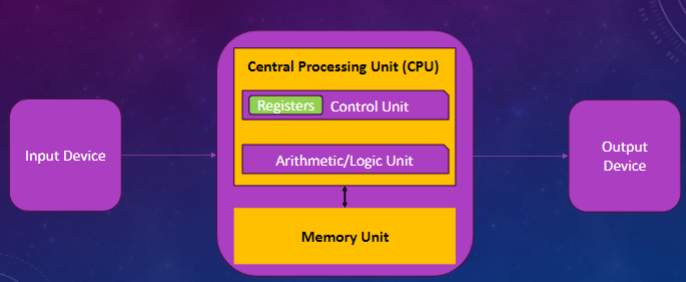
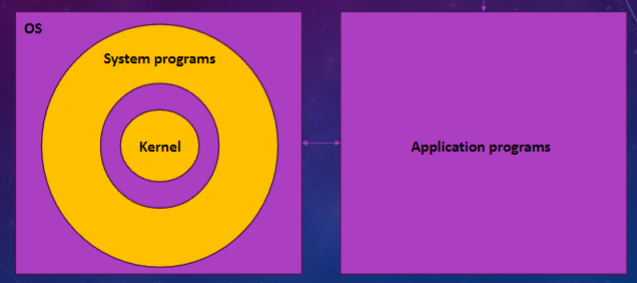
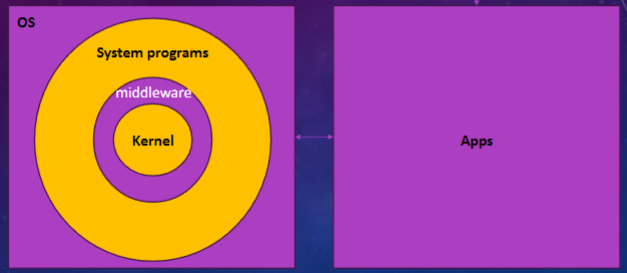
It’s Fine. It’ll Be Fine.

Lecture 02

Terms:

* Operating System (OS): A software system that manages a computer’s hardware resources and allocates those resources to programs
* Memory Controller: Orderly access to shared memory
* Bootstrap: A small program stored within the computer hardware in firmware that initializes all aspects of the computer system when powered on
* System Daemons: Services loaded into the memory at boot time that run in the background the entire time
* Multiprogramming: The ability of an OS to run multiple programs, which increases CPU utilization
* Multi-tasking: A logical extension of multiprogramming. The CPU switches among tasks frequently.
* Virtualization: Running multiple OS instances on one machine - Hardware support
* Containerization: Running a single OS instance on one machine with multiple user spaces to isolate processes - software support

Notes:

* CPU Instruction Cycle: Fetch, Decode, Execute
* The Central Processing Unit (CPU) contains a control unit with registers and an arithmetic/logic unit  
  
* Device driver for each device controller → Offers a uniform interface
  + CPU and device controller run in parallel and compete for memory cycles
* OS Components  
  
* Mobile/IoT OS Components  
  
* Most Internet of Things (IOT) have no OS
* Bootstrap loads OS kernel into memory
* The Interrupts Mechanism: The device controller ***raises*** an interrupt by asserting a signal on the interrupt request line, the CPU ***catches*** the interrupt and ***dispatches*** it to the interrupt handler, and the handler ***clears*** the interrupt by servicing the device
* A process is the unit of work in a system
* OS keeps several processes in memory simultaneously and switches between them when it needs to wait
* Multimode Operation: Separating the execution of the OS code and the user-defined code by hardware support
  + Mode Bit: 0 (Kernel/Superuser), 1 (User)
  + Advantage: A malicious program cannot hurt other programs or the OS
* The OS checkers for illegal instruction or illegal memory access → Terminates the program abnormally w/ an error message
* The OS Timer: Starts before giving control to the user mode, and once it reaches zero control is transferred to the OS
  + In Linux, ***HZ*** specifies frequency of timer and variable ***jiffies*** is the number of timer interrupts

Lecture 03

Terms:

