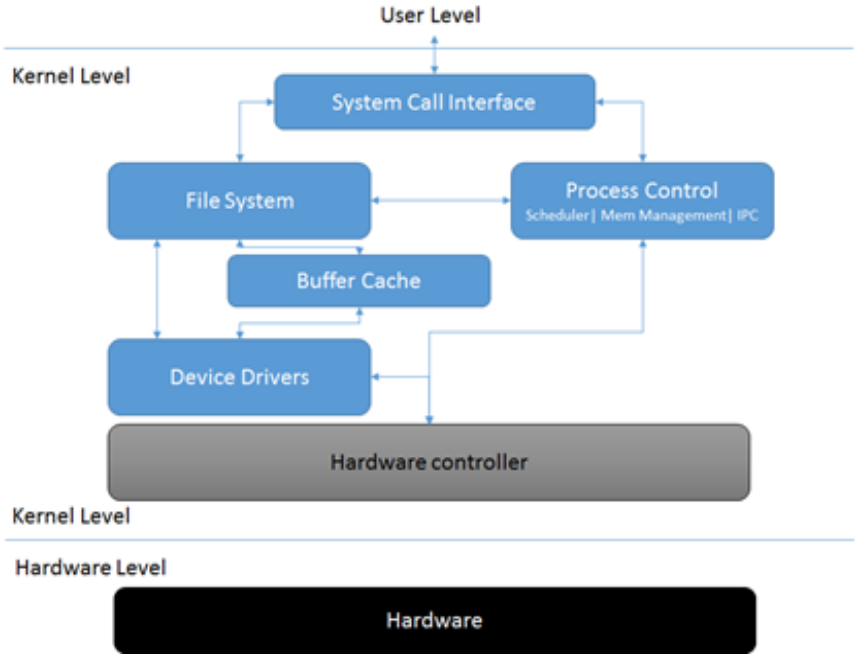
MODULE 2 :
UNIX Files and File System

Unix File System Architecture



The above figure shows the relationships between the major file-system related components in both user space and the kernel.

Unix File System Architecture(contd.)

- User space contains the applications (for this example, the user of the file system) and the GNU C Library (glibc), which provides the user interface for the file system calls (open, read, write, close). The system call interface acts as a switch, funneling system calls from user space to the appropriate endpoints in kernel space.

- Everything below the system call interface and above the hardware is the Kernel. The kernel provides the file system, the process control subsystem, the memory management and the operating system functions over system calls.

- System calls for file manipulation are for example: creat, open, read, write, close, link, unlink and trunc.

I/O in Unix

There are three main categories of I/O in UNIX: Block devices, character device, and sockets for network communication.

- Block devices: This category includes disks and tape. A block device is essential to isolate disk details from the rest of the kernel. Block devices are accessible thru the system file (example: /dev/sda the first scsi drive on the first scsi bus)

- Character devices: this category includes printers, terminals (example: /dev/null a bit bucket or bottomless sink for data)

I/O in Unix(contd.)

- Sockets: This category of device helps connecting to databases, and serving web pages as well as network utilities such as rlogin for remote login, http for hypertext transfer protocol and ftp for file transfer.

- Almost all the files under /dev are device files. Whereas reading and writing to a regular file stores data on a disk or other filesystem, accessing a device file communicates with a driver in the kernel, which generally in turn communicates with a piece of hardware (a hardware device, hence the name).

In order to display the block and character devices in a UNIX system, execute the following command in the shell.

```
_ ls -la /dev | more
total 4
drwxr-xr-x  23 root  root            4520 Oct 13 14:20 .
drwxr-xr-x  33 root  root            4096 Oct  1 10:30 ..
crw-------   1 root  root      10,    54 Oct 13 14:20 acpi_thermal_rel
crw-r--r--   1 root  root      10,   235 Oct 13 14:20 autofs
drwxr-xr-x   2 root  root             320 Oct 13 14:20 block
drwxr-xr-x   2 root  root              80 Oct 13 14:20 bsg
crw-rw----   1 root  disk      10,   234 Oct 13 14:20 btrfs-control
drwxr-xr-x   3 root  root              60 Oct 13 14:20 bus
lrwxrwxrwx   1 root  root               3 Oct 13 14:20 cdrom -> sr0
lrwxrwxrwx   1 root  root               3 Oct 13 14:20 cdrw -> sr0
drwxr-xr-x   2 root  root            4520 Oct 16 10:23 char
crw-------   1 root  root       5,      1 Oct 13 14:21 console
lrwxrwxrwx   1 root  root              11 Oct 13 14:20 core -> /proc/kcore
drwxr-xr-x   2 root  root              60 Oct 13 14:20 cpu
crw-------   1 root  root      10,    59 Oct 13 14:20 cpu_dma_latency
crw-------   1 root  root      10,   203 Oct 13 14:20 cuse
drwxr-xr-x   8 root  root             160 Oct 13 14:20 disk
```

Basics of Files

- From a user perspective in a Unix system, everything is treated as a file. Even such devices such as printers and disk drives.

