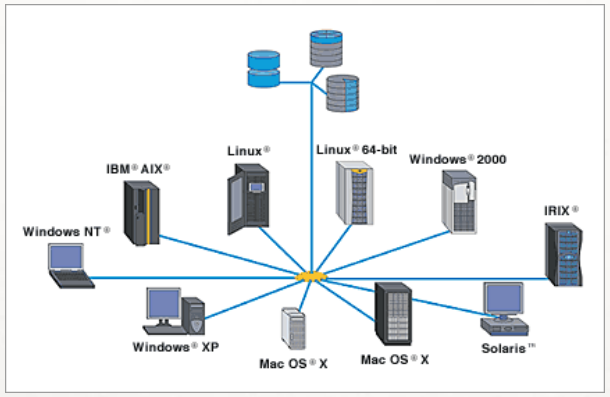
Unix Training



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

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

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

- 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, 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.

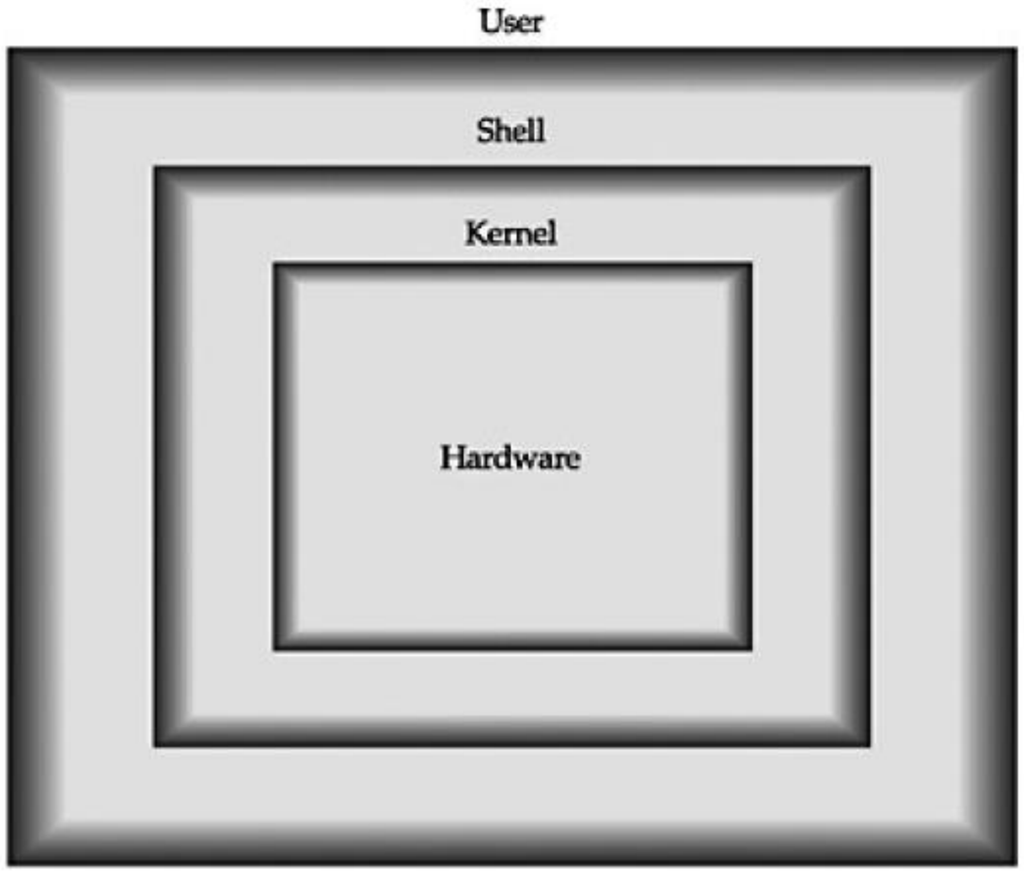
Why Is UNIX Important

- 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

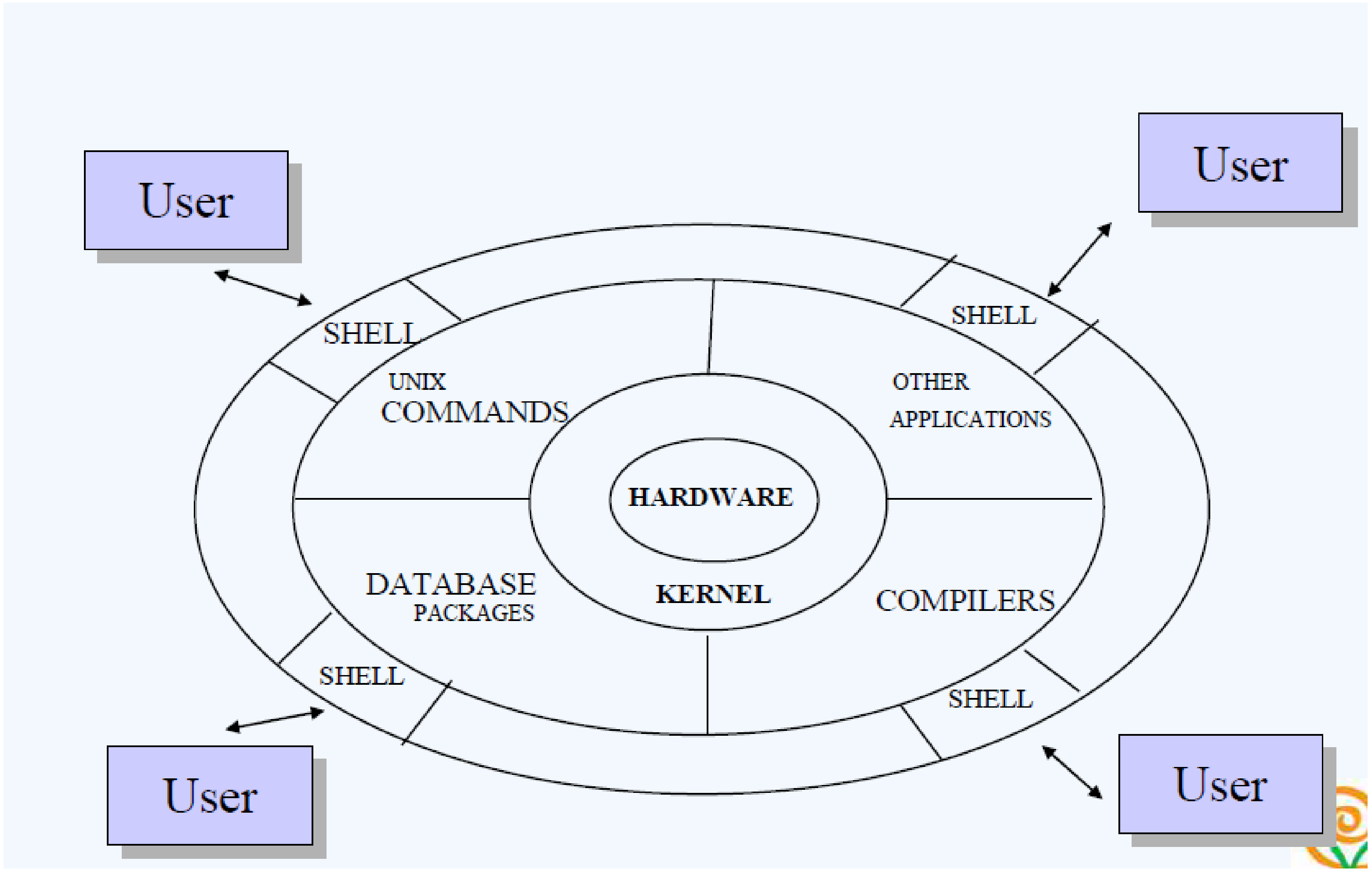
- Hierarchical File System
- Shells

The Structure of the UNIX Operating System



UNIX

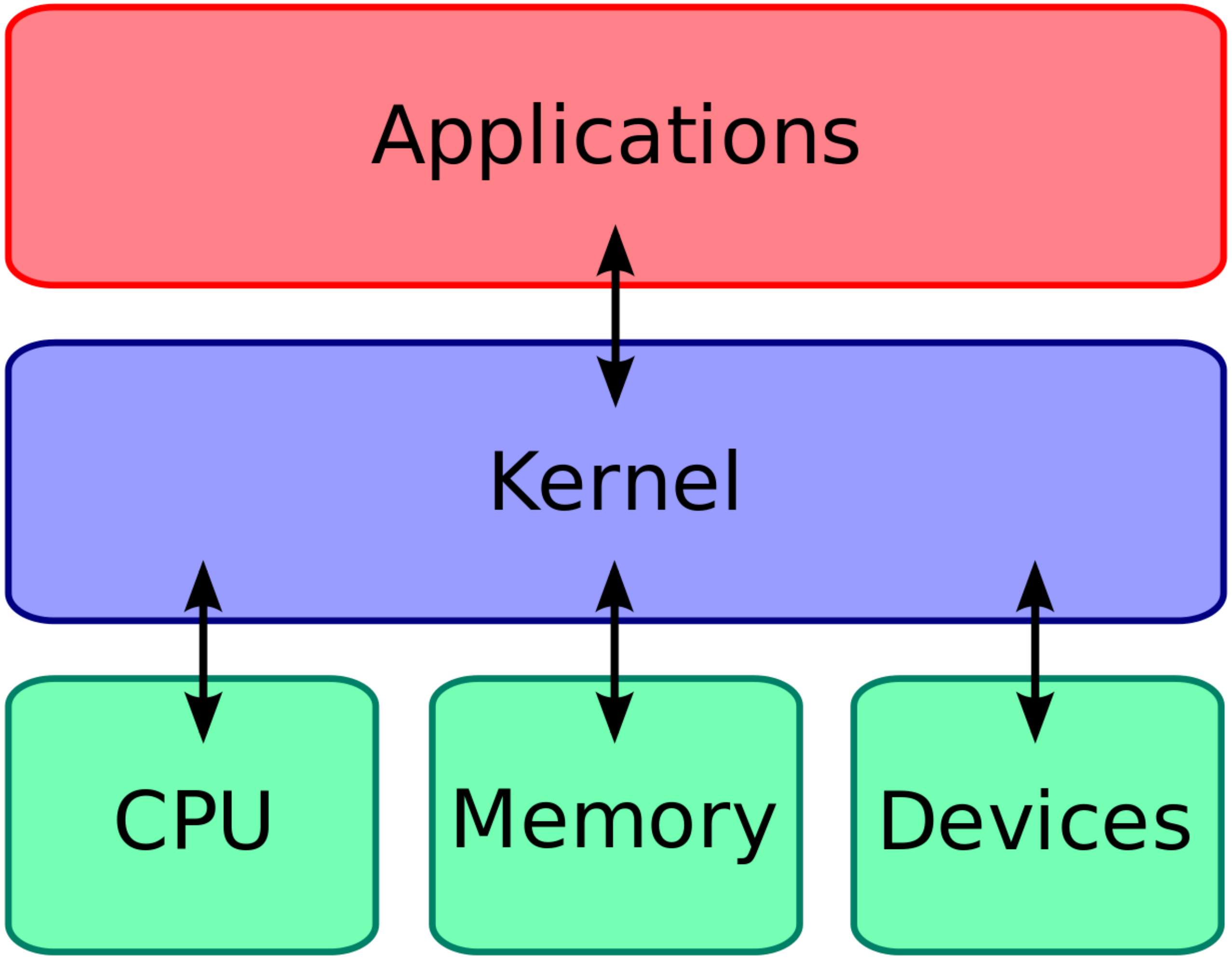
The Structure of the UNIX Operating System



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



Kernel

- 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

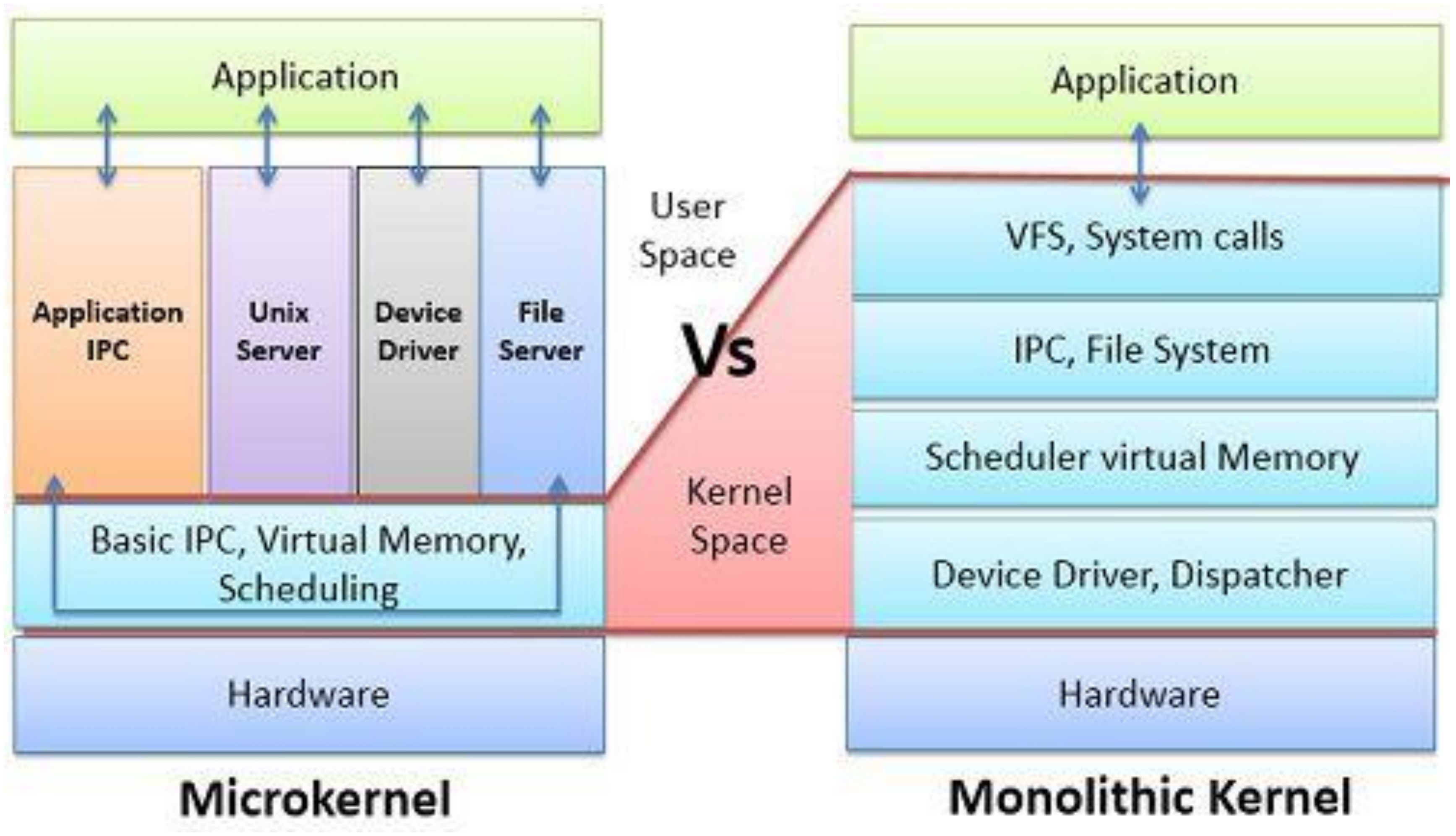
- – handle interrupts (signals to terminate execution)
- – handle errors
- – perform input and output services (which allow computers to interact with terminals, storage devices, and printers)
- – allocate the resources of the computer (such as the CPU or input/output devices) among users.

