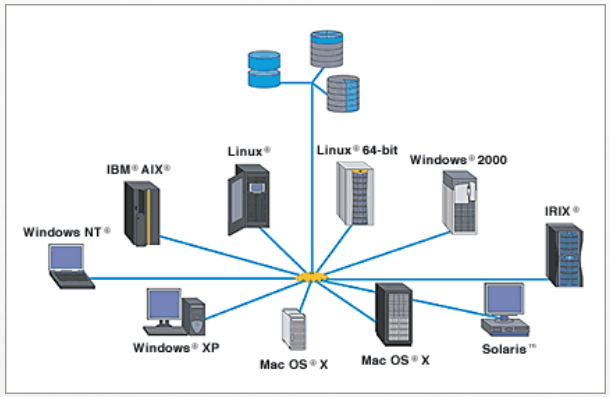
Unix Training



High performance. Delivered.

consulting | technology | outsourcing

Unix

Goals

- Unix Architecture
- System Bootup
- Login Management
- Directories and Files
- Unix File Management
- Standard Unix Streams
- Unix Directories

Unix

Goals

- File Permissions
- Unix Environment Commands
- Java Basic Utilities
- Unix Pipes and Filters
- Grep Command
- Unix Process Management
- Ping, FTP utility

Unix

UNIX

Goals

- Vi Editor
- Shell Commands
- Shell Scripting

What is Unix

- UNIX once referred to a specific operating system.
- However, today it is not a single operating system, but rather a large family of closely related operating systems.
- These different operating systems are sometimes known as UNIX variants, or UNIX-like operating systems.
- All these operating systems are built using a collection of enabling technologies that were originally developed in the 1970s at AT&T Bell Laboratories and at the University of California, Berkeley They have much in common and share a set of utilities and programs.

What is Unix

- The Unix system is a multi-user, multi tasking operating system which means that it allows a single or multiprocessor computer to simultaneously execute several programs by one or several users.
- It has one or several command interpreters (shell) as well as a great number of commands and many utilities (assembler, compilers for many languages, text processing, email, etc.).

What is Unix

- Furthermore, it is highly portable, which means that it is possible to implement a Unix system on almost all hardware platforms.
- Currently, Unix systems have a strong foothold in professional and university environments thanks to their stability, their increased level of security and observance of standards, notably in terms of networks.

Why Is UNIX Important

UNIX

- During the past 35 years, the operating system known as UNIX has evolved into a powerful, flexible, and versatile operating system.
- The different variants of UNIX conform to a variety of standards and are closely related.
- To understand how to use any or all of them, you need to only understand the basic conceptual model upon which UNIX is built.
- Once this conceptual model is understood, it is straightforward to learn the peculiarities of a variant of UNIX or to learn how to use a new variant of UNIX if you already know how to use another.

Why Is UNIX Important

UNIX

- UNIX, as it is implemented in its many variants, serves as the operating system for all types of computers, including personal computers and engineering workstations, multiuser microcomputers, minicomputers, mainframes, and supercomputers, as well as special-purpose devices.
- The number of computers running a variant of UNIX has grown explosively with more than 40 million computers now running a variant of UNIX and more than 300 million people using these systems.
- This rapid growth, especially for computers running Linux, is expected to continue, according to most computer industry experts.
- The success of UNIX is due to many factors, including its portability to a wide range of machines, its adaptability and simplicity, the wide range of tasks that it can perform, its multiuser and multitasking nature, and its suitability for networking, which has become increasingly important as the Internet has blossomed.

Why Is UNIX Important

UNIX

- Open Source Code
- Cooperative Tools and Utilities
- Multiuser and Multitasking Abilities
- Excellent Networking Environment(excellent platform for web servers)
- Portability(**people using the desktop environment of Mac OS X without knowing that it is built on UNIX**)
  - less work is needed to adapt it to run on a new hardware platform.

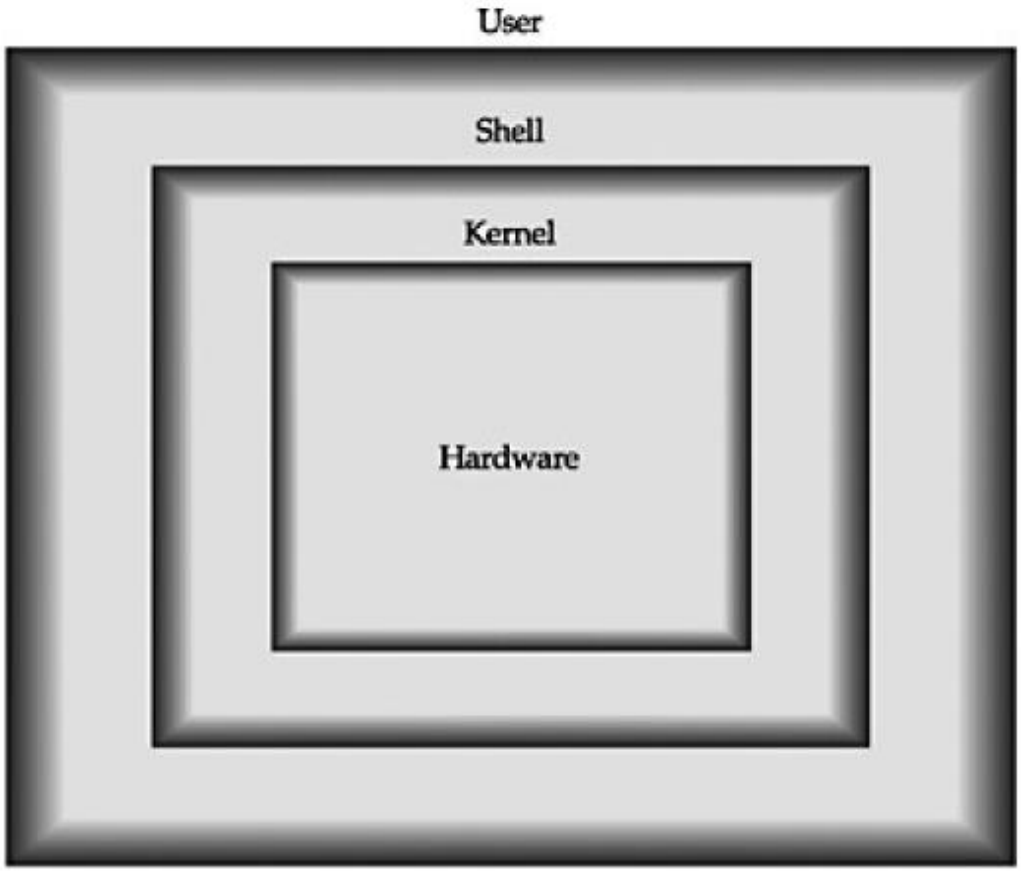
Why Is UNIX Important

- Security
- Background Processing (Many jobs/tasks are executed in bg without human intervention).
- Pipes (chain od commands)
- Redirection tools
- Software Development Tools (any language which has interpreter and compiler).
- Communication

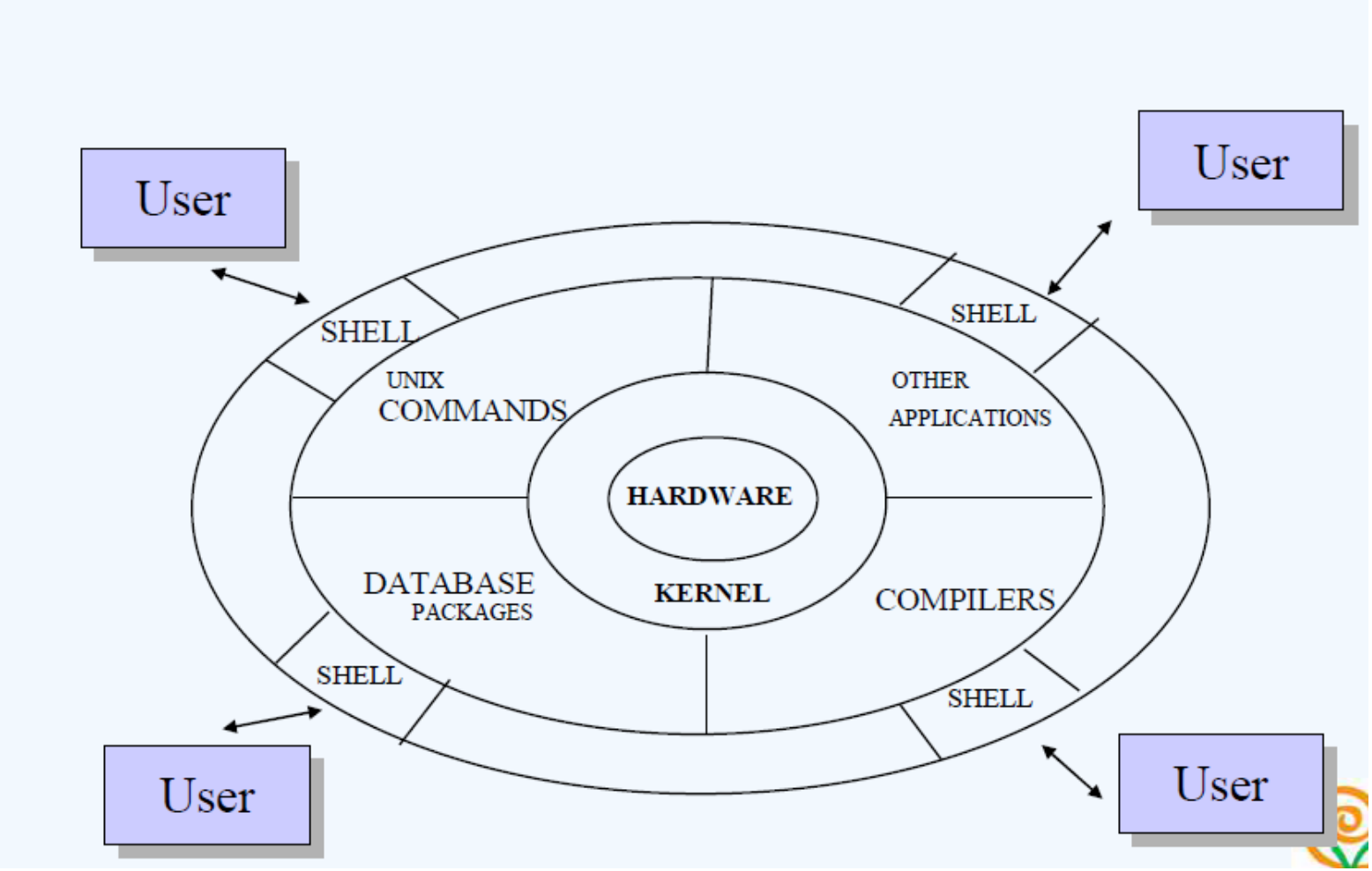
Why Is UNIX Important

- Hierarchical File System
- Shells

The Structure of the UNIX Operating System



The Structure of the UNIX Operating System

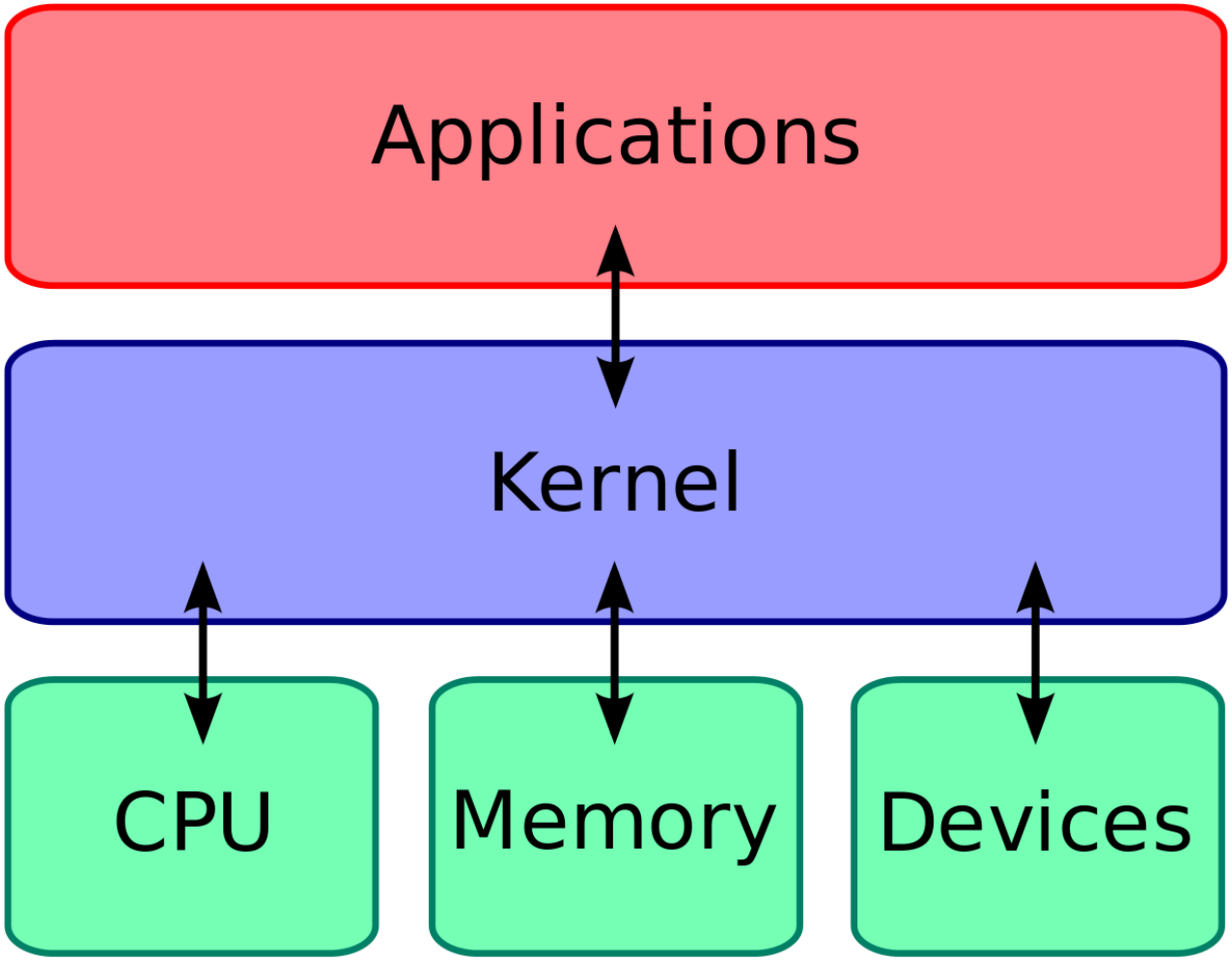


The Structure of the UNIX Operating System

- The UNIX operating system is made up of several major components.
- These components include the kernel, the shell, the file system, and the commands (or user programs).

Kernel

UNIX



Kernel

UNIX

- The kernel is the part of the operating system that interacts directly with the hardware of a computer, through device drivers that are built into the kernel.
- It provides sets of services that can be used by programs, insulating these programs from the underlying hardware.
- The major functions of the kernel are to
  - manage computer memory
  - control access to the computer
  - maintain the file system

Kernel

UNIX

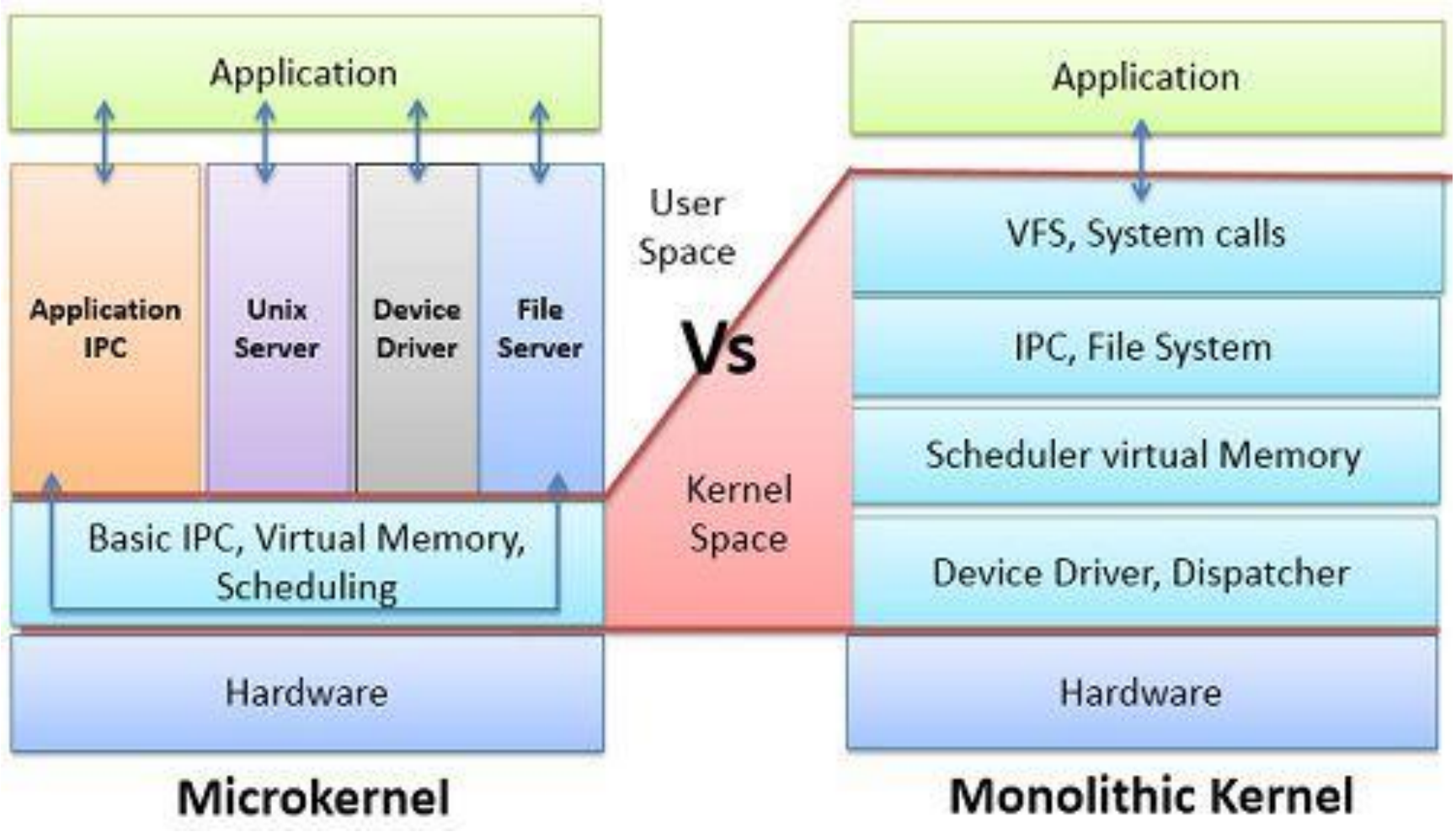
- handle interrupts (signals to terminate execution)
- handle errors
- perform input and output services (which allow computers to interact with terminals, storage devices, and printers)
- allocate the resources of the computer (such as the CPU or input/output devices) among users.

Kernel

UNIX

- Programs interact with the kernel through system calls.
- System calls tell the kernel to carry out various tasks for the program, such as
- Opening a file, writing to a file, obtaining information about a file, executing a program, terminating a process, changing the priority of a process.
- Getting the time of day Different implementations of a variant of the UNIX system may have compatible system calls, with each call having the same functionality.
- However, the internals, programs that perform the functions of system calls (usually written in the C language), and the system architecture in two different UNIX variants or even two different implementations of a particular UNIX variant may bear little resemblance to one another.

Kernel


Application
Application IPC
Unix Server
Device Driver
File Server
Basic IPC, Virtual Memory, Scheduling
Hardware
Microkernel
User Space
Vs
Kernel Space
Application
VFS, System calls
IPC, File System
Scheduler virtual Memory
Device Driver, Dispatcher
Hardware
Monolithic Kernel

Kernel

UNIX

BASIS FOR COMPARISON	MICROKERNEL	MONOLITHIC KERNEL
Basic	In microkernel user services and kernel, services are kept in separate address space.	In monolithic kernel, both user services and kernel services are kept in the same address space.
Size	Microkernel are smaller in size.	Monolithic kernel is larger than microkernel.
Execution	Slow execution.	Fast execution.
Extendible	The microkernel is easily extendible.	The monolithic kernel is hard to extend.
Security	If a service crashes, it does effect on working of microkernel.	If a service crashes, the whole system crashes in monolithic kernel.
Code	To write a microkernel, more code is required.	To write a monolithic kernel, less code is required.
Example	QNX, Symbian, L4Linux, Singularity, K42, Mac OS X, Integrity, PikeOS, HURD, Minix, and Coyotos.	Linux, BSDs (FreeBSD, OpenBSD, NetBSD), Microsoft Windows (95,98,Me), Solaris, OS-9, AIX, HP-UX, DOS, OpenVMS, XTS-400 etc.

Utilities

UNIX

- The UNIX System contains several hundred utilities or user programs.
- Commands are also known as tools, because they can be used separately or put together in various ways to carry out useful tasks.
- You execute these utilities by invoking them by name through the shell; this is why they are called commands.
- A critical difference between UNIX and earlier operating systems is the ease with which new programs can be installed-the shell need only be told where to look for commands, and this is user-definable

Utilities

UNIX

- You can perform many tasks using the standard utilities supplied with UNIX.
- There are utilities for text editing and processing, for managing information, for electronic communications and networking, for performing calculations, for developing computer programs, for system administration, and for many other purposes.