- All files in the Unix file system can be loosely categorized into 3 types, specifically:
  - ✔ Ordinary files
  - ✔ Directory files
  - ✔ Device files

Ordinary Files

- **Text file or Binary file**

  Both can be executable files (.sh) with suitable permissions.
  Ex: scripts, documentation, GIF, JPEG, etc.

- **How to create?** using editors (vi editor)

- **How to delete?** using rm command

Directory Files

- Can contain any kind of files

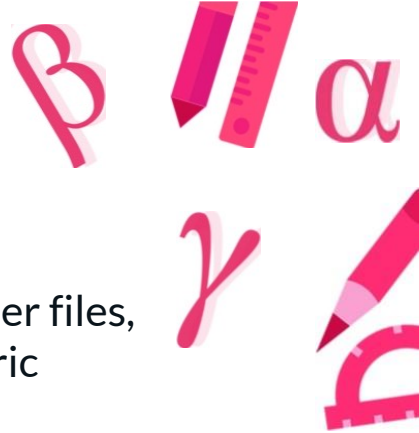
  What is “.” and “..”??

  - **How to create?** using mkdir command (mkdir /tmp/demofolder)
  - **How to delete?** using rmdir command (rmdir /tmp/demofolder)

Unix File System

- The Unix file system is a methodology for logically organizing and storing large quantities of data such that the system is easy to manage.
- A file can be informally defined as a collection of (typically related) data, which can be logically viewed as a stream of bytes (i.e. characters).
- A file is the smallest unit of storage in the Unix file system.

Unix File System(contd.)

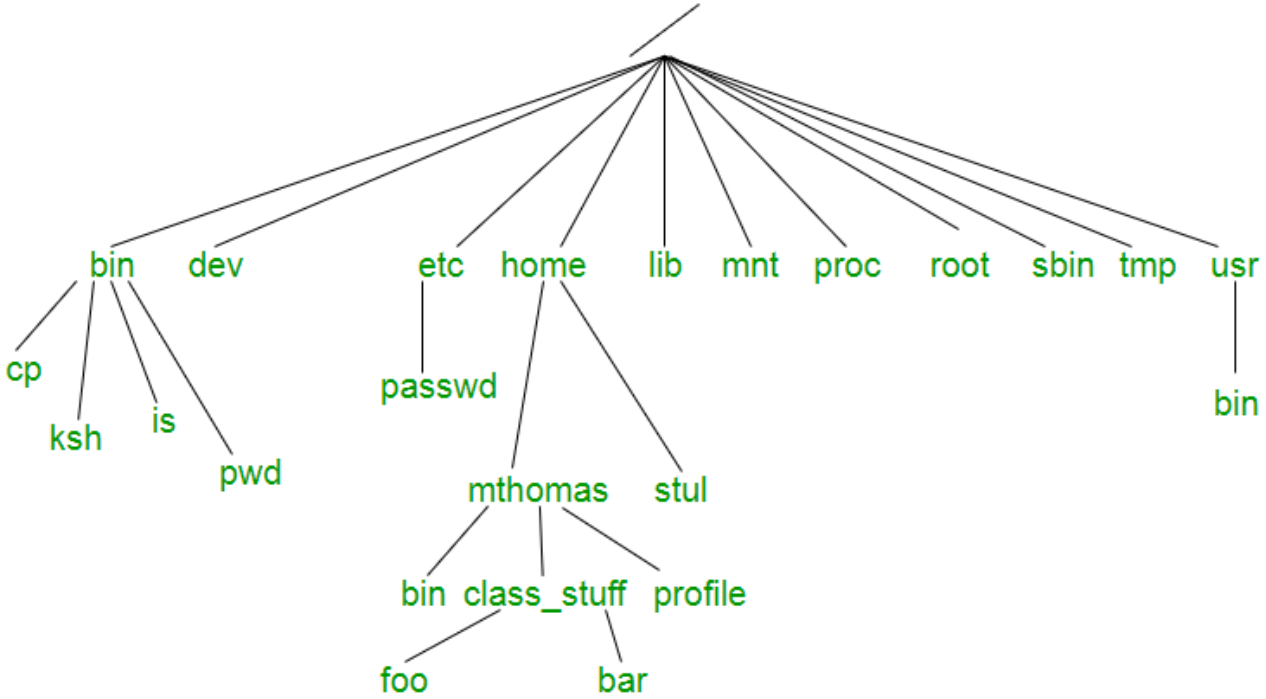


By contrast, a file system consists of files, relationships to other files, as well as the attributes of each file. File attributes for a generic operating system might include (but are not limited to):