* GNU: Recursive acronym for GNU’s Not Unix created in 1984
* GNOME: GNU Network Object Model

Notes:

* UNIX originated at AT&T Bell Labs in early 1970s
  + Since it was portable on various platforms, many UNIX-like OSs emerged: BSD from UC Berkeley, Solaris from Sun Microsystems, Mac OS X from Apple
* The Portable Operating System Interface (POSIX) International Standard (?)
* **The UNIX Philosophy**:  
  - Write programs that do one thing well  
  - Write programs to work together. Expect the output of one program to be the input  
   of another  
  - Design and build software to be tried early, ideally within weeks  
  - Write programs to handle text streams since that is a universal interface
* In 1984, Richard Stallman started developing a free, open-source UNIX-like OS called GNU which pioneered the free (freedom of use) software movement
* All *free* software are open-source, but not all open-source software are *free*
* **The Four Essential Freedoms**:  
  - The freedom to run the program as you wish, for any purpose (freedom 0)  
  - The freedom to study how the program works, and change it so it does your computing as you wish (freedom 1)  
  - The freedom to redistribute copies so you can help others (freedom 2)  
  - The freedom to distribute copies of your modified versions to others (freedom 3)
* Freeware is about the price being zero but free software is about the freedoms
* Permissive vs. Non-Permissive Software
* Not all open-source software is “free”
* “In 1991, a student in Finland, Linus Torvalds, released an open-source kernel using GNU compilers and tools and invited contributions worldwide.”
* Linux Stats:
  + 100% of top 500 supercomputers use Linux
  + 86% global market share for Android smartphones
  + 90% of public cloud workload and 96% of top 1 million web servers use Linux
  + Today’s Linux kernels have 10-30M LOC
  + Almost all military and space programs worldwide use Linux

Commands:

* echo $SHELL - Find what shell is in use
* echo $XDG\_CURRENT\_DESKTOP - Find which desktop environment is in use
* /home/bob - Filesystem structure

Lecture 04

Terms:

* Checksum: Represents the number of bits in a transmission message

Notes:

* Switching User Example:  
  *su alice*
* No need to enter the password for sudo if user is listen in /etc/sudoers
* To add a user to the sudoers list: *usermod -aG sudo username*
* Installing Software Types:  
  Main - Canonical-supported free and open-source software  
  Universe - Community-maintained free and open-source software  
  Restricted - Proprietary drivers for devices  
  Multiverse - Software restricted by copyright or legal issues
* Man Example:  
  *man su*
* Upgrade Commands:  
  *sudo apt update  
  sudo apt list --upgradable  
  sudo apt upgrade*
* Options for diff:  
  *--ignore-case  
  --ignore-blank-lines  
  –B, ignore-space-change  
  –b, --ignore-all-space  
  –w*
* Changing Access Modes on Files  
  User (u), Group (g), Others (o), Read (r), Write (w), Execute (x)  
  -rwxrwxrwx
* Chmod Examples  
  *chmod u-x -> -rw-rwxrwx  
  chmod o-r -> -rwxrwx-wx*
* Alternative to r, w, and x:  
  0 = No Permissions  
  1 = Execute  
  2 = Write  
  4 = Read
* Extensions for ls  
  *ls -l  
  ls -a*
* Examples for csplit (from line 2)  
  *csplit f1.txt 2*
* Concatenate Examples  
  *cat f1  
  cat f1 >> f2  
  cat f1 > f2*
* Banner Examples  
  *banner CS2080*

Commands:

1. su - switch to a different user
2. sudo - switch to the superuser
3. whoami - print name of current user
4. sudo apt install - install a new software package
5. apt search - search for a software package
6. **man** - get information on a command
7. info - the GNU info system for online documentation
8. help - information on built-in commands
9. **ssh** - connect to another system securely via SSH
10. ftp - interactive file transfer program
11. login - sign on to Linux (open a new session)
12. slogin - sign on to remote Linux using secure shell (i.e. SSH)
13. mailx - read or send emails
14. cmp - compare two files byte by byte
15. comm - compare items in two sorted files
16. **diff** - compare two files line by line; or compare two directories
17. sdiff - compare two files side by side
18. diff3 - compare three files
19. **cd** - change directory
20. chgrp - change file group
21. chmod - change access modes on files
22. chown - change file owner
23. cksum - print a file checksum
24. **cp** - copy files (cp –r for copying directories)
25. file - determine a file's type
26. head - show the first few lines of a file
27. tail - show the last few lines of a file
28. more - display files by screenful
29. less - similar to more, but more sophisticated
30. **ls** - list files or directories
31. ln - create file name aliases
32. locate - locate or placate is a fast way to find a file based on its name
33. md5sum - print a file’s checksum using the Message Digest 5 algorithm
34. mkdir - create a directory
35. mv - move or rename files or directories
36. **rm** - remove files (--r or rmdir for removing directories)
37. pwd - print the working directory’s path
38. scp - copy files to a remote system securely using SSH
39. csplit - break files at specific locations
40. split - split files evenly (default size: 1000 lines)
41. wc - count a file’s lines (-l), words (-w), and characters (-m)
42. **cat** - concatenate files or print the content (>> to append, > to substitute)
43. **nano** - simple text editor
44. banner - print a banner of a text

Lecture 05

Terms:

* grep: Short for “global regular expression print”

Notes:

* The nicer the program, the least priority that it gets. The max nicenest is 19 and the least is -20  
  *-n* add an integer to the default niceness of 10
* More on script:  
  If timing data are saved (--log-timing), *scriptreplay* can replay the terminal session
* Spell checking a file:  
  Use the -c option in an interactive mode that allows corrections
* Shutdown Options:  
  -h is equivalent to –poweroff or -P, unless –halt is specified  
  Halt terminates all processes and shuts down the CPU  
  Poweroff turns the entire computer off
* Shutdown Example:  
  *shutdown -h now*
* Options for bzip2 and gzip (installed on most Linux distributions)  
  -k to keep the original file / -d to decompress
* Options for bunzip2:  
  -z option to compress
* Compression is always performed even if the resulting file is larger
* May also use *zip* and *unzip* - for source code or English text, typical 60-70% compression
* Options for tar:  
  -c to create a new archive  
  -v to make it verbose  
  -f to give a file name for the archive  
  *tar -c-v-f f1.txt.tar f1.txt*
* Show the Free Disk Space:  
  *df -h*
* Show Free Memory Options:  
  -m or -mebi to show the amount in mebibyte
* Decimal Prefix: 1 Megabyte (MB) = 1,000,000 bytes  
  Binary Prefix: 1 Mebibyte (MiB) = 1,048,576 bytes
* Note: du -h indicates the disk space usage for a file or directory; ls -l shows the size of a file or directory
* Showing Environment Variable Options:  
  To show a specific environments variable’s value, use echo and $  
  *echo $PATH*  
  To create a new environment variable or set a new value for an existing one, assign a value without using the $ sign  
  *Test=10*
* Killing a Program Example  
  *kill -9 [PID]*
* Information on grep:  
  Use -i to ignore case  
  *grep ‘uccs\|university’ f1.txt  
  grep -e ‘[a-d|A-D]’ f1.txt  
  grep -f pattern.txt f1.txt*
* Find Information  
  find [where to start][what to find]  
  *find . f1.txt  
  find / “\*proc”*
* Piping: indicated by vertical bar | ; read a “pipe”; connects standard input of one process with standard output of another process  
  *command\_1 | command\_2 | command\_3  
  ls | grep f1.txt*  
  *ps -ef | grep java  
  ps -ef | grep python*
  + Pipes are unidirectional: Flow from left to right

Commands:

1. bc - arbitrary precision calculator
2. cal - display calendar
3. calendar - reminder service
4. clear - clear the screen
5. nice - run a command with lower priority, influencing the process schedule
6. nohup - preserve a running job after logging out or receiving a hand-up signal
7. passwd - set your login password
8. script - produce a transcript of your login session
9. spell - report misspelled words
10. aspell - spell check a file
11. w - show who is logged on and what they are doing
12. uptime - tell how long the system has been running
13. reboot - reboot the machine
14. shutdown - power off the machine
15. bzip2 - very high-quality file compression program
16. bunzip2 - decompress (expand) a file previously compressed with bzip2
17. gzip - compress files
18. gunzip - decompress (expand) a .gz archive
19. tar - compress files
20. date - display date and time (UTC)
21. df - show the free disk space
22. free - show the amount of free and used main memory
23. du - estimate file space usage (-h for human readable output)
24. env - show the environment variables
25. finger - display information about users
26. who - who is currently logged in
27. ps -ef - see all system processes using the standard syntax
28. kill - terminate a process
29. grep - searches for one or more patterns in each file; patterns separated by \|
30. egrep - deprecated; equivalent to grep -e; supports regular expressions
31. fgrep - deprecated; equivalent to grep -f; file should be provided with the pattern
32. find - search the filesystem for filenames matching patterns or attributes
33. string - display text strings found in binary files

Lecture 06

Terms:

* sed: Stands for “Stream Editor”; filtering and transforming text
* awk: A powerful pattern matching and text processing language
* Record Separator (RS): Delimiter; by default is the newline character (\n)
* Record Number (NR): Current record number; if RS is default, NR is the line number
* Field Separator (FS): Used for separating the input records into fields
* Output Field Separator (OFS): Joins the fields for the output
* Number of Fields (NF): Number of fields in the current record; if FS is the default, then NF is the number of words
* vim: Modified and advanced version of the vi editor
* LaTeX: A typesetting and document preparation software

Notes:

* Interactive Text Editors - Nano, vi/vim, Emacs  
  Non-Interactive Text Editors - For writing “one-liners” e.g. sed and awk
* Stream editor sed is non-interactive; input can be file or pipeline; works line-by-line in one pass
  + sed [options] commands [input test/file]  
    [command/pipeline] | sed [options]
* Deletions to text files will not save unless you save the modified version
* Gawk is GNU’s version of awk
  + awk ‘pattern { action }’ - Will be executed if the pattern statement is equivalent to a non-zero value (true)
  + default behavior is printing
* The default for FS or OFS is one or more whitespace characters
* Predefined Variables in Awk: RS, NR, FS/OFS, NF
* The vim has two modes: insert (you can write text) and normal (efficient ways to navigate and manipulate text)
  + i for insert mode / esc for normal mode
  + in insert mode, press : (colon) and enter command
  + Set number and push enter to see the line numbers
* G Jump Example  
  *168000000 G*
* LaTeX is based on TeX; greater for scientific papers and technical reports
  + Use the .tex extension for the source file(s)
  + Use the .bib extension for bibliography files
  + latex text.tex → DVI file pdflatex text.tex → PDF file
* LaTeX Hello World:  
  \documentclass{article}  
  \begin{document}  
  Hello, World!  
  \end{document}

Command and Command Examples:

* sed ‘ ‘ file.txt- Read file
* *sed ‘p’ file.txt* - Prints each line twice
* *sed -n ‘p’ file.txt* - Suppresses default behavior
* *sed -n ‘ 2p’ file.txt* - Prints the second line
* *sed -n ‘1,3p’ file.txt* - Prints the first three lines
* *sed* *-n ‘$p’ file.txt* - Prints the last line
* *sed -n ‘3,+3p’ file.txt* - Set an offset
* *sed -n ‘$!p’ file.txt* - Prints every line EXCEPT the last line
* *sed ‘1~2d’ file.txt* - Delete every other line (notice it doesn’t need -n)
* *sed ‘1~2d’ file.txt > modified.txt* - Save a modified version of a file
* *wc -l file.txt* - Check the number of lines
* *awk ‘1 {print}’ file.txt* or *awk ‘2 {print}’ file.txt* - Print all lines as long as pattern isn’t 0
* *awk ‘NR>1 {print}’ file.txt*- Remove the header of a file
* *awk ‘NR>2 && NR<7 {print}’ file.txt* - Prints a range of line (Lines 3-6)
* *awk ‘NF’ file.txt* - Remove whitespace-only lines
* *awk ‘{ print $1, $3}’ FS=, OFS=, file.txt* - Extract fields/columns (1st and 3rd column)
* *sudo apt install texlive* - Install the tex-live package