Kernel

- 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

UNIX


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

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

- 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

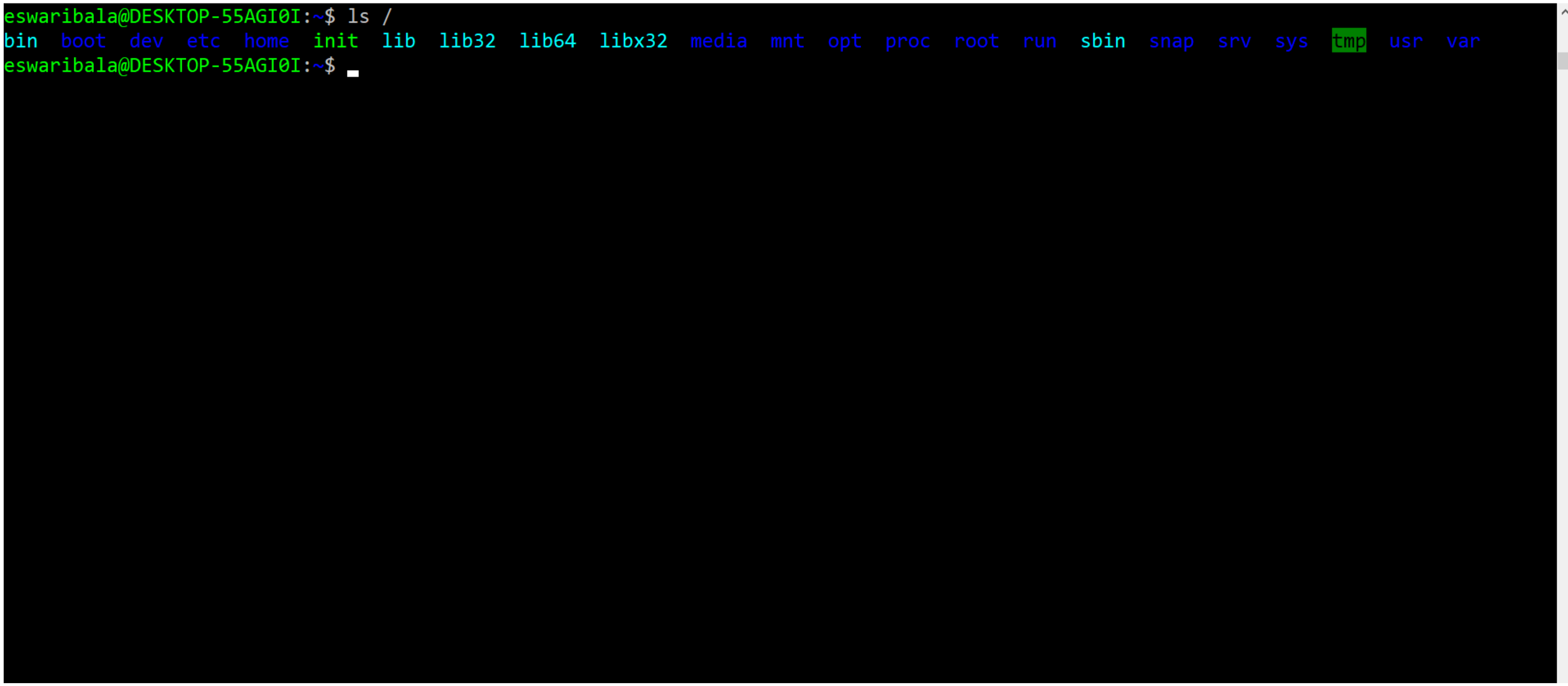
Utilities

- 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

- 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



```
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

- 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

- 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

- 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.

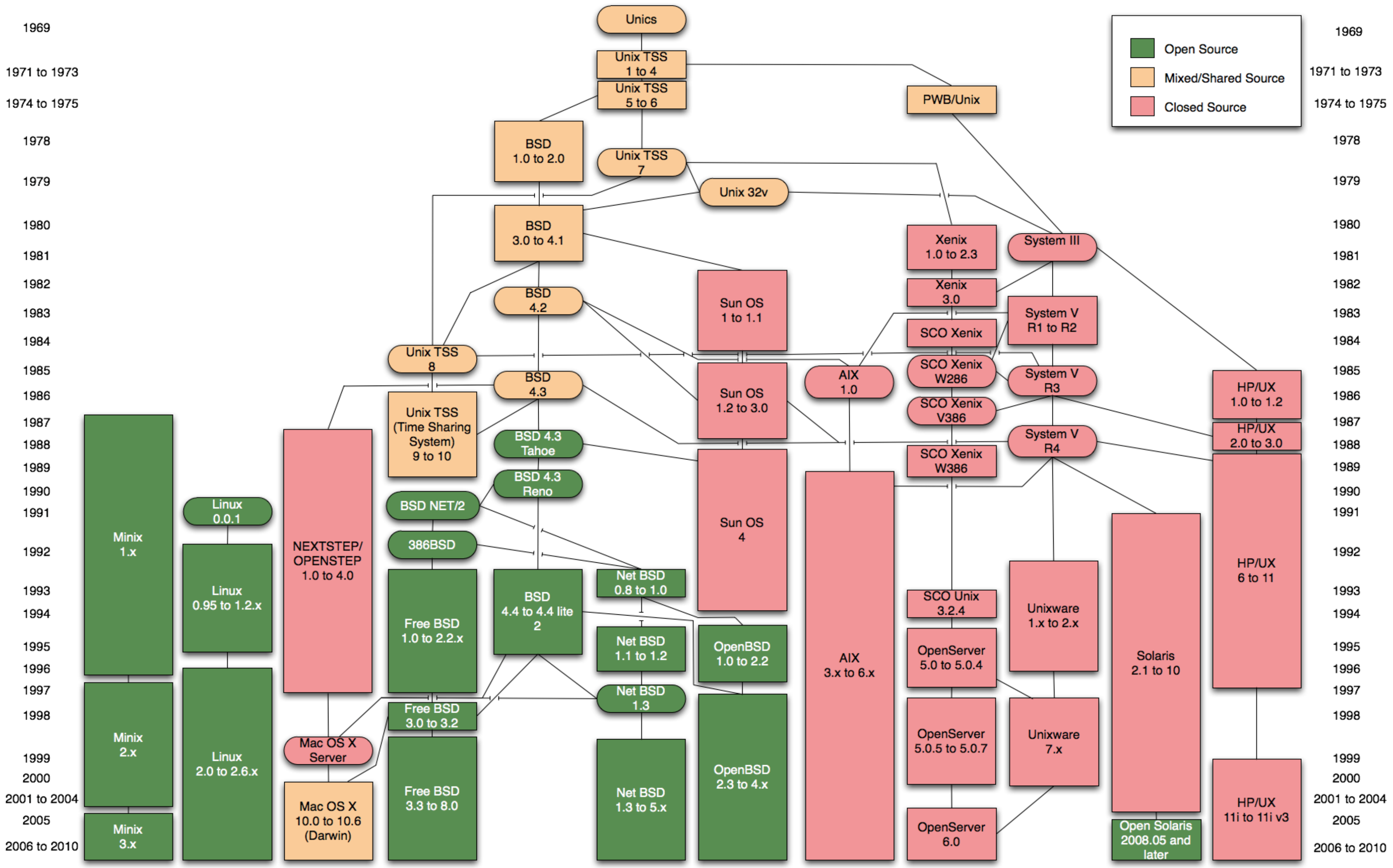
The Birth of the UNIX System

UNIX

- 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

UNIX



A UNIX System Timeline

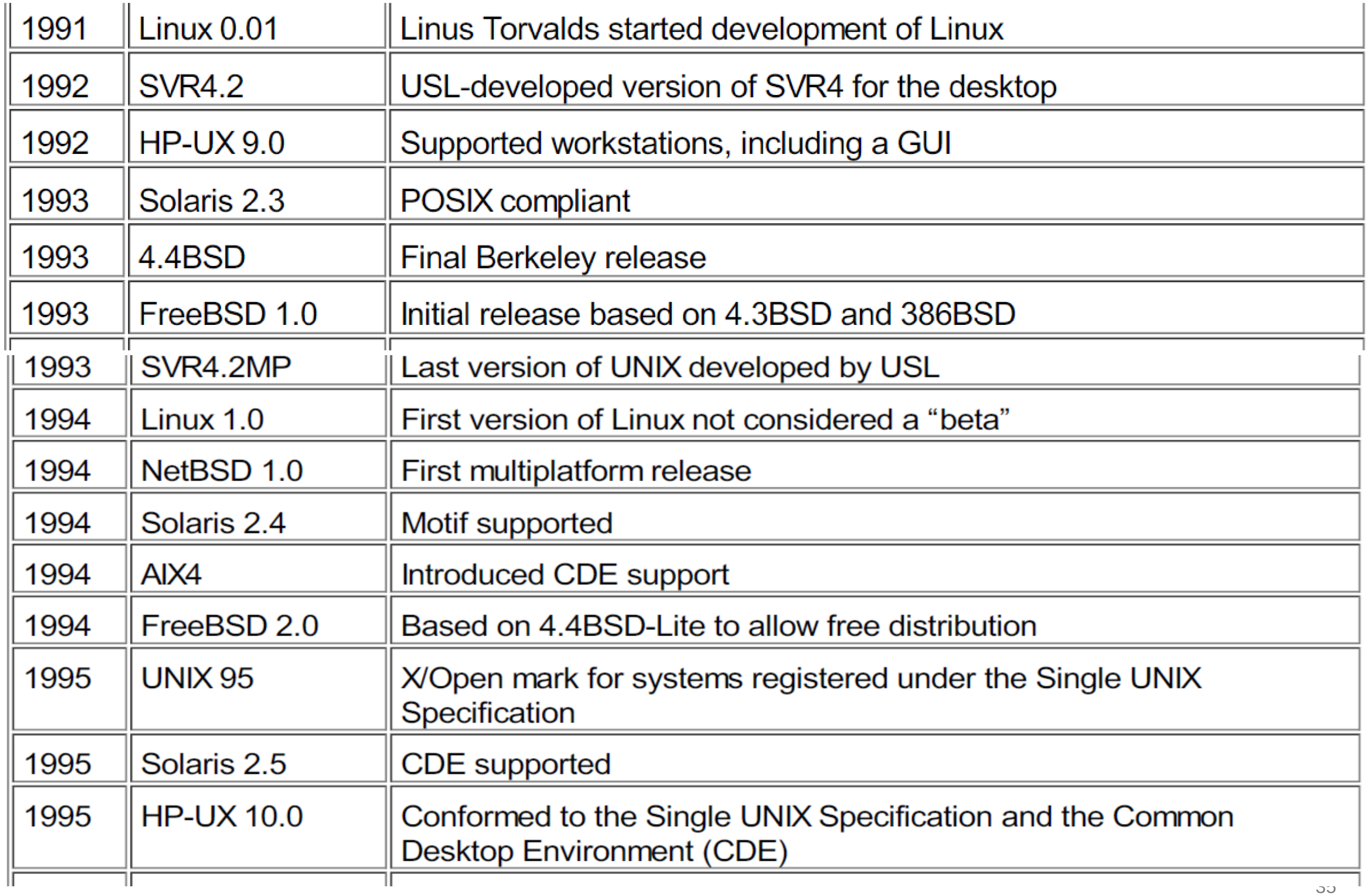
Year	UNIX Variant or Standard	Comments
1969	UNICS (later called UNIX)	A new operating system invented by Ken Thompson and Dennis Ritchie for the PDP-7
1973	Fourth Edition	Written in C programming language; widely used inside Bell Laboratories
1975	Sixth Edition	First version widely available outside 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

UNIX

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

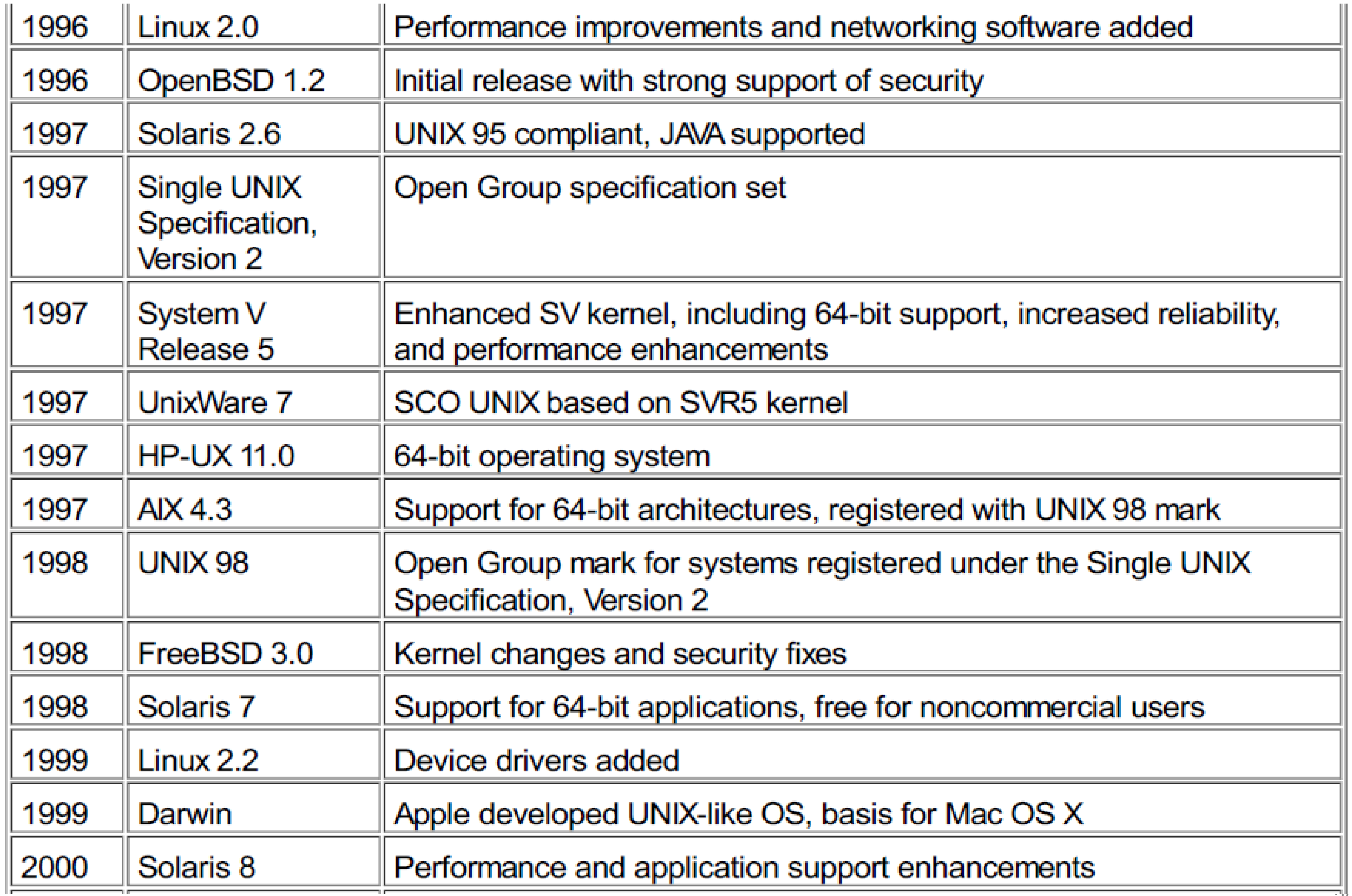
A UNIX System Timeline



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

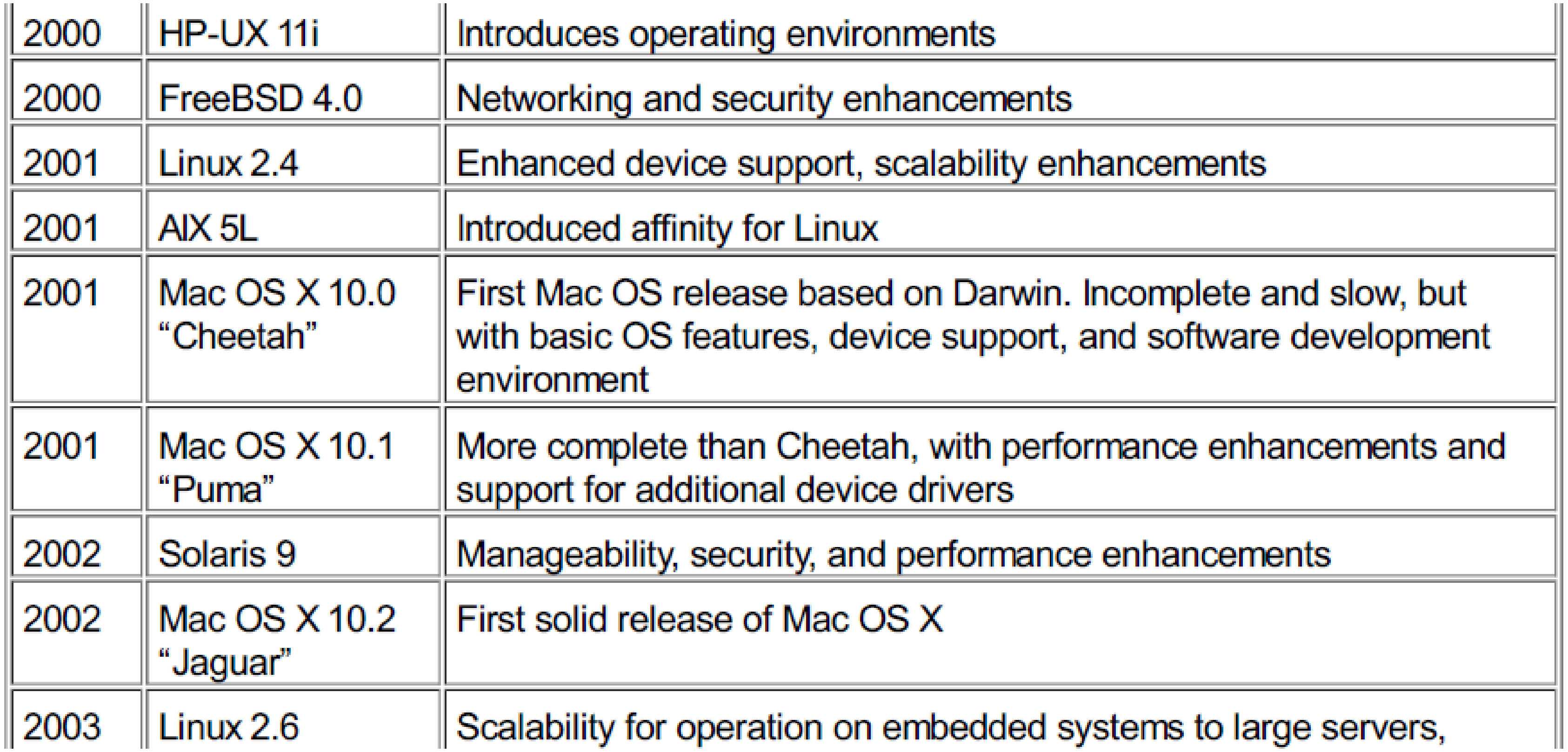
UNIX



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

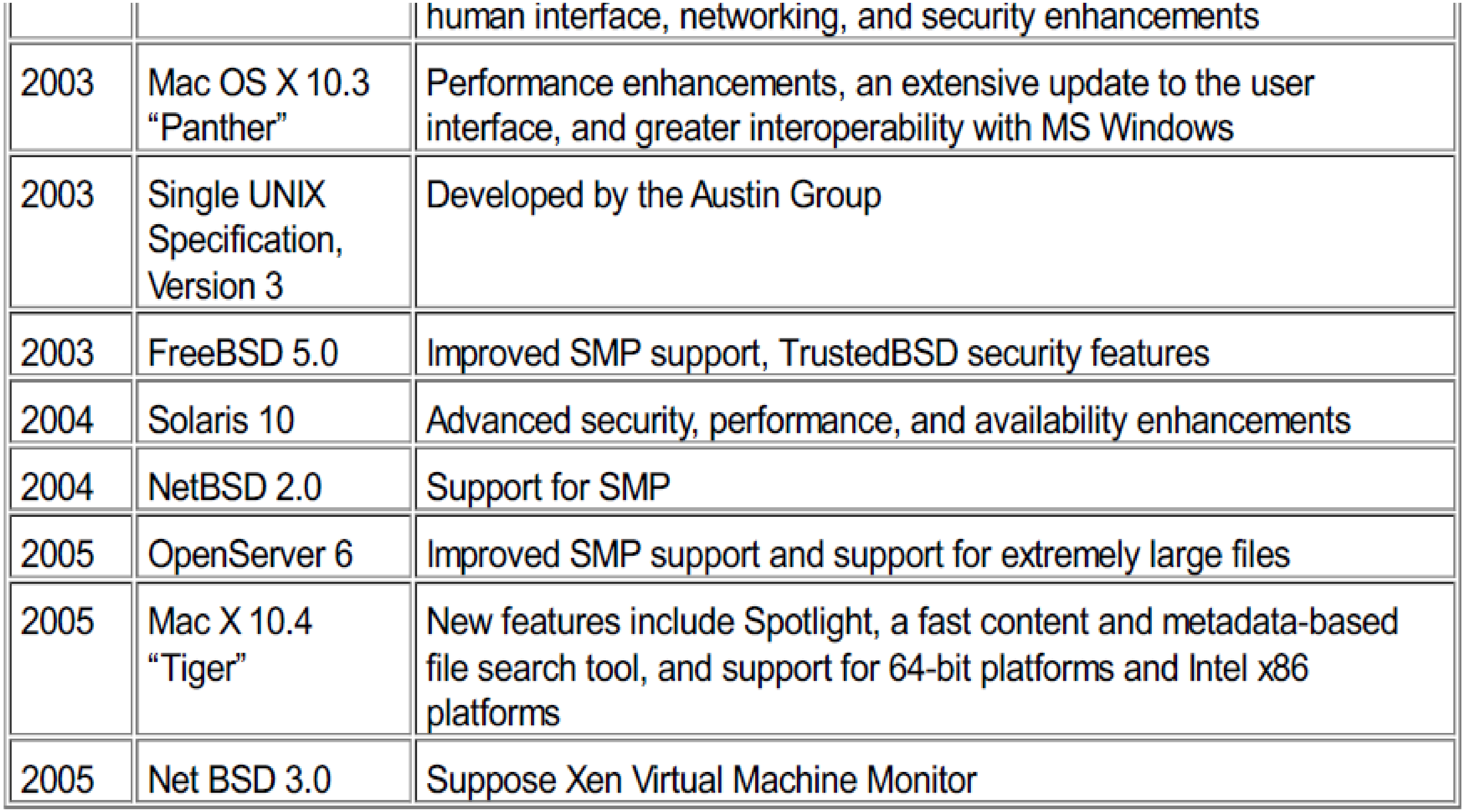
A UNIX System Timeline

UNIX



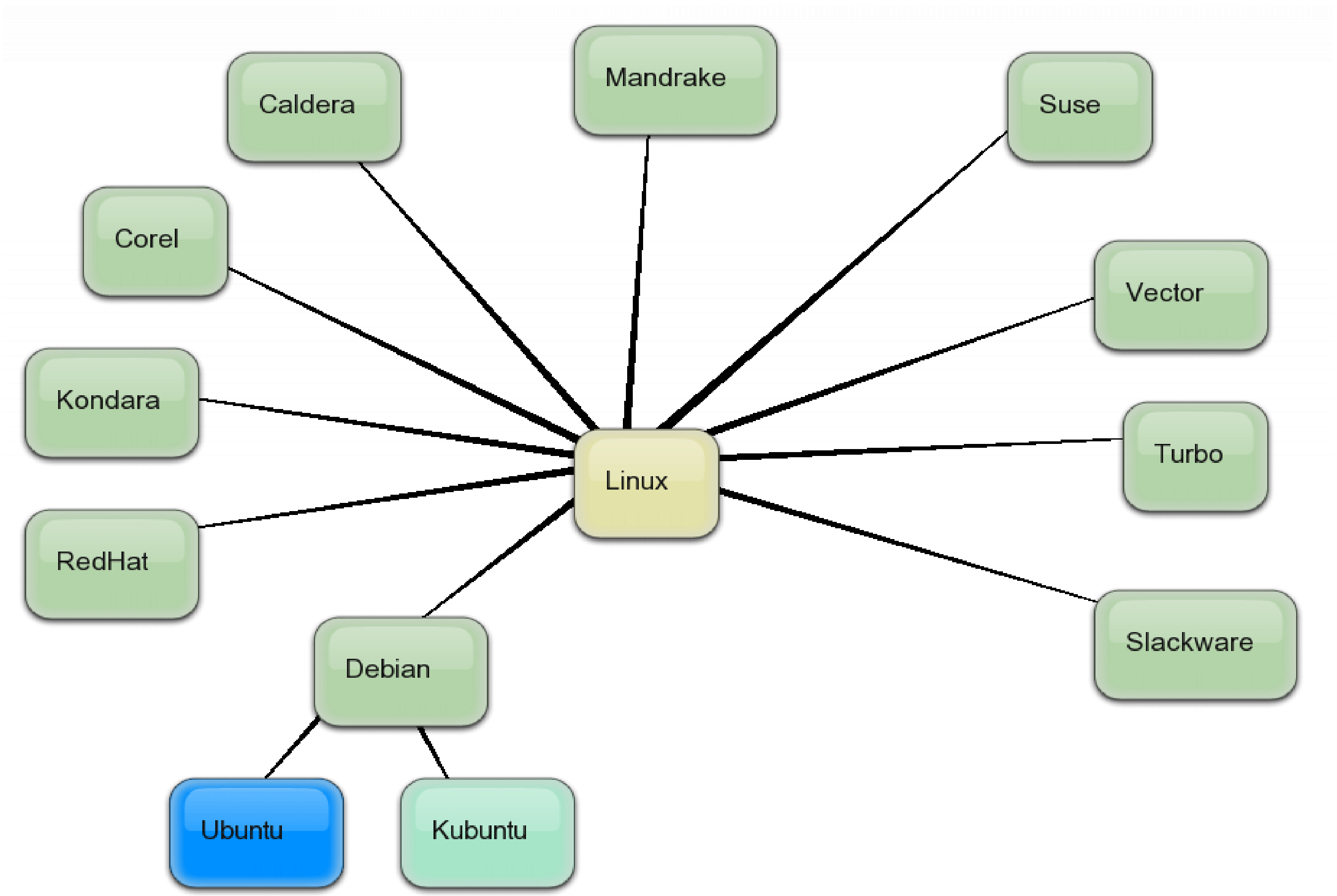
2000	HP-UX 11i	Introduces operating environments
2000	FreeBSD 4.0	Networking and security enhancements
2001	Linux 2.4	Enhanced device support, scalability enhancements
2001	AIX 5L	Introduced affinity for Linux
2001	Mac OS X 10.0 “Cheetah”	First Mac OS release based on Darwin. Incomplete and slow, but 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

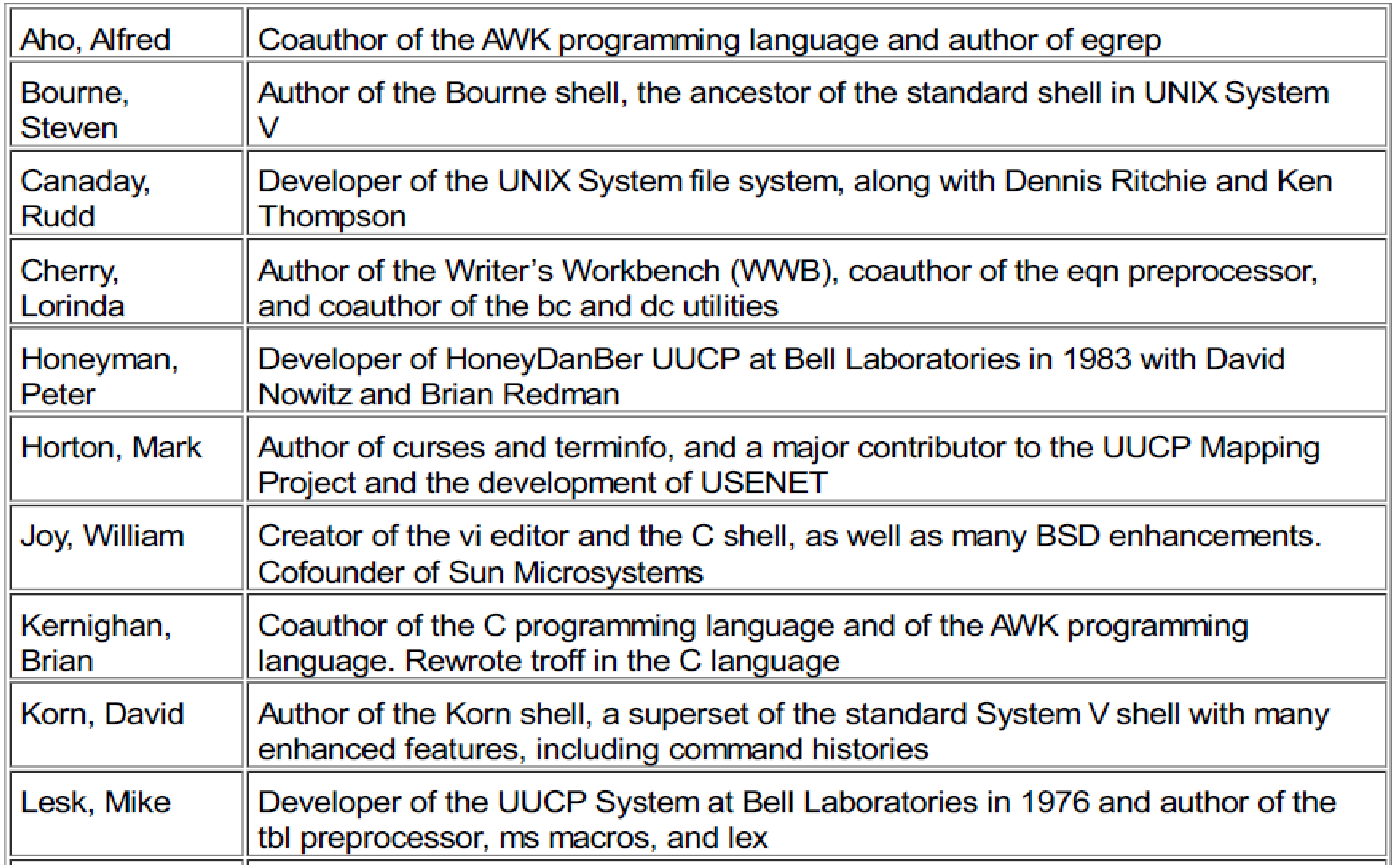


		human interface, networking, and security enhancements
