UNIX/Linux

Operating Systems

UNIX

- Unix is a multi-user, multi-tasking operating system.
- You can have many users logged into a system simultaneously, each running many programs.
- It's the kernel's job to keep each process and user separate and to regulate access to system hardware, including cpu, memory, disk and other I/O devices.

UNIX/Linux Goals

- Designed by programmers, for programmers
- Designed to be:
  - Simple
  - Elegant
  - Consistent
  - Powerful
  - Flexible

History of UNIX

- First Version was created in Bell Labs in 1969 on a PDP-7.
- 1973 Re-written mostly in C, made it easy to port it to new machines.
- 1977 There were about 500 Unix sites world-wide.
- 1980 BSD 4.1 (Berkeley Software Development)
- 1983 SunOS, BSD 4.2, System V
- 1988 AT&T and Sun Microsystems jointly develop System V Release 4.
- 1991 Linux originated.

Standard UNIX

- There have been numerous variants of UNIX over the years.
  - By the end of the 1980s, two different, and somewhat incompatible, versions of UNIX were in widespread use: 4.3BSD and System V Release 3.
- An IEEE Standards Board named **POSIX** was created as an attempt to reconcile the two flavours of UNIX.
- The POSIX committee produced a standard known as **1003.1**. It defines a set of library procedures that every conformant UNIX system must supply.
- Linux is a POSIX compliant operating system