1. w!- save changes in vim
2. *wq!* - save changes and quit in vim
3. *q!* - close without saving changes
4. G - jump to the last line (normal mode)
5. lineNumber G - jump to a specific line (normal mode)
6. dd - delete an entire line (normal mode)
7. yy - copy an entire line to clipboard (normal mode)
8. p - paste a line from the clipboard to a certain location (normal mode)
9. cc - cut a line (i.e., remove and copy to clipboard) (normal mode)
10. /text - to search forward for a text (normal mode)
11. ?text - to search backwards for a text (normal mode)
12. n - find the next occurrence
13. N - find a previous occurrence

Lecture 07

Terms:

* GCC: Compiles the C code into assembly and then generates the machine code
* G++: Compiled the C++ code into assembly and then generates the machine code
* Java Development Kit (JDK): Created by Oracle, formerly Sun Microsystems, for development
* Java Runtime Environment (JRE): An environment to run java programs
* Java Virtual Machine (JVM): It has the java interpreter (line-by-line interpretation of Java Bytecode)
* Just-In-Time (JIT): Converts java bytecode into machine code

Notes:

* Makefiles - The make command; building and maintaining large projects from source code
  + Syntax:  
    targets [file names]: prerequisites [files that need to exist before execution]  
     command-1  
     command-2  
     command-3
* GCC and G++: Compiler → Assembler → Linker → Loader
  + Generates an ELF (Executable and Linkable Format) file (binary)
  + To run the binary file: *./hello.o*
  + Compile with GCC: *gcc -o hello.o hello.c*Compile with G++: *g++ -o hello.o hello.cpp*
* Java Advantage: Platform Independence
* *javac HelloWorld.java* generates the Java Bytecode (class) *HelloWorld.class*
* Java Execution → *java -cp . HelloWorld* → The -cp option sets the CLASSPATH
  + -cp overrides CLASSPATH
  + If you want to add something, remember to keep previous content:  
    *PATH=$PATH…*
* Create a hello world program in various languages: *vim HelloWorld.*(appropriate extension)  
  [Program Code for Hello World]
* Python is both compiled and interpreted (line-by-line) when you run code
  + Pros: Avoids complexity and wide variety of libraries
  + Cons: Too slow
* Anaconda: Freeware but not “Free Software”
* Set up Anaconda Environment:  
  *eval “$(/home/ubuntu/anaconda3/bin/conda shell.bash hook)”*
* Jupyter Notebook Set Up: *jupyter notebook –no-browser --ip=0.0.0.0 --port=8888 &*
* Git is a version of Source Control Management
  + *man git*; *git help*; *man git-log*; *git help log*; *git-clone*; *git pull*; *git add*; *git commit -m “log”*; *git push*; *git fetch*; *git merge*; *git init*

Commands:

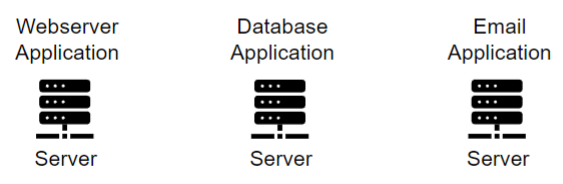
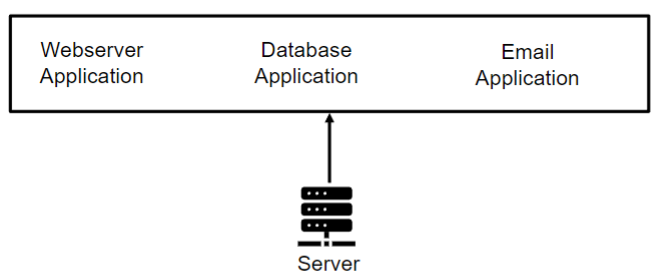
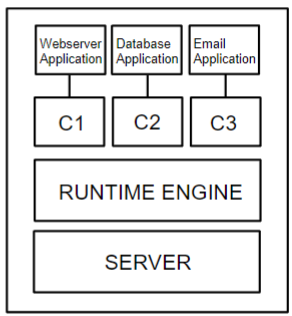
* Makefile Example:  
  *hello:  
   echo “Hello World”  
  Run: make hello*
* Create a new python script: *vim HelloWorld.py*  
  *print(“Hello World”)*  
  Run:  
   *python2 HelloWorld.py  
   python3 HelloWorld.py*