- a file type (i.e. what kind of data is in the file)
- a file name (which may or may not include an extension)
- a physical file size
- a file owner
- file protection/privacy capability
- file time stamp (time and date created/modified)

The Unix file system has a hierarchical (or tree-like) structure with its highest level directory called root (denoted by **/**, pronounced **slash**).

Below is a diagram of a "typical" Unix file system. The top-most directory is **/** (**slash**), with the directories directly beneath being system directories.



System files in Unix FileSystems

Following system files (i.e. directories) are present in most Unix filesystems:

- bin - short for binaries, is the directory where many commonly used executable commands reside
- dev - contains device specific files
- etc - contains system configuration files
- home - contains user directories and files
- lib - contains all library files
- mnt – contains device files related to mounted devices

System files in Unix FileSystems (contd.)

- proc - contains files related to system processes
- root - the root user's' home directory (note this is different than /)
- sbin - system binary files reside here. If there is no sbin directory on your system, these files most likely reside in etc
- tmp - storage for temporary files which are periodically removed from the filesystem
- usr - also contains executable commands

Inode Structure

- We store your information in a file, and the operating system stores the information about a file in an inode, sometimes called as inode number.
- ‘An inode is metadata of the data’.
- 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 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 table.

Inode Structure(contd.)

. (DOT)	9700524
.. (DOT DOT)	9699798
file1.txt	9700532
file2.txt	9700533

file & folder names

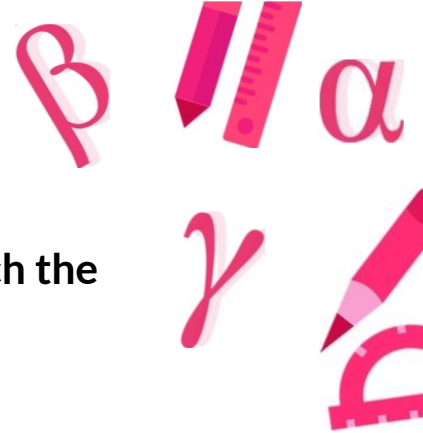
Corresponding inodes

An inode is a list of information related to a particular object (either a file, a directory or a symbolic link). The inode stores the type of object, permissions, ownership and the location where the data is stored.

Lets see an example directory listing

```
_ ls -ia
9700524 .   9700532 file1.txt  9700534 file3.txt
9699798 ..  9700533 file2.txt  9700535 file4.txt
```

Inode Structure(contd.)



Each inode, an entry in the inode table, is a data structure which the system uses to store the following information about a file:

- Type of file (ordinary, directory or special file).
- Access permissions for the file owner, the owner's group members and others (i.e. the general public).
- Number of links to the file.
- File owner's user and group IDs
- File size in bytes.
- The disk addresses of the data blocks where the contents of the file are actually stored.
- Time of last access (read or executed), time of last modification (i.e. written) and time which the inode itself was last changed.

Inode Structure(contd.)

```
  File: important/file1.txt
  Size: 0               Blocks: 0          IO Block: 4096   regular empty file
Device: 803h/2051d      Inode: 9700532     Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2019-10-15 17:09:20.130714453 +0530
Modify: 2019-10-15 17:09:20.130714453 +0530
Change: 2019-10-15 17:09:20.130714453 +0530
 Birth: -
```

File Related Commands

Following are commonly used file related commands in linux:

- cat - display entire file(s) in the terminal window
- cd - change the current directory
- cp - copy file(s)
- chmod - change file(s) protection modes
- lpr - send file(s) to the line printer

File Related Commands(contd.)

- mkdir - create a new directory
- more - nicely view the contents of file(s)
- less - opposite of more
- mv - move (and/or rename) file(s)
- pwd - print absolute pathname of current working directory
- rm - remove file(s) rmdir - remove an empty directory

Comparing files in Unix

```
root in /home/akhan/Documents/demo/important
_  whatis cmp
cmp (1)              - compare two files byte by byte

root in /home/akhan/Documents/demo/important
_ cmp file1.txt file2.txt
file1.txt file2.txt differ: byte 16, line 1

root in /home/akhan/Documents/demo/important
_ whatis comm
comm (1)             - compare two sorted files line by line

root in /home/akhan/Documents/demo/important
_ comm file1.txt file2.txt
hello from file1
        hello from file2

root in /home/akhan/Documents/demo/important
_ whatis diff
diff (1)             - compare files line by line

root in /home/akhan/Documents/demo/important
_ diff file1.txt file2.txt
1c1
< hello from file1
---
> hello from file2
```

To get detailed information on the use of the command use man pages.

man cmp

man is the system's manual pager.

