

Introducción al Sistema Operativo LINUX

Pedro Corcuera

Dpto. Matemática Aplicada y
Ciencias de la Computación

Universidad de Cantabria

corcuerp@unican.es

Index

Introduction

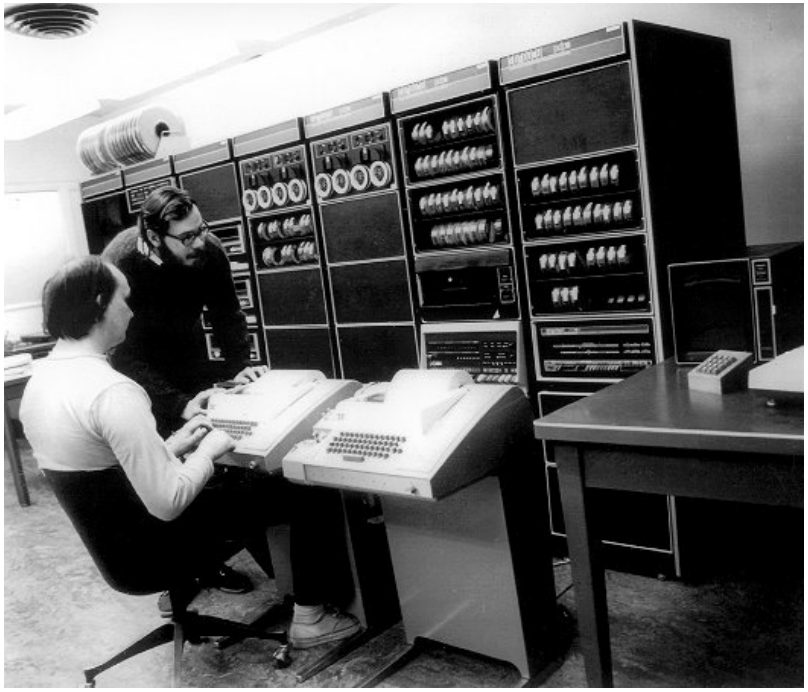
- **Linux** is a free **open source** computer operating system initially developed for Intel x86-based personal computers. It has been subsequently ported to many other hardware platforms.
- Linux borrows heavily from the **UNIX** operating system because it was written to be a free and open source version of UNIX.
- Files are stored in a hierarchical filesystem, with the top node of the system being **root** or simply "/".

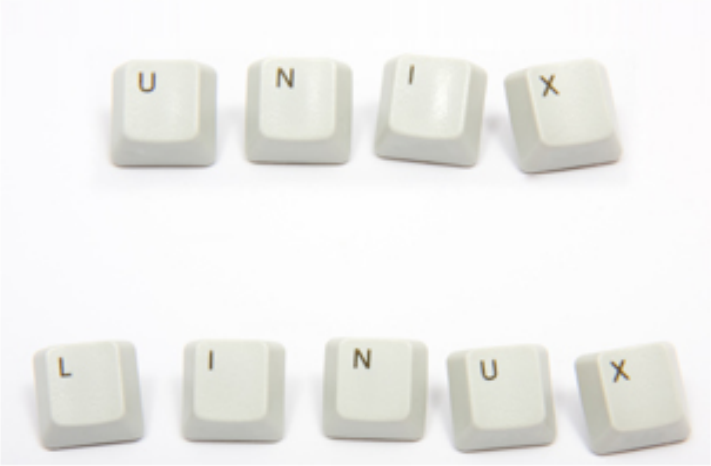
Introduction

- Whenever possible, Linux makes its components available via **files** or objects that look like files. Processes, devices, and network sockets are all represented by file-like objects, and can often be worked with using the same utilities used for regular files.
- Linux is a fully **multitasking** (a method where multiple tasks are performed during the same period of time), **multiuser** operating system, with built-in networking and service processes known as **daemons** in the UNIX world.

Before Linux: UNIX

- **Unix** (mother/father of all operating systems) is a family of multitasking, multiuser computer operating systems developed (1970s at the Bell Labs) by Ken Thompson, Dennis Ritchie, and others.





History of Linux

- **Linux** began in 1991 as a personal project by Finnish student **Linus Torvalds**: to create a new free operating system kernel.
- In 1992, Linux was re-licensed using the General Public License (GPL) by GNU which made it possible to build a worldwide community of developers.
- The Linux distributions created in the mid-90's provided the basis for fully free computing and became a driving force in the open source software movement.
- In 1998, companies like IBM and Oracle announced support for the Linux platform and began major development efforts.

History

Mid 90's — Created for fully free computing and for open source software development

1998 — Major companies like IBM and Oracle announced support

Today — Linux powers more than half of the servers on Internet

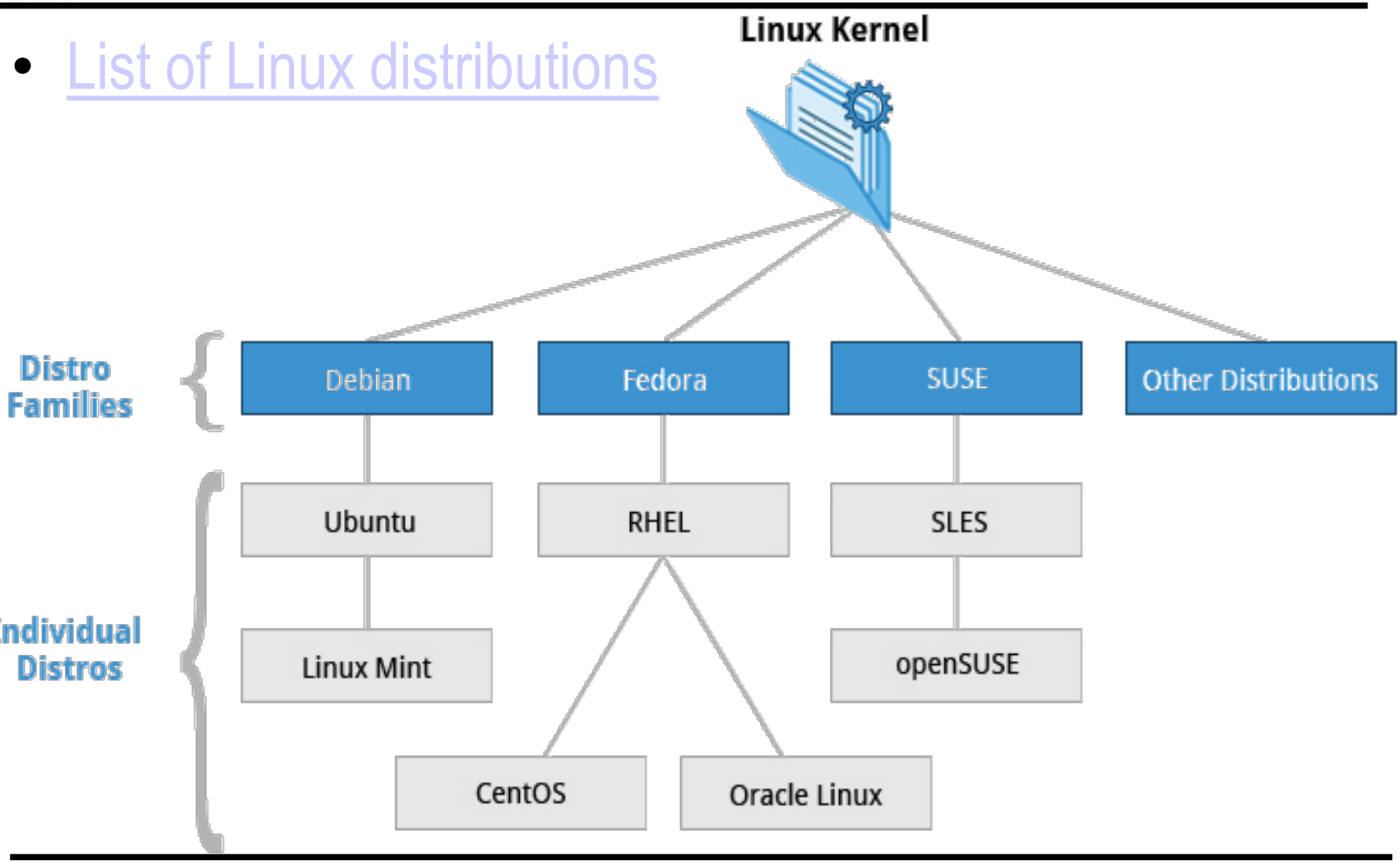
Majority of smart-phones (Via Android)

Linux Distribution Families

- Linux Foundation is a nonprofit consortium dedicated to fostering the growth of Linux.
- The families and representative distributions are:
  1. Debian Family Systems (such as Ubuntu)
  2. SUSE Family Systems (such as openSUSE)
  3. Fedora Family Systems (such as CentOS)

Linux Distribution Families

- List of Linux distributions



Debian Family - Ubuntu

- The Debian family is upstream for **Ubuntu**, and Ubuntu is upstream for Linux Mint and others. It is commonly used on both servers and desktop computers. Debian is a pure open source project and focuses on one key aspect, that is, stability.
- Ubuntu aims at providing a good compromise between long term stability and ease of use. Ubuntu has access to a very large software repository. Ubuntu is a registered trademark of Canonical Ltd.
- Ubuntu uses the **GNOME** graphical interface.
- Ubuntu has been widely used for cloud deployments.