2003	Mac OS X 10.3 “Panther”	Performance enhancements, an extensive update to the user interface, and greater interoperability with MS Windows
2003	Single UNIX Specification, Version 3	Developed by the Austin Group
2003	FreeBSD 5.0	Improved SMP support, 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

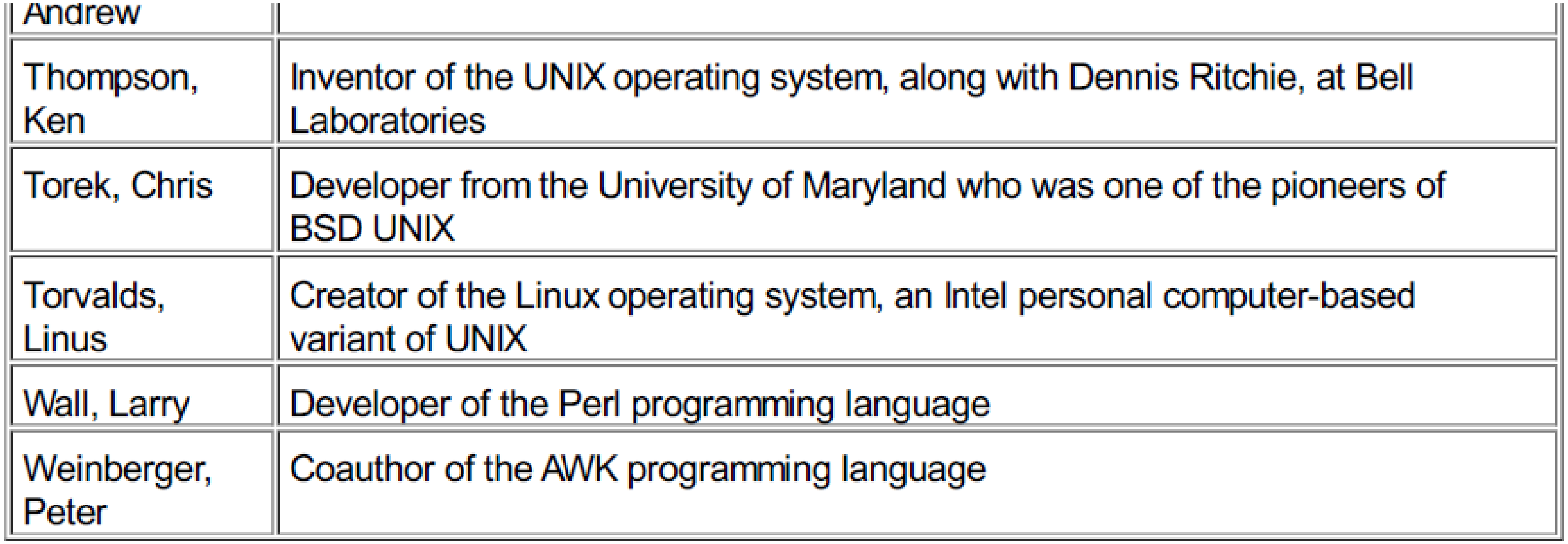


Aho, Alfred	Coauthor of the AWK programming language and author of egrep
Bourne, Steven	Author of the Bourne shell, the ancestor of the standard shell in UNIX System V
Canaday, Rudd	Developer of the UNIX System file system, along with Dennis Ritchie and Ken Thompson
Cherry, Lorinda	Author of the Writer's 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

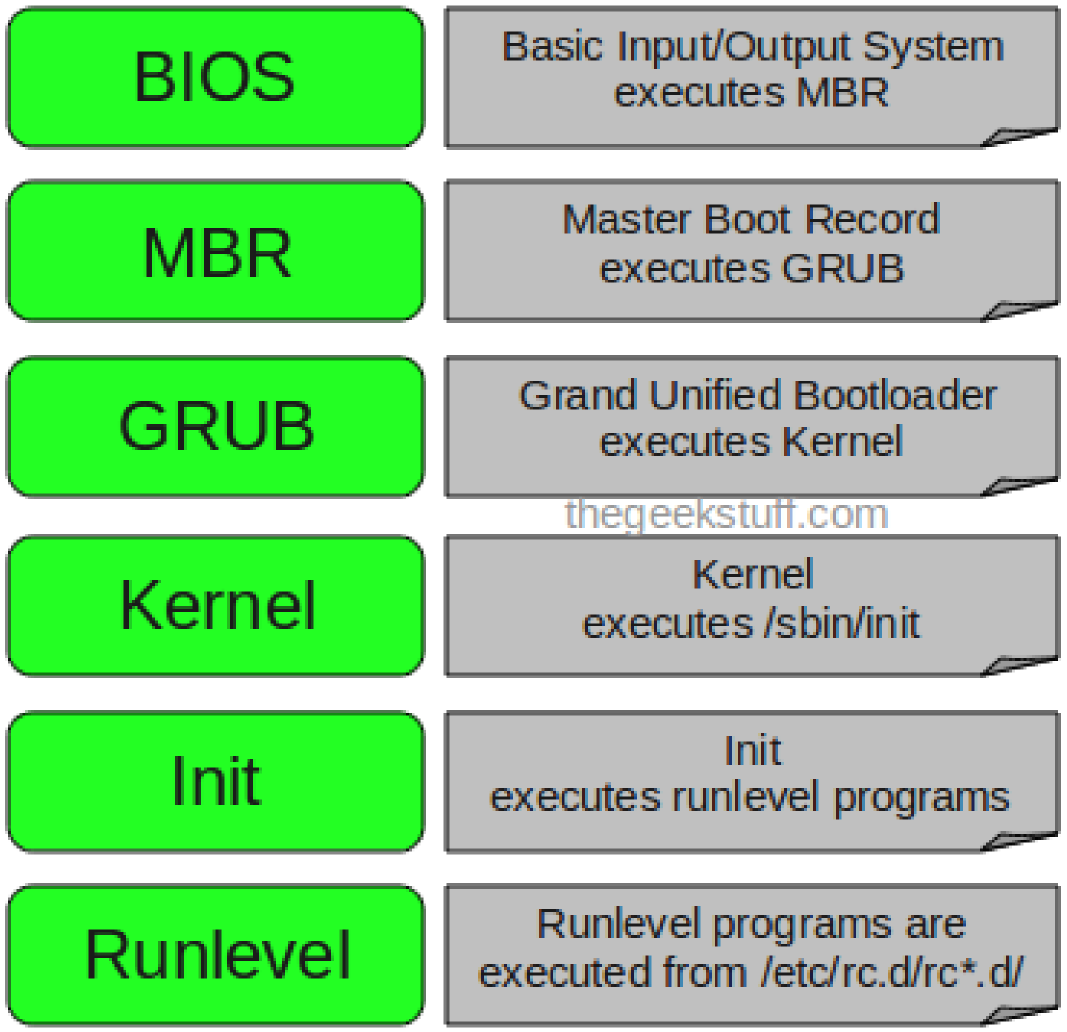
UNIX Contributors



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

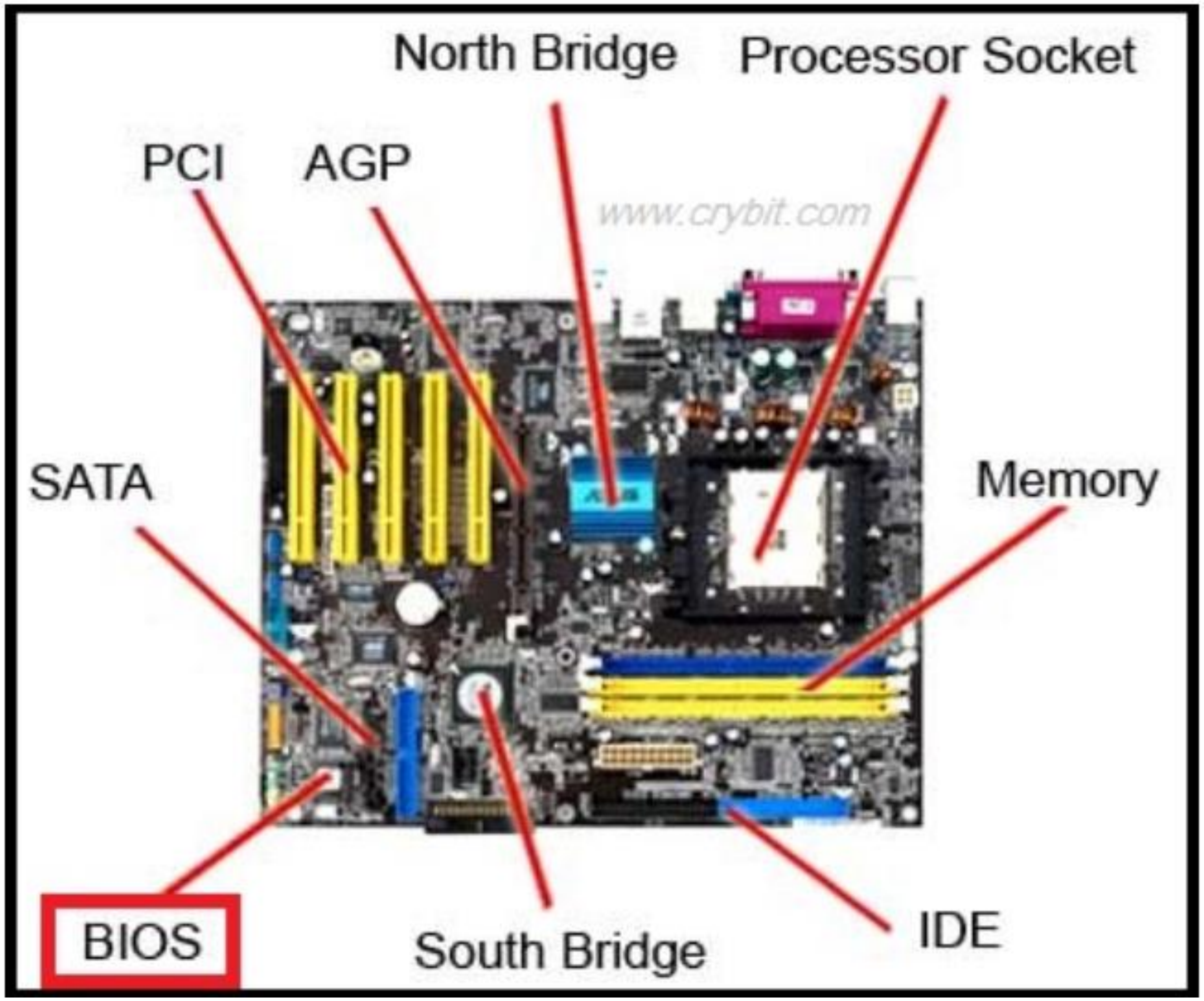
UNIX


BIOS
Basic Input/Output System executes MBR
MBR
Master Boot Record executes GRUB
GRUB
Grand Unified Bootloader executes Kernel
thegeekstuff.com
Kernel
Kernel executes /sbin/init
Init
Init executes runlevel programs
Runlevel
Runlevel programs are executed from /etc/rc.d/rc*.d/

Bios

- 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

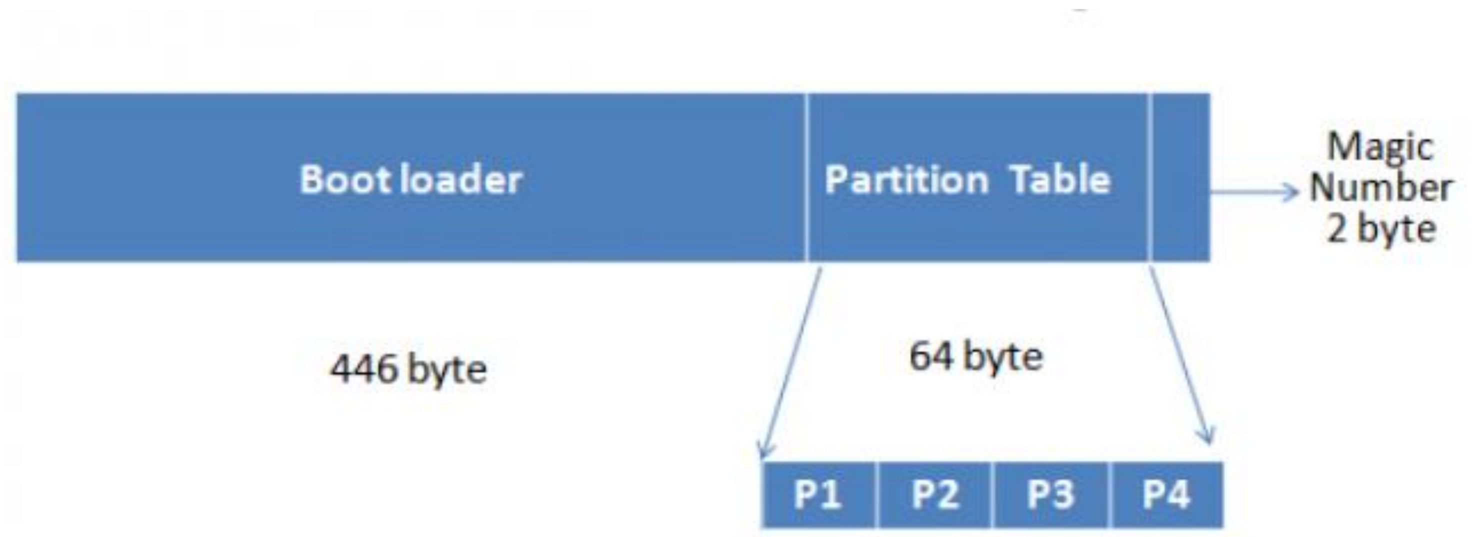

North Bridge
Processor Socket
PCI
AGP
www.crybit.com
SATA
Memory
BIOS
South Bridge
IDE

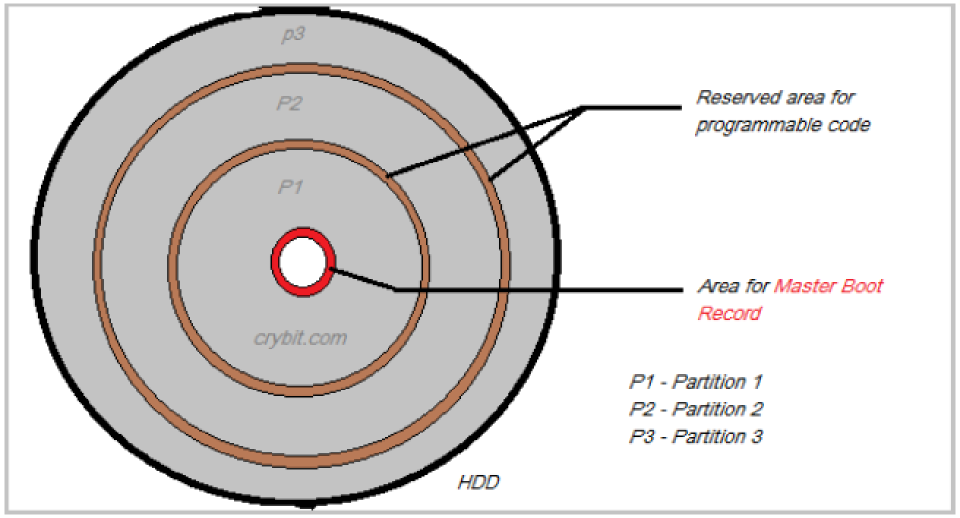
MBR

- 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.

UNIX

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

- 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

- 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

- 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

- 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

- 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.

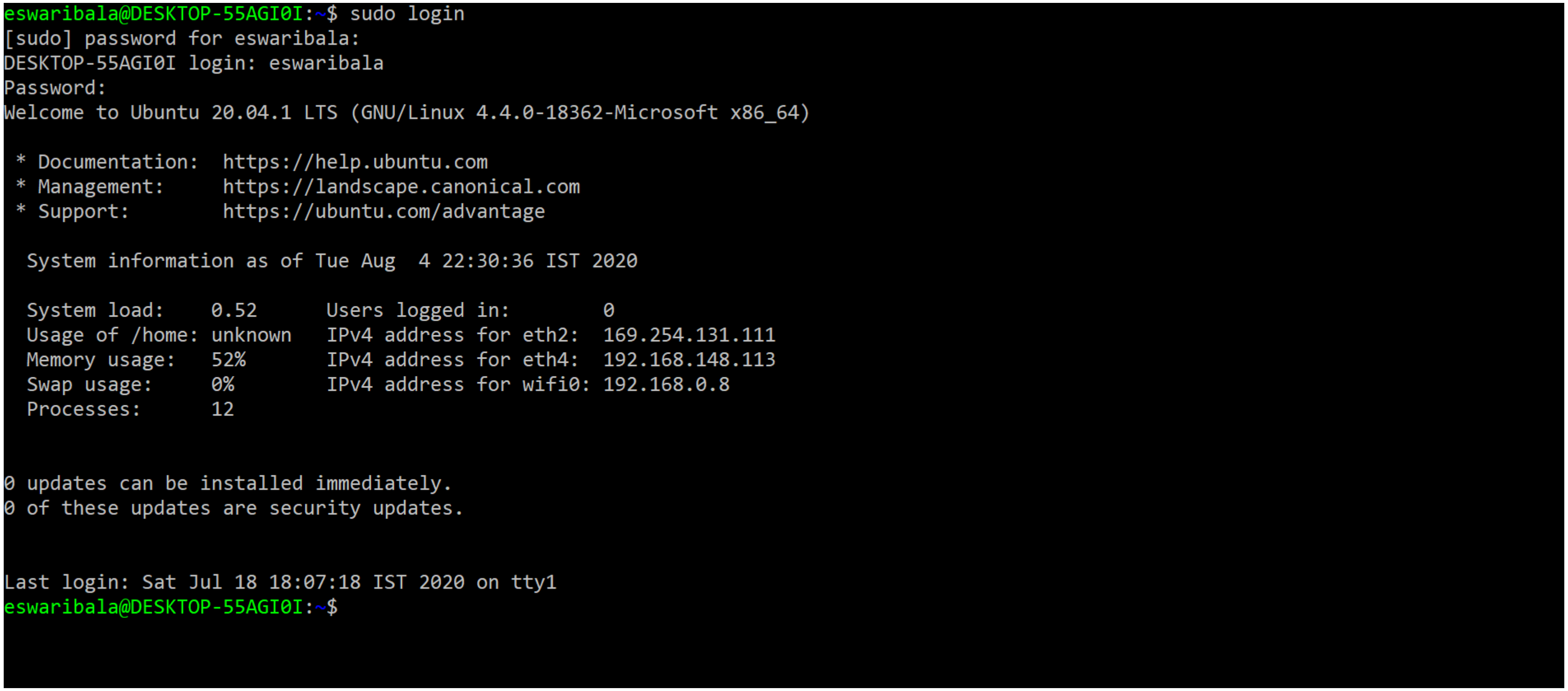
Install Ubuntu in Windows 10

- 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

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.

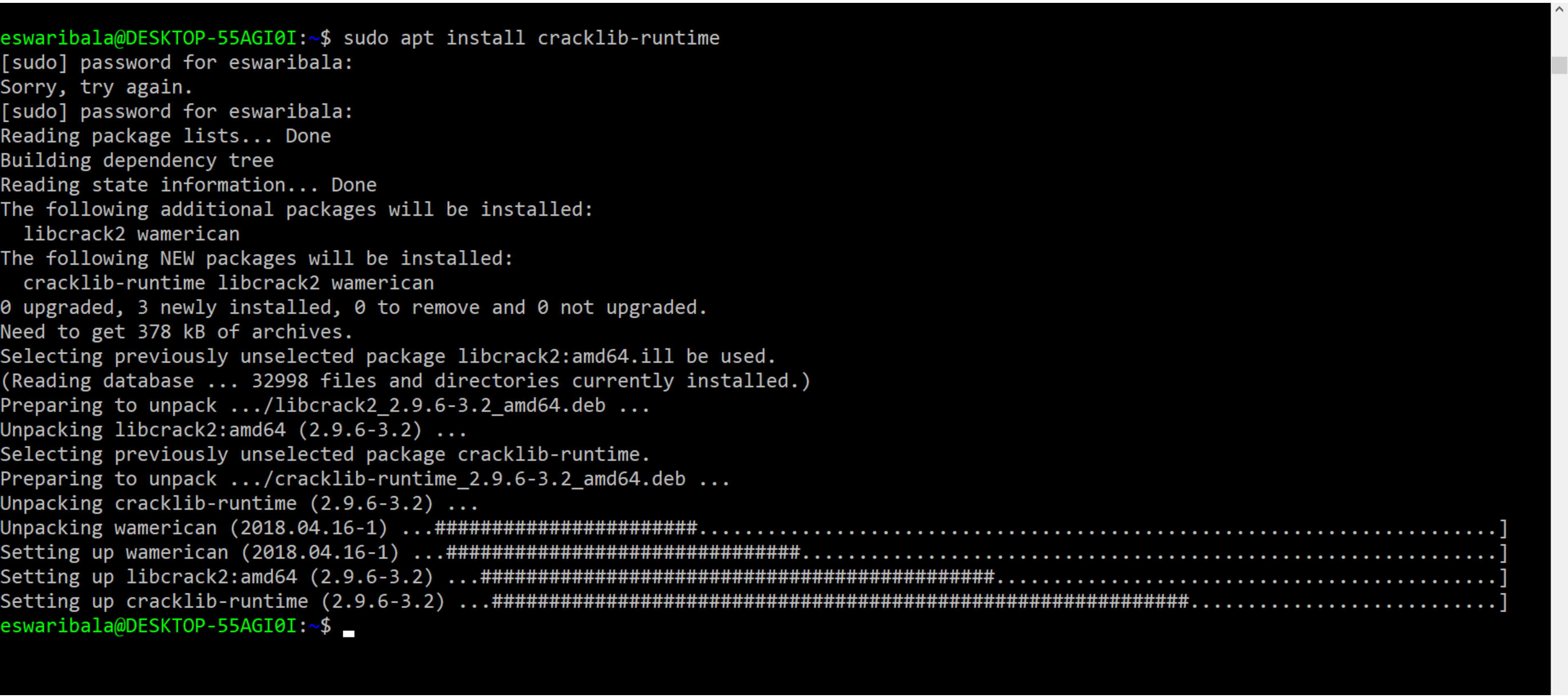
UNIX System Password Security

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

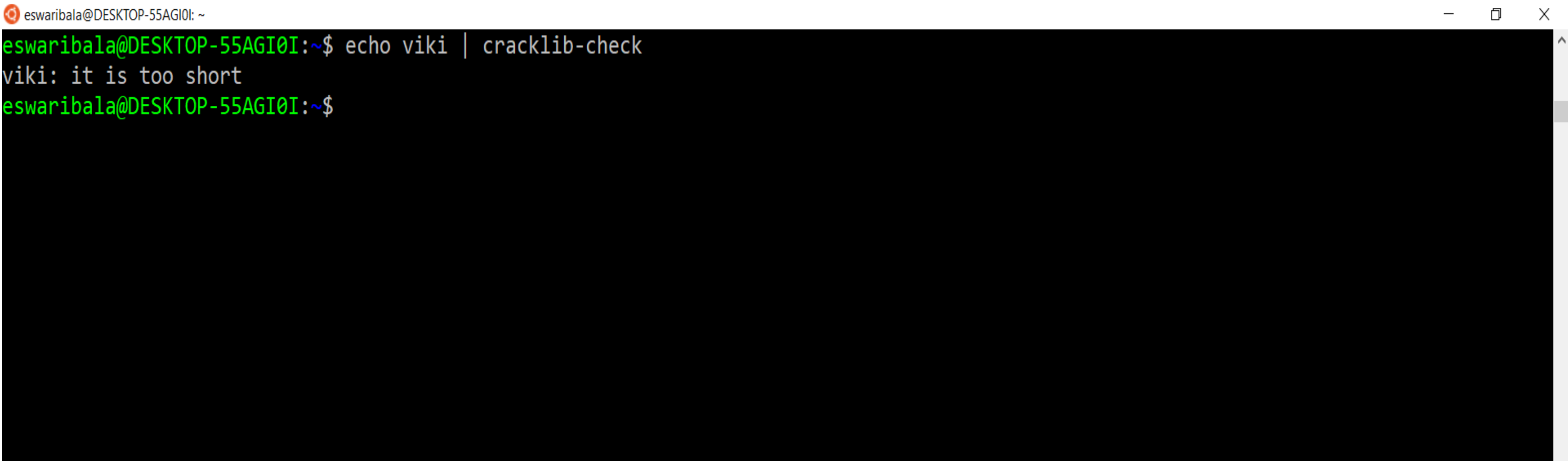
sudo apt install cracklib-runtime