The File System

UNIX

- The basic unit used to organize information in UNIX is called a file.
- The UNIX file system provides a logical method for organizing, storing, retrieving, manipulating, and managing information.
- Files are organized into a hierarchical file system, with files grouped together into directories.
- An important simplifying feature of UNIX is the general way it treats files.
- For example, physical devices are treated as files; this permits the same commands to work for ordinary files and for physical devices; for instance, printing a file (on a printer) is treated similarly to displaying it on the terminal screen.

The File System

UNIX

```
eswaribala@DESKTOP-55AGI0I:~$ ls /
bin  boot  dev  etc  home  init  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  snap  srv  sys  tmp  usr  var
eswaribala@DESKTOP-55AGI0I:~$
```

The Shell

UNIX

- The shell reads your commands and interprets them as requests to execute a program or programs, which it then arranges to have carried out.
- Because the shell plays this role, it is called a command interpreter.
- Besides being a command interpreter, the shell is also a programming language.
- As a programming language, it permits you to control how and when commands are carried out

Applications

UNIX

- You can use applications built using UNIX commands, tools, and programs. Application programs carry out many different types of tasks.
- Some perform general functions that can be used by a variety of users in government, industry, and education.
- These are known as horizontal applications and include such programs as word processors, compilers, database management systems, spreadsheets, statistical analysis programs, and communications programs.
- Others are industry specific and are known as vertical applications.
- Examples include software packages used for managing a hotel, running a bank, and operating point-of-sale terminals.

Applications

UNIX

- Several classes of applications have experienced explosive growth in the past few years.
- The first of these involves network applications, including those that let people make use of the wide range of services available on the Internet.
- Chief among these are web browsers and web server applications.
- Another important class of applications deals with multimedia.
- Such applications let users create and view multimedia files, including audio, images, and video.

The Birth of the UNIX System

- The history of the UNIX System dates back to the late 1960s when MIT, AT&T Bell Labs, and then computer manufacturer GE (General Electric) worked on an experimental operating system called Multics.
- Multics, from Multiplexed Information and Computing System, was designed to be an interactive operating system for the GE 645 mainframe computer.
- Allowed information sharing while providing security Development met with many delays, and production versions turned out to be slow and required extensive memory
- For a variety of reasons, Bell Labs dropped out of the project.
- However, the Multics system implemented many innovative features and produced an excellent computing environment.

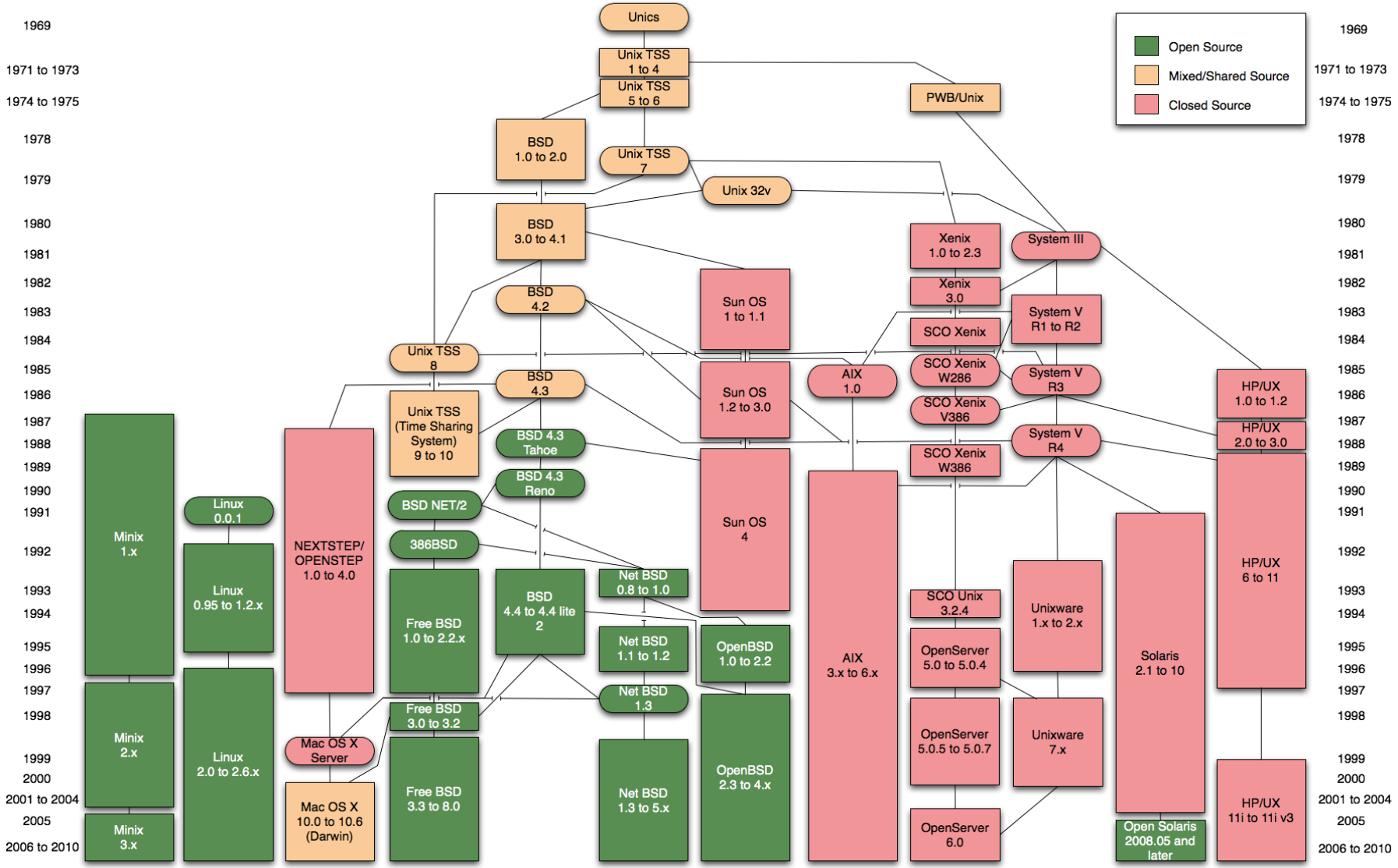
The Birth of the UNIX System

- In 1969, Ken Thompson, one of the Bell Labs researchers involved in the Multics project, wrote a game for the GE computer called Space Travel.
- This game simulated the solar system and a space ship.
- Thompson found that the game ran jerkily on the GE machine and was costly-approximately $75 per run! With help from Dennis Ritchie, Thompson rewrote the game to run on a spare DEC PDP-7.
- This initial experience gave him the opportunity to write a new operating system on the PDP-7, using the structure of a file system Thompson, Ritchie, and Rudd Canaday had designed.
- Thompson, Ritchie, and their colleagues created a multitasking operating system, including a file system, a command interpreter, and some utilities for the PDP-7.

The Birth of the UNIX System

- Because the new multitasking operating system for the PDP-7 could support two simultaneous users, it was humorously called UNICS for the Uniplexed Information and Computing System; the first use of this name is attributed to Brian Kernighan.
- The name was changed slightly to UNIX in 1970, and that has stuck ever since.

The Birth of the UNIX System



A UNIX System Timeline

UNIX

Year	UNIX Variant or Standard	Comments
1969	UNICS (later called UNIX)	A new operating system invented by Ken Thompson and Dennis Ritchie for the PDP-7
1973	Fourth Edition	Written in C programming language; widely used inside Bell Laboratories
1975	Sixth Edition	First version widely available outside of Bell Labs; more than 600 machines ran it
1978	3BSD	Virtual memory
1979	Seventh Edition	Included the Bourne shell, UUCP, and C; the direct ancestor or modern UNIX
1980	Xenix	Introduced by Microsoft
1980	4BSD	Introduced by UC Berkeley
1982	System III	First public release outside of Bell Labs
1983	System V Release 1	First supported release

A UNIX System Timeline

1983	4.1BSD	UC Berkeley release with performance enhancements
1984	4.2BSD	UC Berkeley release with many networking capabilities
1984	System V Release 2	Protection and locking of files, enhanced system administration, and job control features added
1986	HP-UX	First version of HP-UX released for HP Precision Architecture
1986	AIX Version 1	First version of IBM's proprietary version of UNIX, based on SVR3
1987	System V Release 3	STREAMS, RFS, TLI added
1987	4.3BSD	Minor enhancements to 4.2BSD
1988	POSIX	POSIX.1 published
1989	System V Release 4	Unified System V, BSD, and Xenix
1990	XPG3	X/Open specification set
1990	OSF/1	Open Software Foundation release designed to compete with SVR4
1991	386BSD	Based on BSD for Intel 80386

A UNIX System Timeline

1991	Linux 0.01	Linus Torvalds started development of Linux
1992	SVR4.2	USL-developed version of SVR4 for the desktop
1992	HP-UX 9.0	Supported workstations, including a GUI
1993	Solaris 2.3	POSIX compliant
1993	4.4BSD	Final Berkeley release
1993	FreeBSD 1.0	Initial release based on 4.3BSD and 386BSD
1993	SVR4.2MP	Last version of UNIX developed by USL
1994	Linux 1.0	First version of Linux not considered a “beta”
1994	NetBSD 1.0	First multiplatform release
1994	Solaris 2.4	Motif supported
1994	AIX4	Introduced CDE support
1994	FreeBSD 2.0	Based on 4.4BSD-Lite to allow free distribution
1995	UNIX 95	X/Open mark for systems registered under the Single UNIX Specification
1995	Solaris 2.5	CDE supported
1995	HP-UX 10.0	Conformed to the Single UNIX Specification and the Common Desktop Environment (CDE)

A UNIX System Timeline

1996	Linux 2.0	Performance improvements and networking software added
1996	OpenBSD 1.2	Initial release with strong support of security
1997	Solaris 2.6	UNIX 95 compliant, JAVA supported
1997	Single UNIX Specification, Version 2	Open Group specification set
1997	System V Release 5	Enhanced SV kernel, including 64-bit support, increased reliability, and performance enhancements
1997	UnixWare 7	SCO UNIX based on SVR5 kernel
1997	HP-UX 11.0	64-bit operating system
1997	AIX 4.3	Support for 64-bit architectures, registered with UNIX 98 mark
1998	UNIX 98	Open Group mark for systems registered under the Single UNIX Specification, Version 2
1998	FreeBSD 3.0	Kernel changes and security fixes
1998	Solaris 7	Support for 64-bit applications, free for noncommercial users
1999	Linux 2.2	Device drivers added
1999	Darwin	Apple developed UNIX-like OS, basis for Mac OS X
2000	Solaris 8	Performance and application support enhancements

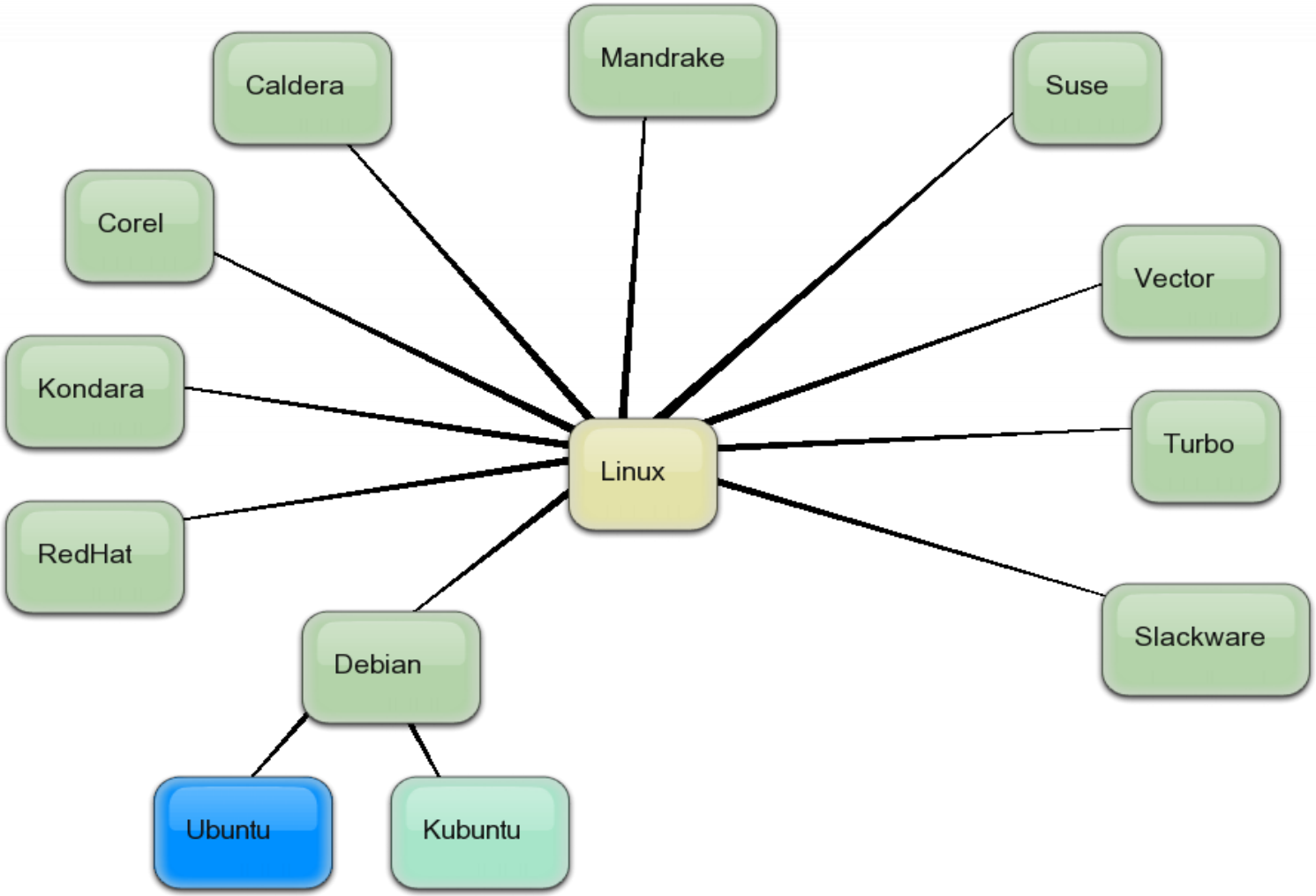
A UNIX System Timeline

2000	HP-UX 11i	Introduces operating environments
2000	FreeBSD 4.0	Networking and security enhancements
2001	Linux 2.4	Enhanced device support, scalability enhancements
2001	AIX 5L	Introduced affinity for Linux
2001	Mac OS X 10.0 “Cheetah”	First Mac OS release based on Darwin. Incomplete and slow, but with basic OS features, device support, and software development environment
2001	Mac OS X 10.1 “Puma”	More complete than Cheetah, with performance enhancements and support for additional device drivers
2002	Solaris 9	Manageability, security, and performance enhancements
2002	Mac OS X 10.2 “Jaguar”	First solid release of Mac OS X
2003	Linux 2.6	Scalability for operation on embedded systems to large servers,

A UNIX System Timeline

		human interface, networking, and security enhancements
2003	Mac OS X 10.3 “Panther”	Performance enhancements, an extensive update to the user interface, and greater interoperability with MS Windows
2003	Single UNIX Specification, Version 3	Developed by the Austin Group
2003	FreeBSD 5.0	Improved SMP support, TrustedBSD security features
2004	Solaris 10	Advanced security, performance, and availability enhancements
2004	NetBSD 2.0	Support for SMP
2005	OpenServer 6	Improved SMP support and support for extremely large files
2005	Mac X 10.4 “Tiger”	New features include Spotlight, a fast content and metadata-based file search tool, and support for 64-bit platforms and Intel x86 platforms
2005	Net BSD 3.0	Suppose Xen Virtual Machine Monitor

Widely Used UNIX Variants



UNIX Contributors

Aho, Alfred	Coauthor of the AWK programming language and author of egrep
Bourne, Steven	Author of the Bourne shell, the ancestor of the standard shell in UNIX System V
Canaday, Rudd	Developer of the UNIX System file system, along with Dennis Ritchie and Ken Thompson
Cherry, Lorinda	Author of the Writer's Workbench (WWB), coauthor of the eqn preprocessor, and coauthor of the bc and dc utilities
Honeyman, Peter	Developer of HoneyDanBer UUCP at Bell Laboratories in 1983 with David Nowitz and Brian Redman
Horton, Mark	Author of curses and terminfo, and a major contributor to the UUCP Mapping Project and the development of USENET
Joy, William	Creator of the vi editor and the C shell, as well as many BSD enhancements. Cofounder of Sun Microsystems
Kernighan, Brian	Coauthor of the C programming language and of the AWK programming language. Rewrote troff in the C language
Korn, David	Author of the Korn shell, a superset of the standard System V shell with many enhanced features, including command histories
Lesk, Mike	Developer of the UUCP System at Bell Laboratories in 1976 and author of the tbl preprocessor, ms macros, and lex

UNIX Contributors

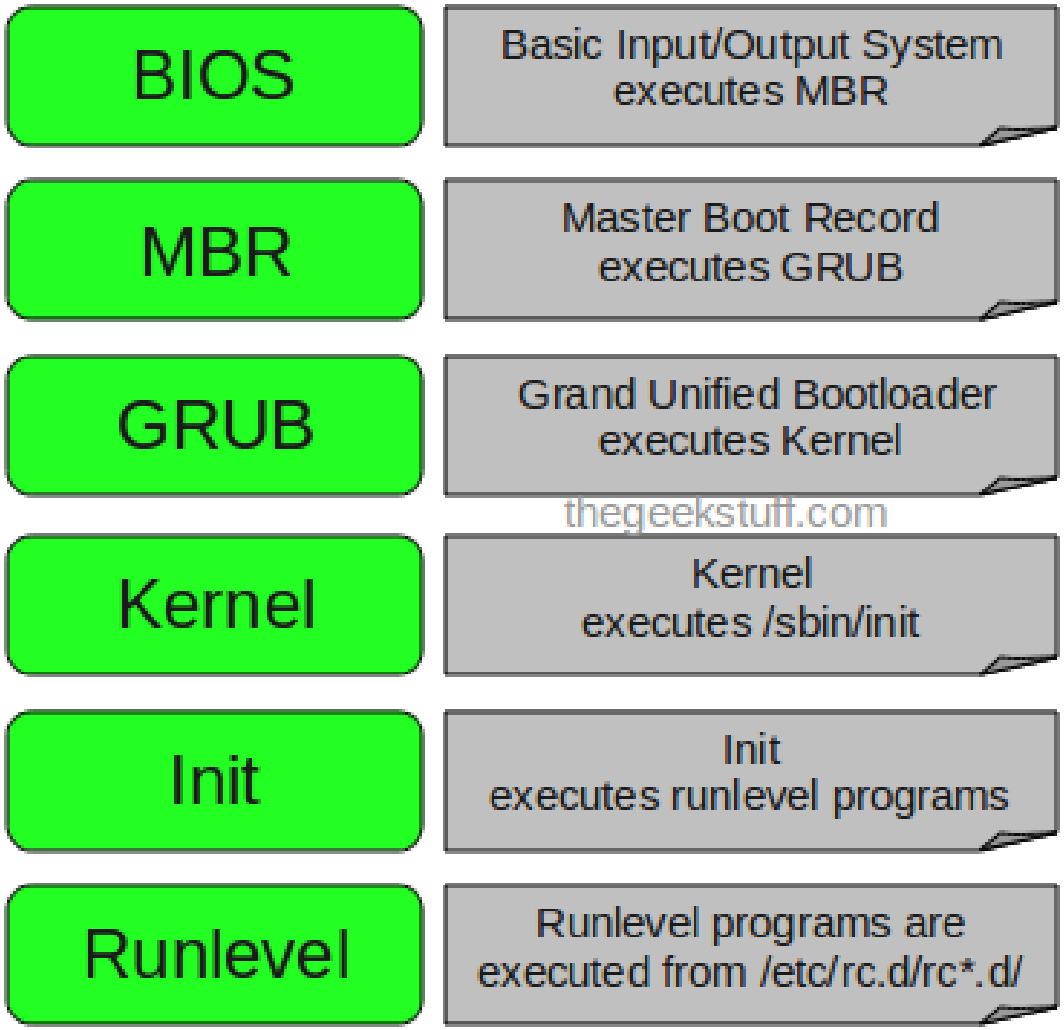
Mashey, John	Author of the early versions of the shell, which were later merged into the Bourne shell
McIlroy, Doug	Developed the concept of pipes and wrote the spell and diff commands
Morris, Robert	Coauthor of the utilities bc and dc
Nowitz, David	Developer of HoneyDanBer UUCP at Bell Laboratories in 1983 with Peter Honeyman and Brian Redman
Ossanna, Joseph	Creator of the troff text formatting processor
Ousterhout, John	Developer of Tcl command language
Redman, Brian	Developer of HoneyDanBer UUCP at Bell Laboratories in 1983 with Peter Honeyman and David Nowitz
Ritchie, Dennis	Inventor of the UNIX Operating System, along with Ken Thompson, at Bell Laboratories. Inventor of the C language, along with Brian Kernighan
Scheifler, Robert	Mentor of the X Window System
Stallman, Richard	Developer of the programmable visual text editor emacs, and founder of GNU project and the Free Software Foundation
Stroustrup, Bjarne	Developer of the object-oriented C++ programming language
Tannenbaum,	Creator of Minix, a program environment that led to the development of Linux

UNIX Contributors

Andrew	
Thompson, Ken	Inventor of the UNIX operating system, along with Dennis Ritchie, at Bell Laboratories
Torek, Chris	Developer from the University of Maryland who was one of the pioneers of BSD UNIX
Torvalds, Linus	Creator of the Linux operating system, an Intel personal computer-based variant of UNIX
Wall, Larry	Developer of the Perl programming language
Weinberger, Peter	Coauthor of the AWK programming language