Linux Terminology

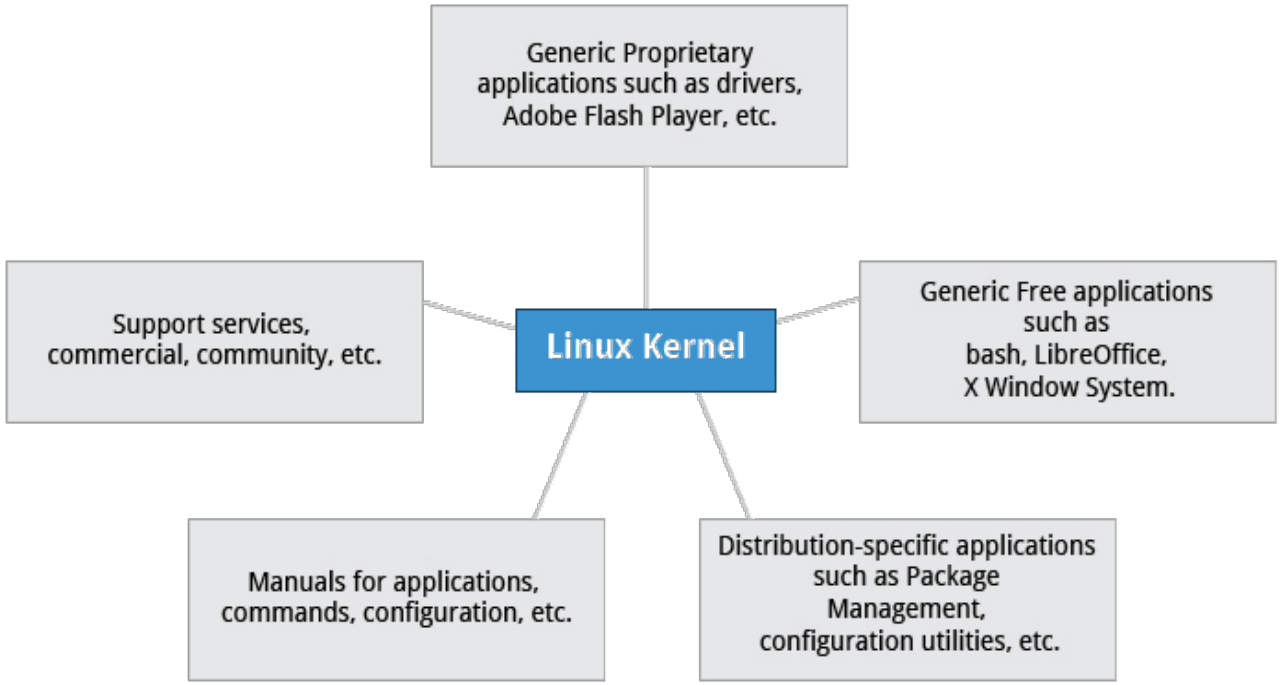
- **Kernel** The glue between hardware and applications.
  - Linux kernel
- **Distribution** Collection of software making up a Linux-based OS.
  - Red Hat Enterprise Linux, Fedora, Ubuntu, Gentoo
- **Boot loader** Program that boots the operating system
  - GRUB and ISOLINUX
- **Service** Program that runs as a background process.
  - httpd, nfsd, ntpd, ftpd and named

Linux Terminology

- **Filesystem** Method for storing and organizing files
  - Ext4, FAT, XFS, NTFS, Btrfs, ReiserFS
- **X Window system** Graphical subsystem of Linux system
- **Desktop environment** Graphical user interface on top of the operating system
  - GNOME, KDE, Xfce, Fluxbox
- **Command line** Interface for typing commands for the operating system to execute
- **Shell** Command line interpreter. Interprets and instruct the OS to perform any necessary tasks and commands.
  - Bash, ksh, csh, tcsh, zsh

Linux distributions

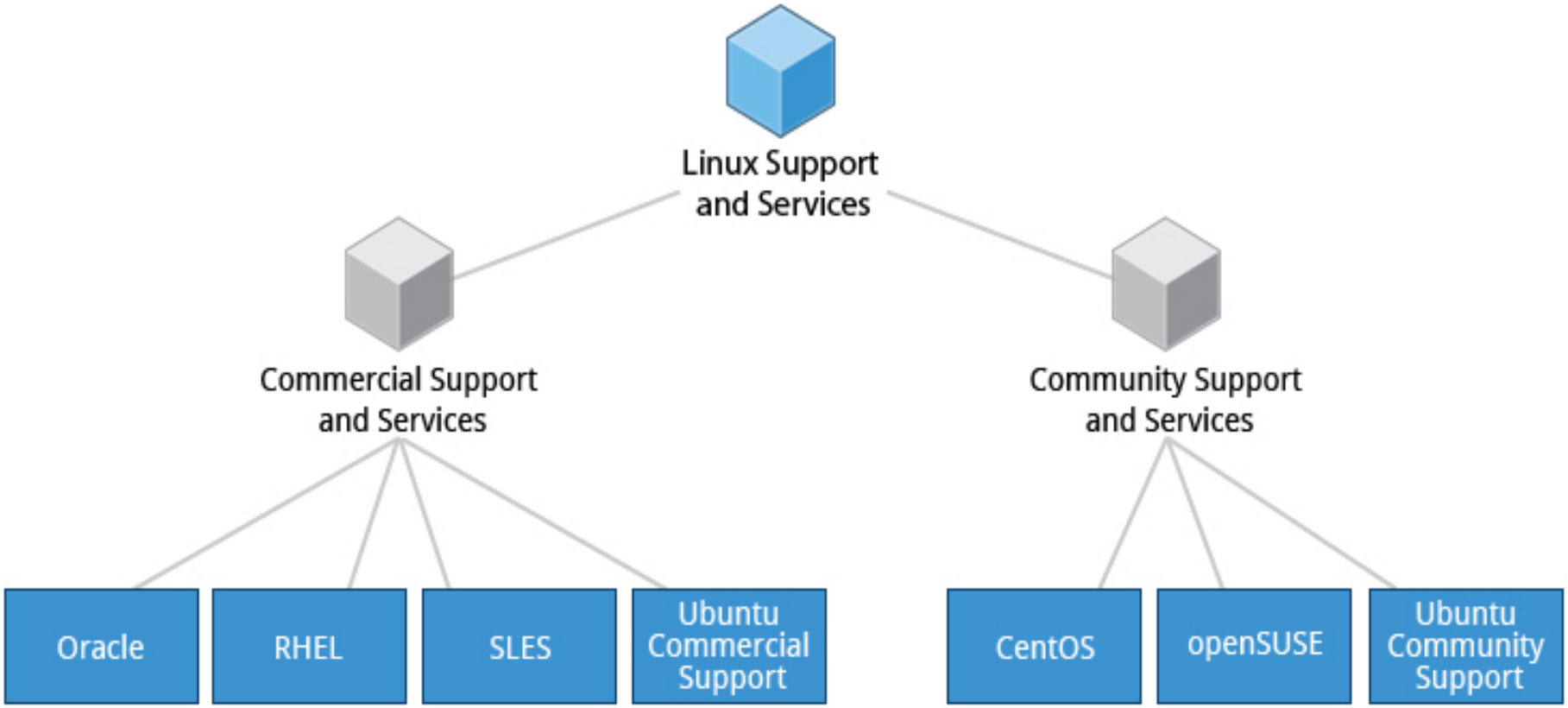
- A full **Linux distribution** consists of the kernel plus a number of other software tools for file-related operations, user management, and software package management. Each of these tools provides a small part of the complete system. Each tool is often its own separate project, with its own developers working to perfect that piece of the system.



Linux distributions

- The current Linux kernel, along with past Linux kernels, can be found at the www.kernel.org web site. The various Linux distributions may be based on different kernel versions.
- Examples of other essential tools and ingredients provided by distributions include: the C/C++ compiler, the gdb debugger, the core system libraries, the low-level interface for drawing graphics on the screen as well as the higher-level desktop environment, and the system for installing and updating the various components including the kernel itself.

Services Associated with Distributions



Linux Filesystems

- Different Types of **Filesystems** Supported by Linux:
  - Conventional disk filesystems: ext2, ext3, ext4, XFS, Btrfs, JFS, NTFS, etc.
  - Flash storage filesystems: ubifs, JFFS2, YAFFS, etc.
  - Database filesystems
  - Special purpose filesystems: procfs, sysfs, tmpfs, debugfs, etc.

Partitions and Filesystems

- A **partition** is a logical part of the disk, whereas a **filesystem** is a method of storing/finding files on a hard disk (usually in a partition).
- Analogy: filesystems are as being like family trees that show descendants and their relationships, while the partitions are like different families (each of which has its own tree).

Partitions and Filesystems

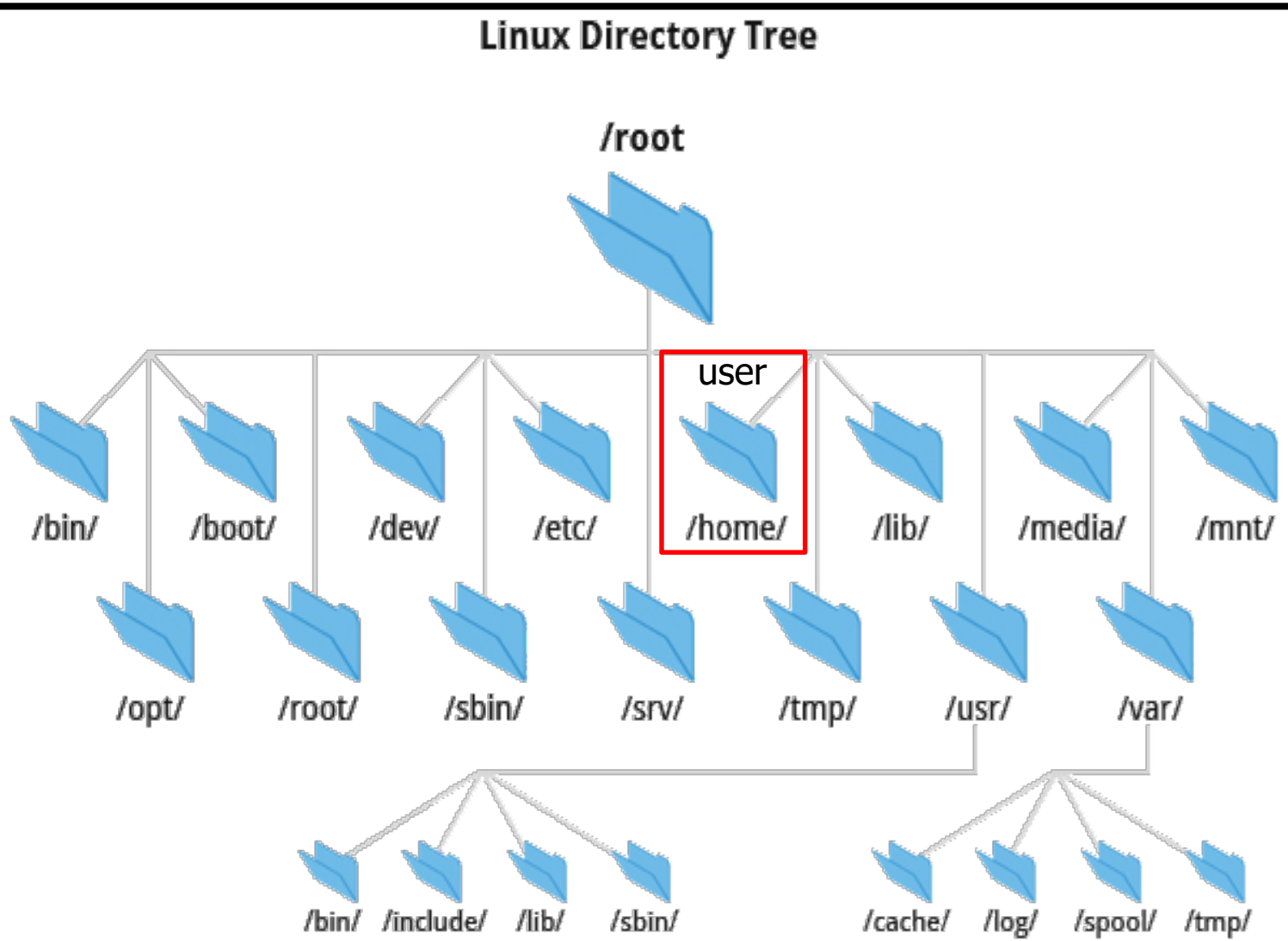
- A comparison between filesystems in Windows and Linux is given in the following table:

	Windows	Linux
Partition	Disk1	/dev/sda1
Filesystem type	NTFS/FAT32	EXT3/EXT4/XFS...
Mounting Parameters	DriveLetter	MountPoint
Base Folder where OS is stored	C drive	/

The Filesystem Hierarchy Standard

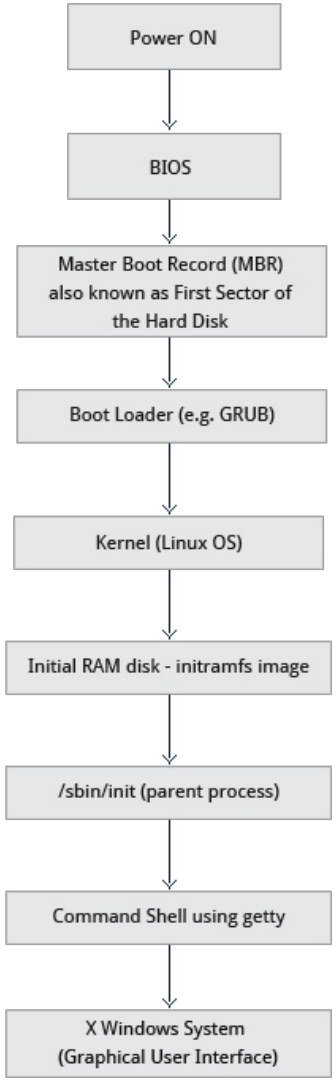
- Linux systems store their important files according to a standard layout called the Filesystem Hierarchy Standard, or **FHS**.
- Linux uses the '*/*' character to separate paths (unlike Windows, which uses '\'), and does not have drive letters. New drives are mounted as directories in the single filesystem, often under /media.
- All Linux filesystem names are **case-sensitive**, so /boot, /Boot, and /BOOT represent three different directories (or folders).
- Many distributions distinguish between core utilities needed for proper system operation and other programs, and place the latter in directories under /usr (think "user").
  - To get a sense for how the other programs are organized, find the /usr directory and compare the subdirectories with those that exist directly under the system root directory (/).

The Filesystem Hierarchy Standard



The Boot Process

- The Linux **boot process** is the procedure for initializing the system. It consists of everything that happens from when the computer power is first switched on until the user interface is fully operational.



Command Line Interface (CLI)

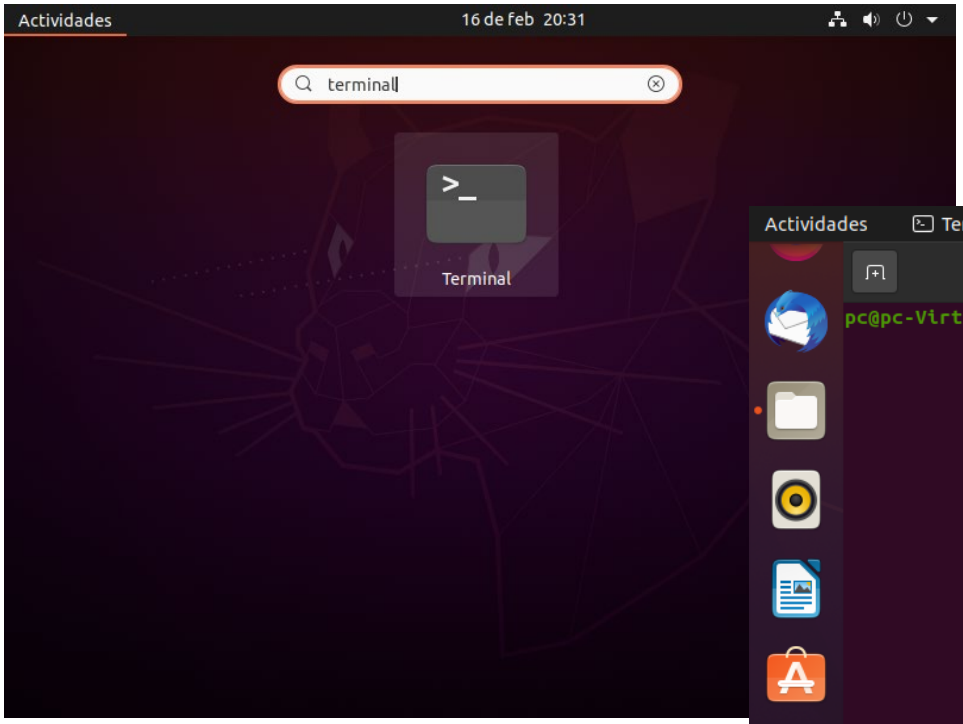
- Open the Terminal Emulator which is often referred to as the **Terminal** for simplicity
- A Terminal Emulator is a program that emulates (mimics) a physical Terminal (Console). The Terminal interacts with the Shell (the Command Line Interface).
- The **Shell** is a command-line interpreter, that is to say, it is a program that processes and executes commands.

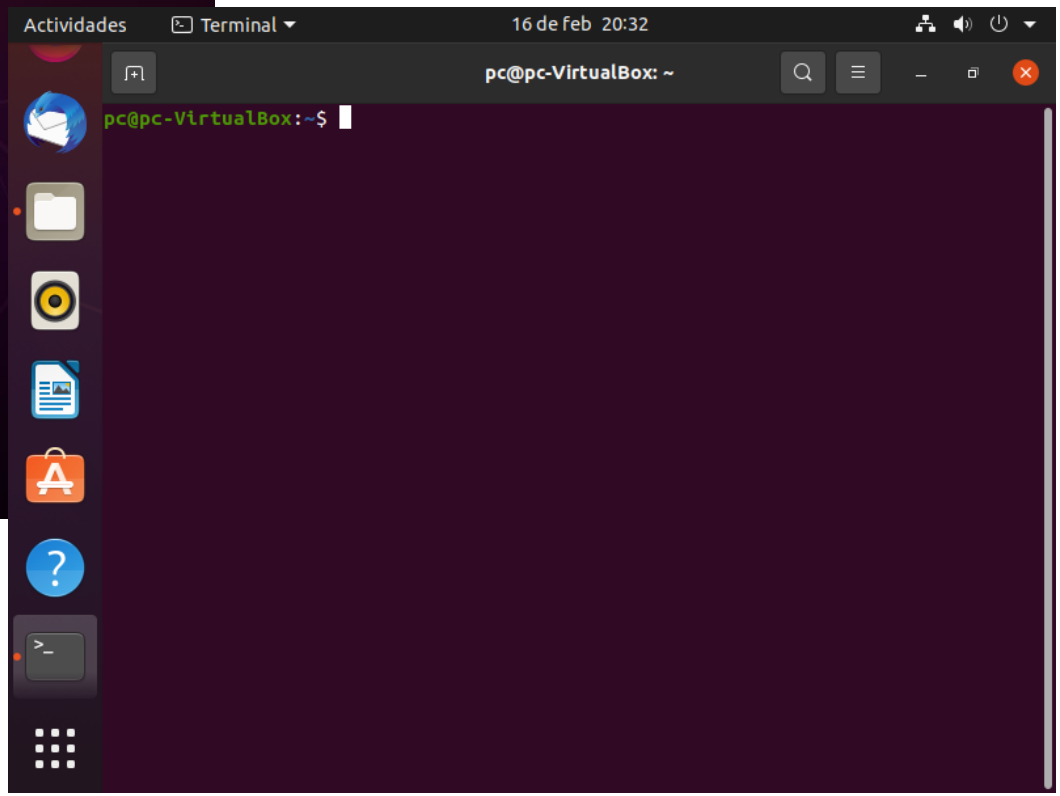
Command Line Operations

- To open a terminal in Ubuntu type **terminal** in the Search box. Inside the Command Line
  - Most input lines entered at the **shell prompt** have three basic elements:
    - Command
    - Options
    - Arguments
- The **command** is the name of the program you are executing. It may be followed by one or more **options** (or switches) that modify what the command may do. Options usually start with one or two dashes, for example,-p or--print, in order to differentiate them from **arguments**, which represent what the command operates on. (Summary Unix Basics)

Command Line Interface (CLI)

Press Ctrl + Alt + T to open terminal





Most tasks can be done faster from the terminal than from the GUI

Graphical Interface

- To display the value of the current desktop:

```
$ echo $XDG_CURRENT_DESKTOP
```

- **GNOME** Desktop Environment.
  - Is a popular desktop environment with an easy to use graphical user interface and menu-based navigation. It is bundled as the default desktop environment for many distributions (Ubuntu).

Default Directories in Ubuntu

- To access your home directory from the graphical user interface in GNOME on Ubuntu, click the **File Cabinet icon** on the left of the screen.
- The File Manager (Files in the case of Ubuntu) will open a window with your Home directory displayed. The left panel of the File Manager window holds a list of commonly used directories, such as Computer, Home, Desktop, Documents, Downloads, and Trash.