```
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

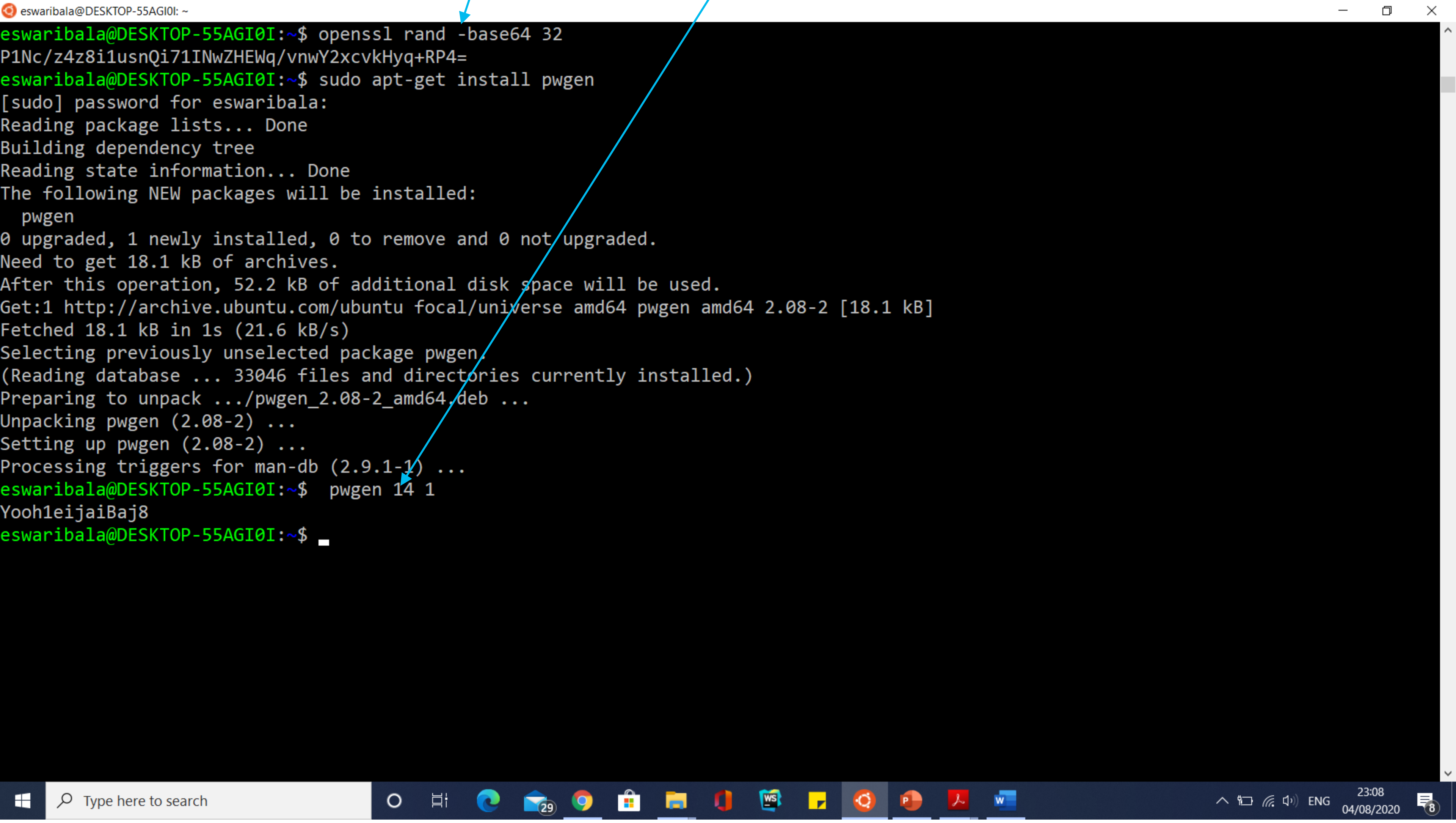


eswaribala@DESKTOP-55AGI0I: ~

```
eswaribala@DESKTOP-55AGI0I:~$ echo viki | cracklib-check
viki: it is too short
eswaribala@DESKTOP-55AGI0I:~$
```

