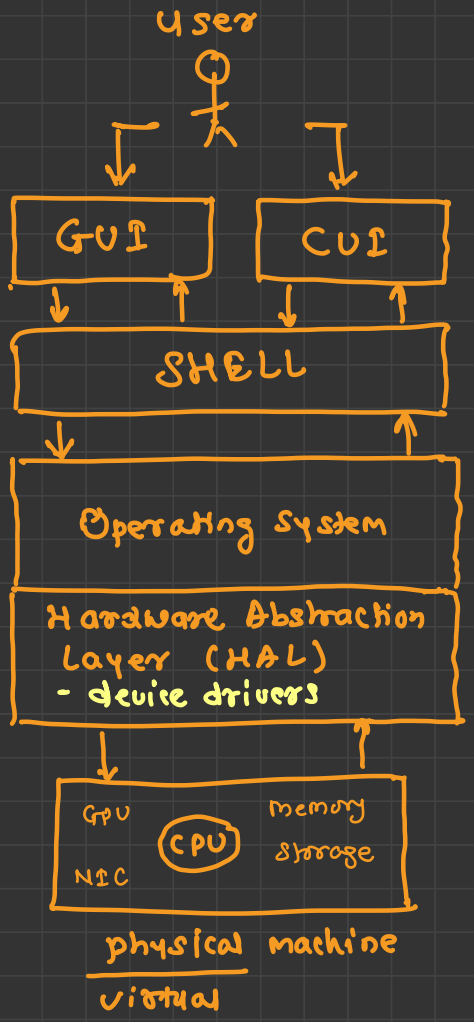

/_inux
It all starts here

① End user

② developer (app) < CUI / GUI } programming language

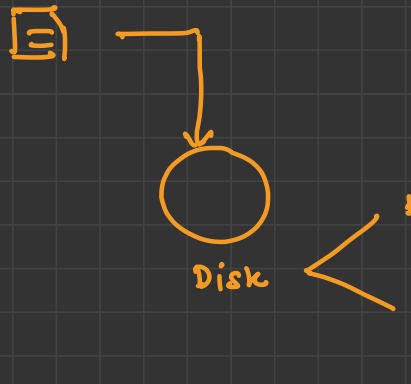
③ administrator

→ ④ OS developer - system dev



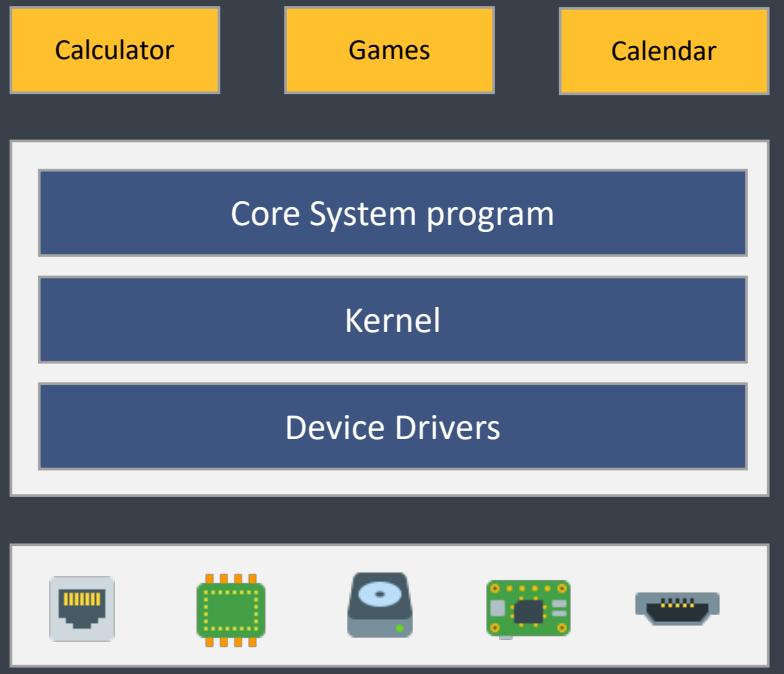
Operating Systems

- desktop OS
  - Windows
  - Linux
  - mac OS
- real time OS
  - RTOS
- embedded OS
  - android
  - ios



What is Operating System ?