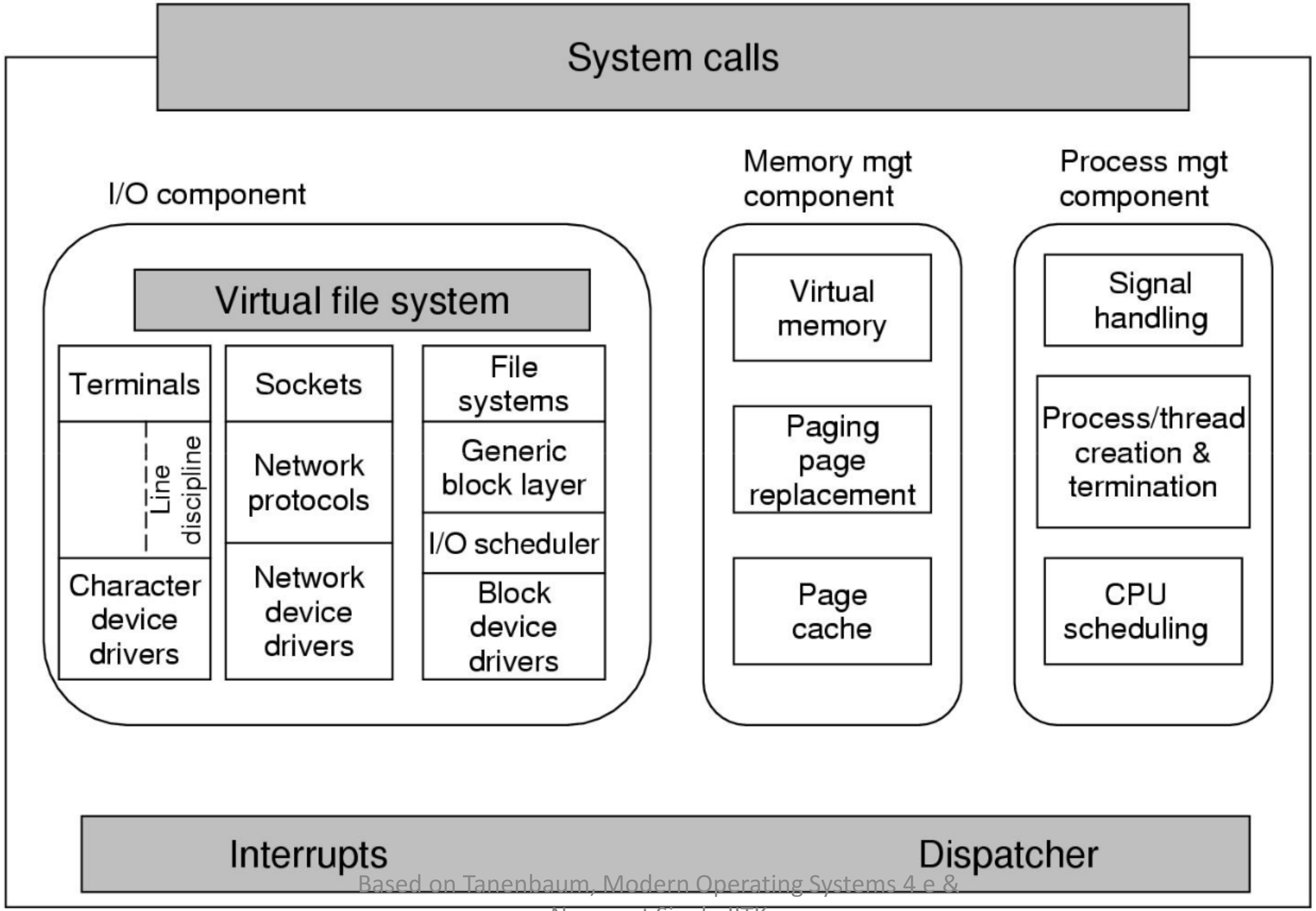
Linux

- A popular variant of UNIX, which runs on a wide variety of computers
- It is one of the dominant operating systems on high-end workstations and servers, but also used on systems ranging from smartphones to supercomputers
- Linux is free software, with a **GPL** (**GNU Public License**)
- It is a monolithic rather than microkernel design, with the entire operating system in the kernel
- Android is based on Linux

Linux Distributions

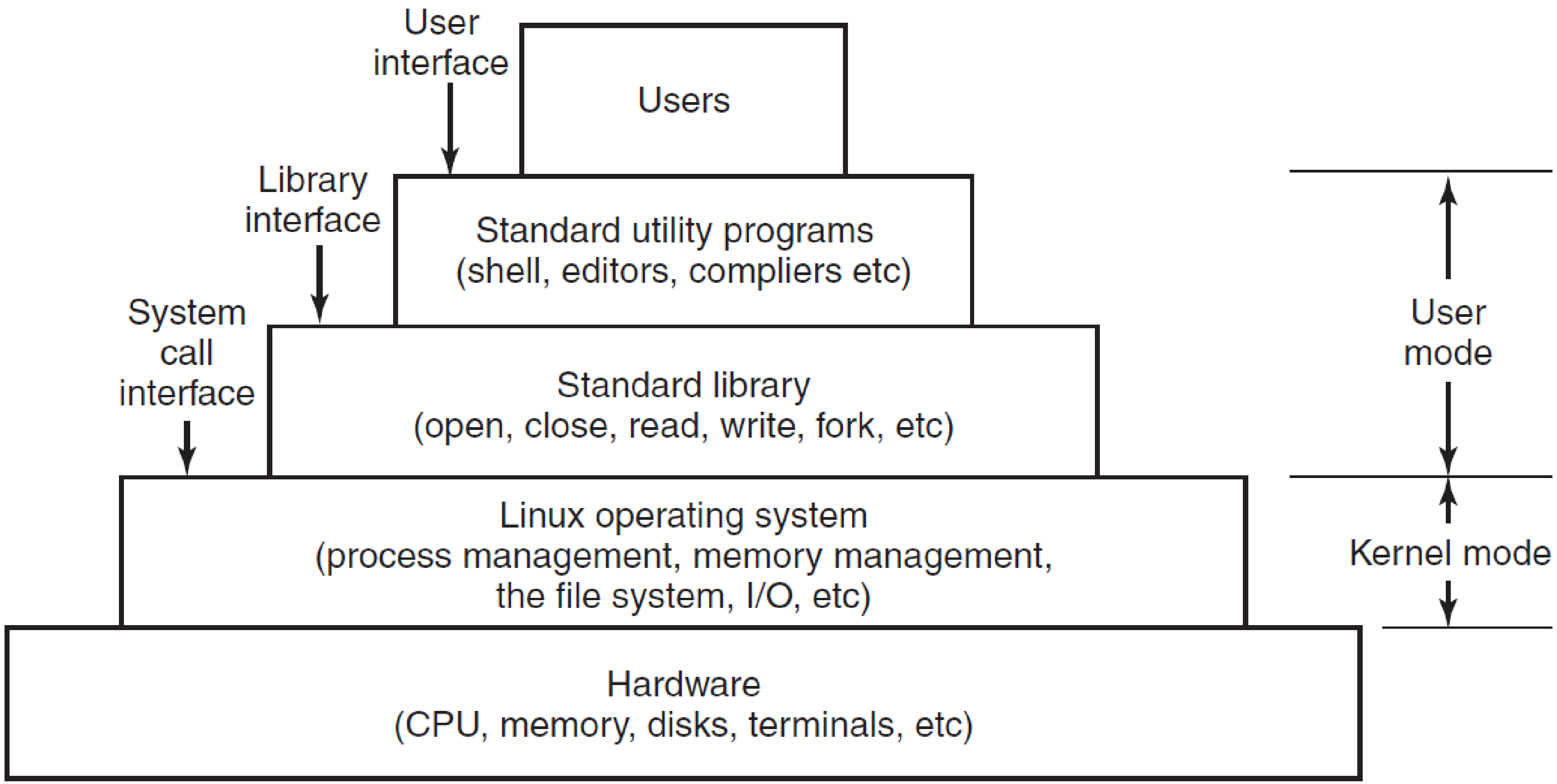
- Ubuntu: https://www.ubuntu.com/
- Debian: http://www.debian.org/
- Linux Mint: https://linuxmint.com/
- RedHat: http://www.redhat.com/
- Fedora: http://fedora.redhat.com/
- SuSE: http://www.suse.com/