Generate Strong Passwords

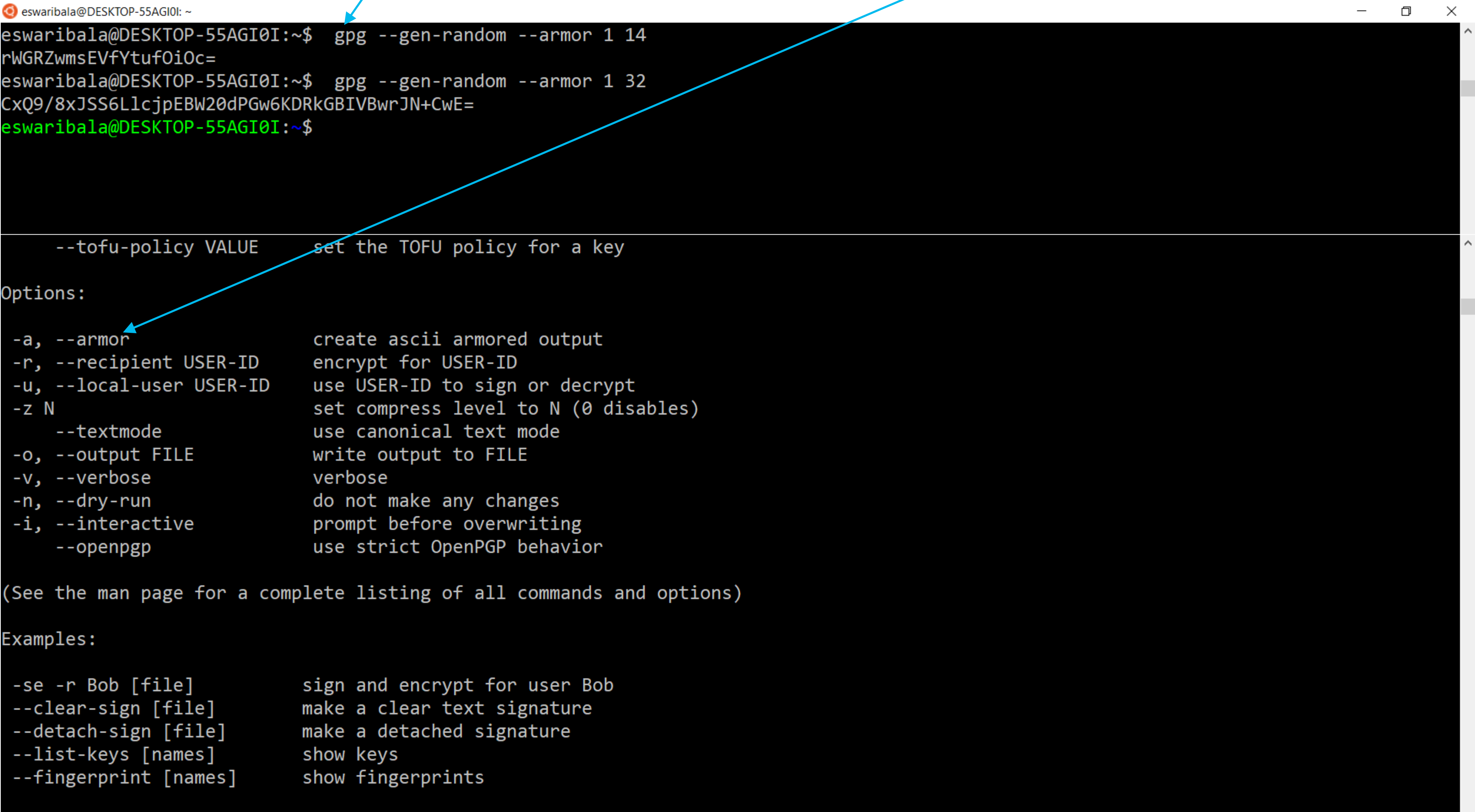
UNIX



eswaribala@DESKTOP-55AGI0I: ~

```
eswaribala@DESKTOP-55AGI0I:~$ openssl rand -base64 32
P1Nc/z4z8i1usnQi71INwZHEWq/vnwY2xcvkHyq+RP4=
eswaribala@DESKTOP-55AGI0I:~$ sudo apt-get install pwgen
[sudo] password for eswaribala:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  pwgen
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 18.1 kB of archives.
After this operation, 52.2 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu focal/universe amd64 pwgen amd64 2.08-2 [18.1 kB]
Fetched 18.1 kB in 1s (21.6 kB/s)
Selecting previously unselected package pwgen.
(Reading database ... 33046 files and directories currently installed.)
Preparing to unpack .../pwgen_2.08-2_amd64.deb ...
Unpacking pwgen (2.08-2) ...
Setting up pwgen (2.08-2) ...
Processing triggers for man-db (2.9.1-1) ...
eswaribala@DESKTOP-55AGI0I:~$  pwgen 14 1
Yooh1eijaiBaj8
eswaribala@DESKTOP-55AGI0I:~$ _
```

Generate Strong Passwords



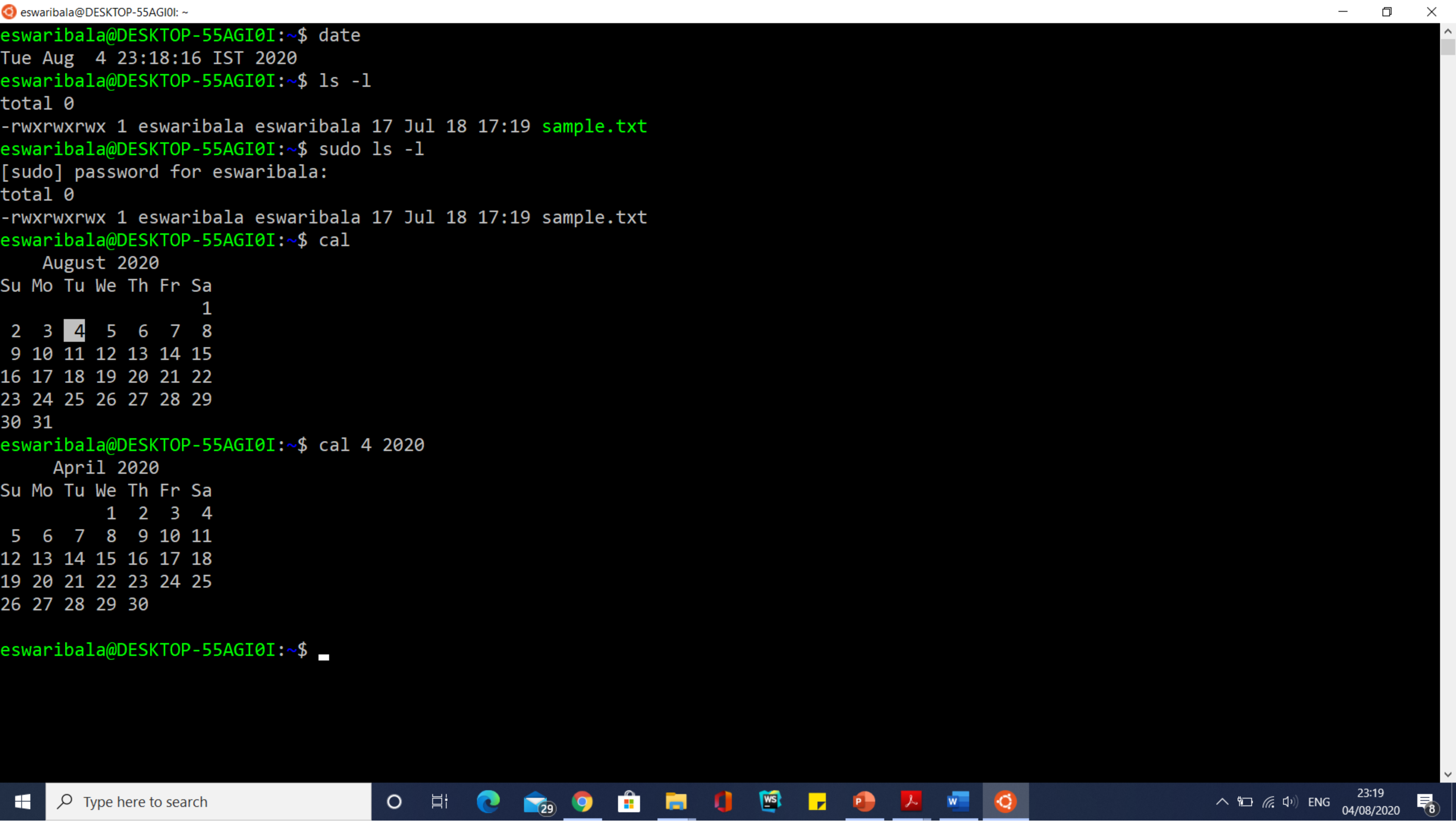
eswaribala@DESKTOP-55AGI0I: ~

```
eswaribala@DESKTOP-55AGI0I:~$  gpg --gen-random --armor 1 14
rWGRZwmsEVfYtufOiOc=
eswaribala@DESKTOP-55AGI0I:~$  gpg --gen-random --armor 1 32
CxQ9/8xJSS6LlcjpEBW20dPGw6KDRkGBIVBwrJN+CwE=
eswaribala@DESKTOP-55AGI0I:~$
```

```
     --tofu-policy VALUE     set the TOFU policy for a key

Options:

 -a, --armor                 create ascii armored output
 -r, --recipient USER-ID     encrypt for USER-ID
 -u, --local-user USER-ID    use USER-ID to sign or decrypt
 -z N                        set compress level to N (0 disables)
     --textmode              use canonical text mode
 -o, --output FILE           write output to FILE
 -v, --verbose               verbose
 -n, --dry-run               do not make any changes
 -i, --interactive           prompt before overwriting
     --openpgp               use strict OpenPGP behavior

(See the man page for a complete listing of all commands and options)

Examples:

 -se -r Bob [file]          sign and encrypt for user Bob
 --clear-sign [file]        make a clear text signature
 --detach-sign [file]       make a detached signature
 --list-keys [names]        show keys
 --fingerprint [names]      show fingerprints
```

Entering Commands

UNIX

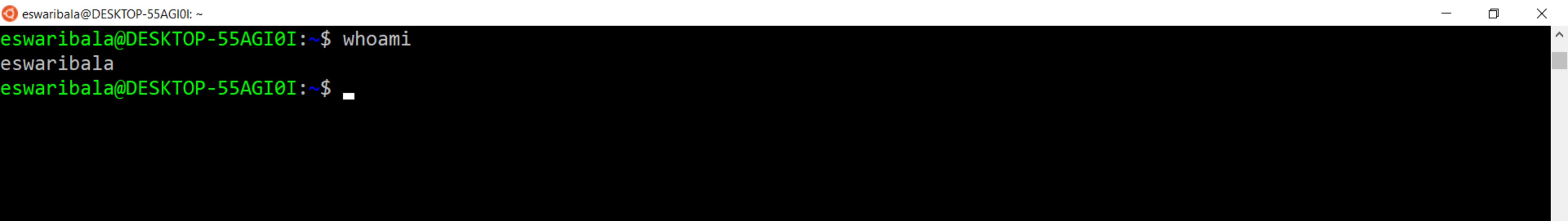


eswaribala@DESKTOP-55AGI0I: ~

```
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

Who logged in whoami

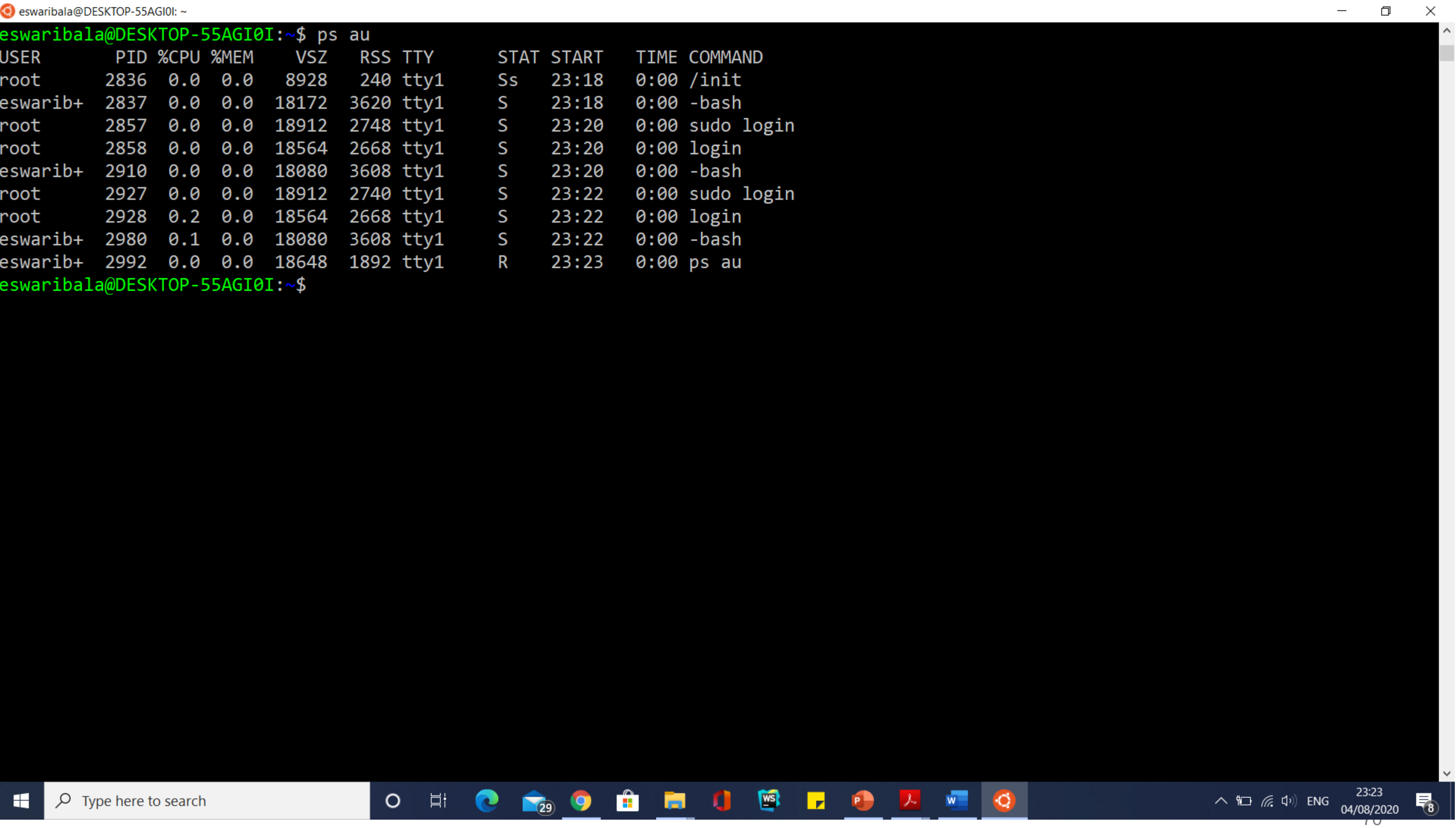

eswaribala@DESKTOP-55AGI0I: ~

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

Entering Commands

UNIX

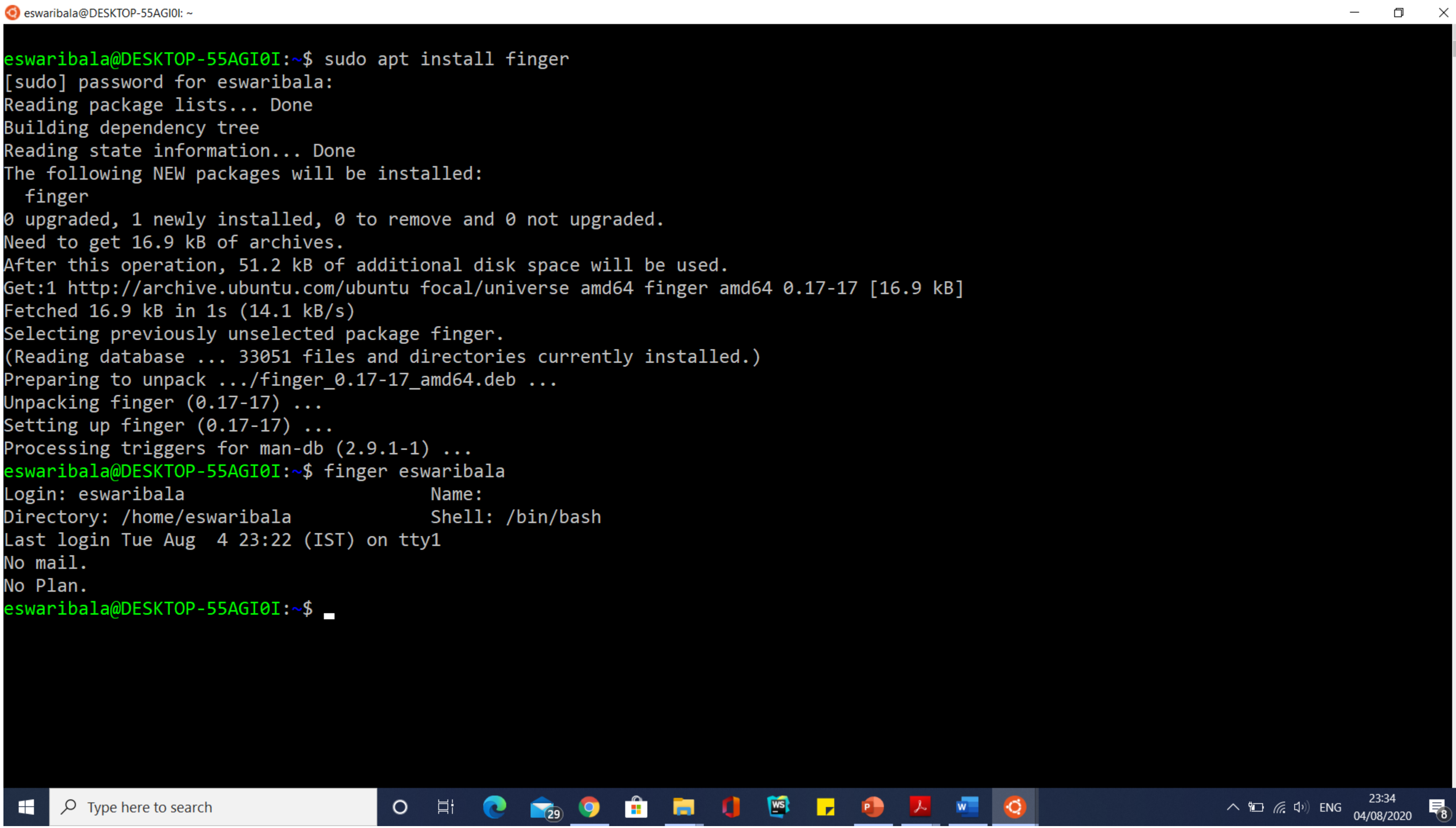
Who logged in who –H, w, who –u, ps au



eswaribala@DESKTOP-55AGI0I: ~