- An operating system is a system software that manages computer hardware, software resources and provides common services for computer program
- Responsibilities
  - Memory management
  - Process management
  - File System management (FS driver)
  - CPU Scheduling
  - Hardware abstraction layer



Server

→ software which serves different requests

→ types:

1] Web servers → apache (httpd), nginx, IIS

2] database servers →
   → RDBMS → MySQL, PostgreSQL, Oracle, MS SQL Server
   → NoSQL → MongoDB, CouchDB, Couchbase, firebase, redis

3] mail servers → SMTP (postfix), IMAP (Dovecot)

4] proxy server → Squid proxy server

5] file server → FTP/TFTP, NFS, Samba (SMB)

6] DHCP servers → DHCPD

7] DNS server → BIND

OS Administration

- Operating System Administration has become a solid criterion for an organization that requires a IT foundation
- Hence, the need for efficient administrators is the requirement of the time
- The job profile might change from each organization as there may be added responsibilities to the role
- Below are some responsibilities of a Administrator:
  - Maintain servers like DNS, RADIUS, Apache, MySQL etc
  - Taking regular back up of data, create new stored procedures and listing back-up is one of the duties
  - Analyzing all error logs and fixing along with providing excellent customer support for Webhosting, ISP and LAN Customers on troubleshooting increased support troubles.
  - Communicating with the staff, vendors, and customers in a cultivated, professional manner at all times has to be one of his characteristics
  - Enhance, maintain and creating the tools for the OS environment and its users
  - Detecting and solving the service problems ranging from disaster recovery to login problems
  - Installing the necessary systems and security tools
  - Working with the Data Network Engineer and other personnel/departments to analyze hardware requirements and makes acquiring recommendations
  - Troubleshoot, when the problem occurs in the server

What are we going to cover ?

hardware virtualization → vmware

Virtualization

ip, ifconfig, nmcli,

Configuring Networking

Managing Processes

simple commands : date, ls

Using essential tools

formats, partitions, VDM, RAID

Disk Management

kernel upgrade

Basic Kernel management

path
- absolute
- relative

virtual FS : Root hierarchy, mkdir, ls, mv, cp

File System Management

package manager → yum, apt, DNF,

Managing Packages

bootloader → GRUB, LiLO

Configure Booting

cat, head, tail, less, more, vim

Working with text files

services, systemctl

Mastering Systemd

python / shell script

Automation

useradd, groupadd,

Users and Groups

automation → cron, atq, timer

Scheduling Tasks

print, time, telnet, ssh

Managing Network Services

chmod, chown, chroot, rwx, ugo

Managing Permissions

process mgmt → top, htop, ps, /var/log

Monitoring & Logging

SELinux firewall

Security Implementation

Introduction

Unix

- Unix is one of the oldest operating systems still in use
- It was created in 1969 by Ken Thompson and Dennis Ritchie
- It was not released as open-source software, instead, Unix versions were associated with many different tech organizations, including IBM, Hewlett-Packard, and AT&T
- These various Unix versions are referred to as Unix "flavors" and were proprietary to each company
- E.g
  - HP-UX
  - SunOS and Solaris
  - IRIX
  - AIX
  - Digital Unix
  - BSD : Berkeley Software Distribution, University of California, Berkley
  - SCO Unix
  - NeXTSTEP and OpenStep