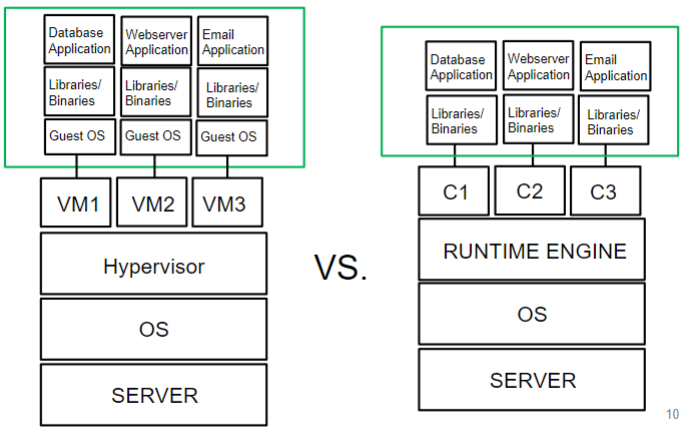
Lecture 08

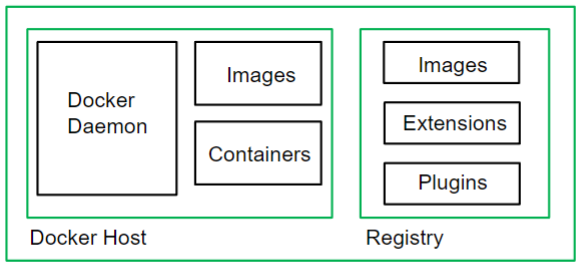
Terms:

* Virtualization: Hardware-level virtualization; possible to run multiple operating systems (OSs) on a single machine; runs of top of physical server
* Containerization: Software-level virtualization; packaged with all the files, dependencies, and configurations; runs on top of OS
* Command Line Interface (CLI): Commands, e.g., docker run, docker build, docker pull
* Docker-Compose: Define and share multi-container application

Notes:

* One Application Per Server: Inefficient resource utilization; High cost for hardware and maintenance  
  
* Multiple Applications Per Server: Coexistence problem for different applications; security issues  
  
* When using Virtual Machines (VMs), you allocate physical resources (RAM, CPU, storage)
* Types of Hypervisor  
  Type-1: Interacts directly with machine hardware; used in data centers, web servers, etc.  
  Type-2: Interacts with the hardware through the OS; hosted hypervisor; used in desktop and development environments
* Benefits of Virtualization:
  + Increase hardware utilization
  + Decrease the number of physical servers
  + Cost savings
  + Easy and quick set up of new environment
  + Less recovery time
* Containerization removes the necessity of an OS and includes a run-time engine that created isolated environments for applications and allocates resources
* Benefits of Containerization: Lightweight; consume less storage, RAM, and CPU power, portable, fast development of apps, more hardware utilization and cost saving; less boot up time



* Docker’s Popularity: Open source, can run on a single node, supports multiple environments, user-friendly interface, easy and fast setup and deployment
* Docker Key Components:
  + Docker Engine: Client-server app; contains client, server, and communication channel (Rest API, network interface, unix socket); client and server can run on same machine or different machines
  + Docker Client: Used to run commands to communicate with the server; CLI or docker-compose; can communicate with multiple docker daemons
  + Docker Server  
    
