- Slides revised from CS 252 Systems Programming

- Gustavo Rodriguez-Rivera
    - Computer Science Department
        - Purdue University

The UNIX Operating System

What is an Operating System

- An Operating System (OS) is a program that sits in between the hardware and the user programs.
- It provides:
  - Multitasking - Multiple processes running in the same
    computer
  - Multiuser - Multiple users using the same computer
  - File system – Storage
  - Networking – Access to the network and internet

What is an Operating System

- Window System – Graphical use interface
- Standard Programs – Programs such as a web browser, task manager, editors, compilers etc.
- Common Libraries – Libraries common to all programs running in the computer such as math library, string library, window library, c library etc.
- It has to do all of the above in a secure and reliable manner.

A Tour of UNIX

- We will start by describing the UNIX operating system (OS).
- Understanding one instance of an Operating System will help us understand other OSs such as Windows, Mac OS, Linux etc.
- UNIX is an operating system created in 1969 by Ken Thompson, Dennis Ritchie, Brian Kernighan, and others at AT&T Bell Labs.
- UNIX was a successor of another OS called MULTICS that was more innovative but it had many problems.
- UNIX was smaller, faster, and more reliable than MULTICS.

A Tour of UNIX

- UNIX was initially created to support typesetting (edition of documents).
- By having the programmers being the users themselves of the OS. UNIX became the robust, practical system that we know today.
- UNIX was written in "C" (95%) and assembly language (5%).
- This allowed UNIX to be ported to other machines besides Digital Equipment (DEC)'s PDP11.

BSD UNIX

- UNIX was a success in the universities.
- Universities wanted to modify the UNIX sources for experimentation. Berkeley created its own version of UNIX called BSD- UNIX.
- POSIX (The Portable Operating System Interface) is a family of standards specified by the IEEE Computer Society for maintaining compatibility between operating systems
- Sockets, FTP (File Transfer Protocol) which is is a standard network protocol used for the transfer of computer files from a server to a client on a computer network., and Mail etc came from BSD UNIX.

The UNIX File System

UNIX File System

- UNIX has a hierarchical File System
- Important directories

/ Root Directory

/etc OS Configuration files

/etc/passwd – User information

/etc/groups – Group information

/etc/inetd.conf – Configuration of Internet

Inetd: inter**net** service **d**aemon

/etc/rc.*/ - OS initialization scripts for different services

- Deamons – Programs running in the background implementing a service. For each configured service, it listens for requests from connecting clients.

UNIX File System

- /dev – List of devices attached to the computer
- /usr – Libraries and tools
  - /usr/bin – Application programs such as grep, ls et
  - /usr/lib – Libraries used by the application programs
  - /usr/include – Include files (.h) for the libraries
- /home – Home directories

Users

```
$ cat /etc/passwd
```

Usually, the first line describes the root user, followed by the system and normal user accounts. New entries are appended at the end of the file.

Each line of the `/etc/passwd` file contains seven comma-separated fields:

```
Output

mark:x:1001:1001:mark,,,:/home/mark:/bin/bash
[--] - [--] [--] [-----] [--------] [--------]
|    |   |    |     |         |        |
|    |   |    |     |         |        +-> 7. Login shell
|    |   |    |     |         +----------> 6. Home directory
|    |   |    |     +--------------------> 5. GECOS
|    |   |    +--------------------------> 4. GID
|    |   +-------------------------------> 3. UID
|    +-----------------------------------> 2. Password
+----------------------------------------> 1. Username
```

UNIX was designed as a multiuser system.

The database of users is in /etc/passwd

The encrypted password used to be stored also here. Now it is stored in /etc/shadow

From https://linuxize.com/post/etc-passwd-file/

Users

- Commands for users
  - adduser – Adds a new user
  - passwd – Change password.
- There exist a special user called "root" with special privileges.
- Only root can modify files anywhere in the system.
- To login as root (superuser) use the command
- "su".

Only root can add users or reset passwords.

Groups

- A “group” represents a group of users.
- A user can belong to several groups.
- The file /etc/group describes the different
  groups in the system.