Linux

minix

CLI

Richard stallman

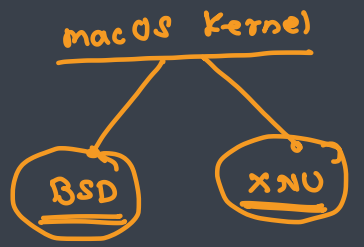
free software movement

FOSS → GNU

GNU is not Unix

GNU Public License

- In 1991, Linus Torvalds created a new Unix-like operating system kernel
- He released this kernel, which he called Linux, under the GPL license
- The Linux kernel, as well as much of the software released with it, is open-source; it can be modified, shared freely, and re-released
- This collaborative approach allows Linux to grow and evolve rapidly
- As a result of this approach, there are now more than 200 Linux versions, or distributions (abbreviated "distros") available
- Of the three primary operating systems in the marketplace today (Linux, macOS, and Windows), two can trace their roots back to Unix
- The macOS kernel evolved from a Unix flavor named BSD and shares many of the same standards and some software as Linux
- However, Apple's OS is not FOSS
- Microsoft Windows also uses a proprietary kernel with a more restrictive licensing method



Unix → flavor

Linux → distro

Features

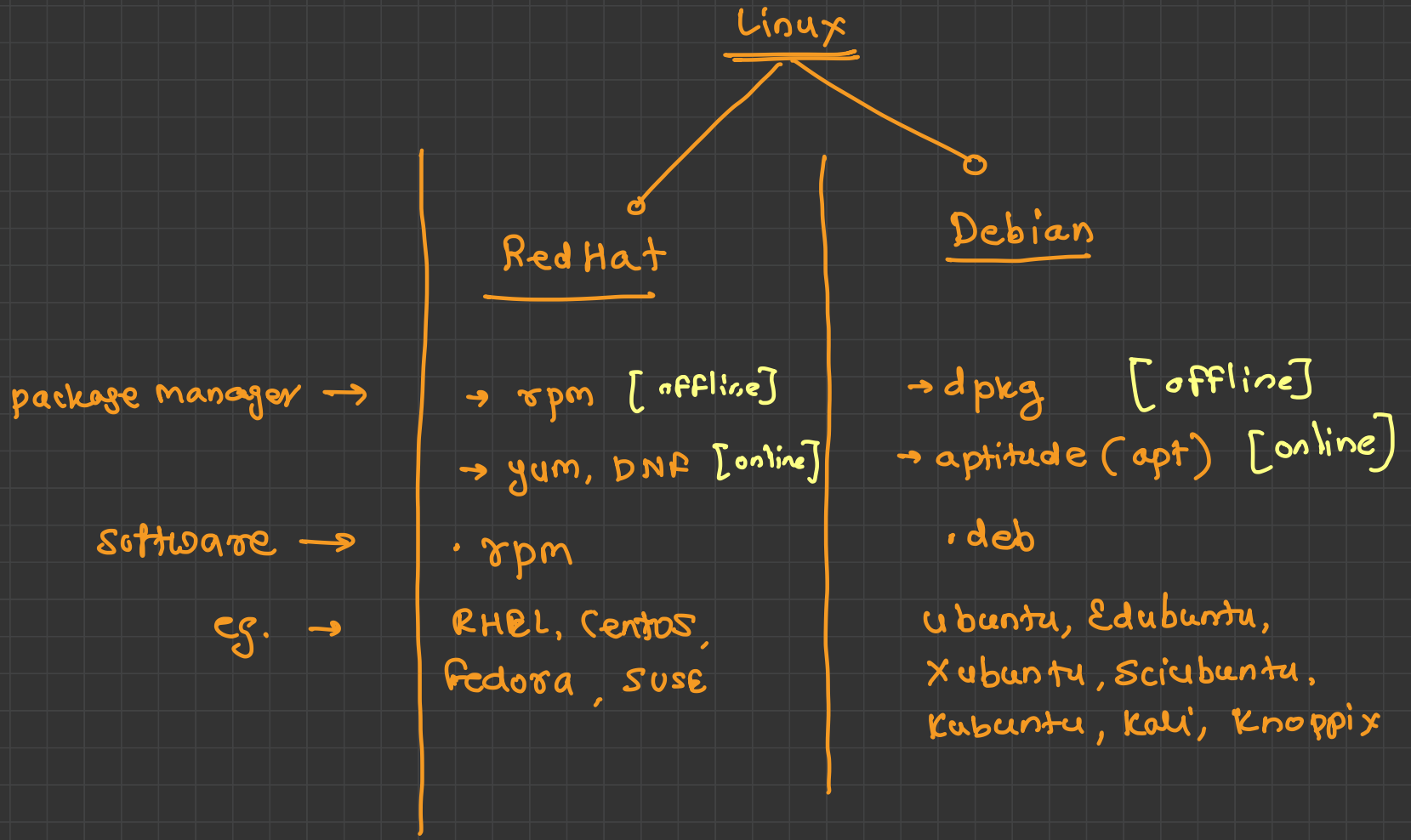
Red Hat

- Free: No licensing fees or tracking associated with most Linux distributions
- Security: Because of the open-source nature of Linux and its associated software, many developers can and do review code for vulnerabilities. Such vulnerabilities tend to be addressed quickly
- Support: Community-driven support may provide easy, efficient, and cost-effective solutions. However, support may be limited to the community, without a strong corporate support structure implemented by the distribution's vendor
- Performance: Linux often provides greater performance and stability compared to other operating systems
- Software availability: Fewer or less familiar software options may exist, especially for nonbusiness applications, such as games
- Hardware requirements: Linux may consume fewer hardware resources, making it easier to retain older systems for longer