Kernel Structure



Based on Tanenbaum, Modern Operating Systems 4 e & Navpreet Singh, IITK

Interfaces to Linux


User interface
Users
Library interface
Standard utility programs
(shell, editors, compliers etc)
System call interface
Standard library
(open, close, read, write, fork, etc)
Linux operating system
(process management, memory management,
the file system, I/O, etc)
Hardware
(CPU, memory, disks, terminals, etc)
User mode
Kernel mode

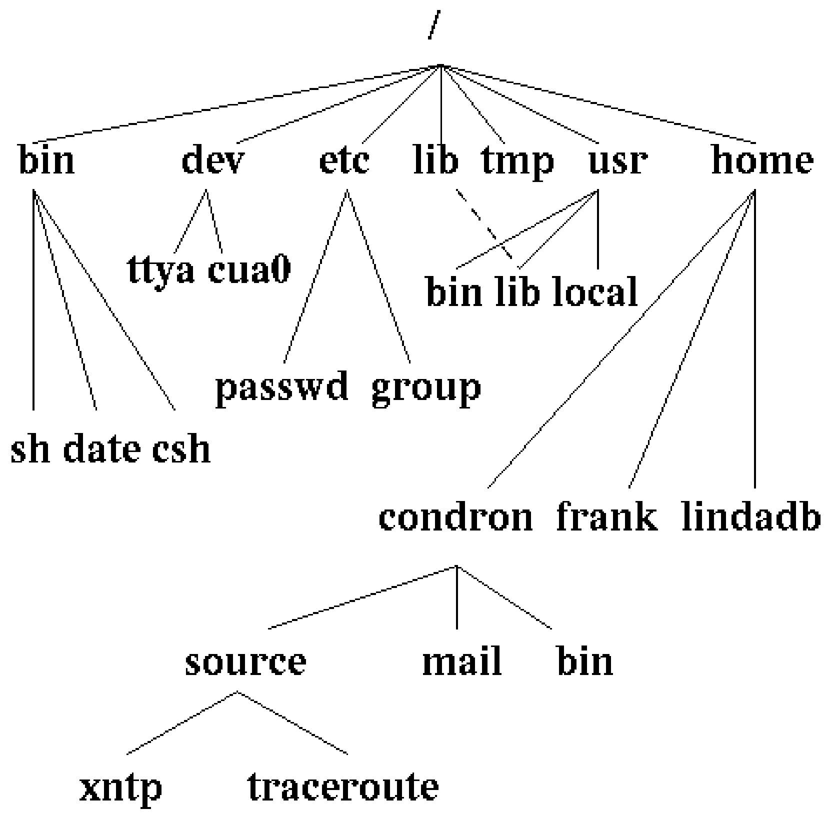
Based on Tanenbaum, Modern Operating Systems 4 e &
Navpreet Singh, IITK

Interfaces to Linux

- Most of the common personal computer distributions of Linux now have a mouse-oriented graphical user interface.
- GUIs on Linux are supported by the X Windowing System, which defines communication and display protocols for manipulating windows on bitmap displays for UNIX and UNIX-like systems.
- Popular desktop environments for Linux include GNOME (GNU Network Object Model Environment) and KDE (K Desktop Environment).