File Systems

- The storage can be classified from fastest to slowest in the following
  - Registers
  - Cache
  - RAM
  - Flash Memory
  - Disk
  - CD/DVD
  - Tape
  - Network storage

Disk File Systems

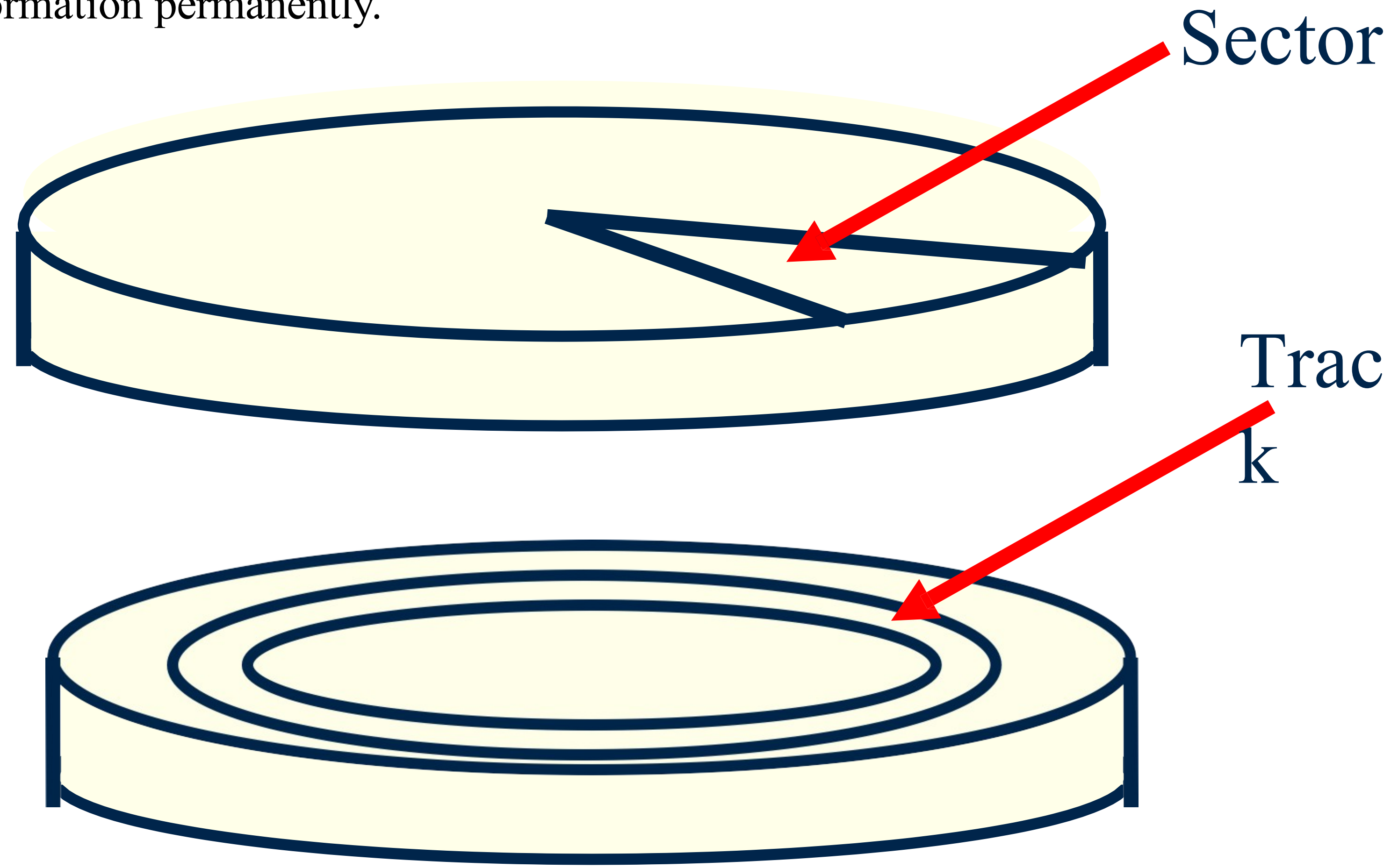
- A file system is a logical collection of files on a partition or disk.
  - Unix divided physical disks into logical disks called partitions
  - A partition is a container for information