- Hardware flexibility: Linux runs on a wide variety of hardware platforms, adding to its flexibility in areas such as Internet of Things (IoT). Specialized hardware may require specific drivers that may not exist for Linux
- Learning curve: Some find that Linux has a steeper learning curve than Windows or macOS does
- Distribution creation: If existing Linux distributions do not fit your needs, you are welcome (and encouraged) to create your own. The sheer number and purpose of Linux distributions can be confusing and overwhelming. There is not a big name in the marketplace that represents Linux and lends it a sense of stability

Linux Distributors

- Because anyone can create and release their own version of Linux, there are thousands of different options
- These individual releases are called distributions
- Distributions are purpose-specific versions of Linux that address a specific need, such as system security or application hosting
- Many distributions trace their history back to one of two specific Linux distributions
  - Red Hat Linux
  - Debian Linux
- One of the main differentiators between these two distros is how they manage software
- Those distros derived from Red Hat Linux use different software managers than those derived from Debian Linux
- The software is also packaged differently

Purpose	Distribution
Security	Kali Linux, Parrot, Mint
Consumer Desktop	Min, Elementary OS, Ubuntu
Lightweight	Puppy Linux, LXLE, Alpine
IoT administration	Snappy Ubuntu Core
Enterprise servers	Red Hat Enterprise Linux, Centos, Fedora Server
Cloud Computing	Amazon Linux, Ubuntu Server
All purpose	Ubuntu


Linux
Red Hat
Debian
package manager →
→ rpm [offline]
→ yum, DNF [online]
→ dpkg [offline]
→ aptitude (apt) [online]
software →
• rpm
• deb
eg. →
RHEL, CentOS,
Fedora, SUSE
Ubuntu, Edubuntu,
Xubuntu, Scibuntu,
Kubuntu, Kali, Knoppix