Boot up Process

UNIX

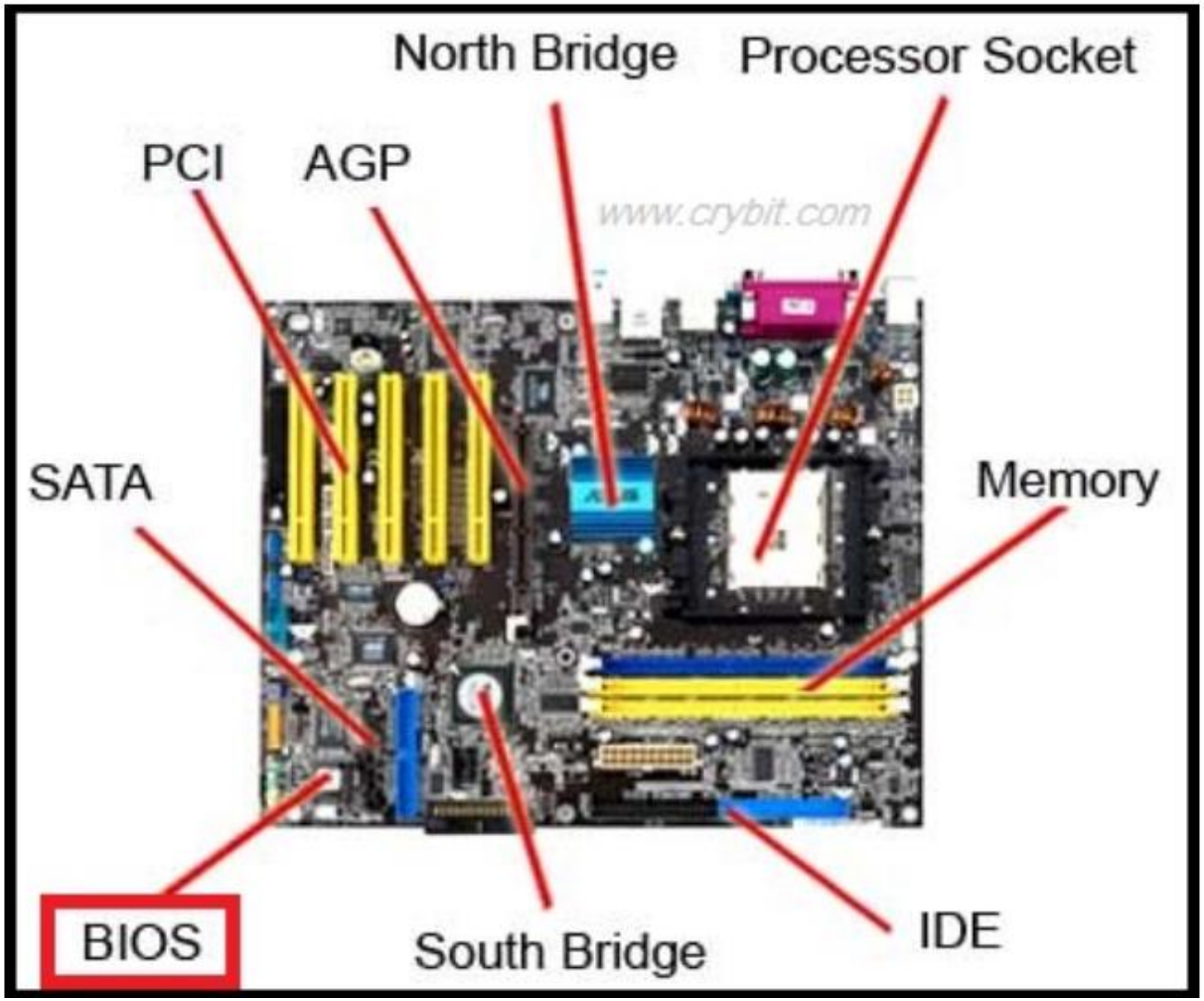


Bios

UNIX

- BIOS stands for Basic Input/Output System
- Performs some system integrity checks
- Searches, loads, and executes the boot loader program.
- It looks for boot loader in floppy, cd-rom, or hard drive. You can press a key (typically F12 of F2, but it depends on your system) during the BIOS startup to change the boot sequence.
- Once the boot loader program is detected and loaded into the memory, BIOS gives the control to it.
- So, in simple terms BIOS loads and executes the MBR boot loader.

Bios

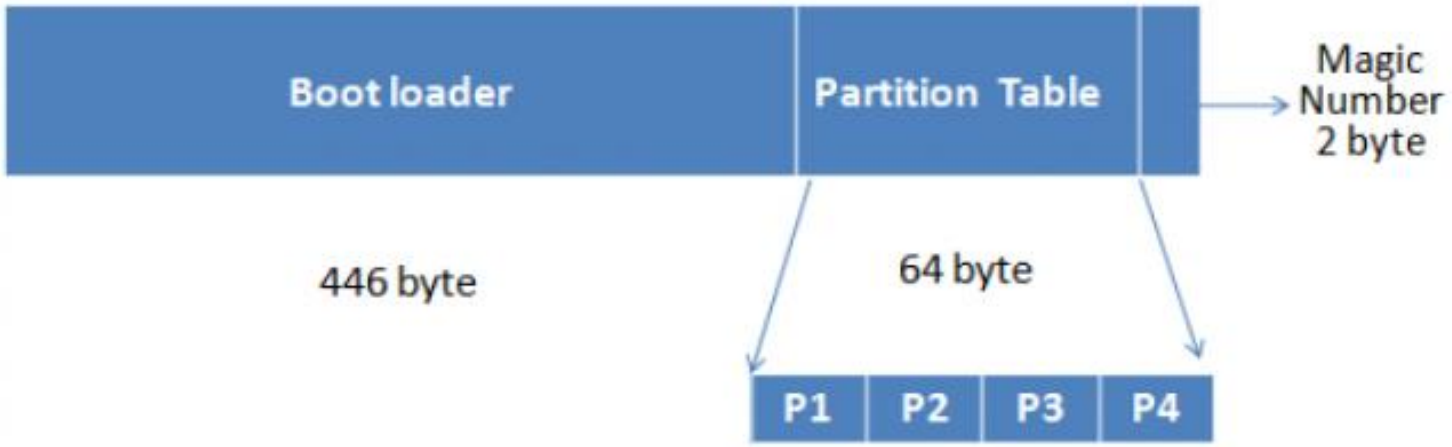

North Bridge
Processor Socket
PCI
AGP
www.crybit.com
SATA
Memory
BIOS
South Bridge
IDE

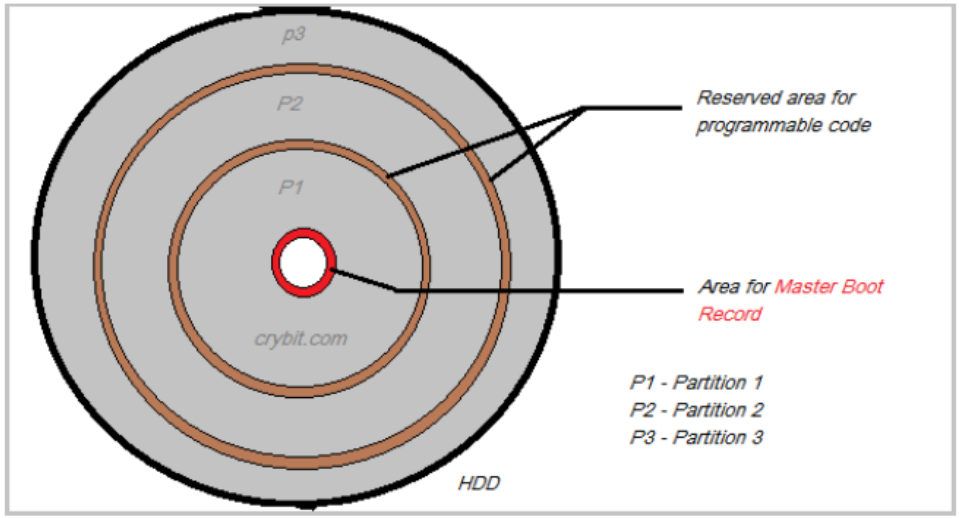
MBR

UNIX

- MBR stands for Master Boot Record.
- It is located in the 1st sector of the bootable disk. Typically /dev/hda, or /dev/sda
- MBR is less than 512 bytes in size. This has three components 1) primary boot loader info in 1st 446 bytes 2) partition table info in next 64 bytes 3) mbr validation check in last 2 bytes.
- It contains information about GRUB (or LILO in old systems).
- So, in simple terms MBR loads and executes the GRUB boot loader.

MBR


Boot loader
Partition Table
Magic Number 2 byte
446 byte
64 byte
P1
P2
P3
P4


p3
P2
P1
crybit.com
Reserved area for programmable code
Area for Master Boot Record
P1 - Partition 1
P2 - Partition 2
P3 - Partition 3
HDD

GRUB

UNIX

- GRUB stands for Grand Unified Bootloader.
- If you have multiple kernel images installed on your system, you can choose which one to be executed.
- GRUB displays a splash screen, waits for few seconds, if you don’t enter anything, it loads the default kernel image as specified in the grub configuration file.
- GRUB has the knowledge of the filesystem (the older Linux loader LILO didn’t understand filesystem).
- Grub configuration file is /boot/grub/grub.conf (/etc/default/grub.d is a link to this).

Kernel

UNIX

- Mounts the root file system as specified in the “root=” in grub.conf
- Kernel executes the /sbin/init program
- Since init was the 1st program to be executed by Linux Kernel, it has the process id (PID) of 1. Do a ‘ps -ef | grep init’ and check the pid.
- initrd stands for Initial RAM Disk.
- initrd is used by kernel as temporary root file system until kernel is booted and the real root file system is mounted. It also contains necessary drivers compiled inside, which helps it to access the hard drive partitions, and other hardware.

Init

UNIX

- Looks at the /etc/inittab file to decide the Linux run level.
- Following are the available run levels
- 0 – halt
- 1 – Single user mode
- 2 – Multiuser, without NFS
- 3 – Full multiuser mode
- 4 – unused
- 5 – X11
- 6 – reboot
- Init identifies the default initlevel from /etc/inittab and uses that to load all appropriate program.

Init

UNIX

- Execute ‘grep initdefault /etc/inittab’ on your system to identify the default run level
- If you want to get into trouble, you can set the default run level to 0 or 6. Since you know what 0 and 6 means, probably you might not do that.
- Typically you would set the default run level to either 3 or 5.

Runlevel programs

UNIX

- When the Linux system is booting up, you might see various services getting started. For example, it might say “starting sendmail …. OK”. Those are the runlevel programs, executed from the run level directory as defined by your run level.

Runlevel programs

UNIX

- Depending on your default init level setting, the system will execute the programs from one of the following directories.
- Run level 0 – /etc/rc.d/rc0.d/
- Run level 1 – /etc/rc.d/rc1.d/
- Run level 2 – /etc/rc.d/rc2.d/
- Run level 3 – /etc/rc.d/rc3.d/
- Run level 4 – /etc/rc.d/rc4.d/
- Run level 5 – /etc/rc.d/rc5.d/
- Run level 6 – /etc/rc.d/rc6.d/

Runlevel programs

UNIX

- Please note that there are also symbolic links available for these directory under /etc directly. So, /etc/rc0.d is linked to /etc/rc.d/rc0.d.
- Under the /etc/rc.d/rc*.d/ directories, you would see programs that start with S and K.
- Programs starts with S are used during startup. S for startup.
- Programs starts with K are used during shutdown. K for kill.
- There are numbers right next to S and K in the program names. Those are the sequence number in which the programs should be started or killed.
- For example, S12syslog is to start the syslog deamon, which has the sequence number of 12. S80sendmail is to start the sendmail daemon, which has the sequence number of 80. So, syslog program will be started before sendmail.

Install Ubuntu in Windows 10

UNIX

- https://docs.microsoft.com/en-us/windows/wsl/install-win10
- In the next step, it's time to install the graphical desktop programs. You do this by running these programs from Bash:
- https://www.zdnet.com/article/how-to-run-run-the-native-ubuntu-desktop-on-windows-10/
- apt-get install ubuntu-desktop
- apt-get install unity
- apt-get install compiz-core
- apt-get install compizconfig-settings-manager
- export DISPLAY=localhost:0
- Ccsm

Install Ubuntu in Windows 10

UNIX

- sudo apt-get install xfce4
- Xfce4-session

How to access and log in to a UNIX system

- https://cocalc.com/doc/terminal.html

How to access and log in to a UNIX system


```
eswaribala@DESKTOP-55AGI0I:~$ sudo login
[sudo] password for eswaribala:
DESKTOP-55AGI0I login: eswaribala
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 4.4.0-18362-Microsoft x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Tue Aug  4 22:30:36 IST 2020

  System load:    0.52      Users logged in:        0
  Usage of /home: unknown   IPv4 address for eth2:  169.254.131.111
  Memory usage:   52%       IPv4 address for eth4:  192.168.148.113
  Swap usage:     0%        IPv4 address for wifi0: 192.168.0.8
  Processes:      12


0 updates can be installed immediately.
0 of these updates are security updates.


Last login: Sat Jul 18 18:07:18 IST 2020 on tty1
eswaribala@DESKTOP-55AGI0I:~$
```

How to access and log in to a UNIX system

- Every UNIX system has at least one person, called the system administrator, whose job is to maintain the system.
- The system administrator is also responsible for adding new users to the system, and for setting up the initial work environment.
- If you are on a multiuser system, you will have to ask the system administrator to set up a login for you.
- If you are the only user on the system, you will be the system administrator.
- During the installation of your UNIX variant, you will be asked to select a login name and password.

How to access and log in to a UNIX system

- In general, your login name can be almost any combination of letters and numbers, although there are a few constraints:
- Your login name must be more than two characters long.
- If it is longer than eight, only the first eight characters are relevant.
- It must contain only lowercase letters and numbers and must begin with a lowercase letter.
- No symbols or spaces are allowed.
- It cannot be the same as another login name already in use. Some login names are customarily reserved for certain uses; for example, root is often a login name for the system administrator (sometimes called the superuser).

How to access and log in to a UNIX system

- Choosing a Password
- If you begin by installing a UNIX variant on your own system, it will ask you to choose a password when you select a login name.
- If your account is on a remote system, your system administrator will probably assign you a temporary password, which you should change the first time you log in.

How to access and log in to a UNIX system

- Passwords must have at least six characters.
- Passwords must contain at least two alphabetic characters (uppercase or lowercase letters), and at least one number or symbol. Note that UNIX is sensitive to case, so WIZARD is a different password than w1zard.
- Your login name with its letters reversed or shifted cannot be used as a password.
- For example, if your login name is msilver, you cannot choose silverm or revlism as a password.
- The passwords 3hrts&3lyonz and R0wkS+@r are both valid, but kilipuppy (no numeric or special characters) and Red1 (too short) are not.

UNIX System Password Security

UNIX

- Your first contact with security on your UNIX system is choosing a password.
- Simple passwords are easily guessed. A large commercial dictionary contains about 250,000 words, which can be checked as passwords in less than two minutes of computer time.
- All dictionary words spelled backward takes about another minute.
- All dictionary words preceded or followed by the digits 0–99 can be checked in just a few more minutes.

UNIX System Password Security

UNIX

- Here are some guidelines:
- Avoid easily guessed passwords, such as your name or the names of family members or pets.
- Also avoid your address, your car’s license plate, and any other phrase that someone might associate with you.
- Avoid words or names that exist in a dictionary (in any language, not just English).
- Avoid trivial modifications of dictionary words. For example, normal words with replacement of certain letters with numbers: mid5umm3r, sn0wball, and so forth.

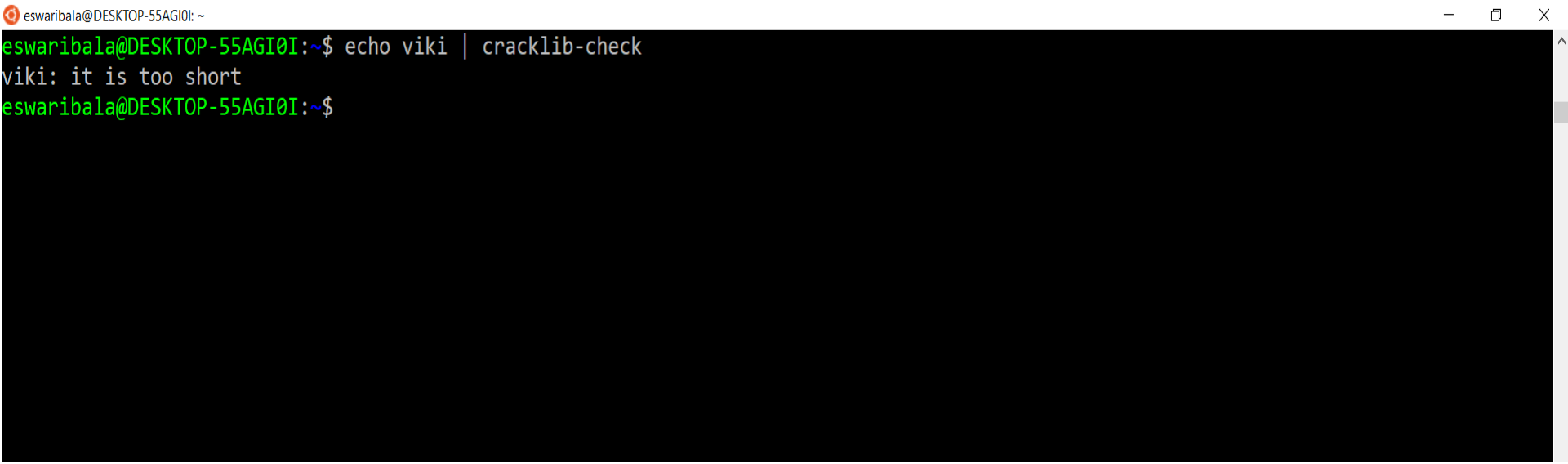
UNIX System Password Security

sudo apt install cracklib-runtime

```
eswaribala@DESKTOP-55AGI0I:~$ sudo apt install cracklib-runtime
[sudo] password for eswaribala:
Sorry, try again.
[sudo] password for eswaribala:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libcrack2 wamerican
The following NEW packages will be installed:
  cracklib-runtime libcrack2 wamerican
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 378 kB of archives.
Selecting previously unselected package libcrack2:amd64.ill be used.
(Reading database ... 32998 files and directories currently installed.)
Preparing to unpack .../libcrack2_2.9.6-3.2_amd64.deb ...
Unpacking libcrack2:amd64 (2.9.6-3.2) ...
Selecting previously unselected package cracklib-runtime.
Preparing to unpack .../cracklib-runtime_2.9.6-3.2_amd64.deb ...
Unpacking cracklib-runtime (2.9.6-3.2) ...
Unpacking wamerican (2018.04.16-1) ...#####################.....................................................................]
Setting up wamerican (2018.04.16-1) ...###############################.....................................................................]
Setting up libcrack2:amd64 (2.9.6-3.2) ...#############################################.....................................]
Setting up cracklib-runtime (2.9.6-3.2) ...#####################################################################.....................]
eswaribala@DESKTOP-55AGI0I:~$
```