- Everything in Unix is considered to be a file, including

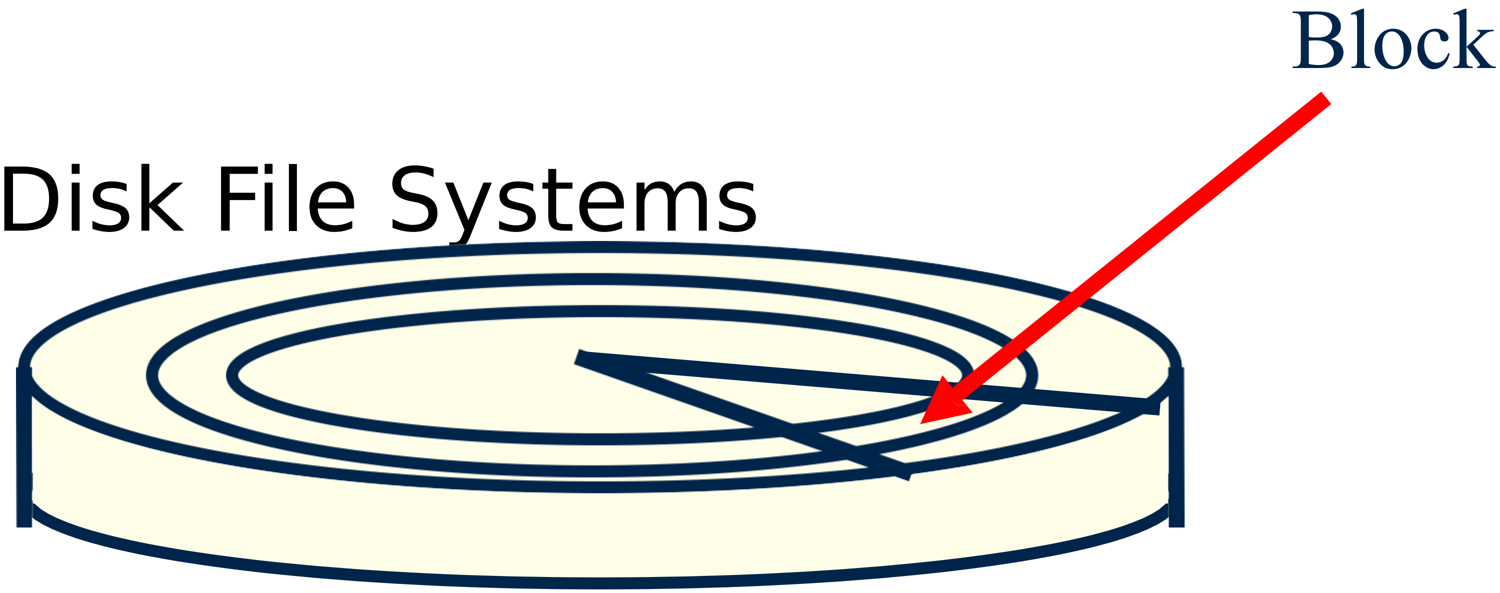
  documents, hard-drives, modems, keyboards, printers and even some inter-process and network communications as simple streams of bytes exposed through the filesystem

Disk File Systems

The disk is a an electromagnetic and mechanical device that is used to store information permanently.



Disk File Systems



A Block is the intersection between a sector and a track

Disks when formatted are divided into **sectors, tracks and blocks**.

Disks are logically divided into partitions.

A partition is a group of blocks.

Each partition is a different file system.

Disk File Systems

- The command **df -h (disk free)** displays the **disk space usage in human readable.**

```
$ df --human-readable
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       110G   45G   61G  43% /
devtmpfs         12G     0   12G   0% /dev
tmpfs            12G  848K   12G   1% /run
/dev/sda1       1.6T  1.3T  191G  87% /home
/dev/sdb1       917G  184G  687G  22% /penguin
/dev/sdc1        57G   50G  4.5G  92% /sneaker
/dev/sdd1       3.7T  2.4T  1.3T  65% /tux
```

- The **du (disk usage) command** enables you to specify directories to show disk space usage on a particular directory in blocks.