Linux Deployments

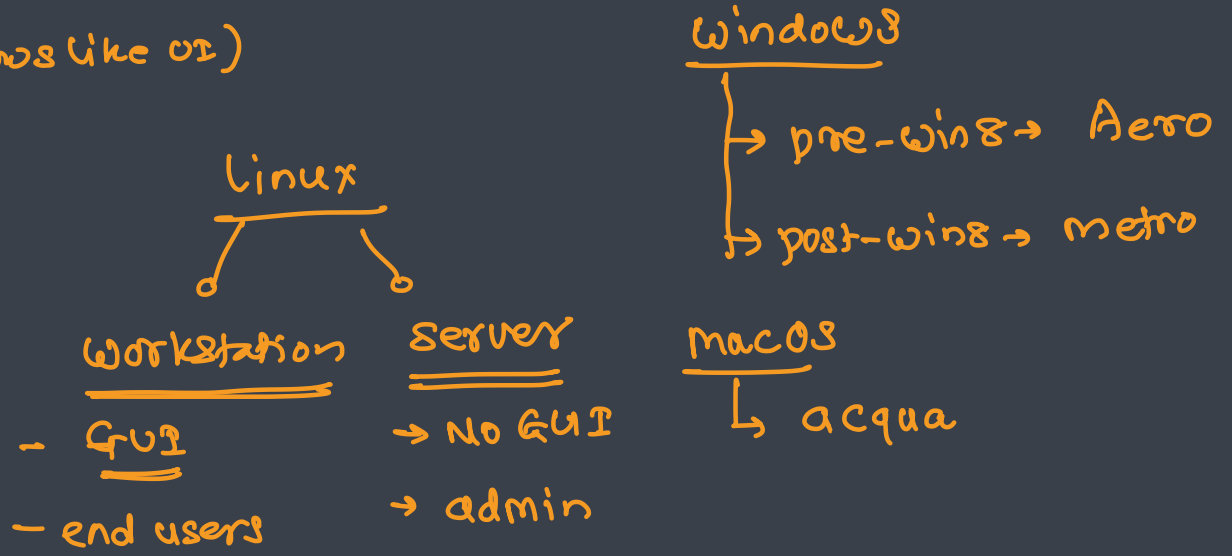
- Webserver: Hosts one or more websites
- Name resolution: Hosts Domain Name System (DNS) name resolution services
- File: Stores business data, usually in some form of text document
- Print: Manages the print process and access to print services
- Log: Centralizes and stores log files from other systems
- Virtualization/container: Hosts virtual machine or container virtualization software
- Database: Hosts one or more databases
- Cluster: Works with other cluster nodes to host high-performance, fault-tolerant services

Interacting with Linux

- Linux provides two ways for user interaction
- Graphical User Interface (GUI)
  - Targeted at end user
  - Simple to use
  - No or less learning curve
- Command Line Interface (CLI)
  - Targeted at administrators and developers
  - Harder compared to GUI
  - Requires you to remember many commands

Graphical User Interface

- Just as there are many different Linux distributions, there are also many different Linux graphical environments
- Windows and macOS users have one GUI available to them
- Linux users have the freedom to install zero, one, or many GUI environments and switch between them
- These GUIs are usually distinguished by two characteristics: user-friendly interface and performance
- In addition, some GUIs consume more processor time and memory than others do
- Common GUI environments
  - GNOME: GNU Network Object Model Environment
  - KDE Plasma : K-Desktop Environment (Windows like UI)
  - Cinnamon
  - MATE
  - XFCE



Command Line Interface

- Linux administrators frequently use the CLI for everyday tasks, while administrators of other platforms often use graphical user interface (GUI) utilities
- In fact, the installation of a GUI is often optional with Linux and may be frowned upon for performance and security reasons
- CLI advantages
  - Quicker: It's usually quicker to execute a series of commands at the CLI
  - Performance: CLI environments consume fewer hardware resources, leaving those resources free to support the server's purpose
  - Scriptable: CLI commands can be written into a text file, which the system then reads and executes in a consistent, efficient, repeatable, and scheduled manner
- CLI disadvantages
  - Learning curve: Remembering many different commands is difficult
  - Nonintuitive: Commands are often difficult to relate to or understand
  - Inconsistent: Many commands differ from each other in small but distinctive ways, making it difficult to recall exactly how to use them
- Common CLI
  - Bash: Default Linux shell
  - ksh: Korn shell
  - zsh: Z shell

csh : C shell

Shell
(Interprets Commands)

Operating System Software
(Processes commands)

Kernel

Hardware

Understanding Commands

- Linux Administrators use commands to interact with OS
- A command is a program which interacts with the kernel to provide the environment and perform the function(s) required by the user
- A command can be
  - Built-in shell commands
    - Internal commands that are built-in the shell
    - Built-in commands are called from the shell and executed directly within the shell itself
    - To list all the built-in commands use: compgen -b
  - External commands
    - Commands are programs which have their own binary and located in the filesystem
    - These are the commands that your system offer and are totally shell independent
    - Mostly these commands reside in /bin, /sbin, /usr/sbin