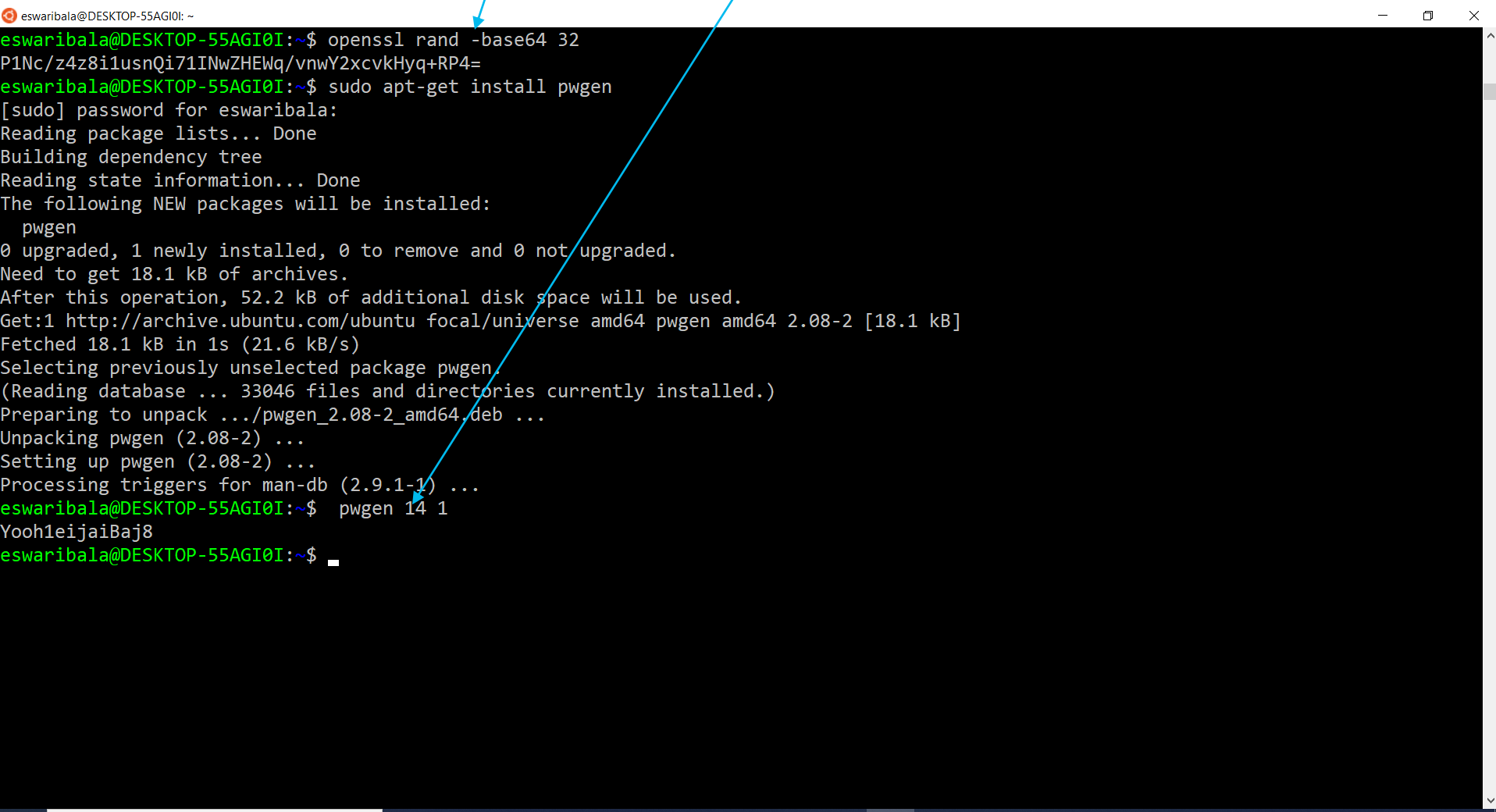
UNIX System Password Security Stength

echo viki | cracklib-check



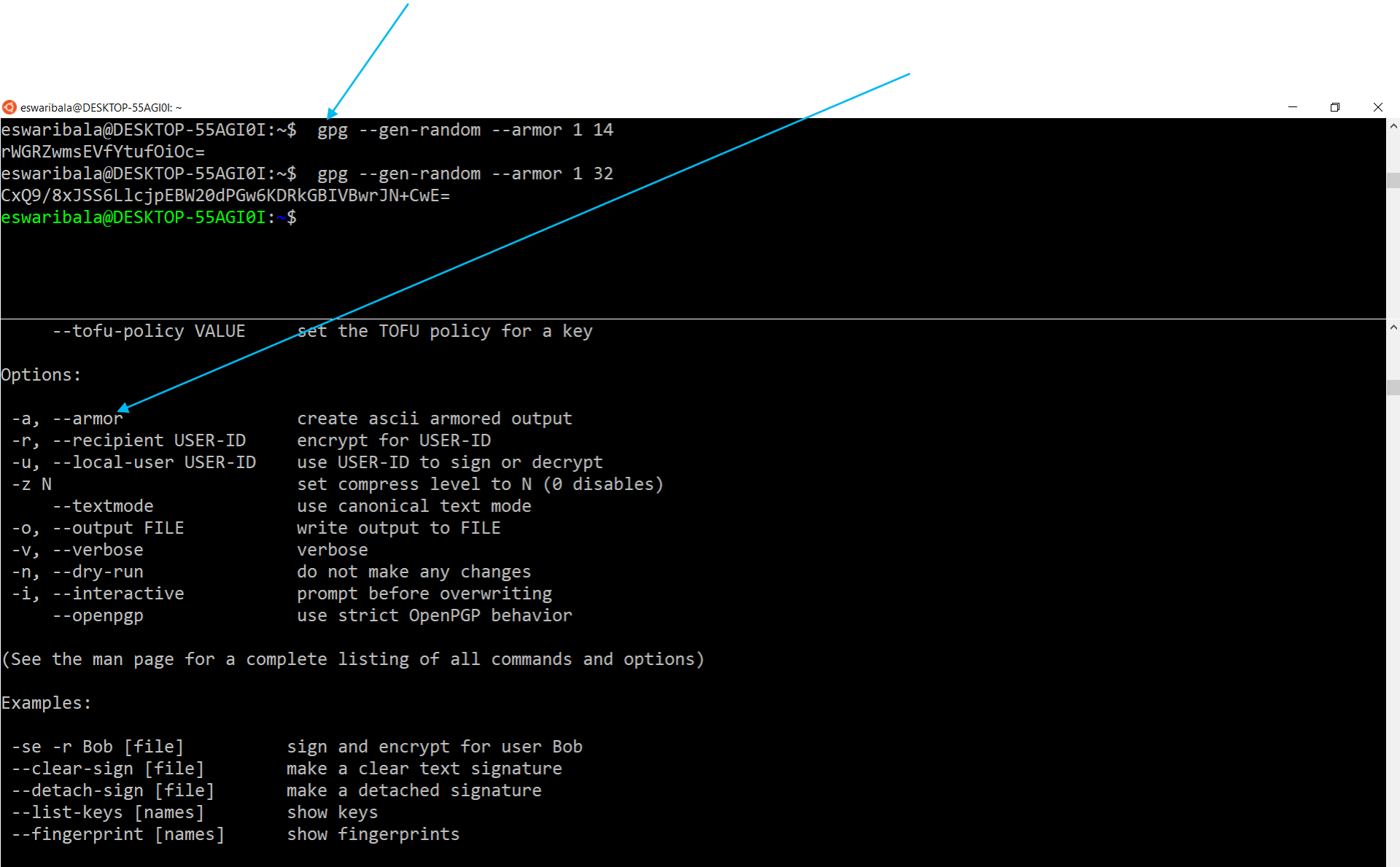
Generate Strong Passwords

UNIX



Generate Strong Passwords

UNIX



Entering Commands

UNIX

```
eswaribala@DESKTOP-55AGI0I: ~
eswaribala@DESKTOP-55AGI0I:~$ date
Tue Aug  4 23:18:16 IST 2020
eswaribala@DESKTOP-55AGI0I:~$ ls -l
total 0
-rwxrwxrwx 1 eswaribala eswaribala 17 Jul 18 17:19 sample.txt
eswaribala@DESKTOP-55AGI0I:~$ sudo ls -l
[sudo] password for eswaribala:
total 0
-rwxrwxrwx 1 eswaribala eswaribala 17 Jul 18 17:19 sample.txt
eswaribala@DESKTOP-55AGI0I:~$ cal
    August 2020
Su Mo Tu We Th Fr Sa
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
eswaribala@DESKTOP-55AGI0I:~$ cal 4 2020
     April 2020
Su Mo Tu We Th Fr Sa
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30

eswaribala@DESKTOP-55AGI0I:~$
```

Entering Commands

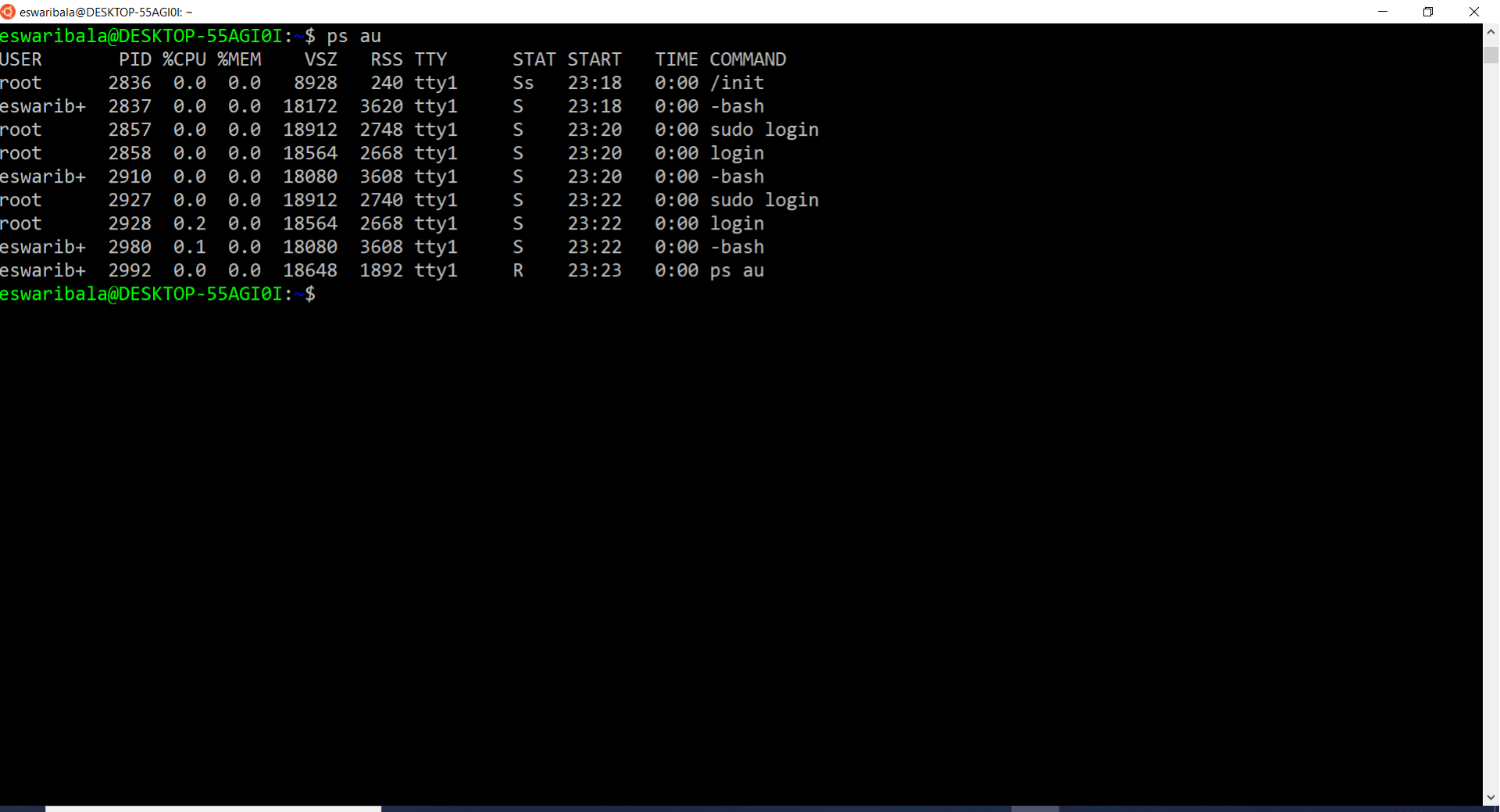
UNIX

Who logged in whoami

Entering Commands

UNIX

Who logged in who –H, w, who –u, ps au



Entering Commands

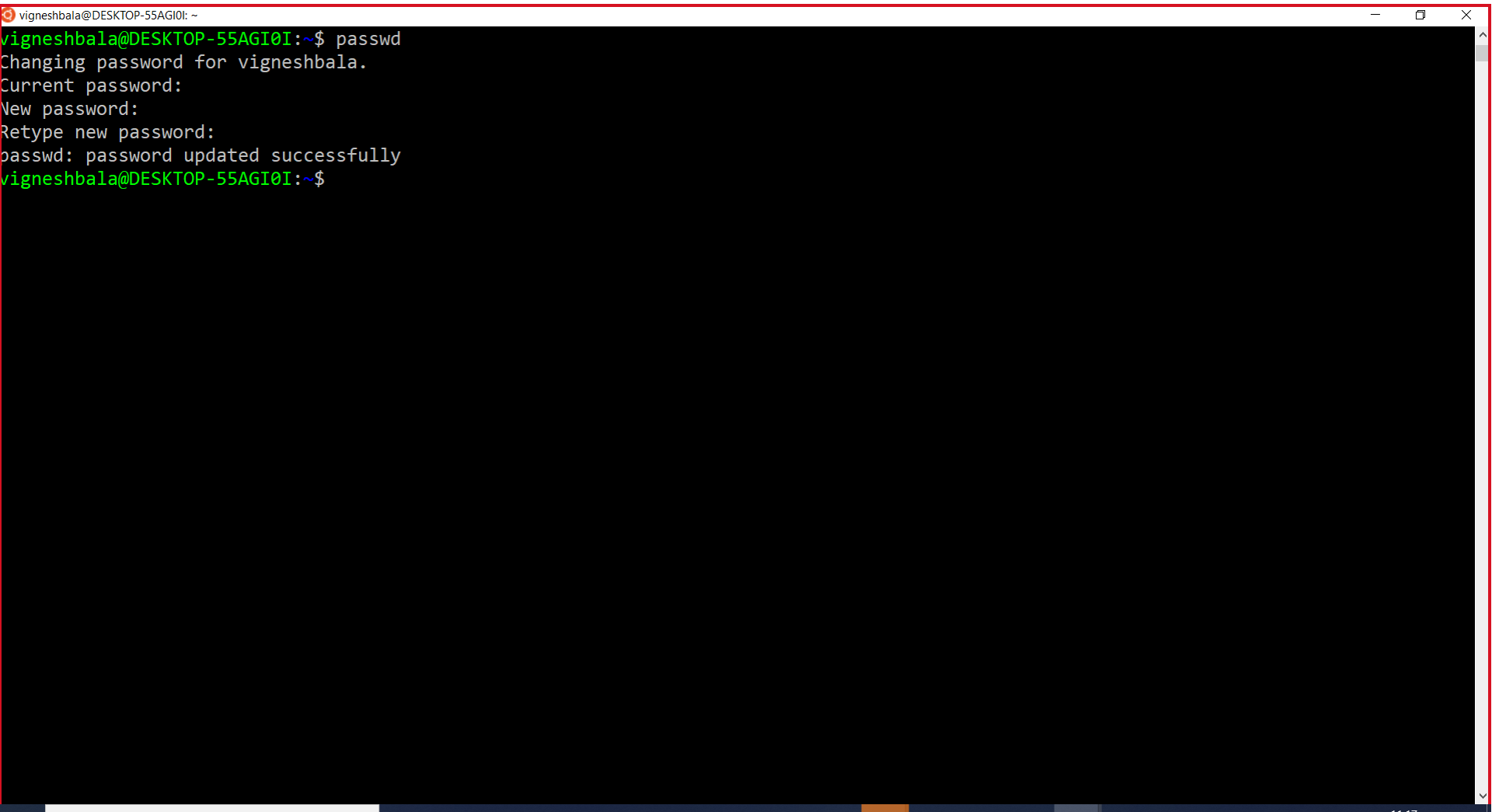
UNIX

eswaribala@DESKTOP-55AGI0I: ~

```
eswaribala@DESKTOP-55AGI0I:~$ sudo apt install finger
[sudo] password for eswaribala:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  finger
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 16.9 kB of archives.
After this operation, 51.2 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu focal/universe amd64 finger amd64 0.17-17 [16.9 kB]
Fetched 16.9 kB in 1s (14.1 kB/s)
Selecting previously unselected package finger.
(Reading database ... 33051 files and directories currently installed.)
Preparing to unpack .../finger_0.17-17_amd64.deb ...
Unpacking finger (0.17-17) ...
Setting up finger (0.17-17) ...
Processing triggers for man-db (2.9.1-1) ...
eswaribala@DESKTOP-55AGI0I:~$ finger eswaribala
Login: eswaribala                       Name:
Directory: /home/eswaribala             Shell: /bin/bash
Last login Tue Aug  4 23:22 (IST) on tty1
No mail.
No Plan.
eswaribala@DESKTOP-55AGI0I:~$
```

Update Password

UNIX



Logout

UNIX

- Logging Out
- When you finish your work session and wish to leave the UNIX system, type exit (or CTRL-D) to log out. After a few seconds, your UNIX system will display the “login:” prompt:
- $ exit
- login:
- This shows that you have logged out, and that the system is ready for another user to log in using your terminal.

Command Summary

UNIX

Command	Use
passwd	Change your password
date	Get the current date and time
cal	Display a calendar
who	List all users who are currently logged in
finger *username*	Get information about *username*
write *username*	Send a chat message to *username*
talk *username*	Open a chat session with *username*
mesg y **mesg n**	Accept or block incoming messages
man *command*	Get information about *command*
mailx **mailx** *address*	Read e-mail messages, or send an e-mail to *address*
exit	Log out of the system
shutdown poweroff	Turn off the machine

Working with Files and Directories

UNIX

- The UNIX file system provides a powerful and flexible way to organize and manage your information.
- Files
- A file is the basic structure that stores information on the UNIX System (and on Windows systems, as well).
- Conceptually, a computer file is similar to a paper document. Technically a file is a sequence of bytes that is stored somewhere on a storage device, such as a hard drive.
- A file can contain any kind of information that can be represented as a sequence of bytes.
- Word processing documents, bitmap images, and computer programs are all examples of files.

Filenames

UNIX

- Every file has a title, called a filename.
- A filename can be almost any sequence of characters, and up to 255 characters long. (On some older versions of UNIX, two filenames are considered the same if the first 14 characters are identical, so be careful if you use long filenames on these systems.)
- You can use any ASCII character in a filename except for the null character (ASCII NUL) or the slash (/), which has a special meaning in the UNIX file system.
- The slash acts as a separator between directories and files.

Filenames

UNIX

- Even though UNIX allows you to choose filenames with special characters, it is a good idea to stick with alphanumeric characters (letters and numbers) when naming files.
- You may encounter problems when you use or display the names of files containing nonalphanumeric characters. In particular, although the following characters can be used in filenames, it is better to avoid them.
- Many of these characters have special meanings in the command shell, which makes them difficult to work with in filenames.

Filenames

UNIX

! (exclamation point)	* (asterisk)	{,} (brackets)
# (pound sign)	? (question mark)	; (semicolon)
& (ampersand)	\ (backslash)	^ (caret)
| (pipe)	(,) (parentheses)	tab
@ (at sign)	‘,” (single or double quotes)	space
$ (dollar sign)	< , > (left or right arrow)	backspace

Filenames

UNIX

Capitalization

- Windows does not distinguish between uppercase and lowercase letters in filenames.
- You could save a file with the name Notes.DOC and find it by searching for notes.doc.
- The UNIX file system, however, is case-sensitive, meaning that uppercase and lowercase letters are distinct. In UNIX, NOTES, Notes, and notes would be three different files.
- If you save a file with the name Music, you will not find it by searching for music. This also applies to commands in UNIX.
- If you are trying to log out with the exit command, typing EXIT will not work.
- By the way, this explains why URLs (web addresses) can be case-sensitive, since the first web server was created on a UNIX-based platform, and many web servers still run UNIX.

Filenames

Filename Extensions

- In Windows, filenames typically consist of a base name, followed by a period and a short filename extension.
- Many Windows programs depend on the extension to determine how to use the file.
- For example, a file named solitaire.exe is considered to be a file named solitaire with the extension .exe, where the .exe extension tells Windows that it is an executable program.
- If the file extension is altered or deleted, it will be more difficult to work with the file in Windows.

Filenames

UNIX

Filename Extensions

- In UNIX, file extensions are conveniences, rather than a necessary part of the filename.
- They can help you remember the content of files, or help you organize your files, but they are usually optional.
- In fact, many UNIX filenames do not have an extension.
- For example, an executable program would typically have a name like solitaire rather than solitaire.exe.
- In addition, filename extensions in UNIX can be longer than three characters.
- For example, some people use .backup to indicate a backup copy of a file, so notes.backup would be an extra copy of the file notes.

UNIX

Filename Extensions

Extension	File Type	Extension	File Type
.au	Audio	*.mpg, .mpeg*	MPEG video
.c	C language source code	*.o*	Object file (compiled and assembled code)
.cc	C++ source code	*.pl*	Perl script
.class	Compiled Java file	*.ps*	PostScript file
.conf	Configuration file	*.py*	Python script
.d	Directory	*.sh*	Bourne shell script
.gif	GIF image	*.tar*	**tar** archive
.gz	Compressed with **gzip**	*.tar.Z, .tar.gz*	Files that have been archived with **tar** and then compressed
.h	Header file for a C program	*.tex*	Text formatted with Tex/LaTeX
.html	Webpage	*.txt*	ASCII text
.jar	Java archive	*.uu, .uue*	Uuencoded file
.java	Java source code	*.wav*	Wave audio
.jpg, .jpeg	JPEG image	*.z*	Compressed with **pack**
.log	Log file	*.Z*	Compressed with **compress**

UNIX

Filename Extensions

- UNIX files can have more than one extension.
- For example, the file book.tar.Z is a file that has first been archived using the tar command (which adds the extension .tar) and then compressed using the compress command (which adds the .Z).
- This enables a single script to both decompress the file and untar it, using the filename as input and parsing each of the extensions to perform the appropriate task.
- The flexibility of filename conventions in UNIX allow for some variation in filenames.
- A program written in Perl could have the filename program.perl, the more frequently used program.pl, or even just the name program.
- You can even create your open file extensions.

Directories

UNIX

- Files contain the information you work with.
- Directories provide a way to organize your files.
- A directory is just a container for as many files as you care to put in it.
- If you think of a file as analogous to a document in your office, then a directory is like a file folder.
- In fact, directories in UNIX are exactly like folders in Windows.
- For example, you may decide to create a directory to hold all of your notes.
- You could name it Notes and use it to hold only files that are your notes, keeping them separated from your e-mail, programs, and other files.

Subdirectories

UNIX

- A directory can also contain other directories.
- A directory inside another directory is called a subdirectory.
- You can create as many subdirectories inside a particular directory as you wish.

Choosing Directory Names

UNIX

- It is a good idea to adopt a convention for naming directories so that they can be easily distinguished from ordinary files.
- Some people give directories names that are all uppercase letters, some use directory names that begin with an uppercase letter, and others distinguish directories using the extension .d or .dir.
- For example, if you decide to use names beginning with an uppercase letter for directories and avoid naming ordinary files this way, you will know that Notes, Misc, Multimedia, and Programs are all directories, whereas note3, misc.note, mm_5, and progmmA are all ordinary files.