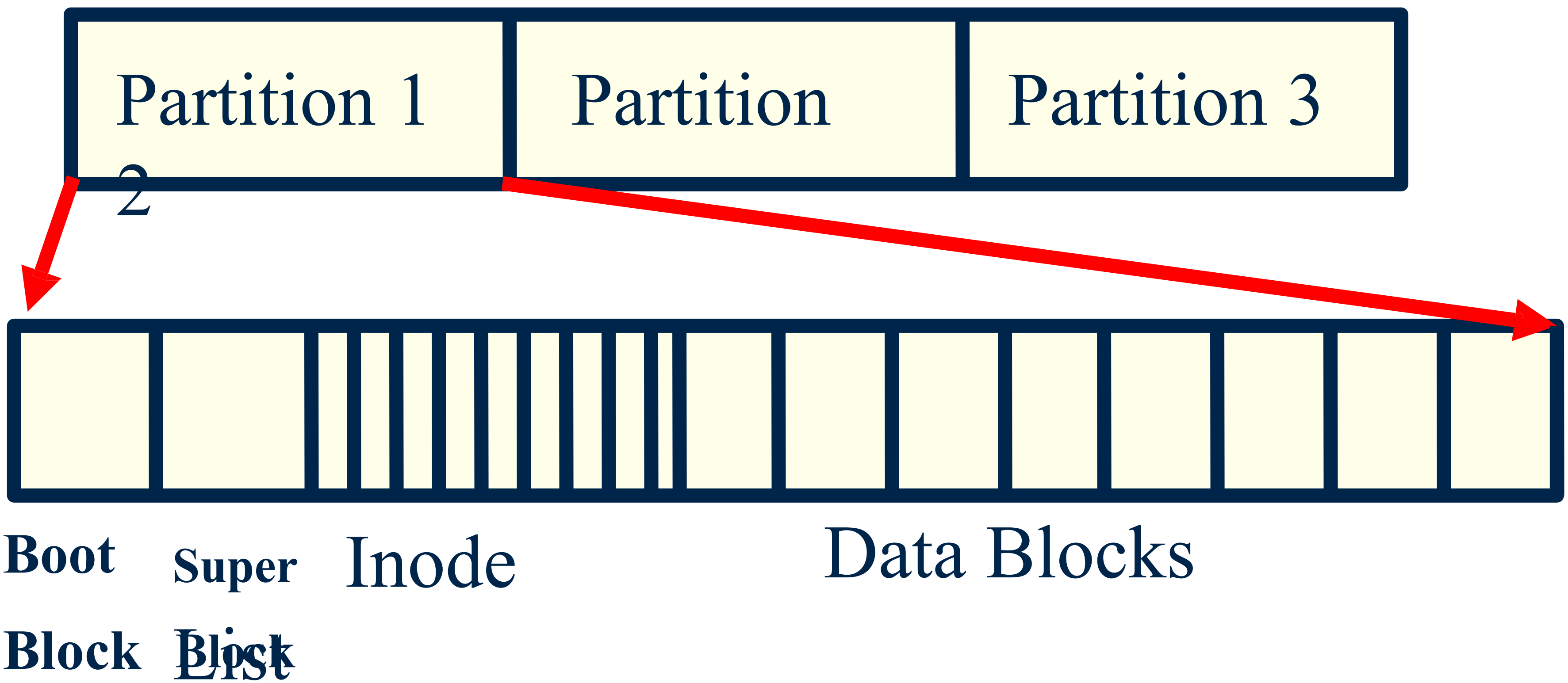
```
$du /etc
10          /etc/cron.d
126         /etc/default
6           /etc/dfs
...
$
```

The **-h** option makes the output easier to comprehend −

```
$du -h /etc
5k      /etc/cron.d
63k     /etc/default
3k      /etc/dfs
...
$
```

From tutorialspoint

Disk File System



Disk File System

- Each partition is divided into:
  - **Boot Block** – Located in the first few sectors of a file system. contains the initial bootstrap program used to load the operating system.
  - **Superblock** – Describes the state of the file system: the total size of the partition, the block size, pointers to a list of free blocks, the inode number of the root directory.
  - **Inode-list** – It is a linear array of I-nodes. One to one mapping of a file to an Inode. An inode has information about a file and what blocks make the file.
  - **Data Blocks** – Store the file data.

Thus, while users think of files in terms of file names, Unix thinks of files in terms of inodes.

File and Inode

- *You store your information in a file, and the operating system stores the information about a file in an inode (sometimes called as an inode number).*

- Whenever a user or a program needs access to a file, the operating system first searches for the exact and unique inode (inode number), in a table called as an inode list/table. In fact the program or the user who needs access to a file, reaches the file with the help of the inode number found from the inode list/table.

https://www.slashroot.in/inode-and-its-structure-linux

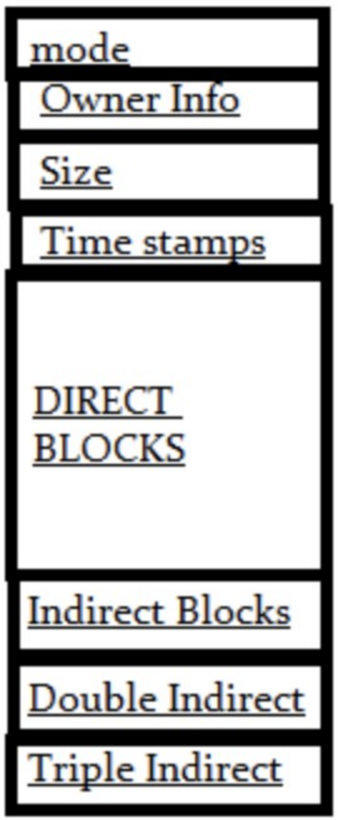
I-node information

- An i-node represents **a file** in disk. Each i-node contains **meta info** of the file:
  1. Flag/Mode
     1. Read, Write, Execute (for Owner/Group/All) RWX RWX RWX Also tells if file is directory, device, symbolic link
  2. Owners
     1. Userid, Groupid
  3. Time Stamps
     1. Creation time, Access Time, Modification Time.
  4. Size
     1. Size of file in bytes
  5. Ref. Count –
     1. Reference count with the number of times the i-node appears in a directory (hard links).
     2. Increases every time file is added to a directory. The file the i-node represents will be removed when the reference count reaches 0.