LearnOA

Directories in Unix

In Unix directories are the entities ,which gave system its hierarchical (tree like) file system structure.

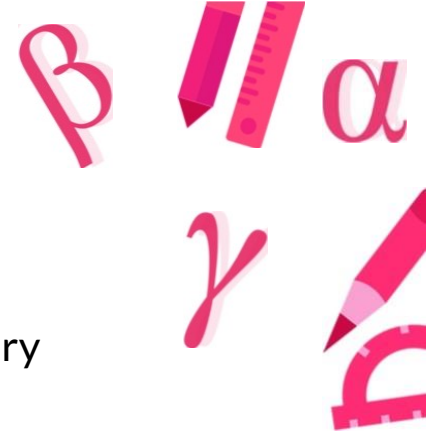
Directory file contains some sequential entries. 3 things are important :

1. **Byte offset :** This is the number which gives location of files inode in that directory .

1. **Inode number:** Inodes entry is made in the directory file along with its name.

1. **File names:** These are the names of files located under present directory.

Directories in Unix(contd.)



The first two file names in every directory are (“.”) and (“..”)

- where “.” Represents present or current directory

and

- “..” Represents parent directory file.

Each entry in a directory file is of 16-byte size

- where 14 bytes for 14 characters in file name
- and 2 bytes for inode number

Access Permissions

Access permissions to the users for directory file :

1. READ - Allow the user to see contents i.e. files in that directory

2. WRITE – Allows the user to create or delete subdirectories and files in that directory

3. EXECUTE – Allows the user to search the directory i.e allow the user to open subdirectories in that directory

For kernel writing permission for directory means , to actually write in directory file.

Files and Directories Attributes

Mode	Links	Owner	Group	File size	Last modified	Filename
dr-xr-xr-x	387	root	root	0	Oct 13 14:20	proc
drwxr-xr-x	48	akhan	akhan	4096	Oct 16 12:19	root
drwxr-xr-x	45	root	root	1420	Oct 16 10:23	run
drwxr-xr-x	2	root	root	12288	Oct 11 12:47	sbin
drwxr-xr-x	5	root	root	4096	Jul 24 12:47	snap
drwxr-xr-x	2	root	root	4096	Jun 24 2016	srv
drwxr-xr-x	5	root	root	4096	May 16 13:47	statping
dr-xr-xr-x	13	root	root	0	Oct 16 13:20	sys
drwxrwxrwt	10	root	root	4096	Oct 16 13:24	tmp
-rw-r--r--	1	root	root	0	Oct 16 13:24	tmp_information
drwxr-xr-x	13	root	root	4096	Jul 26 10:11	usr
drwxr-xr-x	16	root	root	4096	Jul 2 13:11	var
lrwxrwxrwx	1	root	root	30	Oct 1 10:30	vmlinuz -> boot/vmlinuz-4.15.0-65-generic
lrwxrwxrwx	1	root	root	30	Oct 1 10:30	vmlinuz.old -> boot/vmlinuz-4.15.0-64-generic

Files and Directories Attributes(contd.)

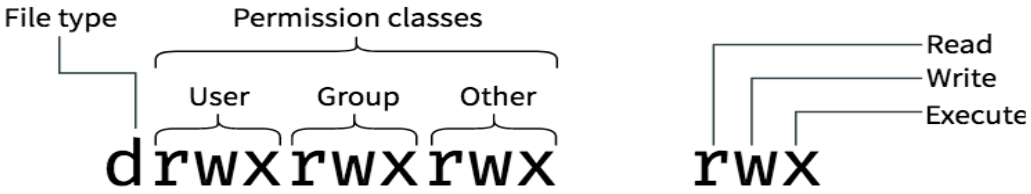
Following are the attributes associated with files and directories:

- file name: the name associated with the file (recall, this can be any type of file)
- modification date: the date the file was last modified, i.e. a "time-stamp". If the file has not been modified within the last year (or six months for Linux), the year of last modification is displayed.
- size: the size of the file in bytes (i.e. characters)
- group: associated group for the file

Files and Directories Attributes(contd.)