```
eswaribala@DESKTOP-55AGI0I:~$ ps au
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root      2836  0.0  0.0   8928   240 tty1     Ss   23:18   0:00 /init
eswarib+  2837  0.0  0.0  18172  3620 tty1     S    23:18   0:00 -bash
root      2857  0.0  0.0  18912  2748 tty1     S    23:20   0:00 sudo login
root      2858  0.0  0.0  18564  2668 tty1     S    23:20   0:00 login
eswarib+  2910  0.0  0.0  18080  3608 tty1     S    23:20   0:00 -bash
root      2927  0.0  0.0  18912  2740 tty1     S    23:22   0:00 sudo login
root      2928  0.2  0.0  18564  2668 tty1     S    23:22   0:00 login
eswarib+  2980  0.1  0.0  18080  3608 tty1     S    23:22   0:00 -bash
eswarib+  2992  0.0  0.0  18648  1892 tty1     R    23:23   0:00 ps au
eswaribala@DESKTOP-55AGI0I:~$
```

Entering Commands



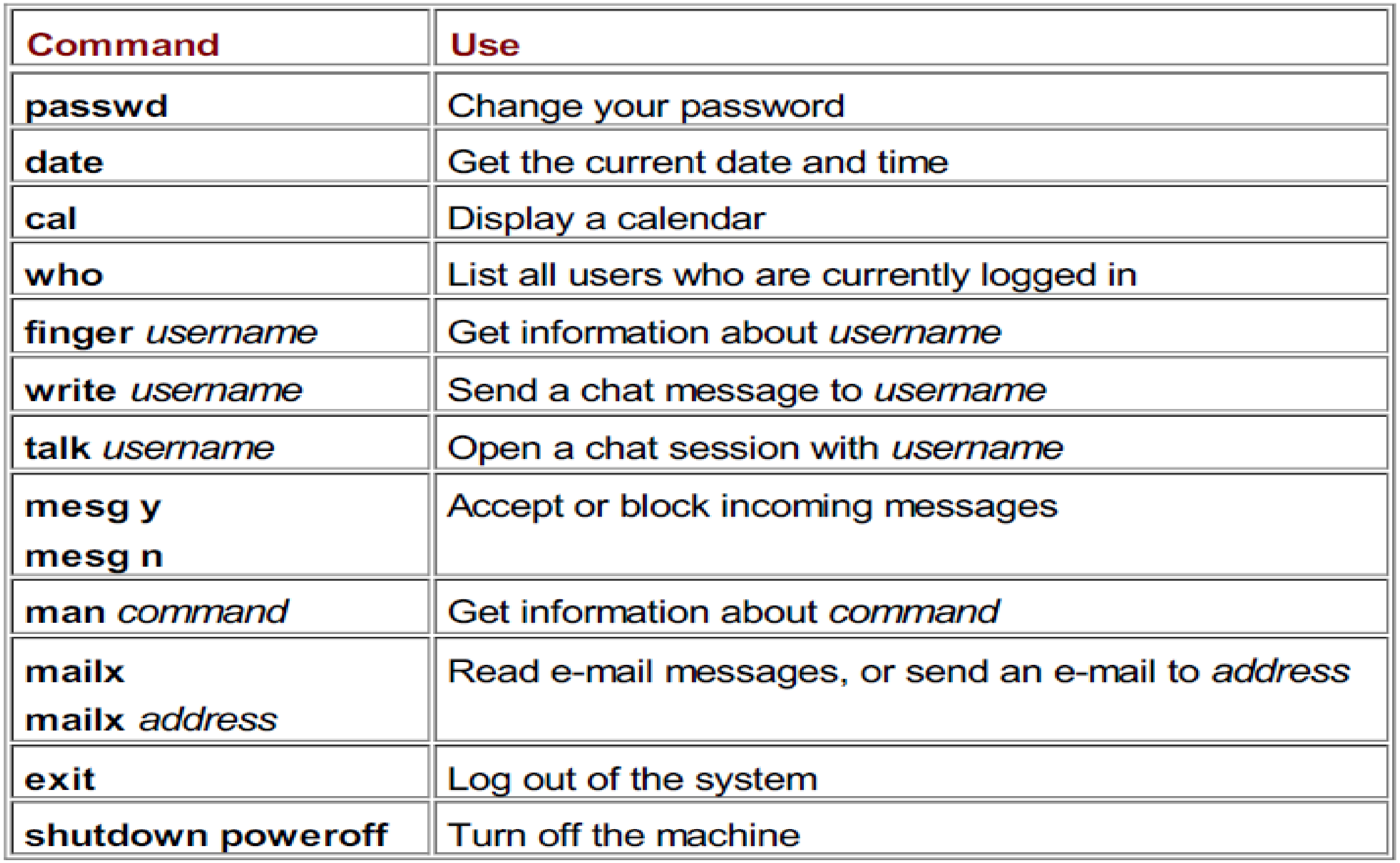
eswaribala@DESKTOP-55AGI0I: ~

```
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:~$
```

Logout

- 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



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

- 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

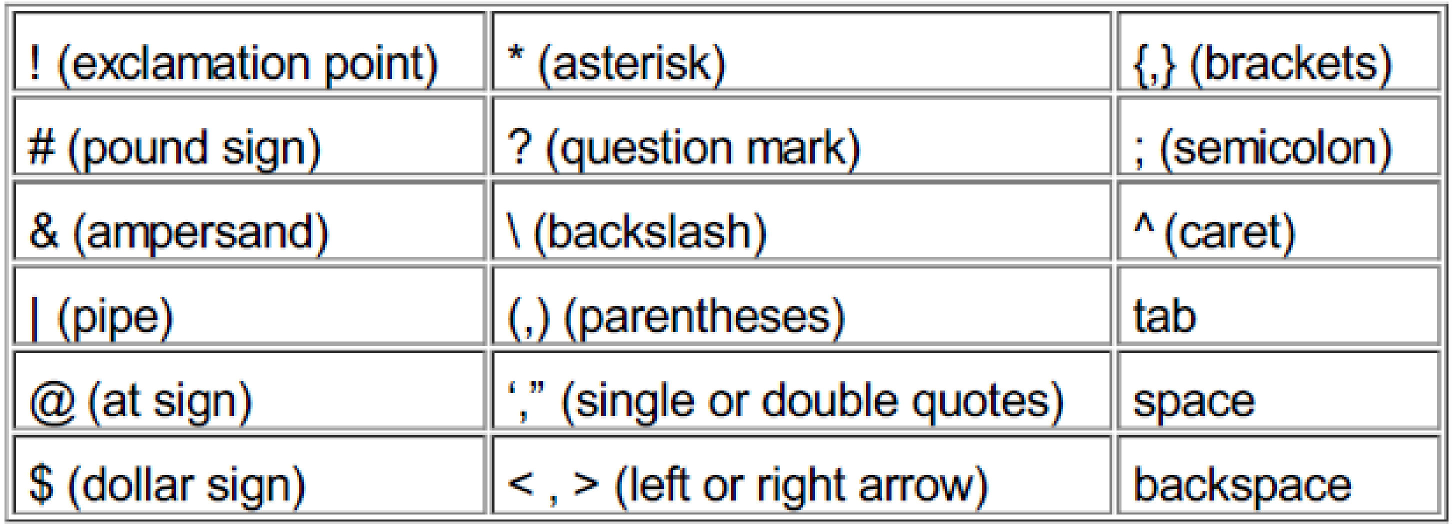
UNIX

- 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

- 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



! (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

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

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.

Filenames

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**

Filenames

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

- 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

- 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

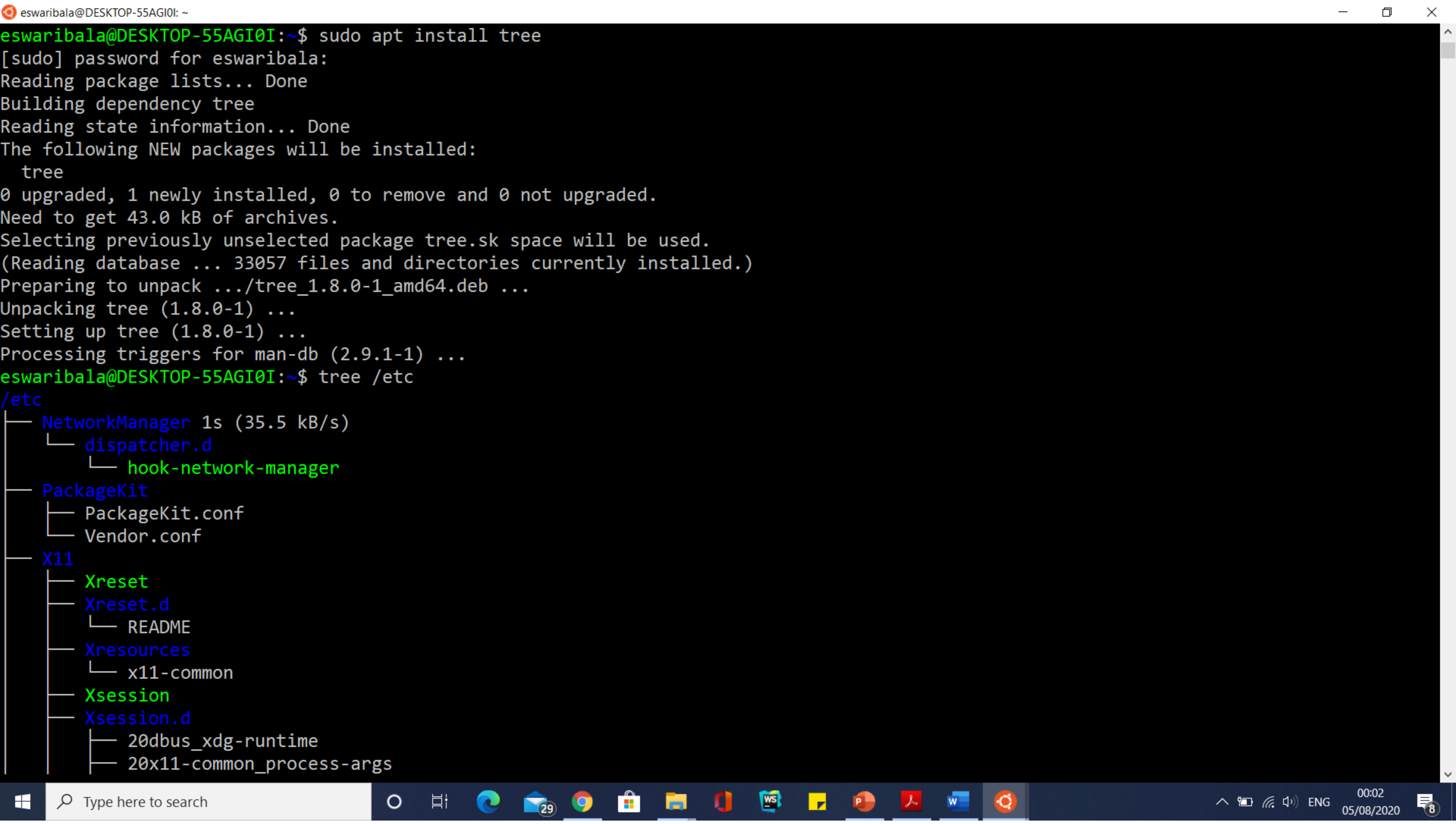
- It is a good idea to adopt a convention for naming directories so that they can be easily distinguished from ordinary files.
- Some people give directories names that are all uppercase letters, some use directory names that begin with an uppercase letter, and others distinguish directories using the 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

- 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

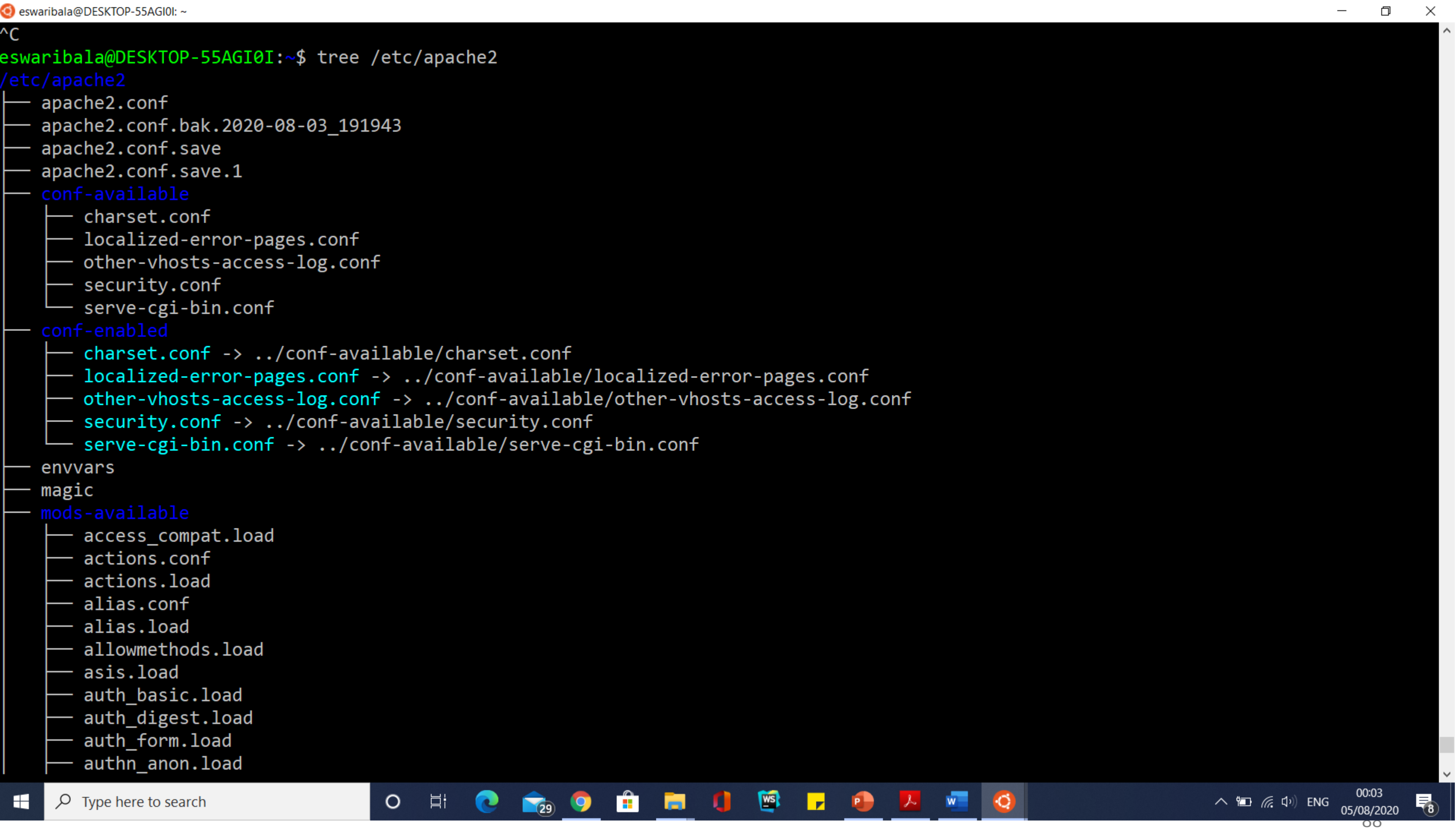


eswaribala@DESKTOP-55AGI0I: ~

```
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

- 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

- 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

- **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

- **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

- 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

- **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

- **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.