I-node information

- The I-node also contains block index with the blocks that form the file.
- To save space, the block index uses indices of different levels.
- This benefits small files since they form the largest percentage of files.
- Small files only uses the direct and single-indirect blocks. When files get bigger, it will use more and more indirect block.
- This saves in space spent in block indices.

Inode as a Data Structure



https://www.slashroot.in/inode-and-its-structure-linux

Mode:
This keeps information about two things, one is the permission information, the other is the type of inode, for example an inode can be of a file, directory or a block device etc.

Owner Info: Access details like owner of the file, group of the file etc.

Size: This location store the size of the file in terms of bytes.

Time Stamps: it stores the inode creation time, modification time, etc.

Now comes the important thing to understand about how a file is saved in a partition with the help of an inode.

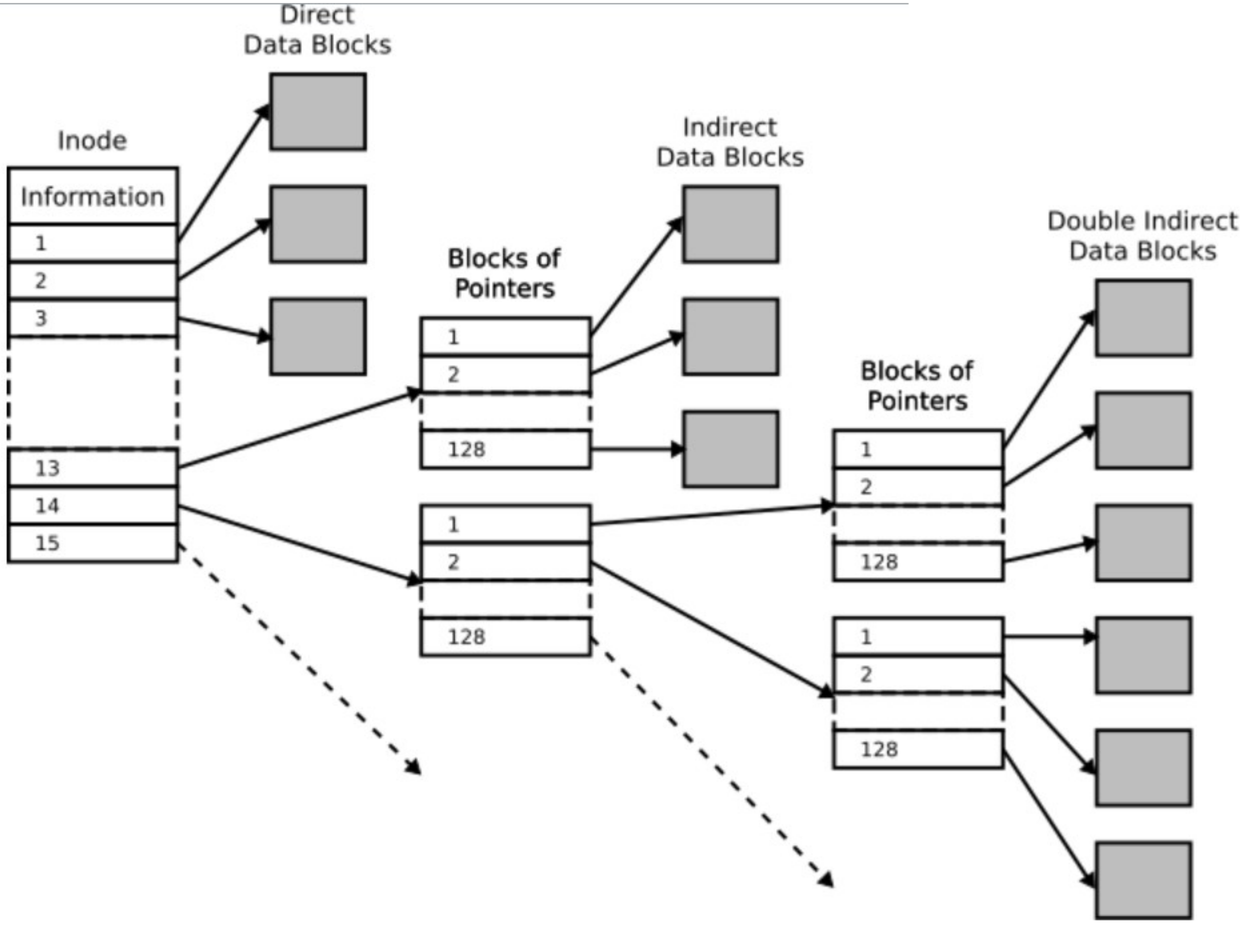
Block Size: Whenever a partition is formatted with a file system. It normally gets formatted with a default block size. Now block size is the size of chunks in which data
will be spread. So if the block size is 4K, then for a file of 15K it will take 4 blocks(because 4K*4 16), and technically speaking you waste 1 K.