Editing a File - gedit

- To edit any text file through the graphical interface in the GNOME desktop environment, simply double-click the file on the desktop window to open the file with the default text editor.
- The default text editor in GNOME is **gedit**. It is simple yet powerful, ideal for editing documents, making quick notes, and programming. Although gedit is designed as a general purpose text editor, it offers additional features for spell checking, highlighting, file listings, and statistics.

System Configuration from the Graphical Interface

- System Settings panel allows you to control most of the basic configuration options and desktop settings.
  - e.g. specifying the screen resolution, managing network connections, or changing the date and time of the system.
- Network Configuration
  - The Network Manager utility allow the choice of a wired, wireless or mobile broadband network, handle passwords, and set up Virtual Private Networks (VPNs).
- Installing and Updating Software
  - The higher-level package management system is the **apt** (Advanced Package Tool) system of utilities. apt-get

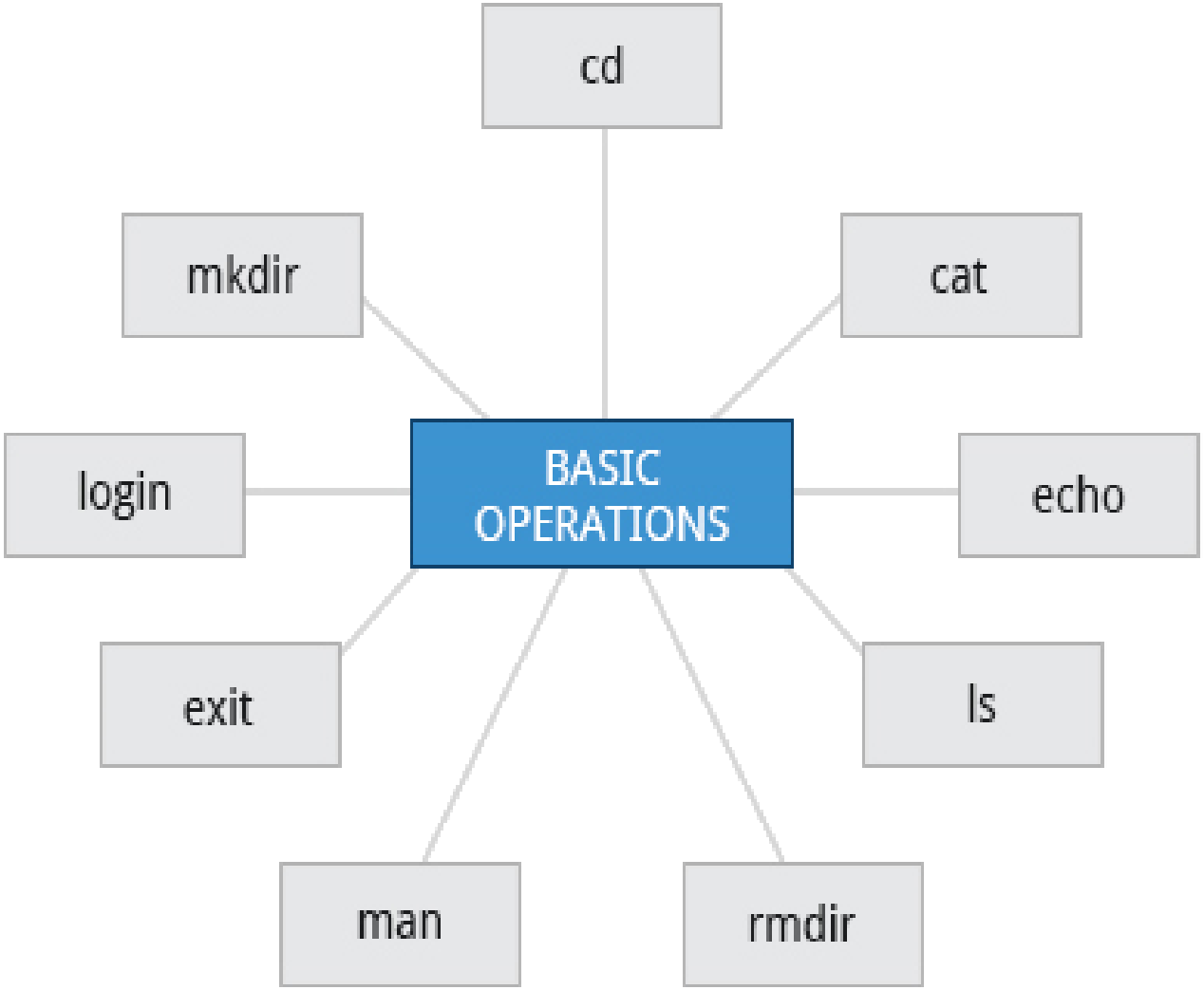
Linux Documentation Sources

- The **man** pages are the most often-used source of Linux documentation (Linux man pages online). They provide in-depth documentation about many programs and utilities as well as other topics, including configuration files, system calls, library routines, and the kernel.
  - man –f generates the same result as typing whatis.
  - man –k generates the same result as typing apropos.
- Typing **info** with no arguments in a terminal window displays an index of available topics.
- The third source of Linux documentation is use of the **help** command or **--help** option.

sudo

- **sudo** provides the user with administrative (admin) privileges when required.
- sudo allows users to run programs using the security privileges of another user, generally root (**superuser**). The functionality of sudo is similar to that of **run as** in Windows.

Basic Operations

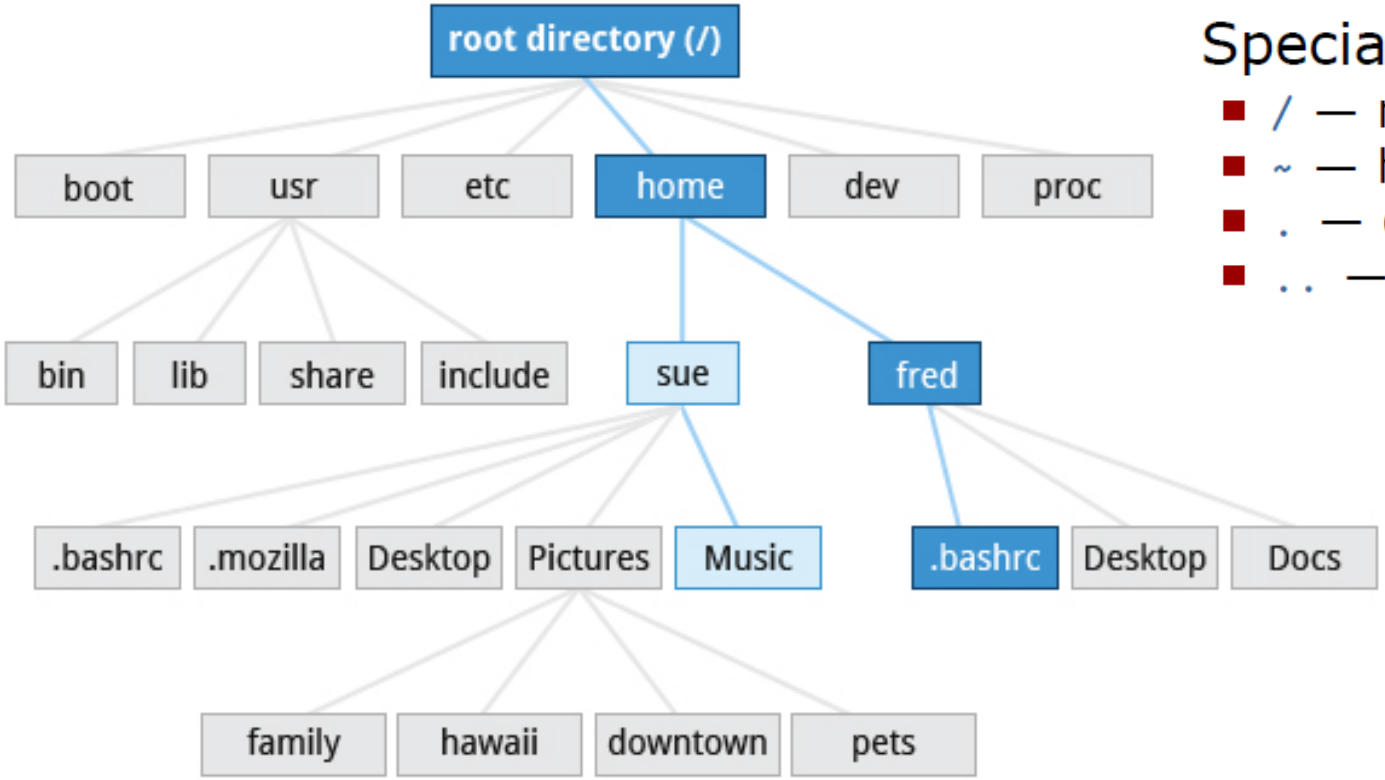


Understanding Absolute and Relative Paths

- There are two ways to identify paths:
  - **Absolute pathname**: An absolute pathname begins with the root directory and follows the tree, branch by branch, until it reaches the desired directory or file. Absolute paths always start with /.
  - **Relative pathname**: A relative pathname starts from the present working directory. Relative paths never start with /.
- Most of the time it is most convenient to use relative paths, which require less typing. Usually you take advantage of the shortcuts provided by: . (present directory), .. (parent directory) and ~ (your home directory).

Understanding Absolute and Relative Paths - examples

- Absolute pathname method: $ cd /usr/bin
- Relative pathname method: $ cd ../../usr/bin



Special folders:

- / — root folder
- ~ — home folder
- . — current folder
- .. — parent folder

Exploring the Filesystem

- Traversing up and down the filesystem tree can get tedious. The **tree** command is a good way to get a bird's-eye view of the filesystem tree. Use tree -d to view just the directories and to suppress listing file names.

Directories under root (/)

/	This is the root of your filesystem, where everything begins.
/etc	This directory contains system configuration files.
/home	This is the default home directory for all users (except the root user).
/root	This is the home directory for the root user.
/dev	This is where your devices such as your hard disks, USB drives, and optical drives reside on your system.
/opt	This is where you can install additional 3rd party software.
/bin	This is where essential binaries (programs) reside on your system.
/sbin	This is where system binaries (programs) that are typically used by the system administrator are stored.
/tmp	This is where temporary files are stored; they are usually deleted after a system reboot, so never store important files here!
/var	This directory contains files that may change in size, such as mail spools and log files.
/boot	All the files required for your system to boot are stored here.
/lib	This directory contains libraries needed by the essential binaries in the /bin and /sbin directories. A library is basically a set of precompiled functions that can be used by a program.
/proc	This is where information about running processes is stored.
/usr	This directory contains files and utilities that are shared between users.