- owner: the owner of the file
- number of links: the number of other links associated with this file
- permission modes: the permissions assigned to the file for the owner, the group and all others

Files and Directories Permissions



	4	2	1	
0	-	-	-	no permissions
1	-	-	x	only execute
2	-	w	-	only write
3	-	w	x	write and execute
4	r	-	-	only read
5	r	-	x	read and execute
6	r	w	-	read and write
7	r	w	x	read, write and execute

Files and Directories Permissions(contd.)

- 9 permission bits used to determine 3 types of accesses: READ, WRITE, EXECUTE.
- Permission can be set based on GROUP, OWNER, ANYONE ELSE
- Use chmod command to change permission
  - Binary 001 for EXECUTE
  - Binary 010 for WRITE
  - Binary 100 for READ

Files and Directories Permissions(contd.)

Examples:

```
chmod 777 /tmp/dir
```

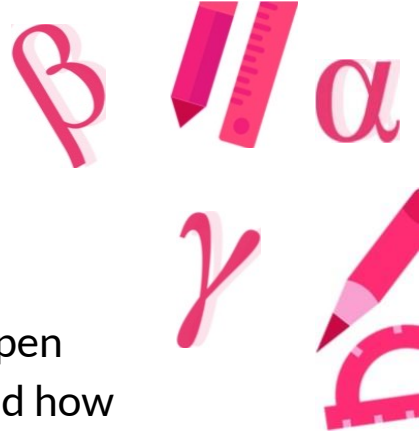
```
chmod u+rwx,g+rwx,o+rwx /tmp/dir
```

```
chmod u+rwx,o+rx /tmp/dir
```

```
chmod g-rx /tmp/dir
```

```
chmod 700 /tmp/dir
```

Open Files and File Descriptor in Unix



- A file descriptor is a number that uniquely identifies an open file in a operating system. It describes a data resource, and how that resource may be accessed.

- When the UNIX kernel starts any process - for example, grep , ls , or a shell - it sets up several places for that process to read from and write to. These places are called “open files” .

- The kernel gives each file a number called a file descriptor.

- On a Unix-like operating system, the first three file descriptors, by default, are STDIN (standard input), STDOUT (standard output), and STDERR (standard error).

Name	File descriptor	Description	Abbreviation
Standard input	0	The default data stream for input, for example in a command pipeline. In the terminal, this defaults to keyboard input from the user.	stdin
Standard output	1	The default data stream for output, for example when a command prints text. In the terminal, this defaults to the user's screen.	stdout
Standard error	2	The default data stream for output that relates to an error occurring. In the terminal, this defaults to the user's screen.	stderr

Go to your terminal command line and type the following command to get the list of file descriptors for your shell process.

```
_ ls -l /proc/$$/fd
total 0
lrwx------ 1 root root 64 Oct 16 19:42 0 -> /dev/pts/4
lrwx------ 1 root root 64 Oct 16 19:42 1 -> /dev/pts/4
lrwx------ 1 root root 64 Oct 16 19:42 2 -> /dev/pts/4
lrwx------ 1 root root 64 Oct 16 19:42 255 -> /dev/pts/4
```

What is file descriptor 255 for?

As an alternative connection to the tty in case fd 1 (/dev/stdout) and fd 0 (/dev/stdin) get blocked.

$$ is the process ID of the current shell instance.

```
_ echo $$
25306
```

Other Types of File Systems

Some of the other common file system types are:

- FAT
- FAT32
- FAT64
- NTFS

FAT File System

- FAT stands for "**File Allocation Table**".
- The file allocation table is used by the operating system to locate files on a disk.
- A file may be divided into many sections and scattered around the disk due to fragmentation.
- FAT keeps track of all pieces of a file.
- FAT provides quick access to files.
- The speed of file access depends on file type, file size, partition size, fragmentation and number of files in a folder.

FAT32 File System

- FAT32 is an advanced version of FAT file system.
- It can be used on drives from 512 MB to 2TB in size.
- One of the most important features of FAT and FAT32 is that they offer compatibility with operating systems other than Windows 2000 also.
- FAT32 provides good file access in partition sizes less than 500 MB or greater than 2 GB.
- FAT32 provides better disk space utilization.

NTFS File System

- NTFS stands for “New Technology File System”.
- Windows 2000 professional fully supports NTFS.
- NTFS is highly reliable.
- It is recoverable file system.
- It uses transaction logs to update the file and folders logs automatically.
- The system also has a great amount of fault tolerance. It means that if transaction fails due to power or system failure, the logged transactions are used to recover the data.

Thank You!