File System

- The Unix file system looks like an inverted tree structure.
- You start with the root directory, denoted by /, at the top and work down through sub-directories underneath it.
- A relative path name specifies the path relative to another, usually the current working directory that you are at.
- Two special directories :
  - . the current directory
  - . . the parent of the current directory

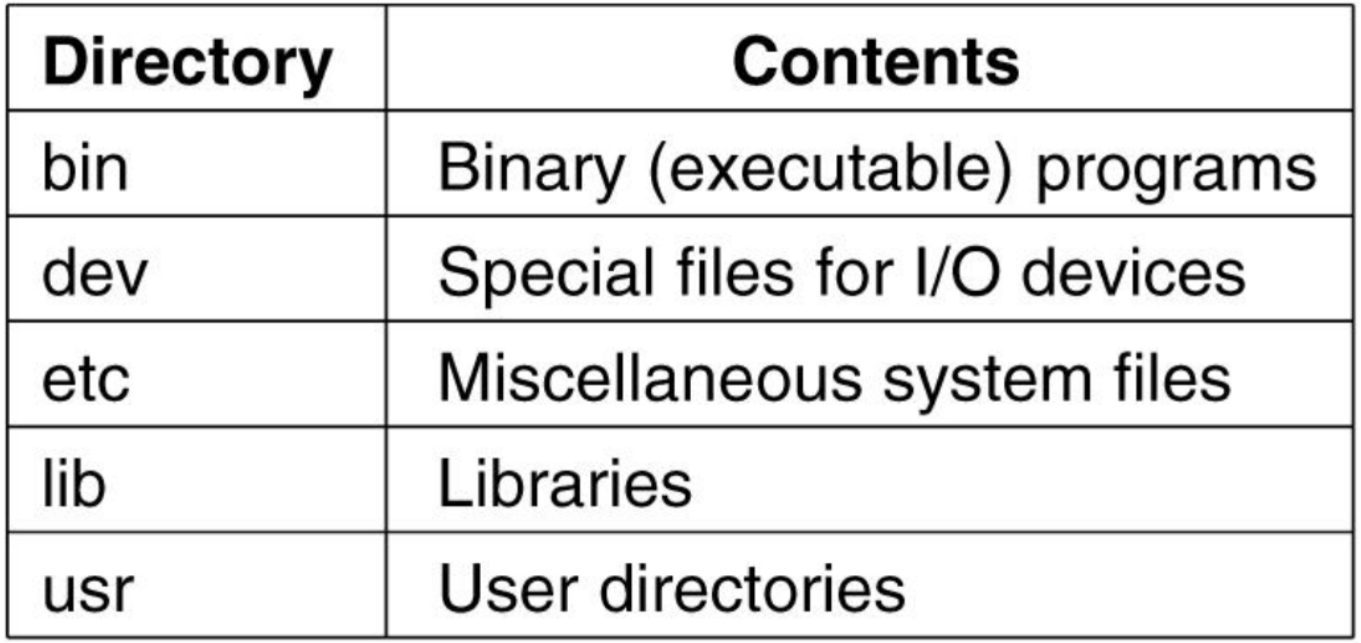


File System

- Types of files
  - Ordinary disk files
  - Special files
    - Each physical device on a Unix system is treated as a special file.
    - Located in the /dev directory.
  - Directory flies
- Naming files and directories
  - File and directory names can include letters, numbers, periods(.), underscores(_), and some other printable characters.
  - Avoid characters with special programming or system meanings, such as /, *?[]<>$'"&!.
  - Generally, a name of file and directory can contain up to 255 characters.