The Hierarchical File Structure

UNIX

- Because directories can contain other directories, which can in turn contain other directories, the UNIX file system is called a hierarchical file system.
- Within the UNIX System, there is no limit to the number of files and directories you can create in a directory that you own.
- File systems of this type are often called tree-structured file systems, because each directory allows you to branch off into other directories and files.
- Tree-structured file systems are usually depicted upside-down, with the root of the tree at the top of the drawing.

The Hierarchical File Structure

UNIX

eswaribala@DESKTOP-55AGI0I: ~

```
eswaribala@DESKTOP-55AGI0I:~$ sudo apt install tree
[sudo] password for eswaribala:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  tree
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 43.0 kB of archives.
Selecting previously unselected package tree.sk space will be used.
(Reading database ... 33057 files and directories currently installed.)
Preparing to unpack .../tree_1.8.0-1_amd64.deb ...
Unpacking tree (1.8.0-1) ...
Setting up tree (1.8.0-1) ...
Processing triggers for man-db (2.9.1-1) ...
eswaribala@DESKTOP-55AGI0I:~$ tree /etc
/etc
├── NetworkManager 1s (35.5 kB/s)
│   └── dispatcher.d
│       └── hook-network-manager
├── PackageKit
│   ├── PackageKit.conf
│   └── Vendor.conf
├── X11
│   ├── Xreset
│   ├── Xreset.d
│   │   └── README
│   ├── Xresources
│   │   └── x11-common
│   ├── Xsession
│   ├── Xsession.d
│   │   ├── 20dbus_xdg-runtime
│   │   ├── 20x11-common_process-args
```

The Hierarchical File Structure

UNIX

eswaribala@DESKTOP-55AGI0I: ~

```
^C
eswaribala@DESKTOP-55AGI0I:~$ tree /etc/apache2
/etc/apache2
├── apache2.conf
├── apache2.conf.bak.2020-08-03_191943
├── apache2.conf.save
├── apache2.conf.save.1
├── conf-available
│   ├── charset.conf
│   ├── localized-error-pages.conf
│   ├── other-vhosts-access-log.conf
│   ├── security.conf
│   └── serve-cgi-bin.conf
├── conf-enabled
│   ├── charset.conf -> ../conf-available/charset.conf
│   ├── localized-error-pages.conf -> ../conf-available/localized-error-pages.conf
│   ├── other-vhosts-access-log.conf -> ../conf-available/other-vhosts-access-log.conf
│   ├── security.conf -> ../conf-available/security.conf
│   └── serve-cgi-bin.conf -> ../conf-available/serve-cgi-bin.conf
├── envvars
├── magic
├── mods-available
│   ├── access_compat.load
│   ├── actions.conf
│   ├── actions.load
│   ├── alias.conf
│   ├── alias.load
│   ├── allowmethods.load
│   ├── asis.load
│   ├── auth_basic.load
│   ├── auth_digest.load
│   ├── auth_form.load
│   ├── authn_anon.load
```

Pathnames

UNIX

- If there are two files with the same name, but in different locations in the file system.
- There is a save file in the Email directory, and another file called save in the Work directory.
- In order to distinguish files with the same name, the UNIX System allows you to specify filenames by including the location of the file in the directory tree.
- This type of name is called a pathname, because it is a listing of the directories you travel through along the path you take to get to the file.
- The path through the file system starts at root (/), and the names of directories and files in a pathname are separated by slashes.
- For example, the pathname for one of the save files is
  - /home/raf/Email/save
  - and the pathname for the other is
  - /home/raf/Work/save

Pathnames

UNIX

- Pathnames that trace the path from root to a file are called full (or absolute) pathnames.
- Specifying a full pathname provides a complete and unambiguous name for a file. In a full pathname, the first slash (/) refers to the root of the file system.
- All the other slashes separate the names of directories, until the last slash separates the filename from the name of the directory it's in.
- Using full pathnames can be awkward when there are many levels of directories, as in this filename:
- /home/dkraut/Work/cs106x/Proj_1/lib/Source/strings.c

Pathnames

UNIX

- **Relative Pathnames**
- You do not always have to specify the full pathnames when you refer to files.
- As a convenient shorthand, you can also specify a path to a file relative to your present directory Such a pathname is called a relative pathname.
- Instead of starting with a / for root, the relative pathname starts with the name of a subdirectory For example, suppose you are in your home directory, /home/raf.
- The relative path for the save file in the Email subdirectory is Email/save, and the relative path for the other save file is Work/save.

Pathnames

UNIX

- **Specifying the Current Directory**
  - A single dot (.) is used as a shortcut to refer to the directory you are currently in.
  - This directory is known as the current directory.
- **Specifying the Parent Directory**
  - Two dots (.., pronounced “dot-dot”) refer to the parent directory of the one you are currently in.
  - The parent directory is the one at the next higher level in the directory tree.
  - Because the file system is hierarchical, all directories have a parent directory
  - The dot-dot references can be used many times to refer to things far up in the file system. The following sequence, for example,
  - ../..
  - refers to the parent of the parent of the current directory.
  - If you are in Work, then ../.. is the same thing as the home directory, since home is the parent of raf, which is the parent of Work.

Pathnames

UNIX

- **Specifying a Home Directory**
- A tilde (~) can be used to refer to your home directory (Strictly speaking, this is a feature of the shell.
- These shortcuts can be combined.
- For example, if your home directory is /home/raf, then
- ~/../liz
- refers to the home directory for the user liz.
- You can also use a tilde followed by a login name to refer to another user's home directory For example, the shortcut ~nate refers to the user nate's home directory

UNIX System File Types

UNIX

- The file is the basic unit of the UNIX System. Within UNIX, there are four different types of files:
  - ordinary files, directories, symbolic links, and special files.
- **Ordinary Files**
  - As a user, most of the information that you work with will be stored as an ordinary file.
  - An ordinary file can contain data, such as text for documents or programs.
  - Image files and binary executables are also examples of ordinary files.

UNIX System File Types

UNIX

- **Links**
- Sometimes it is useful to have a file that is accessible from several directories, without making separate copies of the file.
- For example, suppose you are working with someone else, and you need to share information contained in a single data file that each of you can update.
- It would be convenient for each of you to have a copy in your home directory However, you do not want to make separate copies of the file, because it will be hard to keep them in sync.
- A link is not a kind of file but instead is a second name for a file. With a link, only one file exists on the disk, but it may appear in two places in the directory structure.
- This can allow two users to share the same file.
- Any changes that are made to the file will be seen by both users.
- This type of link is sometimes called a hard link, to distinguish it from a symbolic link

UNIX System File Types

UNIX

- **A hard Link**
  - can't cross the file system boundaries (i.e. A hardlink can only work on the same filesystem),
  - can't link directories,
  - has the same inode number and permissions of original file,
  - permissions will be updated if we change the permissions of source file,
  - has the actual contents of original file, so that you still can view the contents, even if the original file moved or removed.

UNIX System File Types

UNIX

- **A hard Link**
  - The ln command creates a link between files, which enables you to make a single file accessible at two or more locations in the directory system.
  - The following links the file project.main in dkraut's home directory with a new file of the same name in the current directory:
  - $ ln /home/dkraut/project.main project.main

UNIX System File Types

UNIX

- **A soft link**
  - can cross the file system,
  - allows you to link between directories,
  - has different inode number and file permissions than original file,
  - permissions will not be updated,
  - has only the path of the original file, not the contents.

UNIX System File Types

UNIX

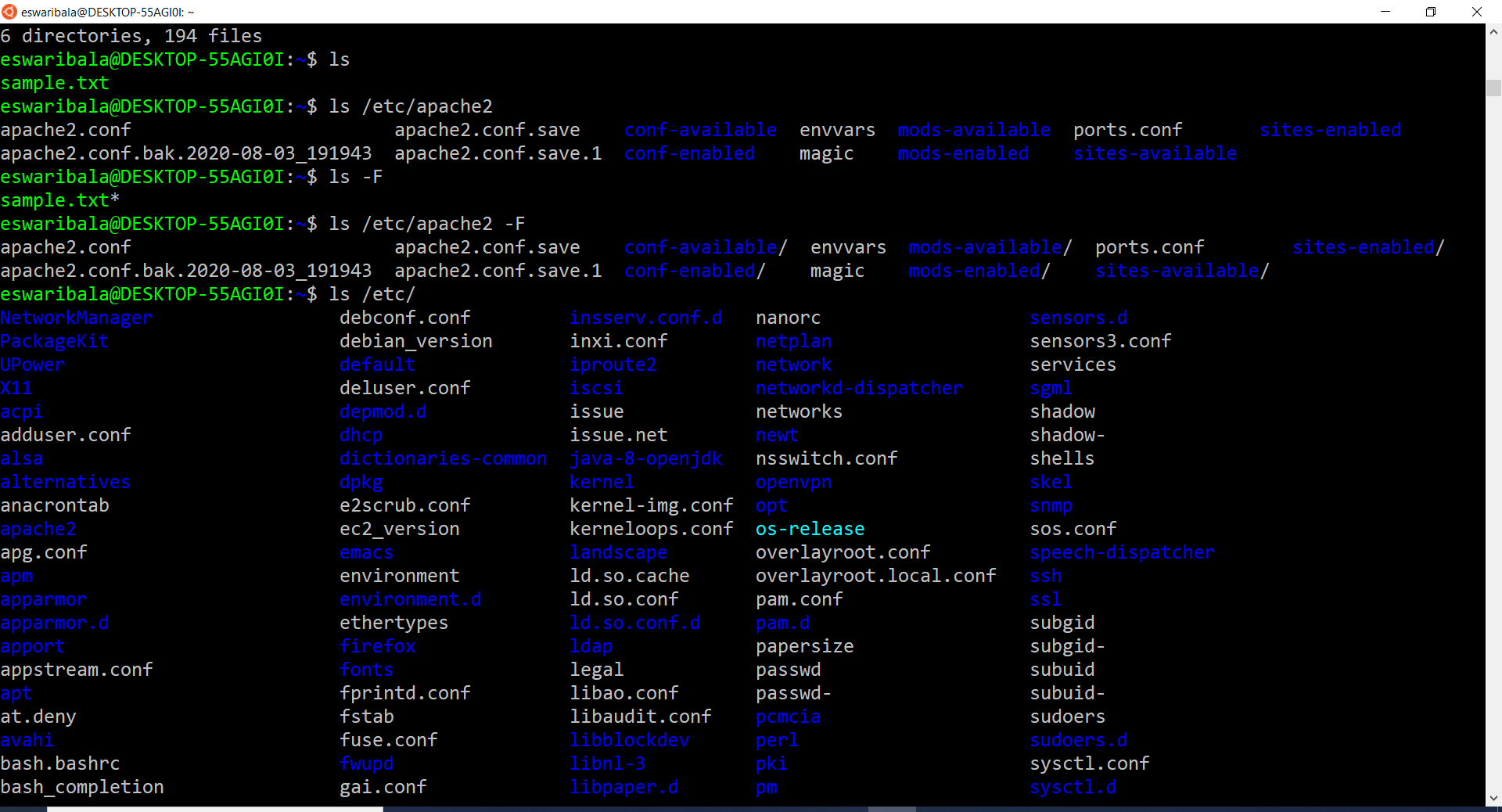
- **A soft link**
  - Symbolic links are created by using the ln command with the -s (symbolic) option.
  - The following example shows how you could use ln to link a file in the /var file system to an entry in one of your directories within the /home file system:
  - $ ln -s /var/X/docs/readme temp/x.readme
  - This will create a symbolic link called x.readme in the temp directory
  - The second argument to ln is optional; if you do not specify the name of the new file, it will create a symbolic link with the same name as the target file. So, for example
  - $ ln -s /usr/bin/firefox/firefox
  - will create a file called firefox in the current directory that is a symbolic link to /usr/bin/firefox/firefox.
  - Symbolic links also enable you to link directories.
  - The command
  - $ ln -s /home/dkraut/work/cs106x/proj1/lib/Source Project

UNIX System File Types

UNIX

- **Listing Directory Contents with Marks**
- When you use the ls command, you do not know whether a name refers to an ordinary file, a program that you can run, or a directory Running the ls command with the -F option produces a list in which the names are marked with symbols that indicate the kind of file that each name refers to.
- Names of directories are listed with / (a slash) following their names. Executable files (those that can be run as programs) are listed with * (an asterisk) following their names.
- Symbolic links are listed with @ (an “at” sign) following their names.
- For instance, suppose that you run ls with the -F option to list the contents of a directory, producing the following result:
- **$ ls -F**
- Email/ notes Projects@
- This example shows that the directory contains the ordinary file notes, the directory Email, and a symbolic link Projects.

UNIX System File Types



UNIX System File Types

UNIX

- **Listing Files in the Current Directory Tree**
- You can add the -R (recursive) option to the ls command to list all the files in your current directory, along with all the files in each of its subdirectories, and so on.
- For example,
  - $ ls -R

UNIX System File Types

UNIX

- **Viewing Files**
- The simplest and most basic way to view a file is with the cat command. cat (short for concatenate) takes any files you specify and displays them on the screen.
- For example, you could use cat to display on your screen the contents of the file review:
- $ cat review

UNIX System File Types

UNIX

- **Viewing Files with Special Characters**
- The cat command recognizes eight-bit characters. In earlier versions of UNIX, it only recognized seven-bit characters.
- This enhancement permits cat to display characters from extended character sets, such as the kanji characters used to represent Japanese words.
- Cat –v sample.txt

UNIX System File Types

UNIX

- **Directing the Output of cat**
- You can send the output of cat to a file as well as to the screen.
- For instance,
  - $ cat physics > physics.backup
- In order to add information to the end of a file, do the following:
  - $ cat notes.august >> notes

UNIX System File Types

UNIX

- **Combining Files and Using Wildcards**
- You can use cat to combine a number of files into one. For example, consider a directory that contains material being used in writing a chapter, as follows:
- $ cat section1 section2 section3 > chapter.3
- $ cat section* > chapter.3
- $ cat *1 *2 > temp

UNIX System File Types

UNIX

- **Creating a File**
- So far, all the examples you have seen involved using cat to copy one or more normal files, either to another file or to your screen. But other possibilities exist.
- Just as your screen is the default output for cat and other commands, your keyboard is the default input.
- If you do not specify a file to use as input, cat will simply copy everything you type to its output.
- This provides a way to create simple files without using an editor.
- The command
  - $ cat > names
  - Nate nate@engineer.com
  - Rebecca rlf@library.edu
  - CTRL-D

UNIX System File Types

UNIX

- Using cat in this way (cat > names) creates the file names if it does not already exist and overwrites (replaces) its contents if it does exist.
- You can use cat to add material to a file as well. For example,
- **$ cat >> names**
- **Dan dkraut@bio.ca.edu**
- **CTRL-D**

UNIX System File Types

- Another command, touch, can also be used to create a file.
- $ touch notes

UNIX System File Types

- Moving Around in Directories

UNIX System File Types

- Moving and Renaming Files and Directories

```
eswaribala@DESKTOP-55AGI0I: ~
eswaribala@DESKTOP-55AGI0I:~$ pwd
/home/eswaribala
eswaribala@DESKTOP-55AGI0I:~$ cd /etc
eswaribala@DESKTOP-55AGI0I:/etc$ cd /etc/apache2
eswaribala@DESKTOP-55AGI0I:/etc/apache2$ cd ..
eswaribala@DESKTOP-55AGI0I:/etc$ cd ../ ..
-bash: cd: too many arguments
eswaribala@DESKTOP-55AGI0I:/etc$ cd ../..
eswaribala@DESKTOP-55AGI0I:/$ cd ~
eswaribala@DESKTOP-55AGI0I:~$ pwd
/home/eswaribala
eswaribala@DESKTOP-55AGI0I:~$ mkdir files
eswaribala@DESKTOP-55AGI0I:~$ ls
files  notes  sample.txt  software
eswaribala@DESKTOP-55AGI0I:~$ mv notes /files
mv: cannot move 'notes' to '/files': Permission denied
eswaribala@DESKTOP-55AGI0I:~$ sudo mv notes /files
[sudo] password for eswaribala:
eswaribala@DESKTOP-55AGI0I:~$ ls
files  sample.txt  software
eswaribala@DESKTOP-55AGI0I:~$
```

For example, the following command moves three files to the subdirectory called TermPaper:
$ mv section1 section2 section3 TermPaper

UNIX System File Types

UNIX

- **To protect mv overwrites the file**
- The following shows what happens if you try to use mv -i to rename the file totals to data when the data file already exists:
- $ mv -i totals data
- mv: overwrite data?

UNIX System File Types

UNIX

- **Moving Directories**
- You can use a single mv command to move a directory and all of its files and subdirectories just as you'd use it to move a single file.
- For example, if the directory Final contains all of your finished work on a document, you can move it to a directory in which you keep all of the versions of that document, Project, as shown here:
- $ ls Project
- Drafts
- $ mv Final Project
- $ ls Project
- Drafts Final

UNIX System File Types

UNIX

- **Copying Files**
- The cp command is similar to mv, except that it copies files rather than moving or renaming them.
- Cp follows the same model as mv: you name the files to be copied first and then give the destination.
- The destination can be a directory, a pathname for a file, or a new file in the current directory.
- The following command makes a backup copy of seattle and names the copy seattle.bk:
- $ cp seattle seattle.bk
- cp -p unix.txt{,.bak.$(date +%F_%H%M%S)}

UNIX System File Types

UNIX

- **Copying the Contents of a Directory**
- If you try to copy a directory, you will get an error message.
- A feature of cp (found on most versions of UNIX) is the -r (r ecursive) option that lets you copy an entire directory structure. Suppose you have a directory called Project, and you wish to make a backup copy
- The following command creates a new directory, called Project.Backup, and copies all of the files and subdirectories in Project to the new directory:
- $ cp -r Project Project.Backup

UNIX System File Types

UNIX

- **Removing Files**
- To get rid of files you no longer want or need, use the rm (remove) command. rm deletes the named files from the file system, as shown in the following example:
- $ ls
- notes research temp
- $ rm temp
- $ ls
- notes research

UNIX System File Types

UNIX

- **Removing Multiple Files**
- The rm command accepts several arguments and takes several options.
- If you specify more than one filename, it removes all of the files you named.
- The following command removes the two files left in the directory:
- $ rm notes research
- $ ls
- $
- The following will remove all files in the current directory:
- $ rm *

UNIX System File Types

UNIX

- **Safely Removing Files**
- Almost every user has accidentally deleted files.
- In the preceding example, if you accidentally hit the SPACEBAR between the * and the extension and type
- $ rm * .rlf
- you will delete all of the files in the current directory As typed, this command says to remove all files (*), and then remove a file named .rlf.

UNIX System File Types

UNIX

- **Safely Removing Files**
- To avoid accidentally removing files, use rm with the -i (interactive) option.
- When you use this option, rm prints the name of each file and waits for your response before deleting it.
- To go ahead and delete the file, type y. Responding n or hitting ENTER will keep the file rather than deleting it.
- For example, in a directory that contains the files notes, research, and temp, the interactive option to rm gives you the following:
- $ rm -i *
- notes: y
- research: <ENTER>
- temp: y
- Your responses cause rm to delete both notes and temp, but not research.

UNIX System File Types

UNIX

- **Restoring Files**
- When you remove a file using the rm command, it is gone. If you make a mistake, you can only hope that the file is available somewhere on a backup file system (on a tape or disk).
- You can call your system administrator and ask to have the file you removed, say /home/you/Work/temp, restored from backup.
- If it has been saved, it can be restored for you.
- Systems differ widely in how, and how often, they are backed up.
- On a heavily supported system, all files are copied to a backup system every day and saved for some number of days, weeks, or months.
- On some systems, backups are done less frequently, perhaps weekly On personal workstations, backups occur when you get around to doing them.