Everything except **file name and data** are stored here

I-node information

- Modern File System has 15 pointers for each I-node structure
  Direct block –
  - Points directly to the block. There are **12** of them in the structure
- Single indirect –
  - Points to a block table that has various entry’s, say 128 or even more, There are **1** of them.
- Double indirect –
  - Points to a block table of 128 entries which then points to another block table of 128. There is **1** of them.
- Triple Indirect -
  - Points to a block table of 128 entries which then points to another block table of 128 that points to another table of 128. There is **1** of them.

I-node Structure



https://en.wikipedia.org/wiki/Inode_pointer_structure

I-node information

- Assume 1KB for each block and 128 entries in each block table.
- Direct block = 12 * 1Kb = 12Kb
- Single indirect = 1 * 128 * 1Kb = 128 Kb
- Double indirect = 1 * 128 * 128 * 1Kb = 16 Mb
- Triple indirect = 1 * 128 * 128 * 128 * 1Kb
  = 2 Gb

  More and more block pointers will be used when the file gets bigger.

I-node information

- Most of the files in a system are small.
- This also saves disk access time since small files need only **direct blocks**.
  - 1 disk access for the I-Node
  - 1 disk access for the datablock.
- An alternative to the multi-level block index is a ***linked list***. Every block will contain a pointer to the next block and so on.
- Linked lists are slow for random access.

Directory Representation and Hard Links

- Directories are just tables that associate file names with inode numbers. Each file name and inode number pair in a directory is called a link. Multiple names for the same inode are called “hard links”.
- Each file name is mapped to only one single inode number, but one file inode number may have many names that map to it.
- An I-node may appear in multiple directories.
- The reference count in the I-node keeps track of the number of directories where the I-node appears.
- When the reference-count reaches 0 by run rm command, the inode has no name and is freed.

```
$ ls -i /usr/bin/perl*
266327 /usr/bin/perl        266329 /usr/bin/perldoc.stub
266327 /usr/bin/perl5.14.2  266330 /usr/bin/perlivp
266331 /usr/bin/perlbug     266331 /usr/bin/perlthanks
266328 /usr/bin/perldoc
```

https://teaching.idallen.com/

Hard Links

- In some OSs, the reference count is incremented when the file is open.
- This prevents the file from being removed while it is in use.
- Hard Links has the same inode value as the original. If the original inode is updated, all hard links are updated.
- It cannot cross partitions, that is, a directory cannot list an I-node of a different partition.

Example. Creating a hard link to a target-file in the current directory

ln target-file name-link

Soft-Links

Example hard link versus soft/symbolic link
https://linuxgazette.net/105/pitcher.html

- Directories may also contain Soft-Links, which acts like shortcut in Windows.
- A soft-link is a pair of the form
  (file name, i-node number-with-file-storing-path)
  Where path may be an absolute or relative path in this or another partition.
- Soft-links can point to files in different partitions. A soft-link/symbolic link is referring to the original file and not its inode value, then replacing into another folder/removing the original file will break the symbolic link, or create a dangling link.

  Example:

  ln –s target-file name-link

File Ownership

- The Group Id and owner's User ID are stored as part of the file information
- Also the creation, modification, and access time are stored in the file in addition to the file size.
- The time stamps are stored in seconds after
  the Epoch (0:00, January 1st, 1970).

File Permissions

- The permissions of a file in UNIX are
- stored in the inode in the flag bits.
- Use “ls –l” to see the permissions.

```
-rw-rw-r--   1 grr           150 Aug 29  1995 calendar
-rw-------   1 grr           975 Mar 25  1999 dew.el
-rwxrwxr-x   1 grr          5924 Jul  9 10:48 chars
-rw-rw-r--   1 grr           124 Jul  9 10:47 chars.c
drwxr-sr-x  10 grr           512 Oct 14  1998 contools
drwxr-sr-x   9 grr           512 Oct  8  1998 contools-new
```

Permission Bits

The permissions are grouped into three groups: User, Group, and Others.

rwx rwx rwx

User Group Other

Permission Bits

- To change the persmissions of a file use the command chmod.

  chmod <u|g|o><+|-><r|w|x> filename

  Where

  <u|g|o> is the owner, group or others.

  <+|-> Is to add or remove permissions

  <r|w|x> Are read, write, execute permissions.

Permission Bits Example

- Make file “hello.txt” readable and writable by user and group but only readable by others

```
chmod u+rw hello.txt
chmod g+rw hello.txt
chmod o+r hello.txt
 chmod o-w
hello.txt
```

❑Scripts and Executable files should have the executable bit set

Permission Bits

- Also you can change the permission bits all at once using the bit representation in octal

```
USER  GROUP  OTHERS
RWX   RWX    RWX
110   110    100     - Binary
 6     6      4      - Octal digits
```

chmod 664 hello.c

"**r**" corresponds with 4, "**w**" with 2, and "**x**" with 1.