Command Syntax

- Commands must be entered using a specific structure, or syntax
- Each component of the syntax has a name to make it easier to understand
- The syntax components are:
  - Command: The primary instruction given to the system
  - Subcommand: A secondary, more detailed instruction supporting the primary command
  - Option: A command modifier that slightly changes the way a command is processed
  - Argument: The object on which the command acts. For example, in a command to delete a file, the argument is the name of the file to be deleted

Basic Commands

Command	Purpose
who	Shows who is logged on the system
whoami	Shows the name of the logged in user
date	Shows the current date and time
cal	Shows the calendar
clear	Clears the terminal
echo	Prints the message given by user
history	Shows the list of commands used till the time
type	Shows whether the command is a built-in command or an external command
which	Shows where in the file system the command executable exists
sleep	Sleeps for number of seconds

Command line basics

(bash | sh | zsh)

Shortcut	Description
Ctrl + a	Jump to the beginning of the command line
Ctrl + e	Jump to the end of the command line
Ctrl + u	Clear from the cursor to the beginning of the command line
Ctrl + k	Clear from the cursor to the end of the command line
Ctrl + Left Arrow	Jump to the beginning of the previous word on the command line
Ctrl + Right Arrow	Jump to the beginning of the next word on the command line
Ctrl + r	Search the history list of commands for a pattern

ctrl + w delete previous word

Getting help

- The historical Linux Programmer's Manual, from which man pages originate, was large enough to be multiple printed books
- Each contained information for specific types of files, which have become the sections
- Articles are referred to as topics, as pages no longer applies
- You can use man command to get the manual information of a command
- E.g.
  - man cal
  - man 8 mount
- You can also search by keyword using –k argument
- E.g.
  - man –k password

man Sections

Section	Content Type
1	User commands (both executable and shell programs)
2	System calls (kernel routines invoked from user space)
3	Library functions (provided by program libraries)
4	Special files (such as device files)
5	File formats (for many configuration files and structures)
6	Games (historical section for amusing programs)
7	Conventions. standards, and miscellaneous (protocols, file systems)
8	System administration and privileged commands (maintenance tasks)
9	Linux kernel API (internal kernel calls)

man program shortcuts

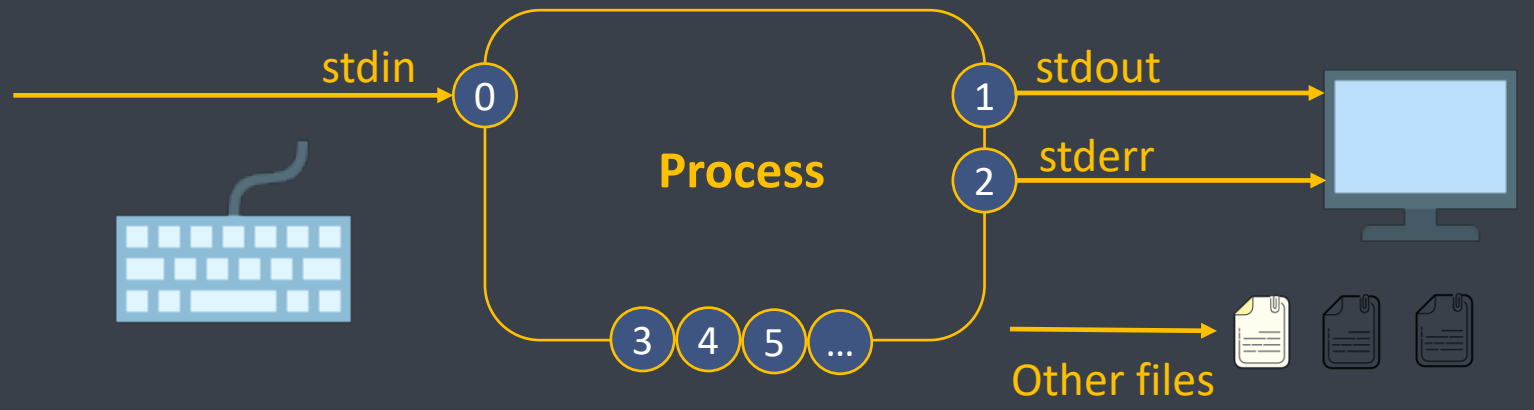
Command	Result
Spacebar or PageDown	Scroll forward (down) one screen
PageUp	Scroll backward (up) one screen
DownArrow	Scroll forward (down) one line
UpArrow	Scroll backward (up) one line
d	Scroll forward (down) on half screen
u	Scroll backward (up) on half screen
/string	Search forward for string
n	Repeat previous search forward
N	Repeat previous search backward
g	Go to the start of the man page
G	Go to end of the man page
q	Exit man and return to terminal