UNIX System File Types

- **Creating a Directory**
- You can create new directories in your file system with the mkdir (make dir ectory) command. It is used as follows:
- $ pwd
- Work
- $ ls
- notes research temp
- $ mkdir New
- $ ls
- notes New research temp

UNIX System File Types

UNIX

- **Removing a Directory**
- There are two ways to remove or delete a directory If the directory is empty (it contains no files or subdirectories), you can use the rmdir (remove dir ectory) command.
- If you try to use rmdir on a directory that is not empty, you'll get an error message.
- The following removes the directory New added in the preceding example:
- $ rmdir New

UNIX System File Types

UNIX

- **Removing a Directory**
- To remove a directory that is not empty, together with all of the files and subdirectories it contains, use rm with the -r (r ecursive) option, as shown here:
- $ rm -r Work
- The -r option instructs rm to delete all of the files it finds in Work and then go to each of the subdirectories and delete all of their files, and so forth, concluding by deleting Work itself.
- Since rm -r removes all of the contents of a directory, be very careful in using it. You can add the –I option to step through all the files and directories, removing or leaving them one at a time.
  - $ rm -ir Work
  - rm: descend into directory 'Work'? y
  - rm: remove regular empty file 'Work/final'? y
  - rm: remove regular empty file 'Work/save'? <RETURN>
  - $ ls Work
  - save

UNIX System File Types

UNIX

- **Getting Information About File Types**
- Sometimes you just want to know what kind of information a file contains.
- For example, you may decide to put all your shell scripts together in one directory You know that several scripts are scattered about in several directories, but you don’t know their names, or you aren’t sure you remember all of them.
- Or you may want to print all of the text files in the current directory, whatever their content.
- You can use several of the commands already discussed to get limited information about file contents.
- For example, ls -l shows you if a file is executable-either a compiled program or a shell script (batch file). But the most complete and most useful command for getting information about the type of information contained in files is file.
- file reports the type of information contained in each of the files you give it.
- The following shows typical output from using file on all of the files in the current directory:
- $ file *

UNIX System File Types

UNIX

- **Searching for Files**
- The command locate searches for a pattern in a database of filenames. For example,
- $ locate pippin
- searches the database for filenames containing the string “pippin”.
- The database contains the full pathname for each file, so this would find files in the directory pippin-photos as well as files such as
- 0915-pippin.jpg.
- The locate command is very fast and easy to use. However, it will only work if the database of filenames is kept up to date.
- On many systems, the database is automatically updated once per day

UNIX System File Types

UNIX

- **Using find**
- The find command searches through the contents of one or more directories, including all of their subdirectories.
- You have to tell find in which directory to start its search.
- The following example
- searches user jmf's directory system for the file new_data and prints the full pathname of any file with that name that it finds:
- $ pwd
- /home/jmf
- $ find . -name new_data -print
- /home/jmf/Dir/Logs/new_data
- /home/j mf/Cmds/new_data

UNIX System File Types

UNIX

- **Using find**
- To search the entire file system, start in the system's root directory, represented by the /:
- $ find / -name new_data –print
- $ find . /tmp/project -name new_data –print
- $ find -name "*data" –print
- $ find / -name new_data -print > found &
- $ find . -name "music" -u sue -mtime +7 -print

UNIX System File Types

- **List Hidden Files**
- To see all files in this directory, use ls -a:
- $ ls -a
- . .. .mailrc .profile Email notes Work

UNIX System File Types

UNIX

- **Controlling the Way Is Displays Filenames**
- You can use the -x option to have names of files displayed horizontally, in as many lines as necessary
- For example,
- $ ls –x
- You also can use the -1 (one) option to have files displayed one line per row (as the old version of ls
- did), in alphabetical order:
- $ ls -l

UNIX System File Types

UNIX

- **Controlling the Way ls Displays Filenames**
- Combining Options to ls
- You can use more than one option to the ls command simultaneously For example, the followingshows the result of using the ls command with the options -F and -a on a home directory:
- $ ls –aF
- $ ls –Fat
- example of what the long format of ls might look like:
- $ ls -1

UNIX System File Types

UNIX

- **Controlling the Way Is Displays Filenames**

The first character in each line tells you what kind of file this is.

-	Ordinary file	**c**	Special character file
d	Directory	**l**	Symbolic link
b	Special block file	**P**	Named pipe special file

UNIX System File Types

UNIX

- **Unix Special Characters Or Metacharacters For File Manipulation**
- **Unix Filename Wildcards – Metacharacters**
- #1) '*' – any number of characters:
- This wild-card selects all the files that matches the expression by replacing the asterisk-mark with any set of zero or more characters.
- Example1: List all files that start with the name 'file'. g. file, file1, file2, filenew
- $ ls file*
- Example2: List all files that end with the name 'file'. g. file, afile, bfile, newfile
- $ ls *file

UNIX System File Types

UNIX

- **Unix Special Characters Or Metacharacters For File Manipulation**
- **Unix Filename Wildcards – Metacharacters**
- #2) ‘?’ – single character:
- This wild-card selects all the files that matches the expression by replacing the question-mark with any one character.
- Example1: List all files that have one character after ‘file’. g. file1, file2, filea
- $ ls file?
- Example2: List all files that have two characters before ‘file’. g. dofile, tofile, a1file
- $ ls ??file

UNIX System File Types

UNIX

- **Unix Special Characters Or Metacharacters For File Manipulation**
- **Unix Filename Wildcards – Metacharacters**
- #3) '[' range ']' – single character from a range:
- This wild-card selects all the files that matches the expression by replacing the marked range with any one character in the range.
- Example1: List all files that have a single digit after 'file'. g. file1, file2
- $ ls file[0-9]
- Example2: List all files that have anyone letter before 'file'. g. afile, zfile
- $ ls [a-z]file

UNIX System File Types

- **Unix Special Characters Or Metacharacters For File Manipulation**
- **Unix Filename Wildcards – Metacharacters**
- # 4) '[' range ']*' – multiple characters from a range:
- This wild-card selects all the files that matches the expression by replacing the marked range with one or more characters from the range.
- Example1: List all files that have digits after 'file'. g. file1, file2, file33
- $ ls file[0-9]*

UNIX System File Types

UNIX

- **Permissions**
- The UNIX file system is designed to support multiple users.
- When many users are sharing one file system, it is important to be able to restrict access to certain files.
- The system administrator wants to prevent other users from changing important system files, for example, and many users have private files that they want to restrict others from viewing.
- File permissions are designed to address these needs.

UNIX System File Types

UNIX

- **Permissions for Files**
- There are three classes of file permissions, for the three classes of users: the owner (or user) of the file, the group the file belongs to, and all other users of the system.
- The first three letters of the permissions field, as seen in the output from ls -l, refer to the owner's permissions;
- the second three letters refer to the permissions for members of the file's group;
- and the last three to the permissions for any other users.

UNIX System File Types

UNIX

- **Permissions for Files**
- In the entry for the file named notes in the ls -l example shown in the preceding section, the first three letters, rwx, show that the owner of the file can read (r) it, write (w) to it, and execute (x) it.
- The second group of three characters, r-x, indicates that members of the group can read and execute the file but cannot write to it.
- The last three characters, r-x, show that all others can also read and execute the file but not write to it.
- If you have read permission for a file, you can view its contents.
- Write permission means that you can alter its contents.
- Execute permission means that you can run the file as a program.

UNIX System File Types

UNIX

- **Special Permissions**
- There are a few other codes that occasionally appear in permission fields.
- For example, the letter s can appear in place of an x in the user's or group's permission field.
- This s refers to a special kind of execute permission that is relevant primarily for programmers and system administrators.

UNIX System File Types

UNIX

- **The chmod Command**
- In the ls -l example, all of the files and directories have the same permissions set.
- Anyone on the system can read or execute any of them, but other users are not allowed to write, or alter, these files.
- Normally you don’t want all your files set up this way.
- You will often want to restrict other users from being able to view your files, for example.
- At times, you may want to allow members of your work group to edit certain files, or even make some files public to anyone on the system.

UNIX System File Types

UNIX

- **The chmod Command**
- The UNIX System allows you to set the permissions of each file you own.
- Only the owner of a file or the superuser can alter the file permissions.
- You can independently manipulate each of the permissions to allow or prevent reading, writing, or executing by yourself, your group, or all users.

UNIX System File Types

UNIX

- **The chmod Command**
- To alter a file's permissions, you use the chmod (change mode) command.
- You specify the changes you want to make with a sort of code.
- First, show which set of permissions you are changing with u for user, g for group, or o for other.
- Second, specify how they should be changed with + (to add permission) or − (to subtract permission).
- Third, list the permissions to alter: r for read, w for write, or x for execute.
- Finally, specify the file or files that the changes refer to.

UNIX System File Types

UNIX

- **The chmod Command**
- $ ls -1 quotations
- -rwxr-xr-x 1 nate group1 346 Apr 27 03:32 quotations
- $ chmod go-rx quotations
- $ ls -1 quotations
- -rwx 1 nate group1 346 Apr 27 03:32 quotations
- $ chmod ugo+rwx quotations
- $ ls -1 quotations
- -rwxrwxrwx 1 nate group1 346 Apr 27 03:32 quotations

UNIX System File Types

UNIX

- **The chmod Command**
- Setting Absolute Permissions
- $ chmod 700 quotations
- $ ls -1 quotations
- -rwxrwxrwx 1 nate group1 346 Apr 27 03:32 quotations

	Owner	Group	Other
Read	4	0	0
Write	2	0	0
Execute	1	0	0
Sum	7	0	0

UNIX System File Types

UNIX

- **The chmod Command**
- $ chmod go-rwx *
- $ chmod 700 *
- $ chmod -R u+r Email

UNIX System File Types

UNIX

- **Using umask to Set Permissions**
- The chmod command allows you to alter permissions on a file-by-file basis.
- The umask command allows you to do this automatically when you create any file or directory.
- Everyone has a default umask setting that is either set up either by the system administrator or included in a shell configuration file.

UNIX System File Types

UNIX

- **Using umask to Set Permissions**
- With the umask command, you specify the permissions that will be given to all files created after issuing the command.
- This means you will not have to worry about the file permissions for each individual file you create.
- Unfortunately, using umask to specify permissions is a little bit complicated.

UNIX System File Types

UNIX

- **Using umask to Set Permissions**
- There are two rules to remember:
- umask uses a numeric code for representing absolute permissions just as chmod does.
- For example, 777 means read, write, and execute permissions for user, group, and others (rwxrwxrwx).
- You specify the permissions you want by telling umask what to subtract from the full permissions value, 777 (rwxrwxrwx).

UNIX System File Types

UNIX

- **Using umask to Set Permissions**
- For example, after you issue the following command, all new files in this session will be given permissions of rwxr-xr-x:
- $ umask 022
- In this example, we want the new files to have the permission value 755. When we subtract 755 from 777, we get 022.
- This is the “mask” we used for the command.

UNIX System File Types

UNIX

- **Changing the Owner of a File**
- Every file has an owner. When you create a file, you are automatically its owner.
- The owner usually has broader permissions for manipulating the file than other users.
- Sometimes you need to change the owner of a file; for example, if you take over responsibility for a file that previously belonged to another user.
- Even if someone else “gives” you a file by moving it to your directory that does not make you the owner.
- One way to become the owner of a file is to make a copy of it-when you make a copy, you are the owner of the new file.

UNIX System File Types

UNIX

- **Changing the Owner of a File**
- However, changing ownership by copying only works when the new owner copies the file from the old owner, which requires the new owner to have read permission on the file.
- A simpler and more direct way to transfer ownership is to use the chown (change owner) command.
- The chown command takes two arguments: the login name of the new owner and the name of the file.
- The following makes liz the new owner of the file contact_info:
- $ chown liz contact_info
- Only the owner of a file (or the superuser) can use chown to change its ownership
- chown -R liz Project

UNIX System File Types

UNIX

- **Changing the Group of a File**
- Groups are meant to help sets of users who need to share files more closely than other users on the system.
- For example, all the students taking a particular class may belong to the same group, so that they can more easily share files when they collaborate on projects

UNIX System File Types

UNIX

- **Changing the Group of a File**
- Every file belongs to a group.
- Sometimes, such as when new groups are set up on a system or when files are copied to a new system, you may want to change the group to which a particular file belongs.
- This can be done using the chgrp (change group) command.
- **The chgrp command takes two arguments, the name of the new group and the name of the file.**
- The following command changes data_file so that it belongs to the group students:
- $ chgrp students data_file

UNIX System File Types

UNIX

- You can use the -R (r ecursive) option of chgrp to change the group to which all the files in a directory belong.
- It works just like the -R option of chown.

UNIX System File Types

UNIX

- **Viewing Long Files**
- When you use cat to display a file, it prints the contents on your screen without pausing, so that long files quickly scroll past.
- A quick solution, when you only need to view a small part of the file, is to use cat and then hit BREAK when the part you want to read comes on the screen.
- This stops the program, but it leaves the output on the screen, so if your timing is good, you may get what you want.

UNIX System File Types

UNIX

- **Viewing Long Files**
- A somewhat better solution is to use the sequence CTRL-S, to make the output pause whenever you get a screen you want to look at, and CTRL-Q to resume scrolling.
- This way of suspending output to the screen works for all UNIX commands, not just cat.

UNIX System File Types

UNIX

- **Viewing Long Files**
- The best solution is to use a pager -a program that is designed specifically for viewing files.
- UNIX gives you a choice of two pagers, pg and more, which are standard with all versions of UNIX, as well as an enhanced pager called less, available for many versions of UNIX, including Linux.
- Less has more features than more and has pretty much replaced it.

UNIX System File Types

UNIX

- **Viewing Long Files (more)**
- To display the file newyork, just enter the command
- $ more newyork
- To tell more to move ahead by a screen, press the SPACEBAR.
- To move ahead one line, press ENTER.
- The commands for half-screen motions, d and CTRL-D, are the same as in pg.
- To move backward by a screen, use b or CTRL-B.

UNIX System File Types

UNIX

- **Viewing Long Files (less)**
- The less command, an enhanced version of the more command, is a feature-rich pager that can be used to interactively display portions of a file.
- It can be used to move either forward or backward in a file.
- Since less reads in portions of files, rather than entire files, it is very efficient for large files.
- This will display the file newyork with less:
- $ less newyork

UNIX System File Types

UNIX

- **Viewing Long Files (less)**
- The less command has many useful options.
- For example, the -p option can be used to start less at the first occurrence of the pattern you specify (where the pattern is entered after the -p option)
- To display the file sanfrancisco, beginning with the first time the **pattern “SFO" appears**, you can use
- $ **less -p SFO sanfrancisco**
- Among the other options supported by less are -s, which squeezes consecutive blank lines into a single blank line;
- -S, which chops off lines longer than the screen (discarding them instead of folding them into the next line); and -U, which displays backspaces and carriage returns as control characters.

UNIX System File Types

UNIX

- **Viewing Long Files (less)**
- Many commands can be used with the less pager to display different parts of a file.
- For example, you can scroll forward one window by entering SPACEBAR or f, and you can scroll backward one window by entering b.
- You can scroll forward one line by entering e or pressing ENTER, and you can scroll backward one line by entering y.
- You can scroll to the next occurrence of the string “pattern” using the command /pattern, and you can scroll to the preceding occurrence of this string using the command ? pattern.

UNIX System File Types

UNIX

- **Viewing Long Files (less)**
- You can find the next and preceding lines that do not contain the string “pattern” using the commands /!pattern and ?!pattern, respectively.
- You can use various commands to move the cursor when using less to display files.
- For example, you can move one space left using the LEFT ARROW key or ESC-H; you can move one space right using the RIGHT ARROW key or ESC-L.
- You can move one word to the left with ESC-B and one word to the right with ESC-W.
- You can also do some editing: BACKSPACE deletes the character to the left of the cursor, DELETE deletes the character under the cursor, CTRL-BACKSPACE deletes the word to the left of the cursor, and CTRL-DELETE deletes the word under the cursor.

UNIX System File Types

UNIX

- **Viewing the Beginning or End of a File**
- The head and tail commands are specifically designed for these jobs.
- head shows you the beginning of a file, and tail shows you the end.
- For example, the command shown here displays the first ten lines of transactions.
- $ head transactions

and the following command displays the last ten lines:

- $ tail transactions

UNIX System File Types

UNIX

- **Viewing the Beginning or End of a File**
- To display some other number, say the last three lines, you give head or tail a numerical argument.
- This command shows only the last three lines:
- $ tail -3 transactions
- A useful feature of tail is the -f (f ollow) option.
- **This lets you use tail to check on the progress of a program that writes its output to a file.**
- Suppose a file transfer program is getting information from a remote system and putting it in the file newdata.
- In this example, tail displays the last three lines of newdata, waits (sleeps) for a short time, looks to see if there has been any new input, displays any new lines, and so on:
- $ tail -3 -f newdata

UNIX System File Types

UNIX

- **Printing Files**
- The UNIX System includes a collection of programs, called the lp system, for printing files and documents.
- You can use it to print everything from simple text files to large documents with complex formats.
- It provides a simple, uniform interface to a wide variety of printers.
- The lp system is itself large and complex, but fortunately its complexity is well hidden from users.

UNIX System File Types

UNIX

- **Printing Files**
- In fact, three basic commands, lp, lpstat, and cancel, are all you need to know to use this system.
- (On Linux, these three basic commands have different names; they are lpr, lpq, and lprm, respectively)

UNIX System File Types

UNIX

- **Sending Output to the Printer**
- The basic command for printing a file is lp (line printer) (or on Linux, lpr). This command prints the file research.nov as follows:
- $ lp research.nov
  - request id is lsr1–142 (1 file)
- You can print several files at once by including all of them in the arguments to lp. For instance,
- $ lp res*
  - request id is lsr1–154 (3 files)

UNIX System File Types

UNIX

- **Specifying a Printer**
- The lp command does not ask you which printer to use.
- There may be several printers on the system, but one of them will be the system default.
- Unless you specify otherwise, this is the printer that lp uses.
- To find out which printers are available, you can ask your system administrator

UNIX System File Types

UNIX

- **Specifying a Printer**
- The lp command does not ask you which printer to use.
- There may be several printers on the system, but one of them will be the system default.
- Unless you specify otherwise, this is the printer that lp uses.
- To find out which printers are available, you can ask your system administrator

UNIX System File Types

UNIX

- **Specifying a Printer**
- To specify a particular printer, use the –d (destination) option, followed by the printer's name.
- For example,
- $ lp -d laser2 flightinfo
- sends flightinfo to the printer named laser2.

UNIX System File Types

UNIX

- **Print Spooling**
- When you print a file on the UNIX System, you do not have to wait until the file is printed (or until it is sent to the printer) before continuing with other work, and you do not have to wait until one print job is finished before sending another.
- lp spools its input to the UNIX print system, which means that it tells the print system what file to print and how to print it, and then leaves the work of getting the file through the printer to the system.

UNIX System File Types

UNIX

- **Print Spooling**
- Your job is submitted and spooled, but it is not printed at the precise time you enter the lp command, and lp does not automatically tell you when your job is actually finished.
- If you want to be notified when it is printed, use the -m (mail) option.
- For example,
- $ lp -m -d laser2 flightinfo
- sends you mail when your file is successfully printed.

UNIX System File Types

UNIX

- **Print Spooling**
- If you change the file between the time you issue the lp command and the time it actually goes to the printer, it is the changed file that will be printed.
- In particular, if you delete the file, or rename it, or move it to another directory the print system will not find it, and it will not be printed.
- To avoid this, use the -c (copy) option.
- The command
- $ lp -c -d laser2 flightinfo
- copies flightinfo to a temporary file in the print system and uses that copy as the input to the printer.
- Any changes you make to flightinfo after you issue this command will not appear in the printed output.

UNIX System File Types

UNIX

- **Using lpstat to Monitor the Print System**
- The lpstat command (on LINUX this is the lpq command) provides a way to get this and other useful information, such as which printers are currently available on the system and how many other print jobs are scheduled.
- One of the most important uses of lpstat is to see if your print jobs are being taken care of or if there is some problem with the system.
- The following shows that a job is scheduled for printing but has not yet started printing:
- $ lpstat

UNIX System File Types

UNIX

- **Canceling Print Jobs**
- cancel inkjet-133

UNIX System File Types

Command	Use	Command	Use
ls	List the contents of a directory	**locate**	Search for files by name
cat	Display a short file	**find**	Find files
touch	Create an empty file	**chmod**	Change file permissions
pwd	Show the present directory	**umask**	Set default file permissions
cd	Change present directory	**chown**	Change the owner of a file or directory
mv	Move a file or directory	**chgrp**	Change the group of a file or directory
cp	Copy a file or directory	**pg, more, less**	Display a file
ln	Create a link	**head**	Display beginning of a file
rm	Remove a file or directory	**tail**	Display end of a file
mkdir	Make a directory	**lp/lpr**	Print a file
rmdir	Remove an empty directory	**lpstat/lpq**	Check status of a print job
file	Get file information	**cancel/lprm**	Cancel a print job

A list of the commonly used variables in Linux

System Variable	Meaning	To View Variable Value Type
BASH_VERSION	Holds the version of this instance of bash.	echo $BASH_VERSION
HOSTNAME	The name of the your computer.	echo $HOSTNAME
CDPATH	The search path for the cd command.	echo $CDPATH
HISTFILE	The name of the file in which command history is saved.	echo $HISTFILE
HISTFILESIZE	The maximum number of lines contained in the history file.	echo $HISTFILESIZE
HISTSIZE	The number of commands to remember in the command history. The default value is 500.	echo $HISTSIZE
HOME	The home directory of the current user.	echo $HOME

A list of the commonly used variables in Linux

System Variable	Meaning	To View Variable Value Type
FS	The Internal Field Separator that is used for word splitting after expansion and to split lines into words with the read builtin command. The default value is <space><tab><newline>.	echo $IFS
LANG	Used to determine the locale category for any category not specifically selected with a variable starting with LC_.	echo $LANG

A list of the commonly used variables in Linux

System Variable	Meaning	To View Variable Value Type
PATH	The search path for commands. It is a colon-separated list of directories in which the shell looks for commands.	echo $PATH
PS1	Your prompt settings.	echo $PS1

A list of the commonly used variables in Linux

System Variable	Meaning	To View Variable Value Type
TMOUT	The default timeout for the read builtin command. Also in an interactive shell, the value is interpreted as the number of seconds to wait for input after issuing the command. If not input provided it will logout user.	echo $TMOUT
TERM	Your login terminal type.	echo $TERM export TERM=vt100

A list of the commonly used variables in Linux

System Variable	Meaning	To View Variable Value Type
SHELL	Set path to login shell.	echo $SHELL
DISPLAY	Set X display name	echo $DISPLAY export DISPLAY=:0.1
EDITOR	Set name of default text editor.	export EDITOR=/usr/bin/vim

A list of the commonly used variables in Linux

What is SUID and SGID?