File Systems

- Some important directories found in most Linux systems



Directory	Contents
bin	Binary (executable) programs
dev	Special files for I/O devices
etc	Miscellaneous system files
lib	Libraries
usr	User directories

Directories, Files and inodes

- Every directory and file is listed in its parent directory.
- In the case of the root directory, that parent is itself.
- A directory is a file that contains a table listing the files contained within it, giving file names to the inode numbers in the list.
- An inode (Index Nodes) is an entry in the table containing information about a file (metadata) including file permissions, UID, GID, size, time stamp, pointers to files data blocks on the disk etc.
- The information about all the files and directories is maintained in INODE TABLE

Users, Groups and Access Permissions

- In UNIX/Linux, there is a concept of user and an associated group
- The system determines whether or not a user or group can access a file or program based on the permissions assigned to them.
- Apart from all the users, there is a special user called Super User or the root which has permission to access any file and directory

Access Permissions

- There are three permissions for any file, directory or program:
  - r — Indicates that a given category of user can read a file.
  - w — Indicates that a given category of user can write to a file.
  - x — Indicates that a given category of user can execute the file.
- Each of the three permissions are assigned to three defined categories of users:
  - owner — The owner of the file or application.
  - group — The group that owns the file or application.
  - others — All users with access to the system.

Access Permissions

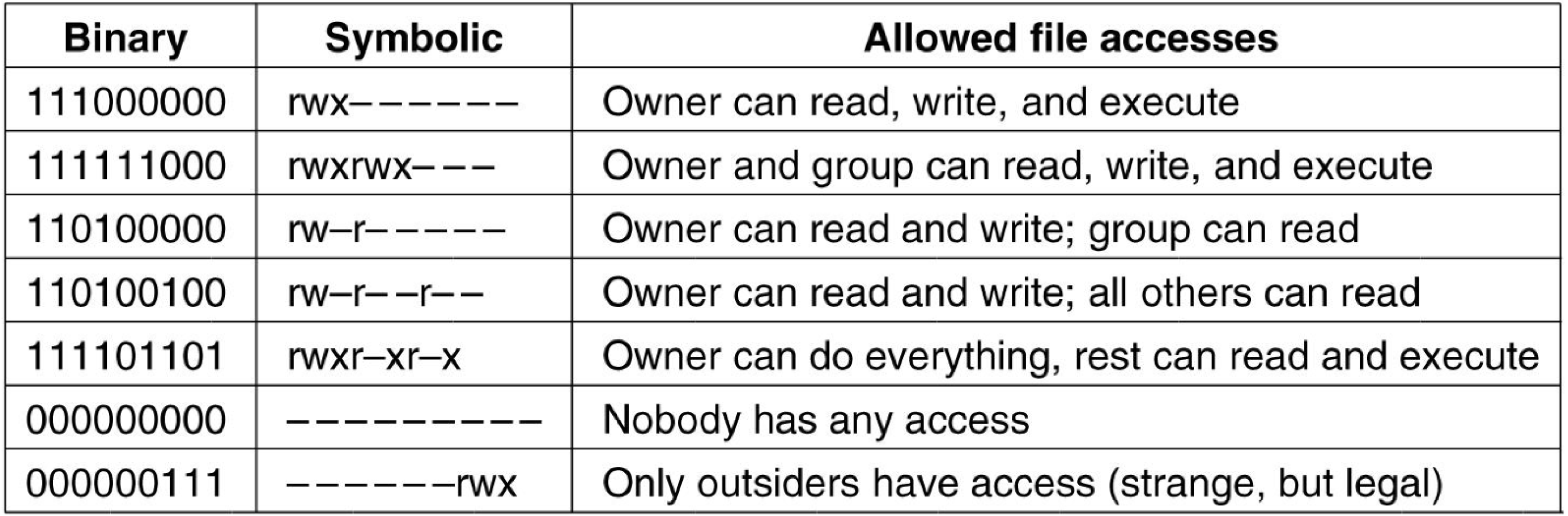
- One can easily view the permissions for a file by invoking a long format listing using the command `ls -l`.

```
drwxr-xr-x  5 nimals pg1493352  4096 Dec  8 09:19 blog
-rw-r--r--  1 nimals pg1493352 18182 Jul 29 06:18 index.html
```

- The permissions are listed at the start of the line, in groups of rwx.
- This first set of symbols define owner access, the next set define group access, and the last set defines access for all other users.