Redirection

Redirection

- A running program, or process, needs to read input from somewhere and write output to somewhere
- A command run from the shell prompt normally reads its input from the keyboard and sends its output to its terminal window
- A process uses numbered channels called file descriptors to get input and send output
- All processes start with at least three file descriptors
  - Standard input (channel 0) reads input from the keyboard
  - Standard output (channel 1) sends normal output to the terminal
  - Standard error (channel 2) sends error messages to the terminal
- If a program opens separate connections to other files, it may use higher-numbered file descriptors.

Redirection



Number	Channel	Description	Default Connection	Usage
0	stdin	Standard Input	Keyboard	Read only
1	stdout	Standard output	Terminal	Write only
2	stderr	Standard Error	Terminal	Write only
3+	filename	Other files	None	Read/Write

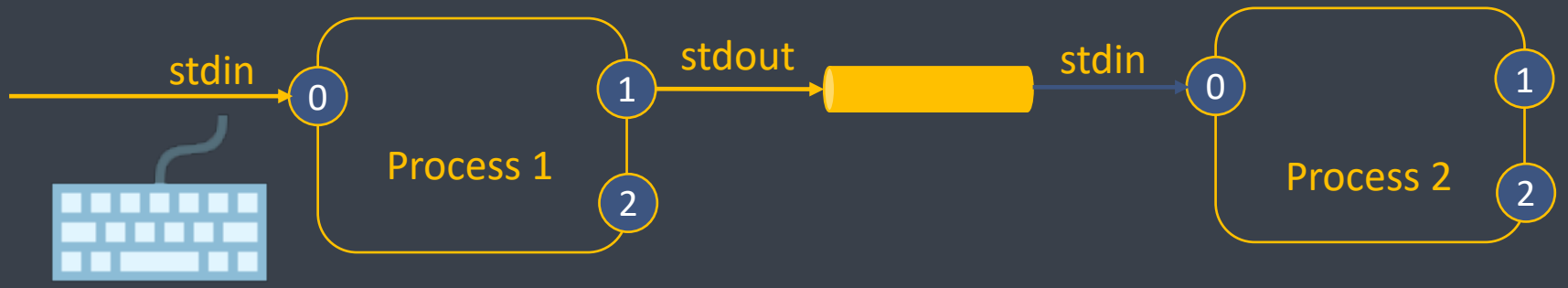
Redirection

- Alternatively, you can use redirection to discard output or errors, so that they are not displayed on the terminal or saved

Usage	Description
> File	Redirect stdout to overwrite a file
>> file	Redirect stdout to append to a file
2> file	Redirect stderr to overwrite file
2> /dev/null	Discard errors
> file1 2> file2	Redirect stdout to file1 Redirect stderr to file2
> file1 2> &1	Redirect stdout and stderr to file1
>> file1 2> &1	Redirect stdout and stderr to append to the same file (file1)

Constructing Pipelines

- A pipeline is a sequence of one or more commands separated by the pipe character (|)
- A pipe connects the standard output of the first command to the standard input of the next command



- **Note**
  - Pipelines and I/O redirection both manipulate standard output and standard input
  - Redirection sends standard output to files or gets standard input from files
  - Pipes send the standard output from one process to the standard input of another process.

Pipe

- Pipeline allows the output of a process to be manipulated and formatted by other process before it outputs to the terminal
- E.g.
  - ls –l /etc | less
  - ls -l /etc | wc -l