Process' Properties

- A process has the following properties:
  - PID: Index in process table
  - Command and Arguments
  - Environment Variables
  - Current Dir
  - Owner (User ID)
  - Stdin/Stdout/Stderr

Process ID

- Uniquely identifies the processes among all live processes.
- The initial process (init process) has ID of 0.
- The OS assigns the numbers in ascending order.
- The numbers wrap around when they reach the maximum and then are reused as long as there is no live process with the same processID.
- You can programmatically get the process id with
  - int getpid();

Command and Arguments

- Every process also has a command that is executing (the program file or script) and 0 or more arguments.
- The arguments are passed to main.

```
int main(int argc, char **argv);
```

- Argc contains the number of arguments including the command name.
- Argv[0] contains the name of the command

Printing the Arguments

```
printargs.c:
int main(int argc, char **argv) {
   int i;
for (i=0; i<argc; i++) {
  printf("argv[%d]=\"%s\"\n", i,
    argv[i]);
}
}

gcc –o printargs printargs.c
./printargs hello world
 argv[0]="./printargs"
argv[1]="hello"
argv[2]="world"
```

Environment Variables

- It is an array of strings that is inherited from the parent process. Some important variables are:
- - PATH=/usr/bin Stores the list of directories that contain commands to execute.
  - USER=<login> Contains the name of the user
  - HOME=/homes/mzhu Contains the home directory.
- You can add Environment variables settings in *.bashrc* which is a Bash shell script that Bash runs whenever it is started interactively. It initializes an interactive shell session.

Environment Variables

- To set a variable from a shell use

  export A=B

  - Modify the environment globally. All processes called will get this change

  A=B

  – Modify environment locally. Only current shell process will get this change.

- Example: Add a new directory to PATH

  export PATH=$PATH:/newdir

Printing Environment

- To print environment from a shell type env

```
mz10 — -bash — 80×24
Last login: Fri Feb  5 09:54:05 on console
CORE0782:~ mz10$ env
TERM_PROGRAM=Apple_Terminal
SHELL=/bin/bash
TERM=xterm-256color
TMPDIR=/var/folders/2p/_y68k3jj28g_7rdkgv0srf58_sbrs2/T/
Apple_PubSub_Socket_Render=/private/tmp/com.apple.launchd.sSEm5l6NR4/Render
TERM_PROGRAM_VERSION=404.1
TERM_SESSION_ID=8E43B928-04F4-49A0-B868-070DEB0064F2
USER=mz10
SSH_AUTH_SOCK=/private/tmp/com.apple.launchd.8h769qAd7f/Listeners
PATH=/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin
PWD=/Users/mz10
LANG=en_US.UTF-8
XPC_FLAGS=0x0
XPC_SERVICE_NAME=0
SHLVL=1
HOME=/Users/mz10
LOGNAME=mz10
SECURITYSESSIONID=186a7
_=/usr/bin/env
CORE0782:~ mz10$
```

Current Directory

- Every process also has a current directory.
- The open file operations such as open() and fopen() will use the current directory to resolve relative paths.
- If the path does not start with "/" then a path is relative to the current directory.

/etc/hello.c – Absolute

path hello.c – Relative

path.

To change the directory use "cd dir" in a shell

or

chdir(dir) inside a program

Process' User ID

- A process always runs in behalf of a user represented by the User ID.
- The UID is inherited from the parent
- process.
- Only root can change the UID of a process at runtime using the "setuid(uid);" call.
- This happens at login time. The login program runs as root but then after identifying the user, the OS switches the UID to that user and runs the shell.
- It also can be done with the comand "su user" that prompts for that user's password.
- Also "sudo user command" runs the command as

Stdin/Stdout/Stderr

- Also a process inherits from the parent a stdin/stdout and stderr.
- They are usually the keyboard and the terminal but they can be redirected.

PIPES

- In UNIX you can connect the output of a command to the input of another using PIPES.

  The Pipe is a command in Linux that lets you use two or more commands such that output of one command serves as input to the next. In short, the output of each process directly as input to the next one like a pipeline. The symbol '|' denotes a pipe.

Example:

cat filename | less

pipe the output of the 'cat' command to 'less' which will show you only one scroll length of content at a time.