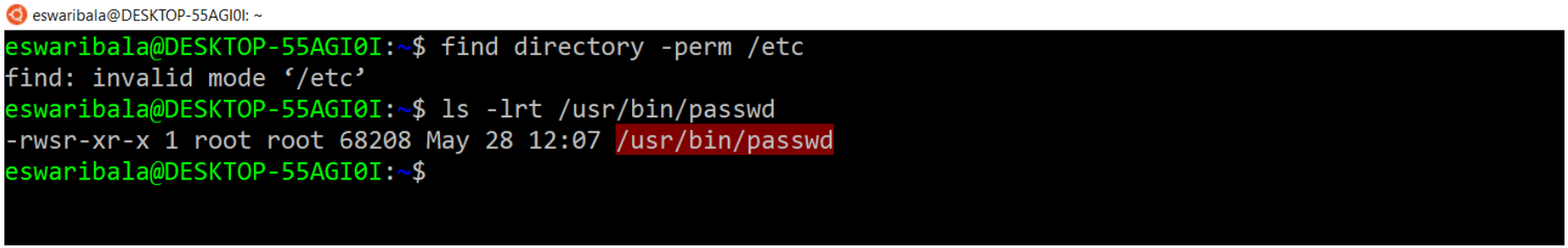
UNIX

- There are 3 special permission that are available for executable files and directories. These are :
- 1. SUID permission
- 2. SGID permission
- 3. Sticky bit

What is SUID and SGID?

UNIX

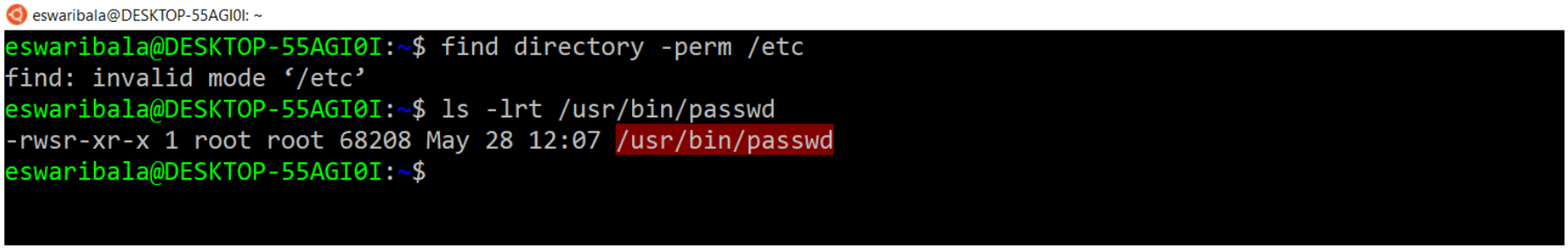
- **Set-user Identification (SUID)**
- Have you ever thought, how a non-root user can change his own password when he does not have write permission to the /etc/shadow file.
- Well to understand the trick check for the permission of /usr/bin/passwd command :
- # ls -lrt /usr/bin/passwd



What is SUID and SGID?

UNIX

- **Set-user Identification (SUID)**
- If you check carefully, you would find the S's in the permission field.
- The first s stands for the SUID
- – When a command or script with SUID bit set is run, its effective UID becomes that of the owner of the file, rather than of the user who is running it.



What is SUID and SGID?

UNIX

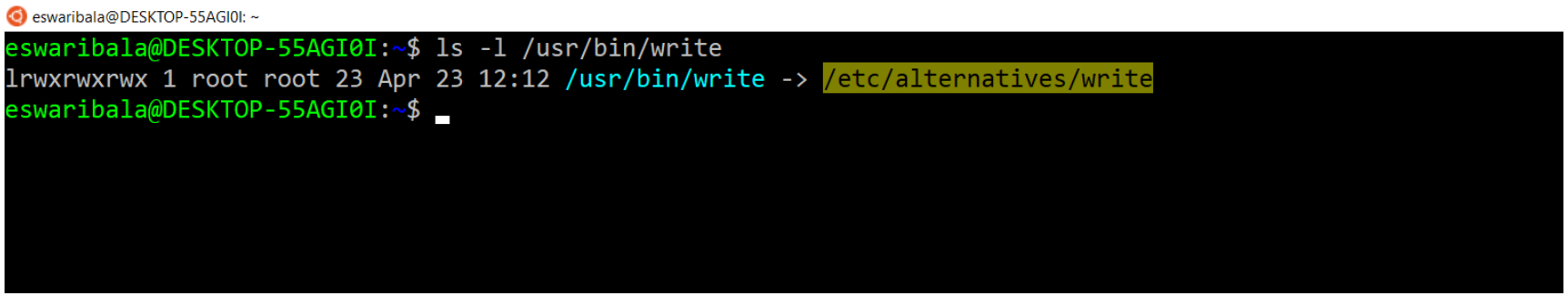
- **Set-user Identification (SUID)**
- How to set SUID on a file?
- # chmod 4555 [path_to_file]
- If a capital “S” appears in the owner’s execute field, it indicates that the setuid bit is on, and the execute bit “x” for the owner of the file is off or denied.

What is SUID and SGID?

UNIX

- **Set-group identification (SGID)**
- SGID permission on executable file
- – SGID permission is similar to the SUID permission, only difference is – when the script or command with SGID on is run, it runs as if it were a member of the same group in which the file is a member.

- # ls -l /usr/bin/write



What is SUID and SGID?

UNIX

- **How to set GUID on a file?**
- **# chmod 2555 [path_to_file]**

What is SUID and SGID?

UNIX

- **SGID on a directory**
- – When SGID permission is set on a directory, files created in the directory belong to the group of which the directory is a member.
- – For example if a user having write permission in the directory creates a file there, that file is a member of the same group as the directory and not the user's group.
- – This is very useful in creating shared directories.
- How to set SGID on a directory
- # chmod g+s [path_to_directory]

What is a profile file?

UNIX

- A profile file is a start-up file of an UNIX user, like the autoexec.bat file of DOS.
- When a UNIX user tries to login to his account, the operating system executes a lot of system files to set up the user account before returning the prompt to the user.
- In addition to the system settings, the user might wish to have some specific settings for his own account.
- To achieve this in UNIX, at the end of the login process, the operating system executes a file at the user level, if present. This file is called profile file.
- The name of the profile file varies depending on the default shell of the user.
- The profile file, if present, should always be in the home directory of the user.

What is a profile file?

UNIX

- The following are the profile files of the commonly used shells:
- The specific settings which an unix user usually does is:
  - Setting of any environment variable
  - Setting of any alias.(Though it is always recommended to keep the aliases in a separate file).
  - Setting of PATH variable or any other path variables.

Shell	Profile File
Ksh	.profile
Bourne	.profile
Bash	.bash_profile
Tcsh	.login
Csh	.login

A typical ksh profile file will look as shown below:
#cat $HOME/.profile

Setting the Terminal Type:

UNIX

- $ echo $TERM
- echo $SHELL
- export TERM=xterm-256color
- echo $LS_COLORS

Shell	Command
csh or tcsh	setenv TERM vt100
sh	TERM=vt100; export TERM
ksh, bash, or zsh	export TERM=vt100

PS and PS Variables

UNIX

- ps - report a snapshot of the current processes.
- This version of ps accepts several kinds of options:
  - UNIX options, which may be grouped and must be preceded by a dash.
  - BSD options, which may be grouped and must not be used with a dash.
  - GNU long options, which are preceded by two dashes.

PS and PS Variables

UNIX

- To see every process on the system using standard syntax:
- ps -e
- ps -ef
- ps -eF
- ps -ely

PS and PS Variables

UNIX

- To see every process on the system using BSD syntax:
  - ps ax
  - ps axu
- To print a process tree:
  - ps -ejH
  - ps axjf
- To get info about threads:
  - ps -eLf
  - ps axms

PS and PS Variables

UNIX

- To get security info:
  - ps -eo euser,ruser,suser,fuser,f,comm,label
  - ps axZ
  - ps -eM
- To see every process running as root (real & effective ID) in user format:
  - ps -U root -u root u
- To see every process with a user-defined format:
  - ps -eo pid,tid,class,rtprio,ni,pri,psr,pcpu,stat,wchan:14,comm
  - ps axo stat,euid,ruid,tty,tpgid,sess,pgrp,ppid,pid,pcpu,comm
  - ps -eopid,tt,user,fname,tmout,f,wchan

PS and PS Variables

UNIX

- Print only the process IDs of syslogd:
  - ps -C syslogd -o pid=
  - Print only the name of PID 42:
  - ps -p 42 -o comm=

PS and PS Variables

UNIX

- man ps

```
SIMPLE PROCESS SELECTION
       a      Lift the BSD-style "only yourself" restriction, which is imposed upon the set of all processes when some BSD-style
              (without "-") options are used or when the ps personality setting is BSD-like.  The set of processes selected in this
              manner is in addition to the set of processes selected by other means.  An alternate description is that this option
              causes ps to list all processes with a terminal (tty), or to list all processes when used together with the x option.

       -A     Select all processes.  Identical to -e.

       -a     Select all processes except both session leaders (see getsid(2)) and processes not associated with a terminal.

       -d     Select all processes except session leaders.

       --deselect
              Select all processes except those that fulfill the specified conditions (negates the selection).  Identical to -N.

       -e     Select all processes.  Identical to -A.

       g      Really all, even session leaders.  This flag is obsolete and may be discontinued in a future release.  It is normally
              implied by the a flag, and is only useful when operating in the sunos4 personality.

       -N     Select all processes except those that fulfill the specified conditions (negates the selection).  Identical to
              --deselect.

       T      Select all processes associated with this terminal.  Identical to the t option without any argument.

       r      Restrict the selection to only running processes.

       x      Lift the BSD-style "must have a tty" restriction, which is imposed upon the set of all processes when some BSD-style
              (without "-") options are used or when the ps personality setting is BSD-like.  The set of processes selected in this
              manner is in addition to the set of processes selected by other means.  An alternate description is that this option
              causes ps to list all processes owned by you (same EUID as ps), or to list all processes when used together with the
 Manual page ps(1) line 80 (press h for help or q to quit)
```

Java Installation

UNIX

sudo apt update
java -version

```
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 12.475/17.194/22.223/3.985 ms
eswaribala@DESKTOP-55AGI0I:~$ sudo apt update
[sudo] password for eswaribala:
Hit:1 http://archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://security.ubuntu.com/ubuntu focal-security InRelease
Hit:3 http://archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:4 http://archive.ubuntu.com/ubuntu focal-backports InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
All packages are up to date.
eswaribala@DESKTOP-55AGI0I:~$ java -version

Command 'java' not found, but can be installed with:

sudo apt install openjdk-11-jre-headless  # version 11.0.8+10-0ubuntu1~20.04, or
sudo apt install default-jre              # version 2:1.11-72
sudo apt install openjdk-13-jre-headless  # version 13.0.3+3-1ubuntu2
sudo apt install openjdk-14-jre-headless  # version 14.0.1+7-1ubuntu1
sudo apt install openjdk-8-jre-headless   # version 8u252-b09-1ubuntu1

eswaribala@DESKTOP-55AGI0I:~$
```

Java Installation

UNIX

sudo apt install openjdk-8-jre-headless

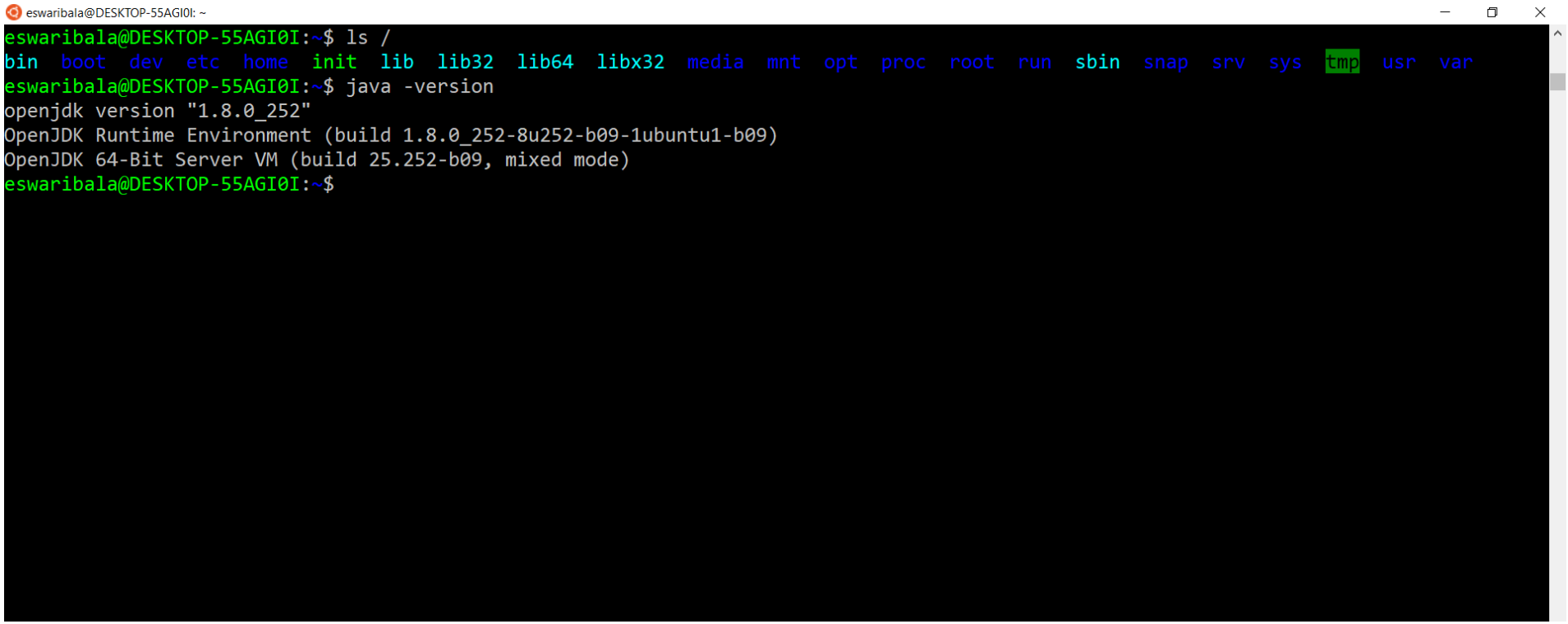
eswaribala@DESKTOP-55AGI0I: ~

```
eswaribala@DESKTOP-55AGI0I:~$ sudo apt install openjdk-8-jre-headless
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  ca-certificates-java java-common libavahi-client3 libavahi-common-data libavahi-common3 libcups2 libjpeg-turbo8 libjpeg8 liblcms2-2
  libnspr4 libnss3 libpcsclite1
Suggested packages:
  default-jre cups-common liblcms2-utils pcscd libnss-mdns fonts-dejavu-extra fonts-ipafont-gothic fonts-ipafont-mincho
  fonts-wqy-microhei fonts-wqy-zenhei fonts-indic
The following NEW packages will be installed:
  ca-certificates-java java-common libavahi-client3 libavahi-common-data libavahi-common3 libcups2 libjpeg-turbo8 libjpeg8 liblcms2-2
  libnspr4 libnss3 libpcsclite1 openjdk-8-jre-headless
0 upgraded, 13 newly installed, 0 to remove and 0 not upgraded.
Need to get 29.4 MB of archives.
After this operation, 107 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu focal/main amd64 java-common all 0.72 [6816 B]
Get:2 http://archive.ubuntu.com/ubuntu focal/main amd64 libavahi-common-data amd64 0.7-4ubuntu7 [21.4 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal/main amd64 libavahi-common3 amd64 0.7-4ubuntu7 [21.7 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal/main amd64 libavahi-client3 amd64 0.7-4ubuntu7 [25.5 kB]
Get:5 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libcups2 amd64 2.3.1-9ubuntu1.1 [233 kB]
Get:6 http://archive.ubuntu.com/ubuntu focal/main amd64 liblcms2-2 amd64 2.9-4 [140 kB]
Get:7 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libjpeg-turbo8 amd64 2.0.3-0ubuntu1.20.04.1 [117 kB]
Get:8 http://archive.ubuntu.com/ubuntu focal/main amd64 libjpeg8 amd64 8c-2ubuntu8 [2194 B]
Get:9 http://archive.ubuntu.com/ubuntu focal/main amd64 libnspr4 amd64 2:4.25-1 [107 kB]
14% [9 libnspr4 71.6 kB/107 kB 67%]                                    17.7 kB/s 27min 6s
```

Java Installation

UNIX

sudo apt install openjdk-8-jre-headless



Installing MySQL in Ubuntu|Linux|Windows Subsystem for Linux from Scratch

- sudo apt-get remove --purge *mysql*
- sudo rm -rf /etc/mysql /var/lib/mysql
- sudo apt-get remove --purge *mariadb*
- sudo apt-get autoremove
- sudo apt-get autoclean
- dpkg -l | grep mariadb
- dpkg -l | grep mysql

Installing MySQL in Ubuntu|Linux|Windows Subsystem for Linux from Scratch

- Now to fix all broken repositories
- sudo apt-get install –f
- Now Upgrading the Repositories
- sudo apt update
- sudo apt upgrade
- Now, to install MySQL 8, simply type:
- sudo apt install mysql-server

Installing MySQL in Ubuntu|Linux|Windows Subsystem for Linux from Scratch

- Now to fix all broken repositories
- sudo apt-get install –f
- Now Upgrading the Repositories
- sudo apt update
- sudo apt upgrade
- Now, to install MySQL 8, simply type:
- sudo apt install mysql-server
- sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf
- (change port)