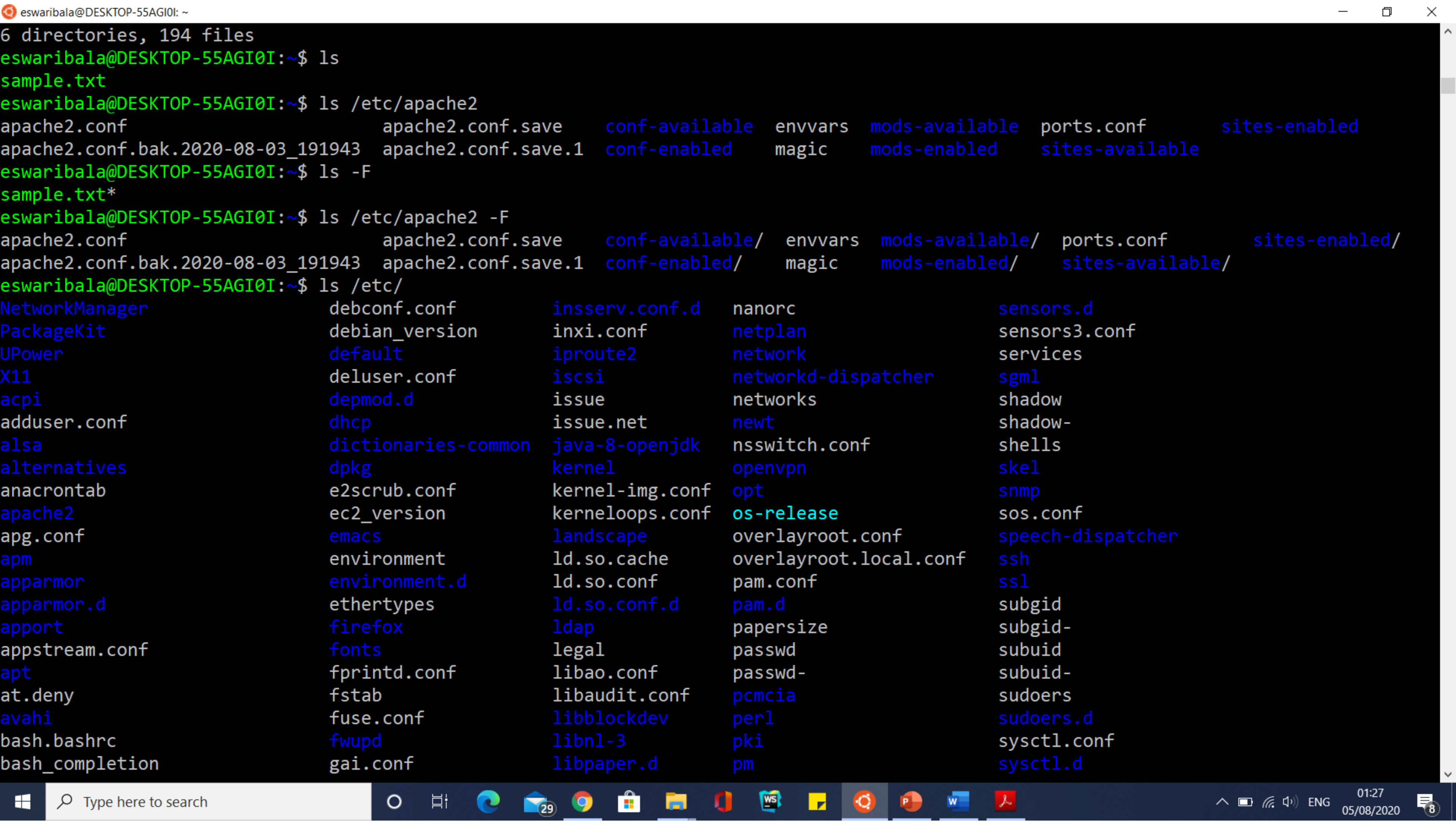
UNIX System File Types

- **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

- **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


eswaribala@DESKTOP-55AGI0I: ~

```
6 directories, 194 files
eswaribala@DESKTOP-55AGI0I:~$ ls
sample.txt
eswaribala@DESKTOP-55AGI0I:~$ ls /etc/apache2
apache2.conf                    apache2.conf.save    conf-available  envvars  mods-available  ports.conf       sites-enabled
apache2.conf.bak.2020-08-03_191943  apache2.conf.save.1  conf-enabled    magic    mods-enabled    sites-available
eswaribala@DESKTOP-55AGI0I:~$ ls -F
sample.txt*
eswaribala@DESKTOP-55AGI0I:~$ ls /etc/apache2 -F
apache2.conf                    apache2.conf.save    conf-available/  envvars  mods-available/  ports.conf        sites-enabled/
apache2.conf.bak.2020-08-03_191943  apache2.conf.save.1  conf-enabled/    magic    mods-enabled/    sites-available/
eswaribala@DESKTOP-55AGI0I:~$ ls /etc/
NetworkManager                 debconf.conf          insserv.conf.d    nanorc                   sensors.d
PackageKit                     debian_version        inxi.conf         netplan                  sensors3.conf
UPower                         default               iproute2          network                  services
X11                            deluser.conf          iscsi             networkd-dispatcher      sgml
acpi                           depmod.d              issue             networks                 shadow
adduser.conf                   dhcp                  issue.net         newt                     shadow-
alsa                           dictionaries-common   java-8-openjdk    nsswitch.conf            shells
alternatives                   dpkg                  kernel            openvpn                  skel
anacrontab                     e2scrub.conf          kernel-img.conf   opt                      snmp
apache2                        ec2_version           kerneloops.conf   os-release               sos.conf
apg.conf                       emacs                 landscape         overlayroot.conf         speech-dispatcher
apm                            environment           ld.so.cache       overlayroot.local.conf   ssh
apparmor                       environment.d         ld.so.conf        pam.conf                 ssl
apparmor.d                     ethertypes            ld.so.conf.d      pam.d                    subgid
apport                         firefox               ldap              papersize                subgid-
appstream.conf                 fonts                 legal             passwd                   subuid
apt                            fprintd.conf          libao.conf        passwd-                  subuid-
at.deny                        fstab                 libaudit.conf     pcmcia                   sudoers
avahi                          fuse.conf             libblockdev       perl                     sudoers.d
bash.bashrc                    fwupd                 libnl-3           pki                      sysctl.conf
bash_completion                gai.conf              libpaper.d        pm                       sysctl.d
```

UNIX System File Types

- **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

- **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

- **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

- **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

- **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

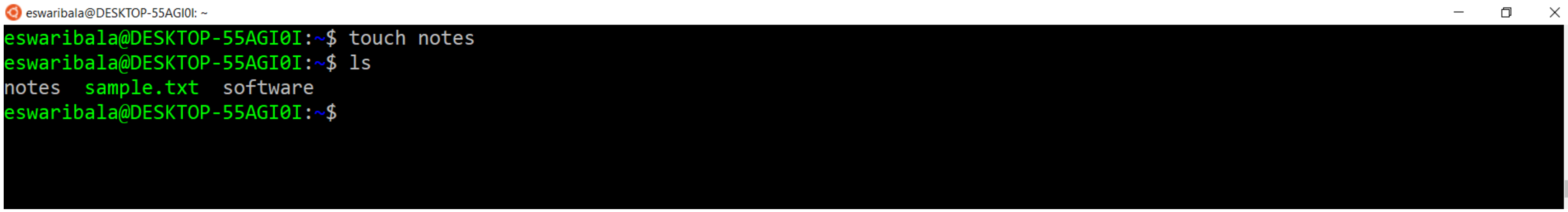
- **Creating a File**
- So far, all the examples you have seen involved using cat to copy one or more normal files, either to another file or to your screen. But other possibilities exist.
- Just as your screen is the default output for cat and other commands, your keyboard is the default input.
- If 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

- 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

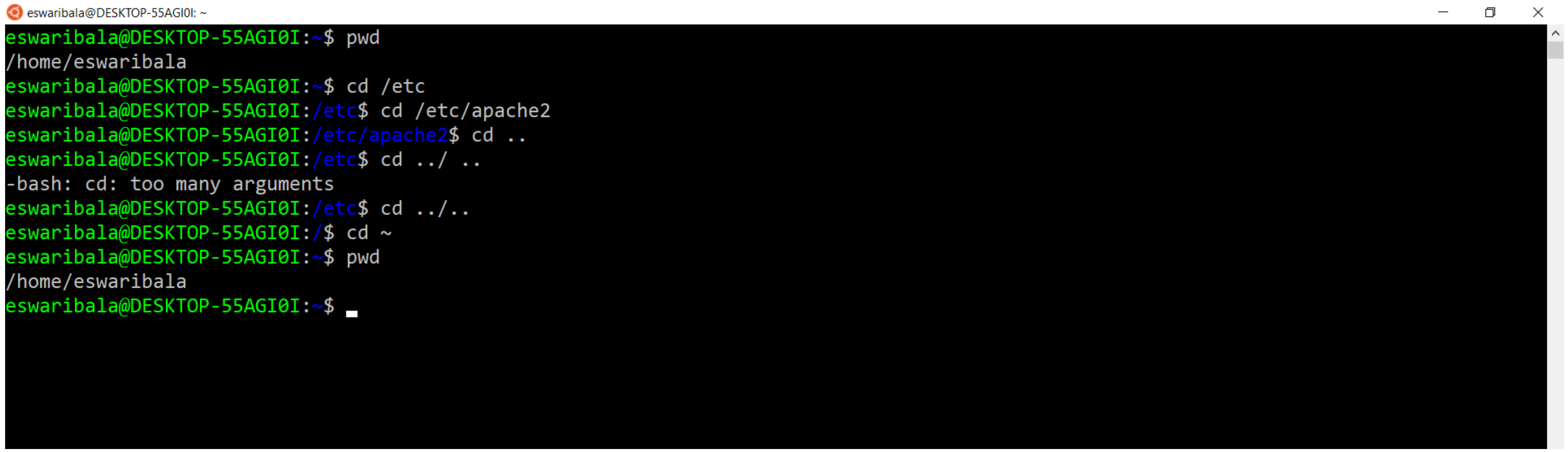


eswaribala@DESKTOP-55AGI0I: ~

```
eswaribala@DESKTOP-55AGI0I:~$ touch notes
eswaribala@DESKTOP-55AGI0I:~$ ls
notes  sample.txt  software
eswaribala@DESKTOP-55AGI0I:~$
```

UNIX System File Types

- Moving Around in Directories



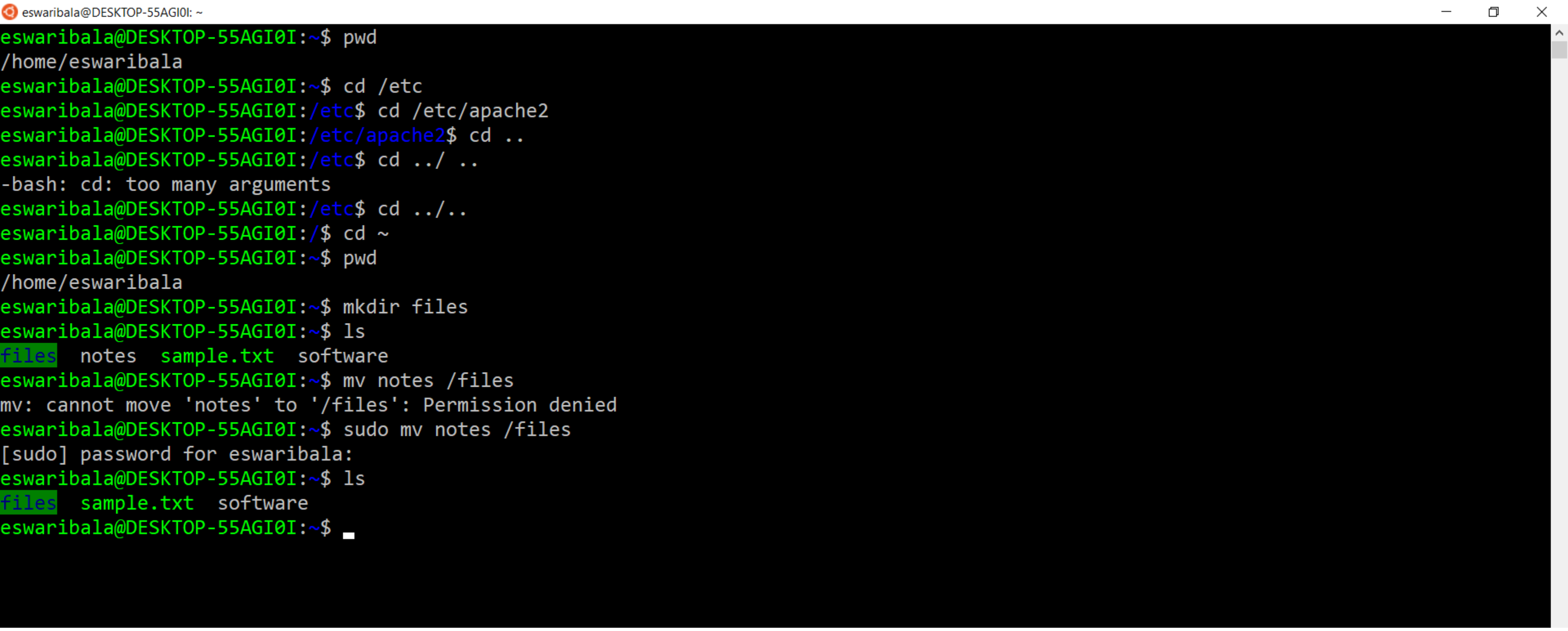
eswaribala@DESKTOP-55AGI0I: ~

```
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:~$ _
```

UNIX System File Types

UNIX

- 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

- **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

- **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

UNIX System File Types

- **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

- **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

- **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

- **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

- **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

- **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

- **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

- **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

- **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

- **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

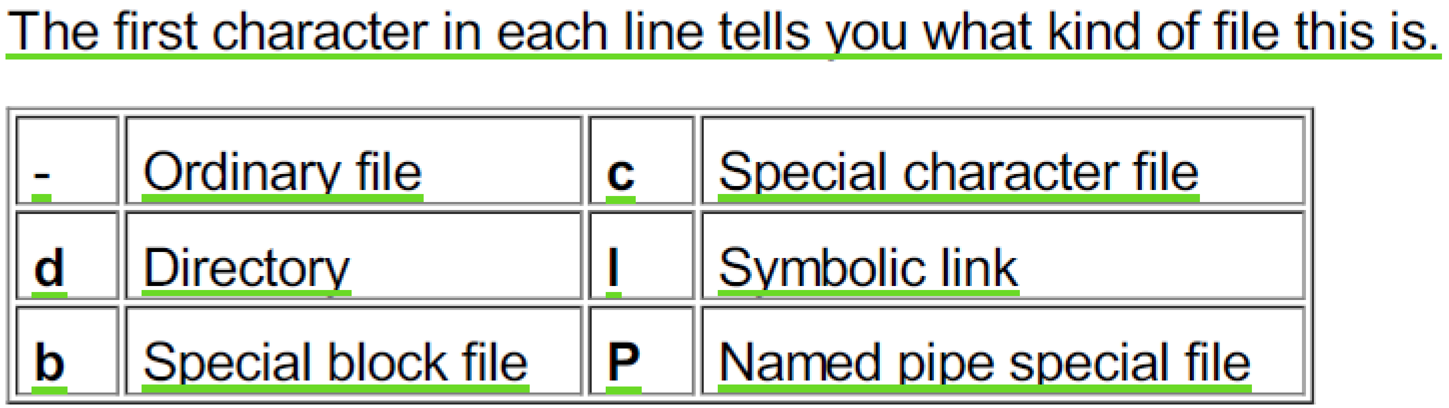
- **Controlling the Way ls Displays Filenames**
- You can use the -x option to have names of files displayed horizontally, in as many lines as necessary
- For example,
- $ ls –x
- You also can use the -1 (one) option to have files displayed one line per row (as the old version of ls
- did), in alphabetical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

UNIX System File Types

- **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

- **Controlling the Way ls 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

- **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

- **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

- **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

- **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

- **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

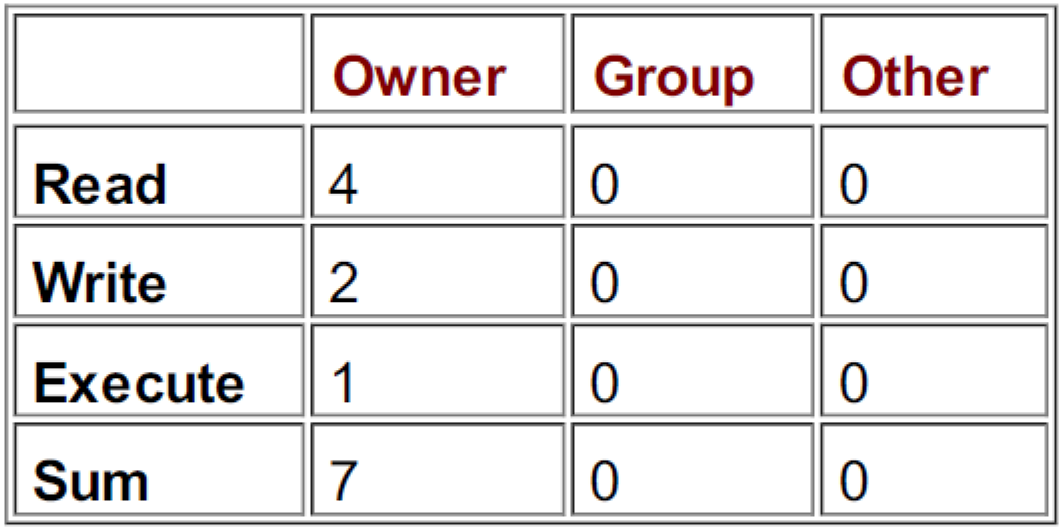
- **The chmod Command**
- To alter a file’s permissions, you use the chmod (change mode) command.
- You specify the changes you want to make with a sort of code.
- First, show which set of permissions you are changing with u for user, g for group, or o for other.
- Second, specify how they should be 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

- **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

- **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

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

UNIX System File Types

- **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

- **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

- **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

- **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

- **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.