Hard and Symbolic Links

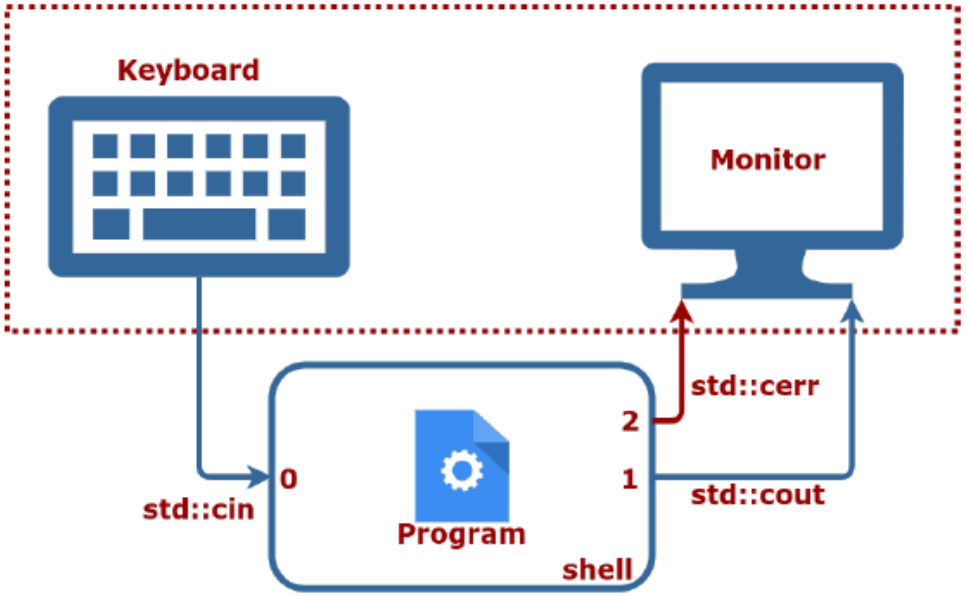
- **ln** can be used to create **hard links** and (with the -s option) **soft links**, also known as symbolic links. Symbolic links take no extra space on the filesystem
- Symbolic (or Soft) links are created with the -s option as in:
- $ ln -s file1 file4

Standard File Streams

- By default there are three standard file streams (or descriptors) always open for use: standard input (standard in or **stdin**), standard output (standard out or **stdout**) and standard error (or **stderr**). Usually, stdin is your keyboard, stdout and stderr are printed on your terminal; often stderr is redirected to an error logging file.
- stdin is often supplied by directing input to come from a file or from the output of a previous command through a pipe. stdout is also often redirected into a file. Since stderr is where error messages are written, often nothing will go there.

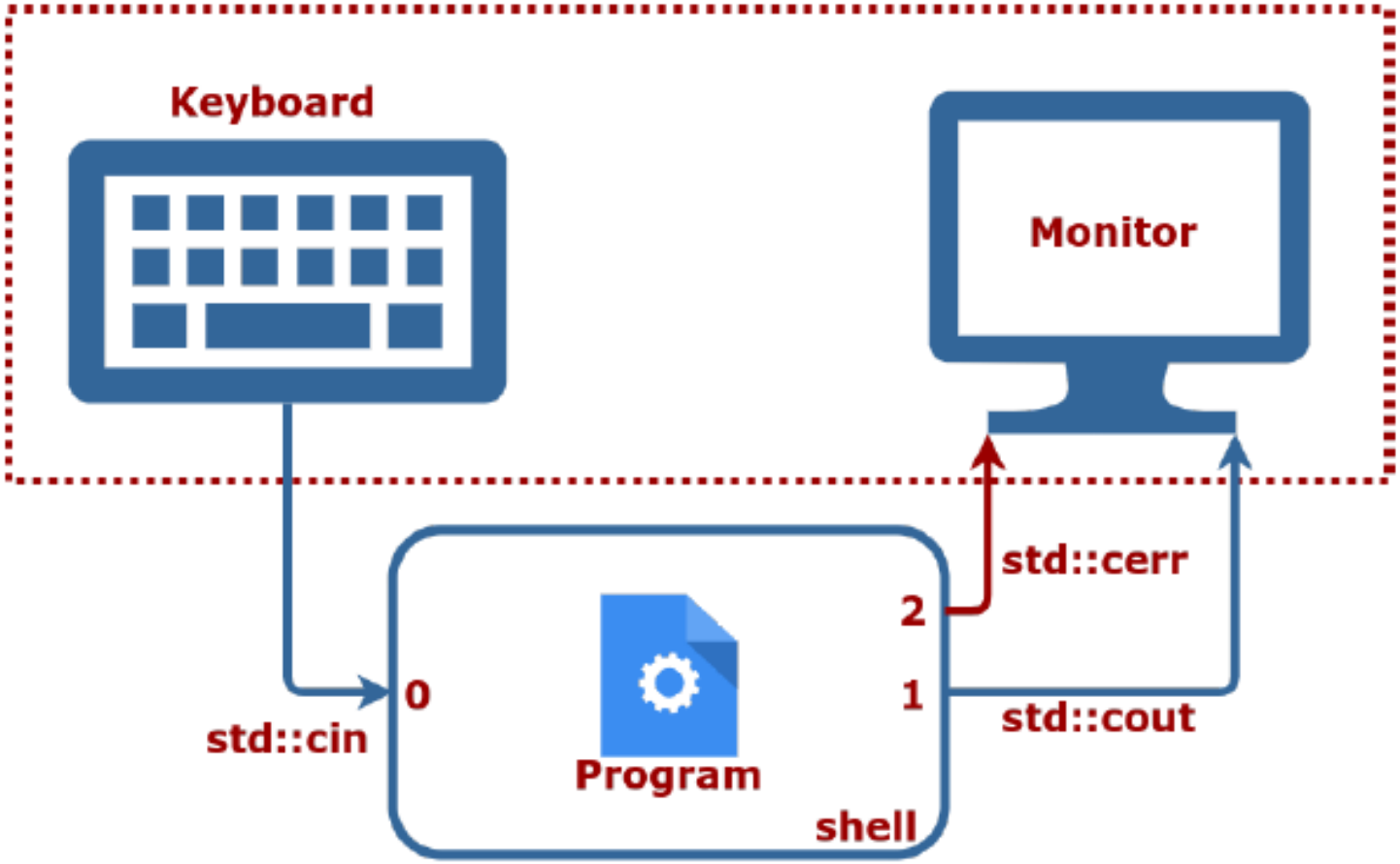
Standard input/output channels

- Single input channel:
  - stdin: Standard input: channel 0
- Two output channels:
  - stdout: Standard output: channel 1
  - stderr: Standard error output: channel 2