Common UNIX Commands

Common UNIX Commands

- There are many UNIX commands that can be very useful
- They output of one can be connected to the input of the other using PIPES
- They are part of the UNIX distribution.
- Open source implementations exist and they are available in Linux.

ls – list files and directories

- ls <options> file list

To show a long format list of files (permissions, ownership, size, and modification date), use:

```
ls -l [path]
```

List all files, including important dotfiles (hidden files), use:

```
ls -a [path]
```

- Examples

  ls –al

  - Lists all files including hidden files (files that start with “.”) and timestamps.

  ls –R dir

  - Lists recursively all directories and subdirectories in dir.

mkdir – Make a directory

- mkdir <options> dir1 …
- Examples

  mkdir dir1

  Create directory dir1

  mkdir –p dir1/dir2/dir3

  Also make parent directory if they do not exist.

cp – Copy files

- cp <options> file1 file2 … destdir

  Copies one or more files to a destination.
- Examples:

  cp a.txt dir1

  Copies file a.txt into dir1

  cp a.txt b.txt

  Create a copy of a.txt called b.txt

  cp –R dir1 dir2

  Copy recursively directories and subdirectories of dir1 into dir2.

mv – Move a file

- mv file1 destdir

  Moves file1 to destdir
- Examples:

  mv a.txt dir1

  Move file a.txt into directory dir1

  mv a.txt b.txt

  Rename file a.txt into b.txt

rm – Remove a file

- rm <options> file1 file2 …

  Removes a list of files

- Examples:

  rm a.txt b.txt

  Remove files a.txt and b.txt

  rm –R dir1

  Remove dir1 and all its contents.

grep – Find lines

- grep <options> pattern file1 file2 …
  - Print lines that contain "pattern"
- Examples:

  grep hello a.txt

    Print the lines in a.txt that contain hello

man – Print manual pages

- man <options> command
  - Print the manual page related to command.
- Examples:

  man cp

  Print the manual pages related to copy

  man –k pthread

  Print all manual pages that contain string "pthread".

  man –s 3 exec

- Print manual page of exec from section 3

  Man Pages are divided into sections:

  Section 1 – UNIX commands (E.g. cp, mv etc) Section 2 – System Calls (E.g. fork)

  Section 3 – C Standard Library

  Example "man –s 1 printf" and "man –s 3c printf" give different man page

whereis – Where a file is located

- Where file
  - Prints the path of where a file is located.
  - It only works if the OS created a database with the files in the file system.
  - Example

    `bash-`

which – Path of a command

- which command
  - Prints the path of the **command** that will be executed by the shell if
    “

ps: list
processe
CPU time

```
CORE0782:~ mz10$ which ps
/bin/ps
CORE0782:~ mz10$ ps
  PID TTY           TIME CMD
18647 ttys000    0:00.09 -bash
19603 ttys000    0:00.02 man sudo
19604 ttys000    0:00.01 sh -c (cd '/usr/share/man' && /usr/bin/tbl '/usr/share
19605 ttys000    0:00.00 sh -c (cd '/usr/share/man' && /usr/bin/tbl '/usr/share
19608 ttys000    0:00.00 sh -c (cd '/usr/share/man' && /usr/bin/tbl '/usr/share
19609 ttys000    0:00.01 /usr/bin/less -is
19614 ttys000    0:00.01 man su
19615 ttys000    0:00.00 sh -c (cd '/usr/share/man' && /usr/bin/tbl '/usr/share
19616 ttys000    0:00.00 sh -c (cd '/usr/share/man' && /usr/bin/tbl '/usr/share
19619 ttys000    0:00.00 sh -c (cd '/usr/share/man' && /usr/bin/tbl '/usr/share
19620 ttys000    0:00.01 /usr/bin/less -is
72312 ttys000    0:00.02 man cp
72313 ttys000    0:00.01 sh -c (cd '/usr/share/man' && /usr/bin/tbl '/usr/share
72314 ttys000    0:00.00 sh -c (cd '/usr/share/man' && /usr/bin/tbl '/usr/share
72317 ttys000    0:00.00 sh -c (cd '/usr/share/man' && /usr/bin/tbl '/usr/share
72318 ttys000    0:00.02 /usr/bin/less -is
```

head – Lists the first few lines

- head file

  List the first 10 lines of a file

Example:

head myprog.c

Lists the first 10 lines of myprog.c

head -5 myprog.c

List the first 5 lines of myprog.

tail – Lists the last few lines

- tail file
  - List the last 10 lines of a file
- Example:

  tail myprog.c

  list the last 10 lines of myprog.c

  tail -3 myprog.c

  list the last 3 lines of myprog.c

  tail –f a.log

  It will periodically print the lines of a.log as they are added.

find – Execute a command for a group of files

- find – Recursively descends the directory hierarchy for each path seeking files that match a Boolean expression.
- Examples:

```
find . -name "*.conf"
```

Find recursively all files with .conf extension.

```
find /home/mz10 -name "*.conf"
```

Find the file in the specific directory