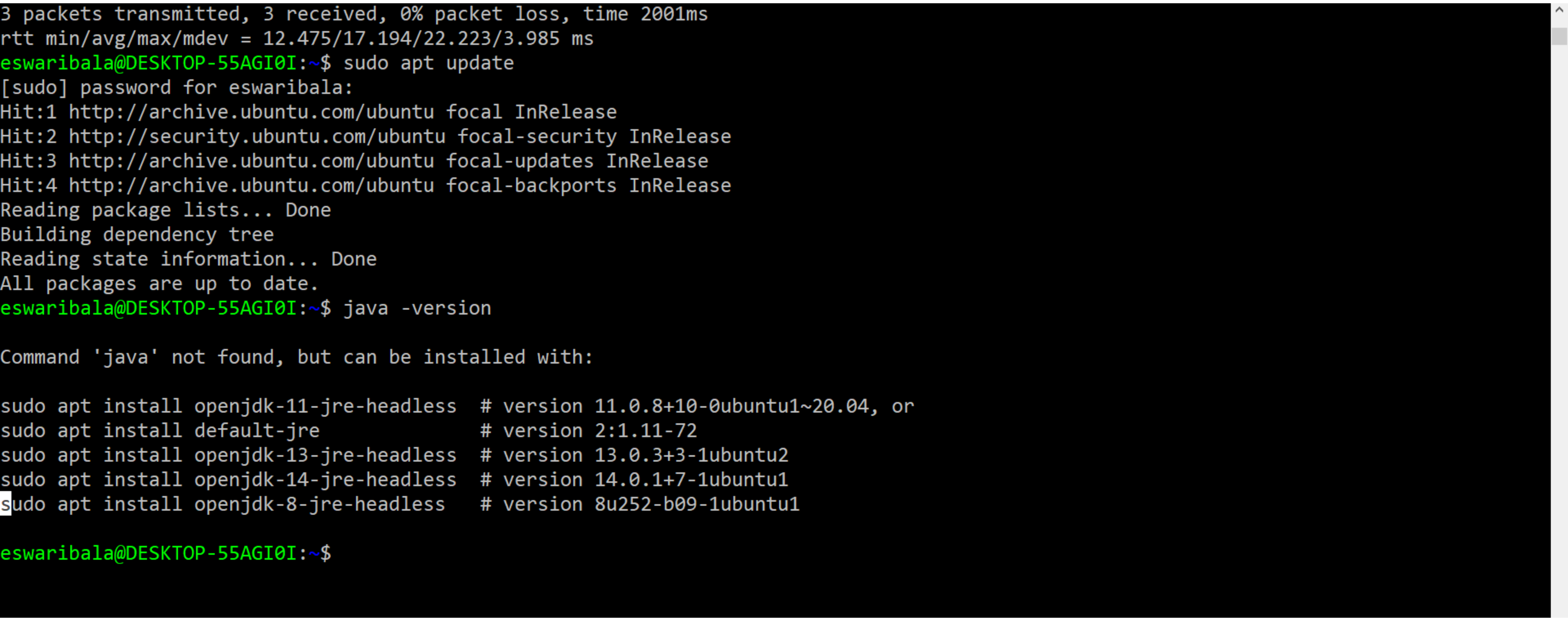
UNIX System File Types

- **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

Java Installation

UNIX

sudo apt update
java -version

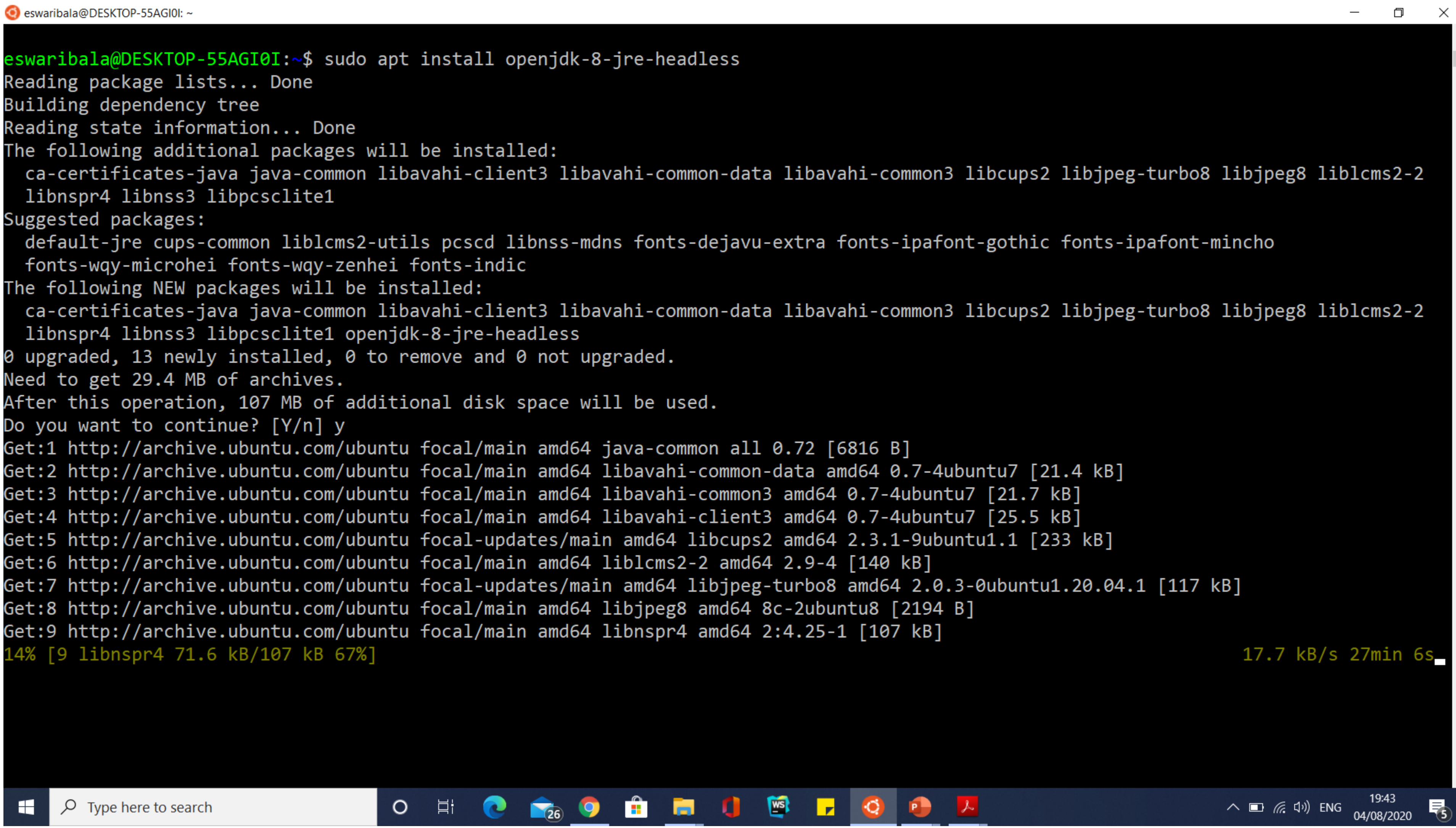


```
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 12.475/17.194/22.223/3.985 ms
eswaribala@DESKTOP-55AGI0I:~$ sudo apt update
[sudo] password for eswaribala:
Hit:1 http://archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://security.ubuntu.com/ubuntu focal-security 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



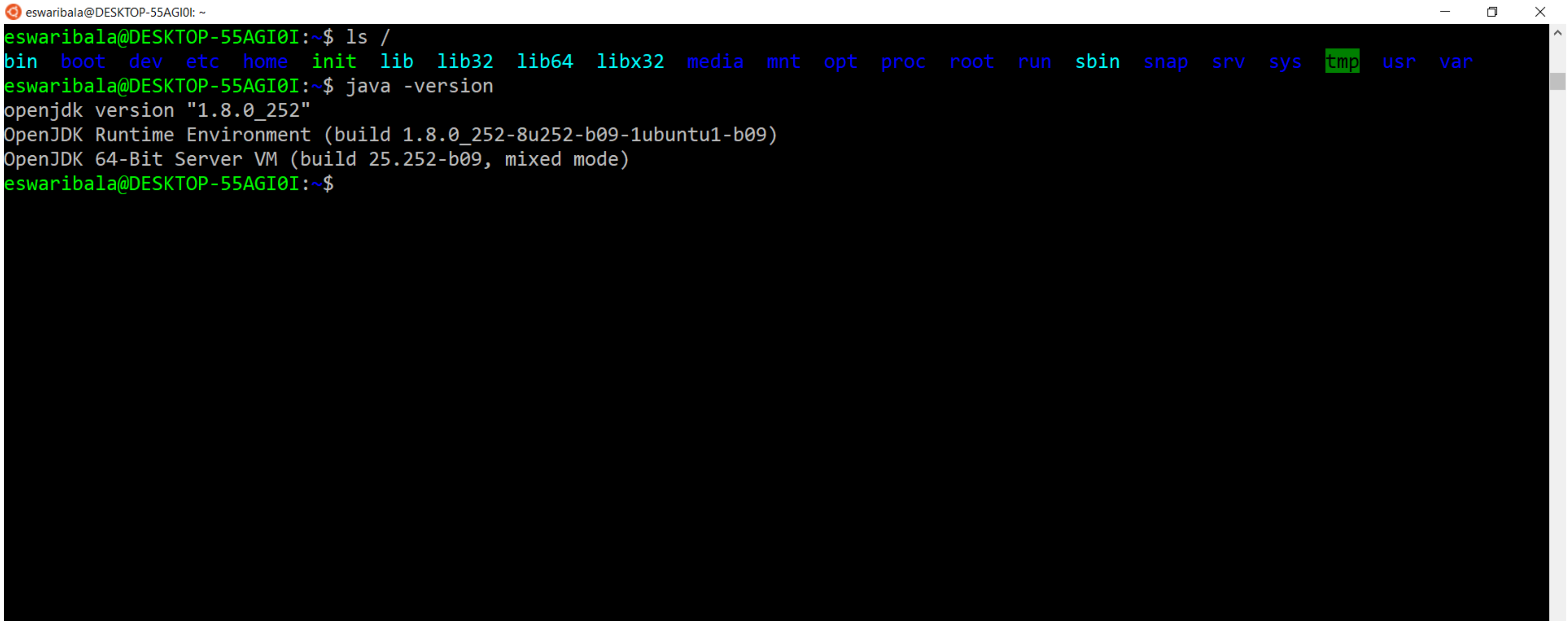
eswaribala@DESKTOP-55AGI0I: ~

```
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

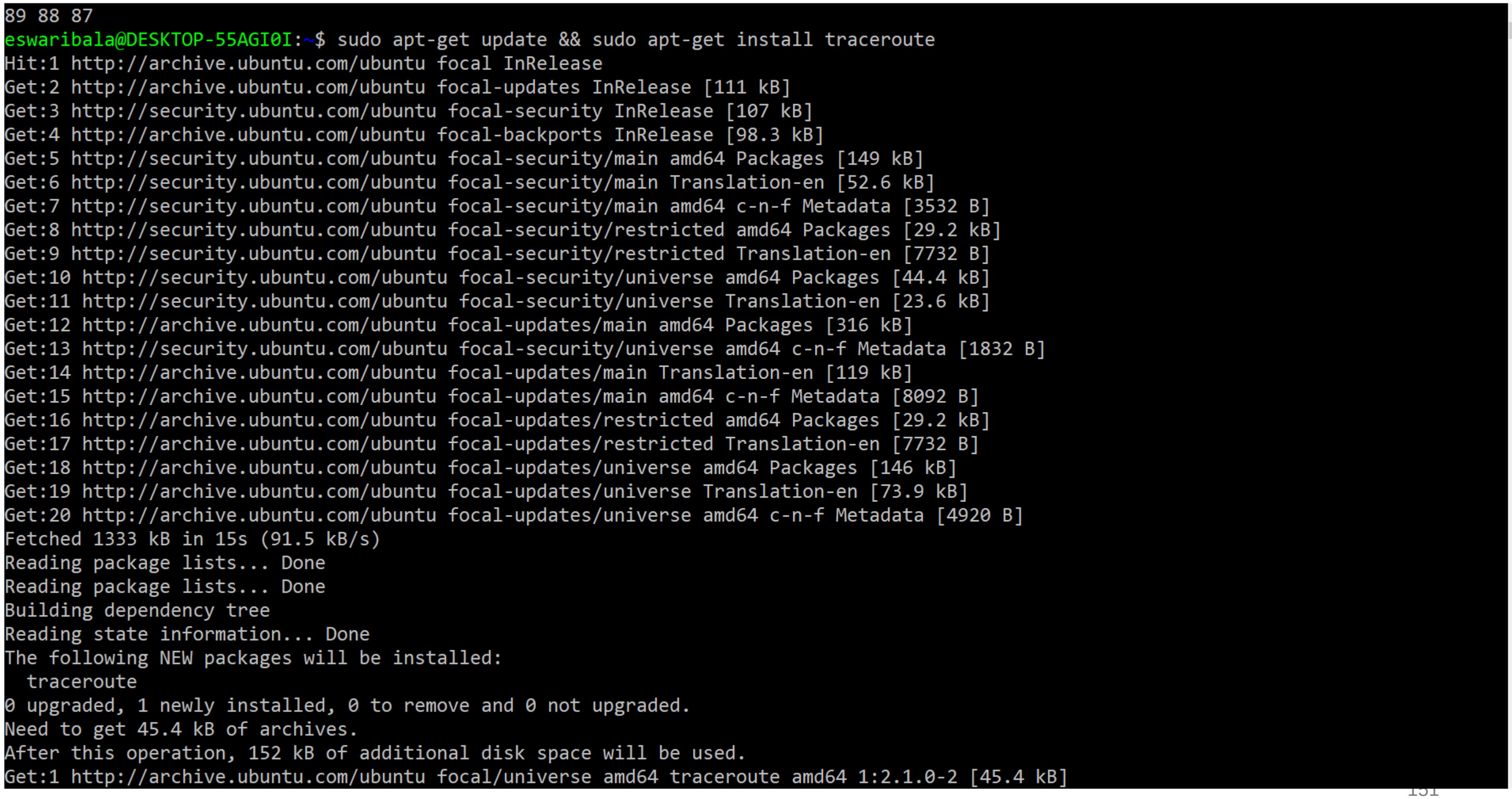


eswaribala@DESKTOP-55AGI0I: ~

```
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:~$ java -version
openjdk version "1.8.0_252"
OpenJDK Runtime Environment (build 1.8.0_252-8u252-b09-1ubuntu1-b09)
OpenJDK 64-Bit Server VM (build 25.252-b09, mixed mode)
eswaribala@DESKTOP-55AGI0I:~$
```

Traceroute Install

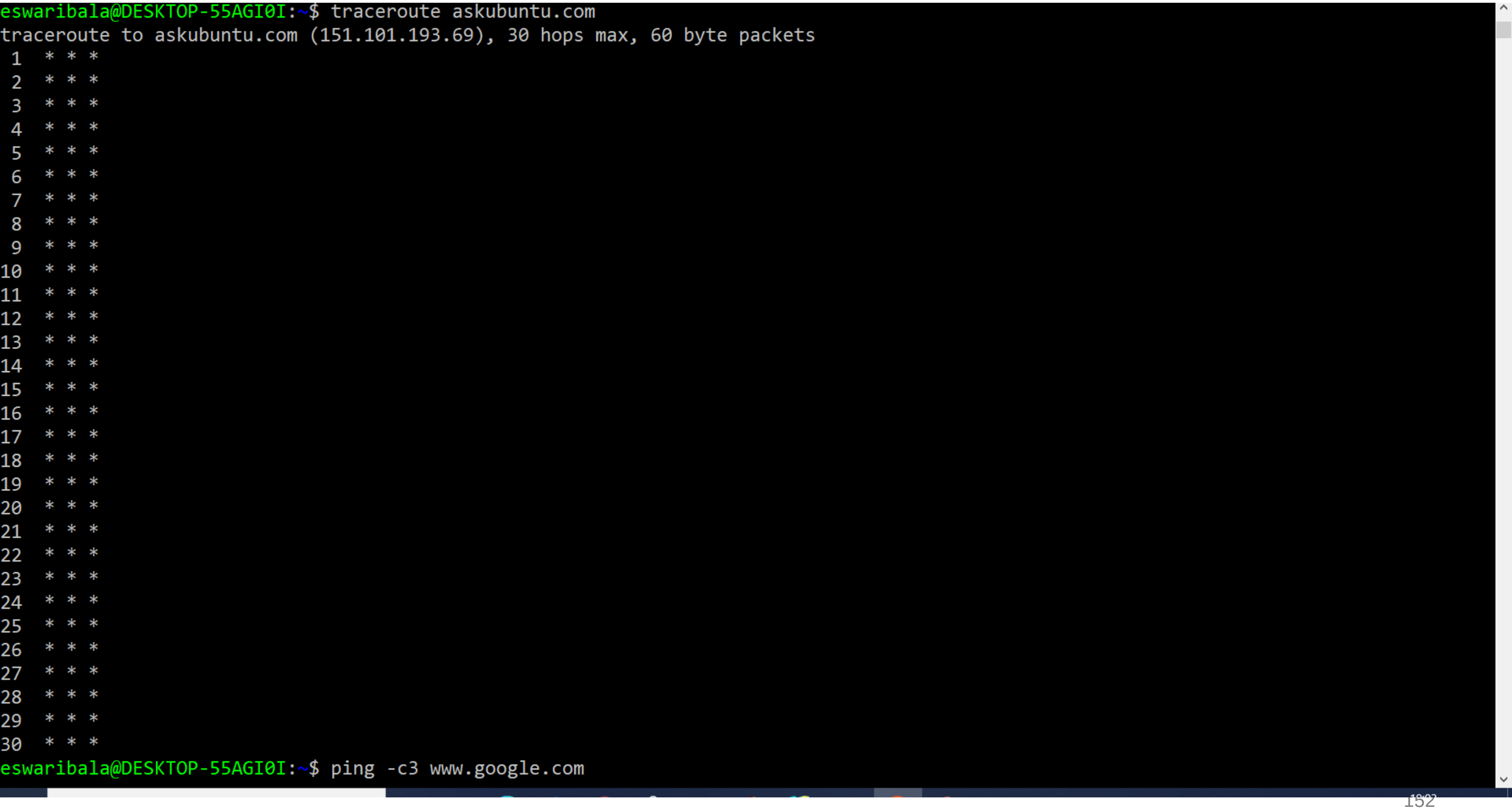
sudo apt-get update && sudo apt-get install traceroute



```
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

traceroute askubuntu.com

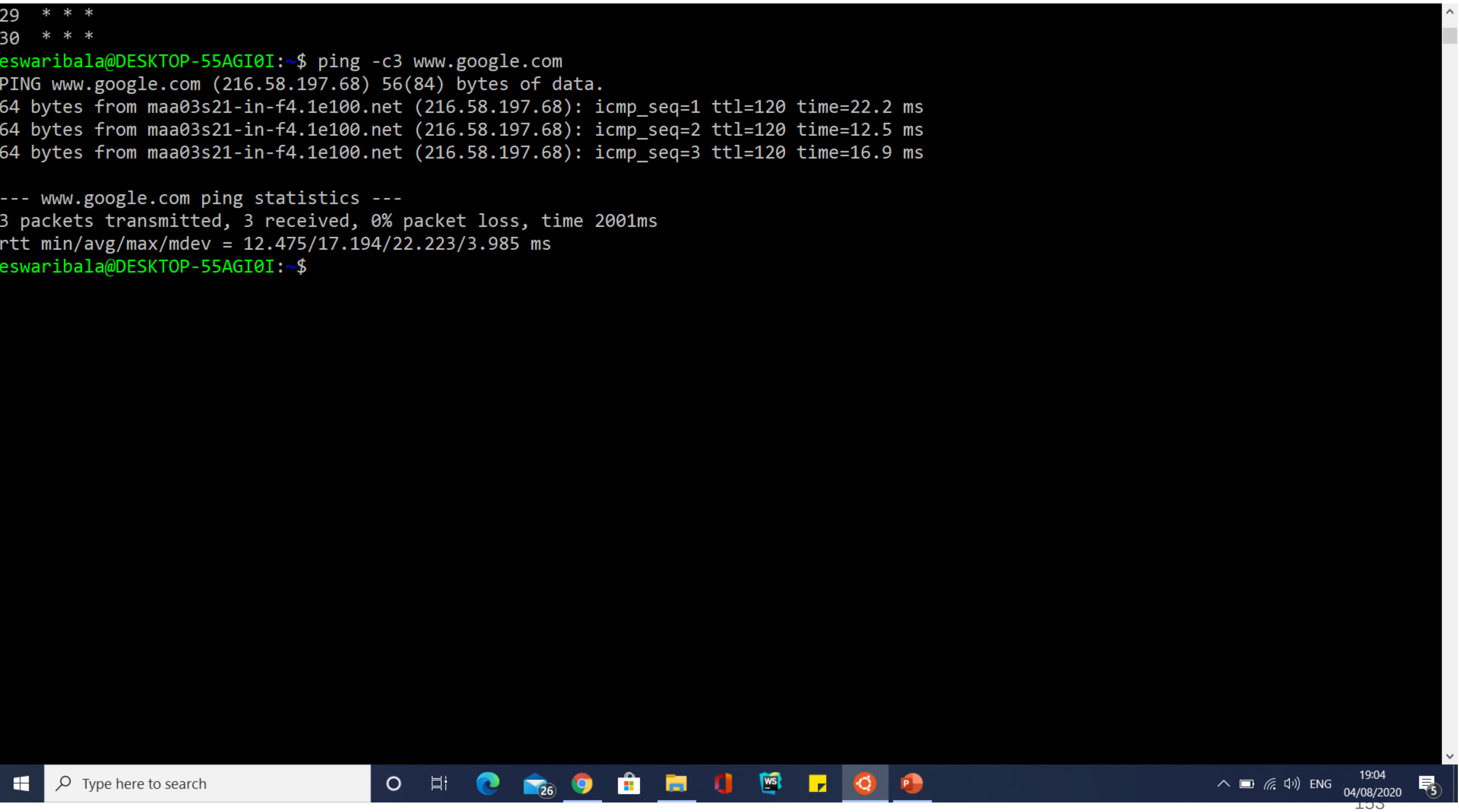


```
eswaribala@DESKTOP-55AGI0I:~$ traceroute askubuntu.com
traceroute to askubuntu.com (151.101.193.69), 30 hops max, 60 byte packets
 1  * * *
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  * * *
 9  * * *
10  * * *
11  * * *
12  * * *
13  * * *
14  * * *
15  * * *
16  * * *
17  * * *
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:~$
```

Questions

Module Summary

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