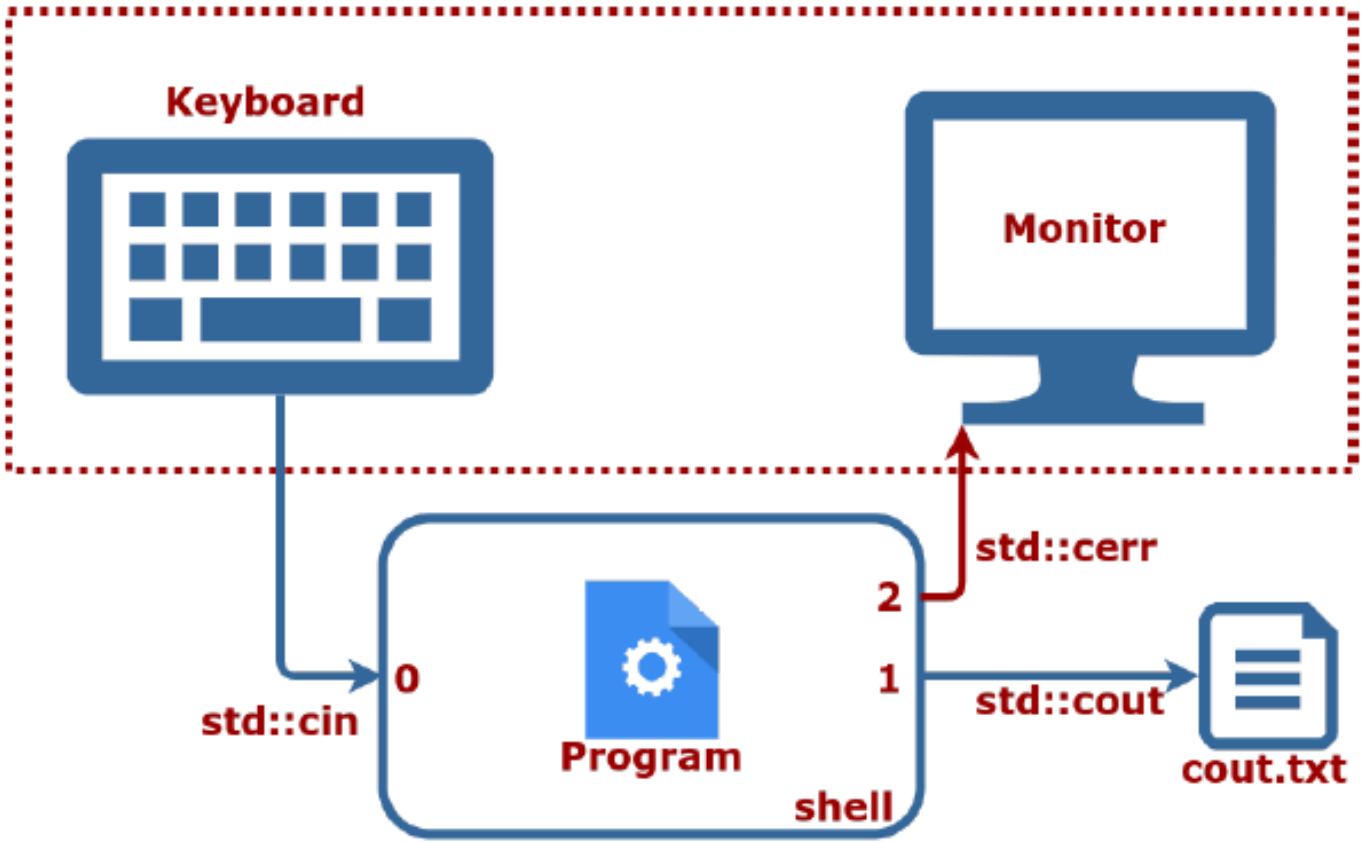
Standard input/output channels

$ program



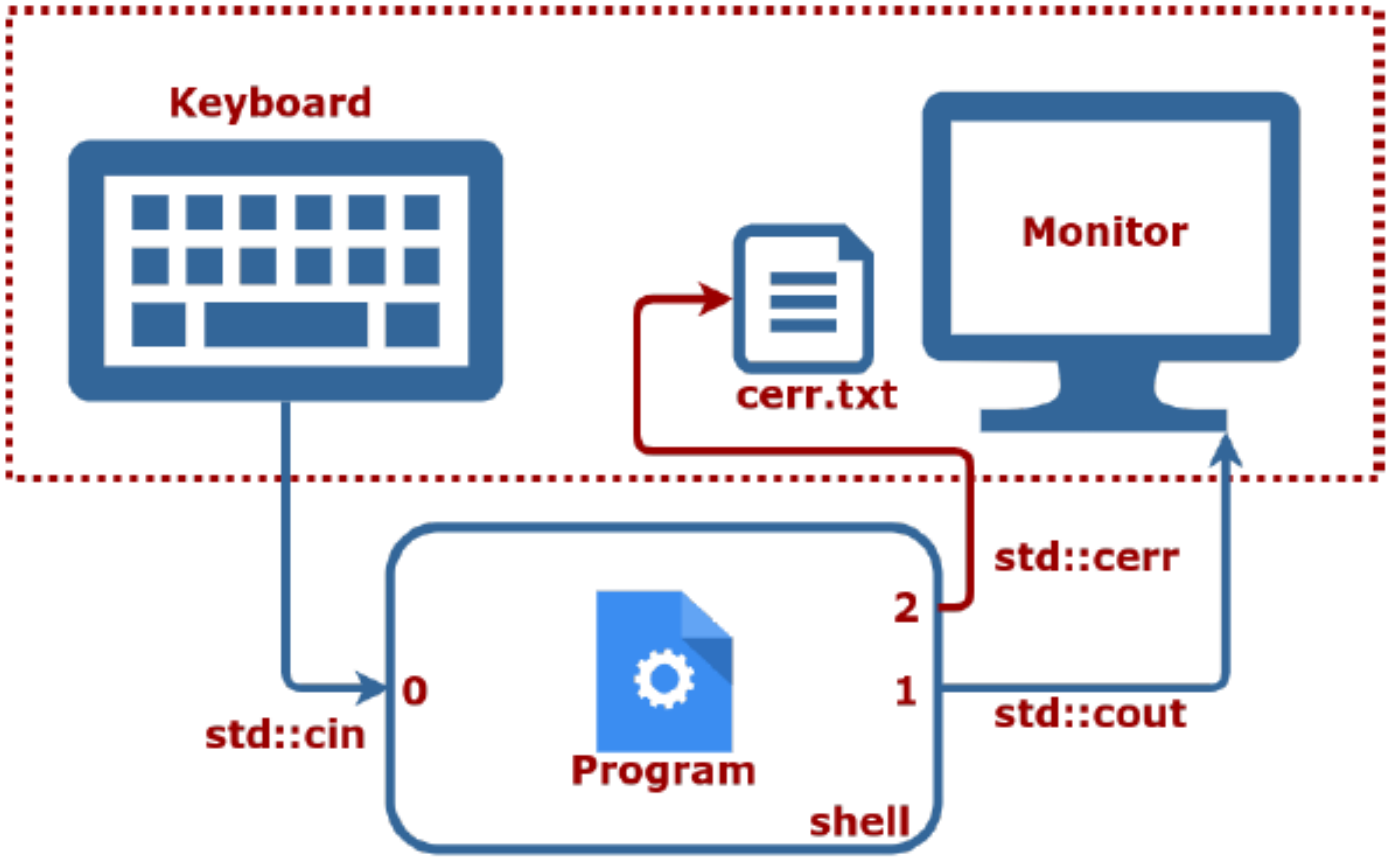
Redirecting stdout

$ program 1> cout.txt



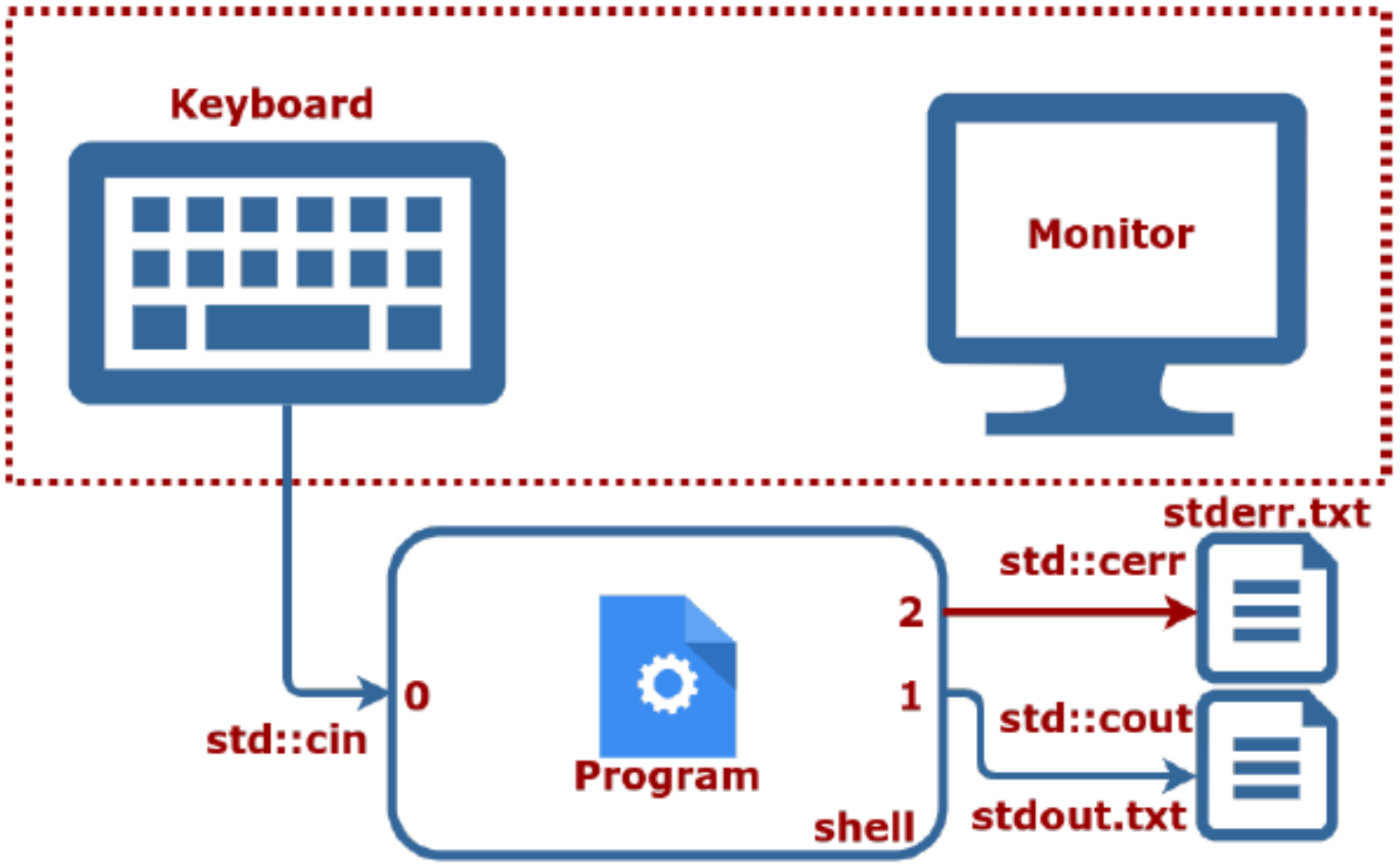
Redirecting stderr

$ program 2> cout.txt



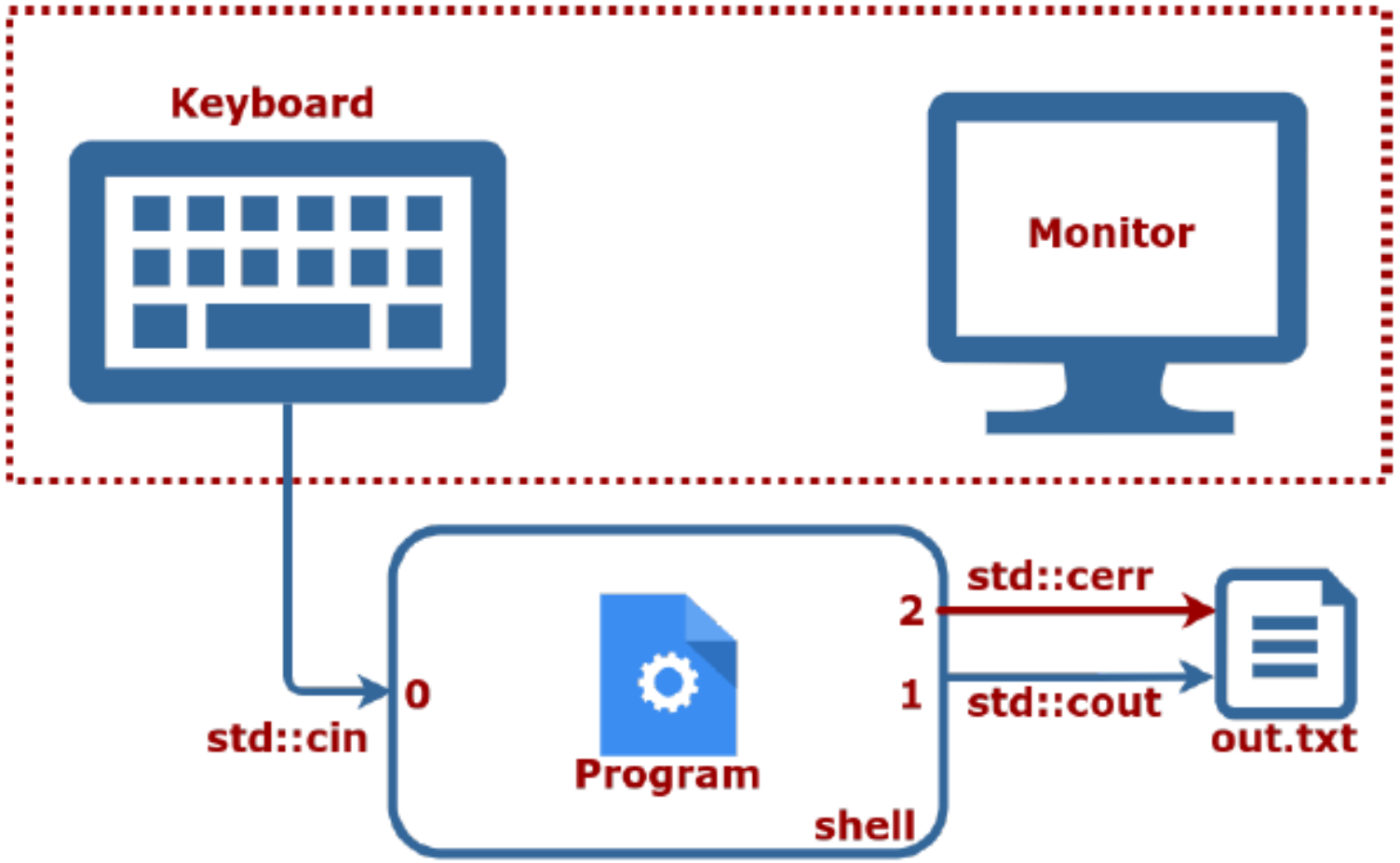
Redirecting stdout and stderr

$ program 1>stdout.txt 2>stderr.txt



Redirecting stdout and stderr

$ program 1>out.txt 2>&1



I/O Redirection

- Through the command shell we can redirect the three standard filestreams so that we can get input from either a file or another command instead of from our keyboard, and we can write output and errors to files or send them as input for subsequent commands.

  $ do_something **<** input-file

  $ do_something **>** output-file

Output Redirection

- Most operating systems accept input from the keyboard and display the output on the terminal. However, in shell scripting you can send the output to a file (called output **redirection**).
- The **>** character is used to write output to a file. For example, the following command sends the output of free to the file /tmp/free.out:

```
$ free > /tmp/free.out
```

- To check the contents of the `/tmp/free.out` file, at the command prompt type `cat /tmp/free.out`.
- Two > characters (**>>**) will append output to a file if it exists, and act just like > if the file does not already exist.

Input Redirection

- Just as the output can be redirected to a file, the input of a command can be read from a file. The process of reading input from a file is called input redirection and uses the **<** character. If you execute the following command:

```
wc -l < /temp/free.out
```

it will count the number of lines from the /temp/free.out file and display the results.

Pipes

- The UNIX/Linux philosophy is to have many simple and short programs (or commands) cooperate together to produce quite complex results, rather than have one complex program with many possible options and modes of operation. In order to accomplish this, extensive use of pipes is made; you can pipe the output of one command or program into another as its input.
- In order to do this we use the vertical-bar, |, (pipe symbol) between commands as in:
- $ command1 | command2 | command3

Chaining commands

- Calls commands one after another:
  `command1; command2; command3`
- Same as above but fails if any of the commands returns an error code:
  `command1 && command2 && command3`
- Pipe stdout of command1 to stdin of command2 and stdout of command2 to stdin of command3:
  `command1 | command2 | command3`
- Piping commonly used with grep:
  `ls | grep smth` look for smth in output of ls

Wildcards and Matching File Names

- To search for files using the **?** wildcard, replace each unknown character with ?
- To search for files using the * wildcard, replace the unknown string with *

Wildcard	Result
?	Matches any single character
*	Matches any string of characters
[set]	Matches any character in the set of characters, for example [adf] will match any occurrence of "a", "d", or "f"
[^set]	Matches any character not in the set of characters

Viewing Files

Command	Usage
cat	Used for viewing files that are not very long; it does not provide any scroll-back.
tac	Used to look at a file backwards, starting with the last line.
less	Used to view larger files because it is a paging program; it pauses at each screenful of text, provides scroll-back capabilities, and lets you search and navigate within the file. Note: Use / to search for a pattern in the forward direction and ? for a pattern in the backward direction.
tail	Used to print the last 10 lines of a file by default. You can change the number of lines by doing -n 15 or just -15 if you wanted to look at the last 15 lines instead of the default.
head	The opposite of tail; by default it prints the first 10 lines of a file.

touch and mkdir

- **touch** is often used to set or update the access, change, and modify times of files. By default it resets a file's time stamp to match the current time.
- **mkdir** is used to create a directory.
  - To create a sample directory named sampdir under the current directory, type mkdir sampdir.
  - To create a sample directory called sampdir under /usr, type mkdir /usr/sampdir.
- Removing a directory is simply done with **rmdir**. The directory must be empty or it will fail. To remove a directory and all of its contents you have to do rm -rf.

Removing a File

Command	Usage
mv	Rename a file
rm	Remove a file
rm –f	Forcefully remove a file
rm –i	Interactively remove a file

Renaming or Removing a Directory

Command	Usage
mv	Rename a directory
rmdir	Remove an empty directory
rm -rf	Forcefully remove a directory recursively

File Operations

- Partitions in Linux
  - Each filesystem resides on a hard disk partition. Partitions help to organize the contents of disks according to the kind of data contained and how it is used.
- Mount Points
  - This is simply a directory (which may or may not be empty) where the filesystem is to be attached (mounted). Sometimes you may need to create the directory if it doesn't already exist.