Access Permissions

- Some example file protection modes.



Binary	Symbolic	Allowed file accesses
111000000	rwx––––––	Owner can read, write, and execute
111111000	rwxrwx–––	Owner and group can read, write, and execute
110100000	rw–r–––––	Owner can read and write; group can read
110100100	rw–r––r––	Owner can read and write; all others can read
111101101	rwxr–xr–x	Owner can do everything, rest can read and execute
000000000	–––––––––	Nobody has any access
000000111	––––––rwx	Only outsiders have access (strange, but legal)

Processes

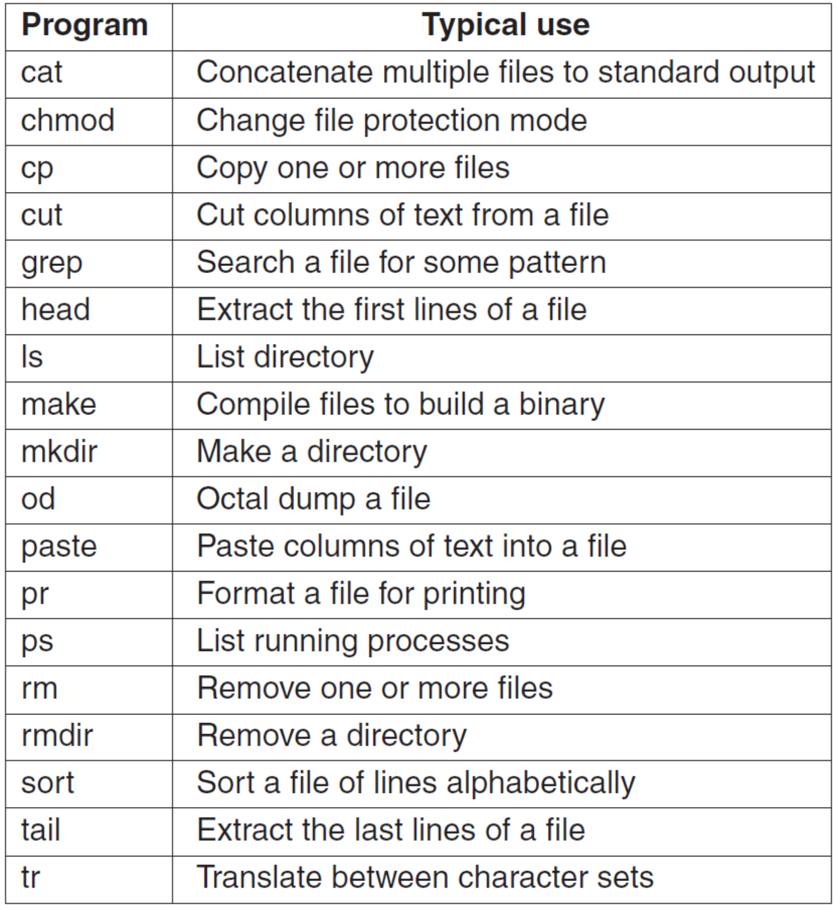
- A process is a program that is currently executing
  - Can be created and destroyed
  - Has resources allocated to it and has an environment associated with it:
    - Process and process group IDs
    - Open files
    - Working directory
    - File creation mask
    - Real and effective user and group IDs
    - Resource limits: maximum file size, maximum amount of memory
    - Signal action settings
    - A set of named variables
  - Can create other processes
  - Can communicate with other processes

Controlling Processes

- Creating a process
  - Running jobs in the foreground: **`command`**
  - Running jobs in the background: **`command &`**
- Obtaining process status
  - **`jobs`** - Displays status of jobs in the current session. *Job number, job status, PID*
  - **`ps`** - Shows current status of processes. *PID, state, accumulated execution time, command, ……*
- Controlling and managing jobs
  - Placing a job in the foreground: **`fg`**
  - Restarting a job in the background: **`bg`**
  - Stopping a process: Ctrl/C, **`kill`**
  - Setting process priority: **`nice`**
  - Scheduling jobs to run at appropriate times: **`at`**, **`crontab`**

The Shell

- Although Linux systems have a graphical user interface, most programmers and sophisticated users still prefer a command-line interface, called the **shell**.
- The shell command-line interface is much faster to use, more powerful and easily extensible.
- The command-line (shell) user interface to Linux consists of a large number of standard utility programs.