Installing MySQL in Ubuntu|Linux|Windows Subsystem for Linux from Scratch

- sudo service mysql start
- sudo mysql_secure_installation
- mysql -u root -p

```
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: NO)
eswaribala@DESKTOP-55AGI0I:~/Ecommerce$ sudo mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 8.0.21-0ubuntu0.20.04.4 (Ubuntu)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Uninstall packages

UNIX

- sudo apt list –installed
- sudo apt remove xfce4

Standard Input and Output

- The command gets its input through the channel labeled “standard input.”
- That input can come from your keyboard (the default), a file, or a command.
- Similarly, the command delivers its output through the channel labeled “standard output.”
- The output might go to your screen (the default), a file, or another command.

Standard Input and Output

UNIX

- The command doesn’t need to know which of these sources the input comes from, or where the output goes.
- It is the shell that sets up these connections, according to the instructions in your command line.
- It does this through the I/O redirection mechanisms, which include pipes and file redirection.

Standard Input and Output

UNIX

- A typical use of the pipe feature is the following command:
- $ man find lp
- This uses a pipe to send the output from the man command to the lp command, in order to print a hard copy of the manual page for find.
- An example of file redirection is the following command:
- $ man find > temp
- This saves the output from the man command as the file temp

Shell Redirection Output

UNIX

Symbol	Example	Function
|	**cmd1 | cmd2**	Run **cmd1** and send output to **cmd2**
>	**cmd > file**	Send output of **cmd** to *file*
>>	**cmd >> file**	Append output of **cmd** to *file*
<	**cmd < file**	Take input for **cmd** from file
/dev/stdin	**cmd /dev/stdin**	Take input from keyboard
2>	**cmd 2> errorfile**	Send standard error to errorfile **(ksh, bash)**
2>&1	**cmd > msgs 2>&1**	Send both output and standard error to *msgs* (**sh**, **ksh**, and **bash**)
/dev/tty	**(cmd > /dev/tty) >& error**	Redirect output to screen, and error to error (**csh**, **tcsh**, and **bash**)
>&	**cmd >& msgs**	Send both output and errors to *msgs* (**csh**, **tcsh**, and **bash**)

Using Pipes

UNIX

- The pipe symbol (|) tells the shell to take the standard output of one command and use it as the standard input of another command.
- Using pipes to join individual commands together in pipelines is an easy way to use a sequence of simple commands to carry out a complex task.
- For example, suppose you want to know if the user named “Ashok” is logged in.
- One way to find out would be to use the who command to list all of the users currently logged in, and to look for a line listing “Ashok” in the output.
- However, on a large system there could be many users-enough to make it difficult to find a specific name in the list.

How to use grep command in Unix

UNIX

- The grep utility searches text file.txt for a pattern and prints all lines that contain that pattern.
- Syntax: grep [-options] limited-regular-expression [filename …]

How to use grep command in Unix

UNIX

- **grep options or grep command options**

  -c Print only a count of the lines that contain the pattern.

  -i Ignore upper/lower case distinction during comparisons.

  -l Print only the names of file.txt with matching lines,separated by NEWLINE characters. Does not repeat the names of file.txt when the pattern is found more than once.

  -n Precede each line by its line number in the file (first line is 1).

  -v Print all lines except those that contain the pattern.

  -r It recursively search the pattern in all the file.txt in the current directory and all it's sub-directory.

  -w It searches the exact word

How to use grep command in Unix

UNIX

- **To find all uses of the word “top” (in any case) in the multiples file like x*, and write with line numbers:**
  - grep -i -n spring a*

- search ‘application.properties for ‘Trading’ anywhere in a line
  - grep Trading application.properties

How to use grep command in Unix

UNIX

- **grep case insensitive command .**
- By default grep command is case sensitive.
- You can use option grep -i to make it case insensitive.
- We can use grep -w option for searching the specific work not sub-string .
- The below example searches adpatch.log for word failure in any case
- grep -i -w files demo.txt

How to use grep command in Unix

UNIX

- **find ‘run time’ or ‘run-time’ in all txt in file.txt**
- grep run[-]time *.txt
- **pipe who to grep, look for appmmgr**
- who | grep appmmgr
- **grep recursive option .It search for oracle string in current directory files and all the files in sub directory**
- grep -r "oracle" *

How to use grep command in Unix

UNIX

- **Grep exclude option (grep -v) . We can use grep -v to exclude the search item item. It will not show the lines which has oracle string in it**
- ps -ef|grep -v mysql

Understanding Regular Expressions

UNIX

Understanding Regular Expressions:

^ (Caret)	match expression at the start of a line, as in ^A.
$ (Question)	match expression at the end of a line, as in A$.
\ (Back Slash)	turn off the special meaning of the next character, as in \^. To look for a Caret "^" at the start of a line, the expression is ^\^.
[ ] (Brackets)	match any one of the enclosed characters, as in [aeiou]. Use Hyphen "-" for a range, as in [0-9].
[^]	match any one character except those enclosed in [ ], as in [^0-9].
. (Period)	match a single character of any value, except end of line. So b.b will match "bob", "bib", "b-b", etc.
* (Asterisk)	match zero or more of the preceding character or expression. An asterisk matches zero or more of what precedes it. Thus [A-Z]* matches any number of upper-case letters, including none, while [A-Z][A-Z]* matches one or more upper-case letters.

Understanding Regular Expressions

UNIX

- **search file.txt for lines with ‘kite’**
  - grep kite file.txt
- **‘kite’ at the start of a line**
  - grep '^kite' file.txt
- **‘kite’ at the end of a line**
  - grep 'kite
- **lines containing only ‘kite’**
  - grep '^kite

Understanding Regular Expressions

UNIX

- **lines starting with ‘^s’, “\” escapes the ^**
- grep '\^s' file.txt
- **search for ‘kite’ or ‘Kite’**
- grep '[Kk]ite' file.txt
- **search for TOM, Tom, TOm or ToM**
- grep 'T[oO][mM]' file.txt
- **search for blank lines**
- grep '^

Understanding Regular Expressions

UNIX

- **search for pairs of numeric digits**
- grep '[0-9][0-9]' file
- **list your mail**
- grep '^From: ' /usr/mail/$USER
- **any line with at least one letter**
- grep '[a-zA-Z]' 1.txt
- **anything not a letter or number**
- grep '[^a-zA-Z0-9]'

Understanding Regular Expressions

UNIX

- **line start with “.” and 2 lower case letters} letters**
- grep '^\.[a-z][a-z]’
- If you want to search multiple words in the same grep command ,then use egrep command in UNIX
- **It search all the three words in the file**
- egrep 'cat|bad|sat' file.txt
- **It discarded all the lines having any of these three word from the output of ps -ef**
- ps -ef| egrep -v 'cat|bad|sat' :

Understanding Regular Expressions

UNIX

- **grep with pipe command**
- pipe command in Linux let u input the output of the one command to the another command.
- **Example**
- ps -ef|grep python

Understanding Regular Expressions

UNIX

- **Sometimes we just want the grep to show out only the file names which matched the given pattern then we use the -l (lower-case L) option. if multiple files are there. This will simply print all the file names**
- grep -l ORA-0600 *.trc
- **Suppose you want to count that how many lines matches the given pattern/string, then use the option -c**
- grep -c "TOM" 1.txt

Understanding Regular Expressions

UNIX

- **When you are searching error using grep on a huge file, it may be useful to see some lines around the match.**
- Lines before the match
- grep -A 10 "TOM" 1.txt

- Lines after the match
- grep -B 10 "TOM" 1.txt

- Lines around the match
- grep -C 10 "TOM" 1.txt

Understanding Regular Expressions

UNIX

- **When we want to show the line number of the matched pattern with in the file.we can use grep -n**
- grep -n "ORA-0600" alert.log
- **Grep exclude directory in recursive search. Some time we want to exclude one directory from grep recursive search**
- grep -r --exclude-dir=log "TOM"

Using Pipes

UNIX

- A better solution is to use a pipe to redirect the output of who to grep.
- The grep command searches through its input and prints the lines that match a target pattern.
- $ ls -l | grep application

Using Pipes

UNIX

- **Output Redirection to a File**
- The > redirection operator sends the output of a command to a file.
- For example,
- $ ls -l > filelist
- causes the shell to save the output of ls -l as the file filelist

Using Pipes

UNIX

- **Output Redirection to a File**
- The >> operator appends data to a file without overwriting it.
- That means that if the file already exists, the new data will be added on to the end.
- In this example,
- $ cat notes.jan >> research
- the shell redirects the standard output from cat and appends it to the file named research.

Using Pipes

UNIX

- **Input Redirection from a File**
- Just as you can use the greater than (or right arrow) symbol, >, to redirect standard output, you can use the less than (or left arrow) symbol, <, to redirect standard input.
- The < symbol tells the shell to interpret the filename that follows it as the standard input to a command.

Using Pipes

UNIX

- **You can use the < redirection operator to replace the keyboard with a file as the standard input.**
- In this case, the mail command will send the contents of that file, instead of waiting for you to enter a message at the keyboard:
- $ mail anita@bio.ca.edu < note
- The < tells the shell to run mail with the contents of note as its standard input.

Using Pipes

UNIX

- **You can redirect input and output at the same time.**
- The following example uses the sort command to take the information in source, alphabetize it, and put the output in dest:
- $ sort < source > dest
- The order in which you indicate the input and output files doesn’t matter, so the following example has the same result:
- $ sort > dest < source

Using Pipes

UNIX

- **Standard Input from the Keyboard**
- You can also force commands to accept input from the keyboard.
- The logical filename /dev/stdin refers to standard input.
- When used as an argument to a command, it causes the shell to send the input from the keyboard to the command, in place of a file.

Using Pipes

UNIX

- **Standard Input from the Keyboard**
- For example, the command sort is used to alphabetize the lines in a file.
- If you give it the argument /dev/stdin, it will sort the lines you type in from the keyboard:
- $ sort /dev/stdin > guestlist

Unix Process Management

- A process refers to a program in execution; it’s a running instance of a program.
- It is made up of the program instruction, data read from files, other programs or input from a system user.

Unix Process Management

UNIX

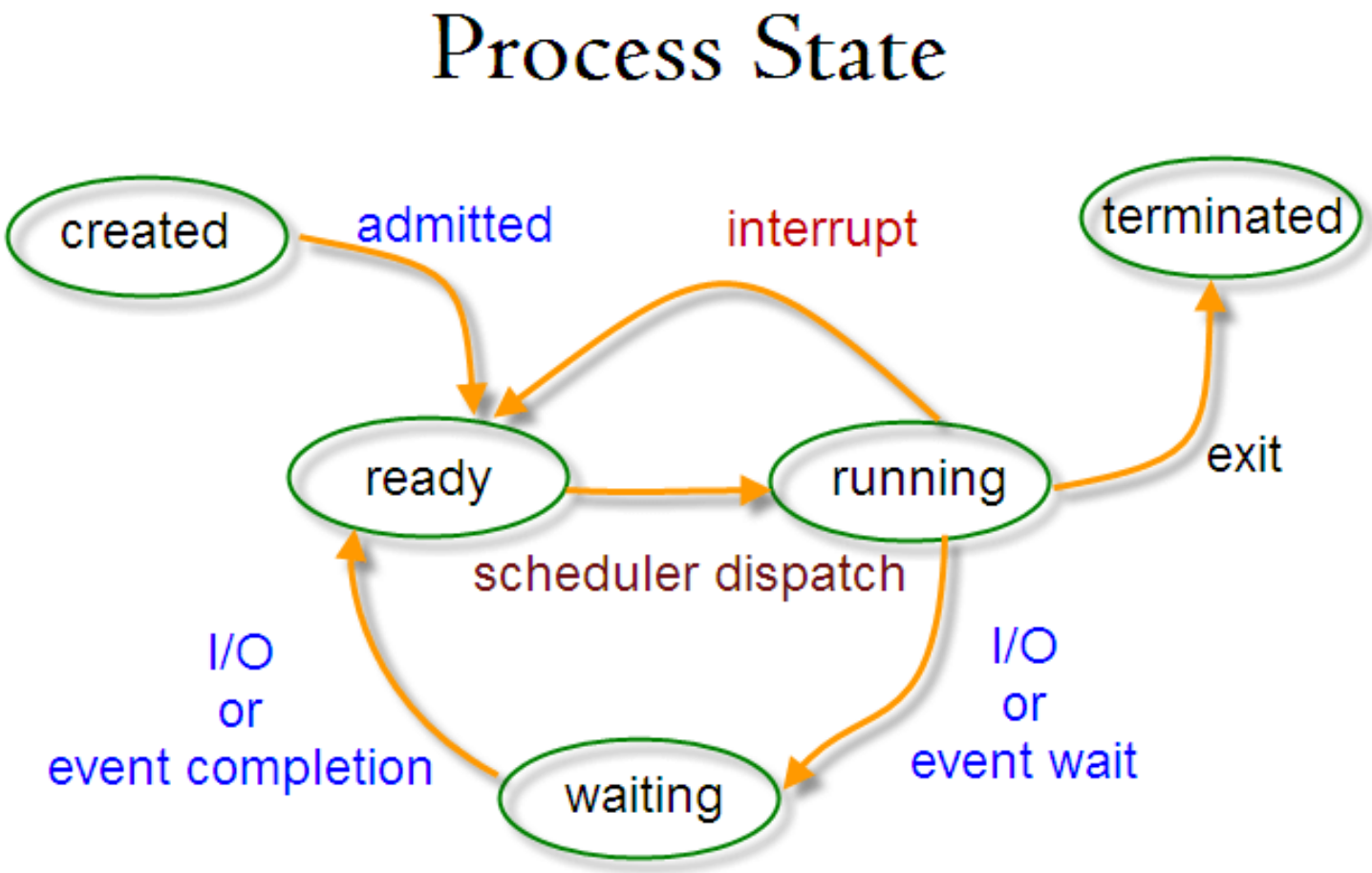
- **Types of Processes**
- There are fundamentally two types of processes in Linux:
  - **Foreground processes** (also referred to as interactive processes) – these are initialized and controlled through a terminal session.
  - In other words, there has to be a user connected to the system to start such processes; they haven't started automatically as part of the system functions/services.
  - **Background processes** (also referred to as non-interactive/automatic processes) – are processes not connected to a terminal; they don't expect any user input.

What is Daemons

UNIX

- These are special types of background processes that start at system startup and keep running forever as a service; they don’t die.
- They are started as system tasks (run as services), spontaneously.
- However, they can be controlled by a user via the init process.

What is Daemons



Creation of a Processes in Linux

UNIX

- A new process is normally created when an existing process makes an exact copy of itself in memory.
- The child process will have the same environment as its parent, but only the process ID number is different.
- There are two conventional ways used for creating a new process in Linux:
- Using The System() Function – this method is relatively simple, however, it's inefficient and has significantly certain security risks.
- Using fork() and exec() Function – this technique is a little advanced but offers greater flexibility, speed, together with security.
- Java process Example

How Does Linux Identify Processes?

UNIX

- Because Linux is a multi-user system, meaning different users can be running various programs on the system, each running instance of a program must be identified uniquely by the kernel.
- And a program is identified by its process ID (PID) as well as it's parent processes ID (PPID), therefore processes can further be categorized into:
- Parent processes – these are processes that create other processes during run-time.
- Child processes – these processes are created by other processes during run-time.

The Init Process

UNIX

- Init process is the mother (parent) of all processes on the system.
- it's the first program that is executed when the Linux system boots up;
- it manages all other processes on the system.
- It is started by the kernel itself, so in principle it does not have a parent process.
- The init process always has process ID of 1. It functions as an adoptive parent for all orphaned processes.
- You can use the pidof command to find the ID of a process:

The Init Process

UNIX

- To find the process ID and parent process ID of the current shell, run:
- $ echo $$
- $ echo $PPID

Starting a Process in Linux

UNIX

- Once you run a command or program (for example ls –l *.txt), it will start a process in the system.
- You can start a foreground (interactive) process as follows, it will be connected to the terminal and a user can send input it:

Linux Background Jobs

UNIX

- To start a process in the background (non-interactive), use the & symbol, here, the process doesn't read input from a user until it's moved to the foreground.
- ls –l *.txt &
- You can also send a process to the background by suspending it using [Ctrl + Z], this will send the SIGSTOP signal to the process, thus stopping its operations; it becomes idle:

Linux Background Jobs

UNIX

- To continue running the above-suspended command in the background, use the bg command:
- # bg
- To send a background process to the foreground, use the fg command together with the job ID like so:
- # jobs
- # fg %1

States of a Process in Linux

UNIX

- During execution, a process changes from one state to another depending on its environment/circumstances.
- In Linux, a process has the following possible states:
- **Running** – here it's either running (it is the current process in the system) or it's ready to run (it's waiting to be assigned to one of the CPUs).
- **Waiting** – in this state, a process is waiting for an event to occur or for a system resource. Additionally, the kernel also differentiates between two types of waiting processes; interruptible waiting processes – can be interrupted by signals and uninterruptible waiting processes – are waiting directly on hardware conditions and cannot be interrupted by any event/signal.

States of a Process in Linux

UNIX

- **Stopped** – in this state, a process has been stopped, usually by receiving a signal. For instance, a process that is being debugged.
- **Zombie** – here, a process is dead, it has been halted but it’s still has an entry in the process table.

How to View Active Processes in Linux

- ps Command
- top – System Monitoring Tool
- glances – System Monitoring Tool
  - glances is a relatively new system monitoring tool with advanced features

How to View Active Processes in Linux


```
eswaribala@DESKTOP-55AGI0I: ~
DESKTOP-55AGI0I - IP 192.168.43.50/24 Pub 2401:4900:234f:6b02:802b:bd4:40fb:f97b                                                Uptime: 4:31:25

CPU  [|      17.7%]   CPU      17.7%  nice:     0.0%  ctx_sw:     0   MEM     49.7%  active:     164M   SWAP      1.0%   LOAD    8-core
MEM  [||     49.7%]   user:     8.0%  irq:      0.0%  inter:      0   total:  15.8G  inactive:   154M   total:   27.9G   1 min:    0.52
SWAP [       1.0%]   system:   7.0%  iowait:   0.0%  sw_int:     0   used:   7.87G  buffers:   33.2M   used:    274M   5 min:    0.58
                     idle:    36.0%  steal:    0.0%                  free:   7.97G  cached:    191M   free:    27.6G   15 min:   0.59

NETWORK              Rx/s   Tx/s   TASKS   7 (65 thr), 1 run, 6 slp, 0 oth sorted automatically by CPU consumption
eth2                   0b     0b
eth3                   0b     0b   CPU%   MEM%   VIRT   RES     PID USER          TIME+ THR  NI S   R/s W/s  Command
lo                     0b     0b   19.7   0.3    440M   47.3M  8784 eswaribal      0:01 6     0 R     ? ?    /usr/bin/python3 /usr/bin/gla
wifi0                  0b     0b   0.0    0.0    17.8M  3.37M    66 eswaribal      0:01 1     0 S     ? ?    -bash
                                   0.0    0.0    14.3M  3.15M  2341 root           0:00 1     0 S     ? ?    /usr/sbin/apache2 -k start
DefaultGateway              39ms   0.0    0.0    1.91G  2.10M  8140 www-data       0:00 27    0 S     ? ?    /usr/sbin/apache2 -k start
                                   0.0    0.0    1.91G  2.09M  8138 www-data       0:00 27    0 S     ? ?    /usr/sbin/apache2 -k start
                                   0.0    0.0    8.72M  544K     65 root           0:00 1     0 S     ? ?    //init
                                   0.0    0.0    8.68M  512K      1 root           0:00 2     0 S     ? ?    //init

2020-08-06 01:47:13 IST
```

Sending Signals To Processes

UNIX

- The fundamental way of controlling processes in Linux is by sending signals to them. There are multiple signals that you can send to a process, to view all the signals run:
- $ kill -l

Traceroute Install

UNIX

sudo apt-get update && sudo apt-get install traceroute

```
89 88 87
eswaribala@DESKTOP-55AGI0I:~$ sudo apt-get update && sudo apt-get install traceroute
Hit:1 http://archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://archive.ubuntu.com/ubuntu focal-updates InRelease [111 kB]
Get:3 http://security.ubuntu.com/ubuntu focal-security InRelease [107 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal-backports InRelease [98.3 kB]
Get:5 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [149 kB]
Get:6 http://security.ubuntu.com/ubuntu focal-security/main Translation-en [52.6 kB]
Get:7 http://security.ubuntu.com/ubuntu focal-security/main amd64 c-n-f Metadata [3532 B]
Get:8 http://security.ubuntu.com/ubuntu focal-security/restricted amd64 Packages [29.2 kB]
Get:9 http://security.ubuntu.com/ubuntu focal-security/restricted Translation-en [7732 B]
Get:10 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages [44.4 kB]
Get:11 http://security.ubuntu.com/ubuntu focal-security/universe Translation-en [23.6 kB]
Get:12 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [316 kB]
Get:13 http://security.ubuntu.com/ubuntu focal-security/universe amd64 c-n-f Metadata [1832 B]
Get:14 http://archive.ubuntu.com/ubuntu focal-updates/main Translation-en [119 kB]
Get:15 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 c-n-f Metadata [8092 B]
Get:16 http://archive.ubuntu.com/ubuntu focal-updates/restricted amd64 Packages [29.2 kB]
Get:17 http://archive.ubuntu.com/ubuntu focal-updates/restricted Translation-en [7732 B]
Get:18 http://archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [146 kB]
Get:19 http://archive.ubuntu.com/ubuntu focal-updates/universe Translation-en [73.9 kB]
Get:20 http://archive.ubuntu.com/ubuntu focal-updates/universe amd64 c-n-f Metadata [4920 B]
Fetched 1333 kB in 15s (91.5 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  traceroute
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 45.4 kB of archives.
After this operation, 152 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu focal/universe amd64 traceroute amd64 1:2.1.0-2 [45.4 kB]
```

Traceroute Install

UNIX

traceroute askubuntu.com

```
eswaribala@DESKTOP-55AGI0I:~$ traceroute askubuntu.com
traceroute to askubuntu.com (151.101.193.69), 30 hops max, 60 byte packets
 1  * * *
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  * * *
 9  * * *
10  * * *
11  * * *
12  * * *
13  * * *
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
19  * * *
20  * * *
21  * * *
22  * * *
23  * * *
24  * * *
25  * * *
26  * * *
27  * * *
28  * * *
29  * * *
30  * * *
eswaribala@DESKTOP-55AGI0I:~$ ping -c3 www.google.com
```

Ping

UNIX

ping -c3 www.google.com

```
29  * * *
30  * * *
eswaribala@DESKTOP-55AGI0I:~$ ping -c3 www.google.com
PING www.google.com (216.58.197.68) 56(84) bytes of data.
64 bytes from maa03s21-in-f4.1e100.net (216.58.197.68): icmp_seq=1 ttl=120 time=22.2 ms
64 bytes from maa03s21-in-f4.1e100.net (216.58.197.68): icmp_seq=2 ttl=120 time=12.5 ms
64 bytes from maa03s21-in-f4.1e100.net (216.58.197.68): icmp_seq=3 ttl=120 time=16.9 ms

--- www.google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 12.475/17.194/22.223/3.985 ms
eswaribala@DESKTOP-55AGI0I:~$
```

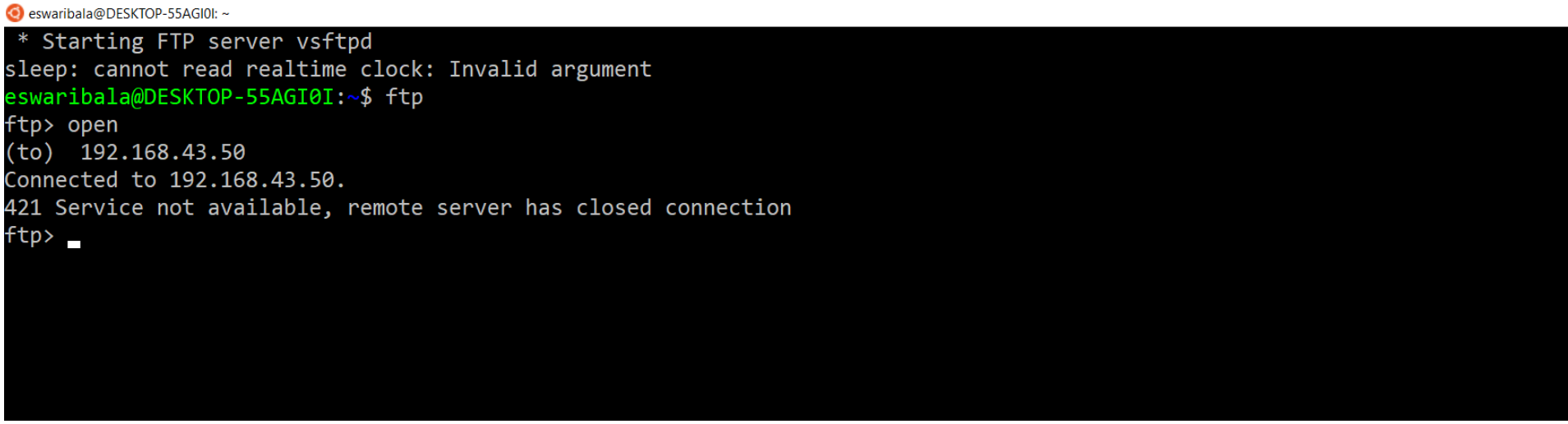
ftp utility

UNIX

- FTP (File Transfer Protocol) is a relatively old and most used standard network protocol used for uploading/downloading files between two computers over a network.
- However, FTP by its original insecure, because it transmits data together with user credentials (username and password) without encryption.
- FTP

ftp utility

UNIX



Vi Editor

UNIX

- Vi Editor
- Vi Cheat sheet

Questions

Module Summary

- Unix Architecture
- Directories and Files
- File Management
- Editors
- Shell Scripting
- Utilities