  - The **mount** command is used to attach a filesystem (which can be local to the computer or, as we shall discuss, on a network) somewhere within the filesystem tree. Arguments include the device node and mount point. For example,

    $ mount /dev/sda5 /home

The /bin directories

- The **/bin** directory contains executable binaries, essential commands used in single-user mode, and essential commands required by all system users, such as:

Command	Usage
ps	Produces a list of processes along with status information for the system.
ls	Produces a listing of the contents of a directory.
cp	Used to copy files.

Compressing Data

- File data is often compressed to save disk space and reduce the time it takes to transmit files over networks.
- Linux uses a number of methods to perform this compression including:

Command	Usage
gzip	The most frequently used Linux compression utility
bzip2	Produces files significantly smaller than those produced by gzip
xz	The most space efficient compression utility used in Linux
zip	Is often required to examine and decompress archives from other operating systems

Archiving and Compressing Data Using tar

- **tar** allows you to create or extract files from an archive file, often called a tarball. At the same time you can optionally compress while creating the archive, and decompress while extracting its contents.
- Here are some examples of the use of tar:

Command	Usage
$ tar xvf mydir.tar	Extract all the files in mydir.tar into the mydir directory
$ tar zcvf mydir.tar.gz mydir	Create the archive and compress with gzip
$ tar jcvf mydir.tar.bz2 mydir	Create the archive and compress with bz2
$ tar Jcvf mydir.tar.xz mydir	Create the archive and compress with xz
$ tar xvf mydir.tar.gz	Extract all the files in mydir.tar.gz into the mydir directory. Note you do not have to tell tar it is in gzip format.

User Environment

- Identifying the Current User
  - To list the currently logged-on users, type **who**
  - To identify the current user, type **whoami**
- Basics of Users and Groups
  - Linux uses groups for organizing users. Groups are collections of accounts with certain shared permissions.
  - All Linux users are assigned a unique user ID (uid), which is just an integer, as well as one or more group ID's (gid), including a default one which is the same as the user ID.

Adding and Removing Users/Groups

- Distributions have straightforward graphical interfaces for creating and removing users and groups and manipulating group membership. However, it is often useful to do it from the command line or from within shell scripts. Only the root user can add and remove users and groups.
- Adding a new user is done with **useradd** and removing an existing user is done with **userdel**.
- Adding a new group is done with **groupadd**.

The root Account

- The root account is very powerful and has full access to the system. Other operating systems often call this the administrator account; in Linux it is often called the superuser account. You must be extremely cautious before granting full root access to a user; it is rarely if ever justified. External attacks often consist of tricks used to elevate to the root account.
- When assigning elevated privileges, you can use the command **su**.
- Granting privileges using **sudo** is less dangerous and is preferred.

Environment Variables

- Environment variables are simply named quantities that have specific values and are understood by the command shell, such as bash. Some of these are pre-set (built-in) by the system, and others are set by the user either at the command line or within startup and other scripts. An environment variable is actually no more than a character string that contains information used by one or more applications.

Environment Variables

- There are a number of ways to view the values of currently set environment variables; one can type **set**, **env**, or **export**. Depending on the state of your system, set may print out many more lines than the other two methods.
- The **history** command recalls a list of previous commands which can be edited and recycled.

Keyboard Shortcuts

Keyboard Shortcut	Task
CTRL-L	Clears the screen
CTRL-D	Exits the current shell
CTRL-Z	Puts the current process into suspended background
CTRL-C	Kills the current process
CTRL-H	Works the same as backspace
CTRL-A	Goes to the beginning of the line
CTRL-W	Deletes the word before the cursor
CTRL-U	Deletes from beginning of line to cursor position
CTRL-E	Goes to the end of the line
Tab	Auto-completes files, directories, and binaries

File Ownership

- In Linux every file is associated with a user who is the **owner**. Every file is also associated with a group which has an interest in the file and certain rights, or **permissions**: read, write, and execute.
- The following utility programs involve user and group ownership and permission setting.

Command	Usage
chown	Used to change user ownership of a file or directory
chgrp	Used to change group ownership
chmod	Used to change the permissions on the file which can be done separately for owner, group and the rest of the world (often named as other.)