Program	Typical use
cat	Concatenate multiple files to standard output
chmod	Change file protection mode
cp	Copy one or more files
cut	Cut columns of text from a file
grep	Search a file for some pattern
head	Extract the first lines of a file
ls	List directory
make	Compile files to build a binary
mkdir	Make a directory
od	Octal dump a file
paste	Paste columns of text into a file
pr	Format a file for printing
ps	List running processes
rm	Remove one or more files
rmdir	Remove a directory
sort	Sort a file of lines alphabetically
tail	Extract the last lines of a file
tr	Translate between character sets

Linux Utility Programs

- Categories of utility programs:
  - File and directory manipulation commands.
  - Filters.
  - Program development tools, such as editors and compilers.
  - Text processing.
  - System administration.
  - Miscellaneous

Navigating across Directories

- **`pwd`** – Prints present working directory
- **`cd`** – Change present working directory
- **`pwd`** – Prints **`/home/nimal`**
- **`cd class`** – Changes the directory to class (/home/nimal/class)
- **`cd ..`** – Change to parent directory (/home/nimal)
- **`cd /home`** – Change to absolute path (/home)

Listing the Content of a Directory

- **`ls`** is used to list the contents of a directory.
- If the command **`ls`** is written with parameter **`-l`** then the command lists contents of the working directory with details.
  - **`ls -l`**
- If the command **`ls`** is written with parameter **`-a`** then the command lists all contents including system or hidden content.
  - **`ls -a`**
- If needed multiple parameters for a command can be combined as:
  - **`ls -al`**

Make Directory

- The command **`mkdir`** makes new directory as a subdirectory of the current directory.
- The path is given relative to the current directory.
  - **`mkdir newdir`** – Creates as subdirectory newdir
  - **`mkdir ../newdir`** – Creates newdir in the parent directory
  - **`mkdir /home/nimal/newdir`** – Creates newdir at the given path
- The command **`rmdir`** removes directory if it is empty.
  - **`rmdir newdir`**

Copy File, Move/Rename File

- The command **`cp file_1 file_2`** copies file_1 to file_2.
  - Here both files must be in the same working directory. If they are in various directories, the path must be given.
  - **`cp file_1 /home/nimal/file_2`**

- The command **`mv file_1 file_2`** moves file_1 to file_2.
  - Here both files must be in the same working directory. If they are in various directories, the path must be given.
  - **`mv file_1 /home/nimal/file_2`**
  - The file_1 is removed from the disk.

Remove File

- The command **`rm file_1`** removes the file_1 from the system
- If you use wildcard, for example **`rm h*c`** you will remove all files beginning with h and ending with c which are in working directory.
- If you write **`rm *`** you will erase all files from your working directory.

Access Permissions for File/Directory

- The ownership of the file or directory can be changed using:
  - **`chown <owner> <file/directory name>`**
- The group of the file or directory can be changed using:
  - **`chgrp <group> <file/directory name>`**
- The permissions of the file can be changed using chmod command
  - **`chmod ### <filename or directory>`**
  - The #'s can be:

0 = Nothing	1 = Execute	2 = Write
3 = Execute & Write (2 + 1)	4 = Read	5 = Execute & Read (4 + 1)
6 = Read & Write (4 + 2)	7 = Execute & Read & Write (4 + 2 + 1)	

Based on Tanenbaum, Modern Operating Systems 4 e &
Navpreet Singh, IITK

Further Learning

- Interactive crash course:
  - https://linuxsurvival.com/
- EdX course:
  - https://www.edx.org/course/introduction-to-linux
- The Unix Shell:
  - https://swcarpentry.github.io/shell-novice/
- Bash Scripting:
  - https://ryanstutorials.net/bash-scripting-tutorial/
- Linux Command Manuals
  - https://www.kernel.org/doc/man-pages/

Discussion

- General Discussion
- Class Feedback
- Final Exam