File Permission Modes and **chmod**

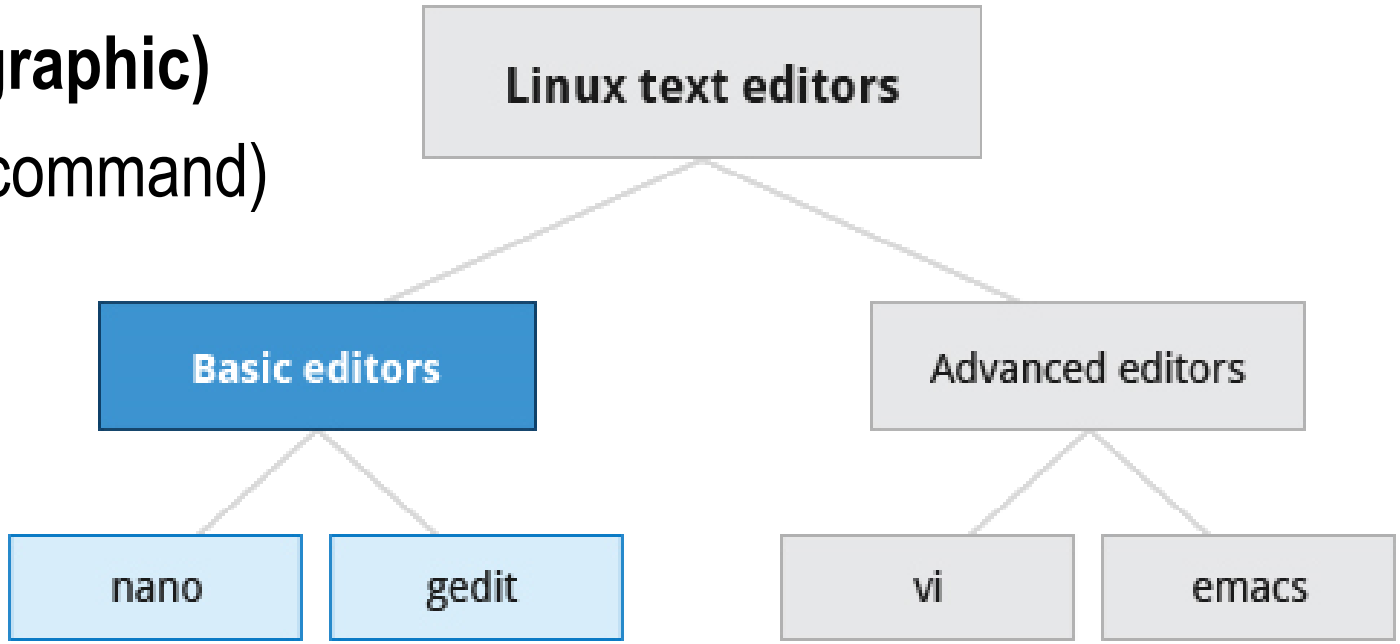
- Files have three kinds of permissions: read (r), write (w), execute (x). These are generally represented as in rwx. These permissions affect three groups of owners: user/owner (u), group (g), and others (o).
- As a result, you have the following three groups of three permissions:

rwx: rwx: rwx

u: g: o

Text Editors

- Linux comes with text editors, there are many choices ranging from quite simple to very complex, including:
  - nano
  - **gedit (graphic)**
  - vi (line command)
  - emacs



Manipulating Text

- Command Line Tools:
  - Display and append to file contents using **cat** and **echo**.
  - Edit and print file contents using sed and awk.
  - Search for patterns using grep.
  - All of the above tools use regular expressions
  - Use multiple other utilities for file and text manipulation.

File Manipulation Utilities

- Linux provides several file manipulation utilities that you can use while working with text files:
  - sort
  - uniq
  - paste
  - join
  - split
- It's common to use this commands with regular expressions and search patterns.

File Manipulation Utilities

- **grep** is extensively used as a primary text searching tool.
- The **tr** utility is used to translate specified characters into other characters or to delete them.
- **tee** takes the output from any command, and while sending it to standard output, it also saves it to a file.
- **wc** (word count) counts the number of lines, words, and characters in a file or list of files.
- **cut** is used for manipulating column-based files and is designed to extract specific columns.

Viewing Return Values

- As a script/command executes, one can check for a specific value or condition and return success or failure as the result. By convention, success is returned as 0, and failure is returned as a non-zero value. An easy way to demonstrate success and failure completion is to execute **ls** on a file that exists and one that doesn't, as shown in the following example, where the return value is stored in the environment variable represented by **$?**:

```
$ ls /etc/passwd
/etc/ passwd
$ echo $?
0
```

In this example, the system is able to locate the file /etc/passwd and returns a value of 0 to indicate success; the return value is always stored in the $? environment variable.

Exporting Variables

- By default, the variables created within a script are available only to the subsequent steps of that script. Any child processes (sub-shells) do not have automatic access to the values of these variables. To make them available to child processes, they must be promoted to environment variables using the **export** statement as in:

```
export VAR=value
```

or

```
VAR=value ; export VAR
```

- While child processes are allowed to modify the value of exported variables, the parent will not see any changes; exported variables are not shared, but only copied.

Redirecting Errors to File and Screen

- In UNIX/Linux, all programs that run are given three open file streams when they are started as listed in the table:

File stream	Description	File Descriptor
stdin	Standard Input, by default the keyboard/terminal for programs run from the command line	0
stdout	Standard output, by default the screen for programs run from the command line	1
stderr	Standard error, where output error messages are shown or saved	2

- Using redirection we can save the stdout and stderr output streams to one file or two separate files for later analysis after a program or command is executed.

Networking

- A network is a group of computers and computing devices connected together through communication channels, such as cables or wireless media.
- IP Addresses
  - Devices attached to a network must have at least one unique network address identifier known as the IP (Internet Protocol) address. The address is essential for routing packets of information through the network.

IPv4 and IPv6

- There are two different types of IP addresses available: IPv4 (version 4) and IPv6 (version 6). IPv4 is older and by far the more widely used, while IPv6 is newer and is designed to get past the limitations of the older standard and furnish many more possible addresses.
- IPv4 uses 32-bits for addresses; there are only 4.3 billion unique addresses available.
- IPv6 uses 128-bits for addresses; this allows for 3.4x 10^{38} unique addresses.

Network Configuration Commands

- To view the IP address:

  $ /sbin/ip addr show

- To view the routing information:

  $ /sbin/ip route show

- **ping** is used to check whether or not a machine attached to the network can receive and send data.
- **route** is used to view or change the IP routing table.
- **traceroute** is used to inspect the route which the data packet takes to reach the destination host which makes it quite useful for troubleshooting network delays and errors.

More Networking Tools

- Networking tools are very useful for monitoring and debugging network problems (connectivity and traffic).

Networking Tools	Description
ethtool	Queries network interfaces and can also set various parameters such as the speed.
netstat	Displays all active connections and routing tables. Useful for monitoring performance and troubleshooting.
nmap	Scans open ports on a network. Important for security analysis
tcpdump	Dumps network traffic for analysis.
iptraf	Monitors network traffic in text mode.

FTP (File Transfer Protocol)

- File Transfer Protocol (FTP) is a well-known and popular method for transferring files between computers using the Internet. This method is built on a client-server model.
- Some command line FTP clients are:
  - ftp
  - sftp
  - ncftp
  - yafc (Yet Another FTP Client)
- **sftp** is a very secure mode of connection, which uses the Secure Shell (ssh) protocol. sftp encrypts its data and thus sensitive information is transmitted more securely.

telnet

- **telnet** is a terminal emulation program for TCP/IP networks that allows you to access another computer on the Internet or local area network by logging in to the remote system.
- telnet is a client-server protocol used to establish a connection to Transmission Control Protocol port number 23. You can also check open ports on a remote system using telnet.
- telnet is an unencrypted and therefore insecure protocol. So it is recommended use ssh instead of telnet.

Processes

- A process is simply an instance of one or more related tasks (threads) executing on your computer.
- **ps** provides information about currently running processes, keyed by PID.
- Background and Foreground Processes
  - Foreground jobs run directly from the shell, and when one foreground job is running, other jobs need to wait for shell access (at least in that terminal window if using the GUI) until it is completed.
  - The background job will be executed at lower priority, which, in turn, will allow smooth execution of the interactive tasks, and you can type other commands in the terminal window while the background job is running. You can put a job in the background by suffixing & to the command

Processes

- **at** executes any non-interactive command at a specified time.
- **cron** is used to schedule tasks that need to be performed at regular intervals.
- **pstree** displays the processes running on the system in the form of a tree diagram showing the relationship between a process and its parent process and any other processes that it created.
- **ps** provides information about currently running processes, keyed by PID.

Process types

Process Type	Description	Example
Interactive Processes	Need to be started by a user, either at a command line or through a graphical interface such as an icon or a menu selection.	bash, firefox, top
Batch Processes	Automatic processes which are scheduled from and then disconnected from the terminal. These tasks are queued and work on a FIFO (First In, First Out) basis.	updatedb
Daemons	Server processes that run continuously. Many are launched during system startup and then wait for a user or system request indicating that their service is required.	httpd, xinetd, sshd
Threads	Lightweight processes. These are tasks that run under the umbrella of a main process, sharing memory and other resources, but are scheduled and run by the system on an individual basis. An individual thread can end without terminating the whole process and a process can create new threads at any time. Many non-trivial programs are multi-threaded.	gnome-terminal, firefox
Kernel Threads	Kernel tasks that users neither start nor terminate and have little control over. These may perform actions like moving a thread from one CPU to another, or making sure input/output operations to disk are completed.	kswapd0, migration, ksoftirqd

Compiling C++ programs

- We need to install g++ compiler

```
$ sudo apt install g++
```

or

```
$ sudo apt install build-essential
```

- Confirm your installation by checking for GCC version:

```
$ g++ --version
```

Compiling and running C++ programs

- Compile a simple C++ "Hello World" code: $

```
#include <iostream>
using namespace std;

int main()
{
  cout << "Hello, World!";
  return 0;
}
```

- Save the above code within hello.cpp file, compile and execute it:

```
$ g++ -o hello hello.cpp
$ ./hello
Hello, World!
```

Make utility

- The **make** utility automates the aspects of building executable from source code using a file called **makefile**, which contains rules on how to build the executables.
- Create the file named makefile in the same directory as the source file. Use "tab" to indent the command (NOT spaces).

```
all: hello.exe
hello.exe: hello.o
    g++ -o hello.exe hello.o
hello.o: hello.cpp
    g++ -c hello.cpp
clean:
    rm hello.o hello.exe
```

- Run the make utility as follows:

```
$ make
```

Examples

- Unix Basics

Referencias

- Linux command line cheat sheet
- Linux commands
- Linux man pages
- Linux Foundation