    - Docker Daemon: Manages docker images, containers, and other components; listens to the docker client requests; start up during booting; run in background; parent process of all container processes
    - Registry: Stores the image files of docker; kept in server/cloud; public registry; docker hub is default; also a private registry
      * Docker pull command – pull the images from registry
      * Docker push command - push the images to registry
    - Image: Instructions for creating containers; read-only template; contains the packages, dependencies, snapshots, and configurations to run apps; can be customized from other’s images
    - Container: Executable software package; running instance of an image; isolated from other containers and host machine
* Images can be built or created using docker file
  + Instructions in a text file for creating images
  + Running docker build command on a dockerfile
  + Dockerfile
* Containers have their own file system (“/” – starting point)
  + Read-Only directories/file (from image)
  + Read-Write directories/file (from application)
* Docker Container Lifecycle
  + Phase 1: Creation - Read/Write layer is added to the Read-Only layer of the image; prepares to run programs
  + Phase 2: Start - Sets up the resources (e.g., memory, CPU, etc.); ready to perform the tasks assigned to it
  + Phase 1 and 2: Run - Create a container, ready to perform the tasks
  + Phase 3: Pause/Unpause/Stop/Kill
    - Pause - Container is running but the processes are paused (troubleshooting or freeing resources) - SIGSTOP signal
    - Stop - Shut down the processes inside the container - SIGTERM (graceful shutdown) - SIGKILL (forceful shutdown)
    - Unpause - SIGCONT signal
* Docker Swarm - Multiple docker hosts (runs one or more containers); form cluster
  + Manager Node: Manages membership; delegates service
  + Worker Node: Runs service (tasks to execute); tasks (running commands inside container)

Commands:

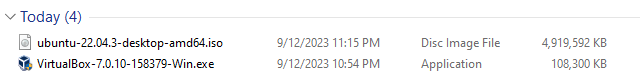
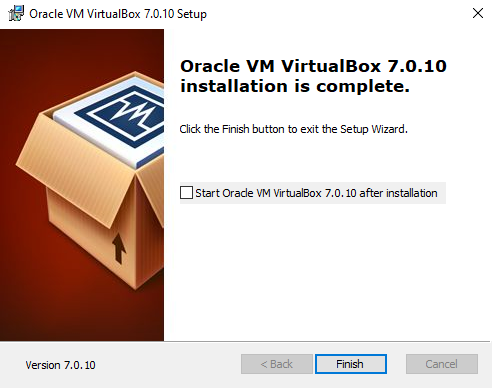
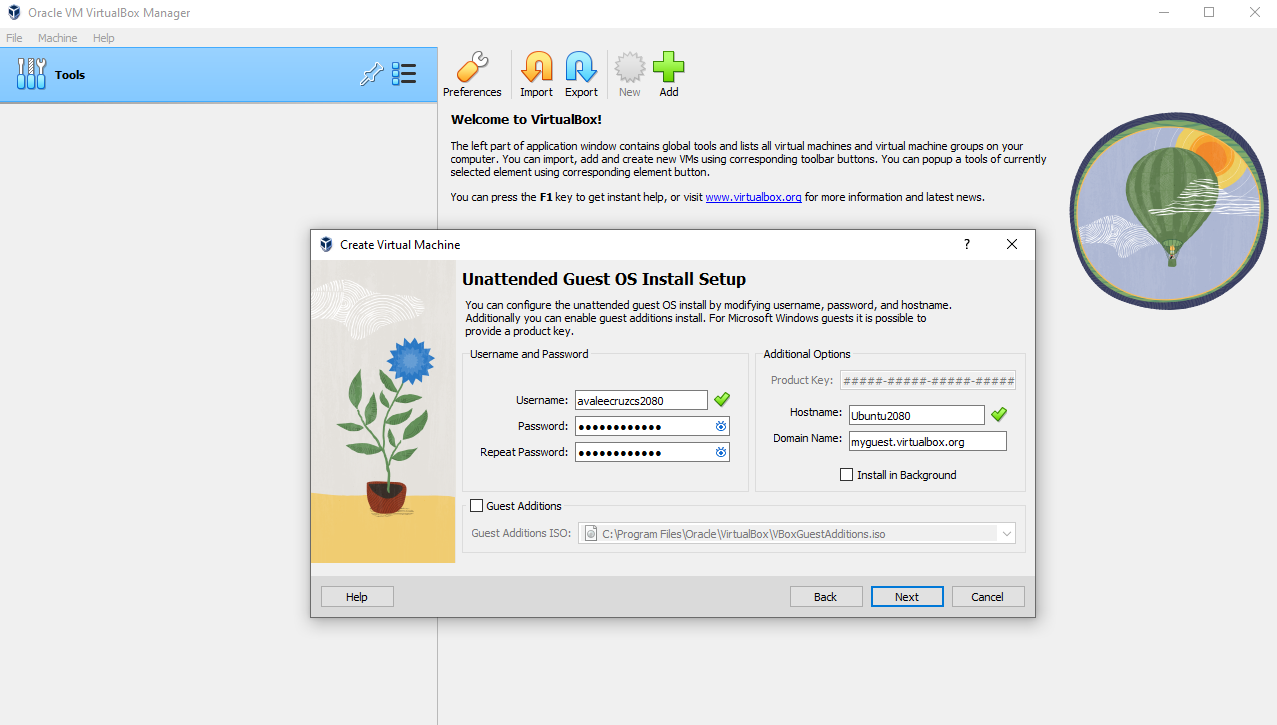
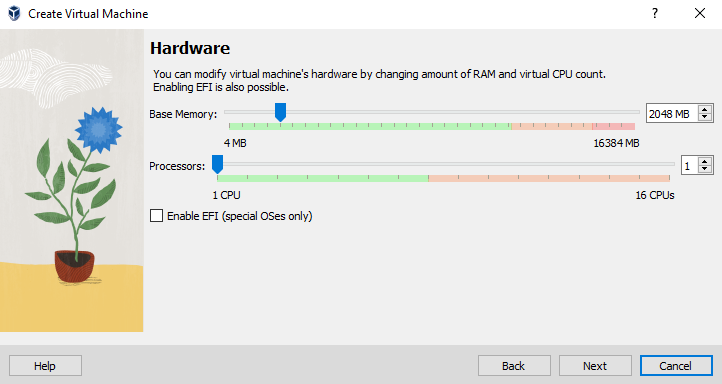
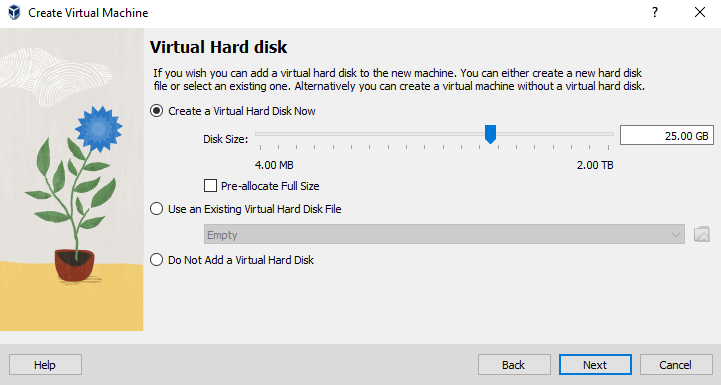
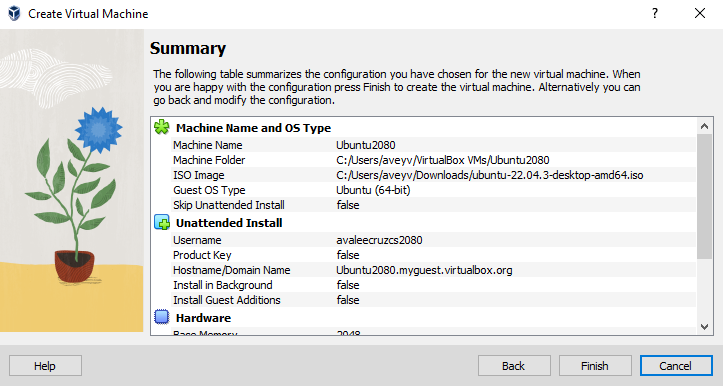
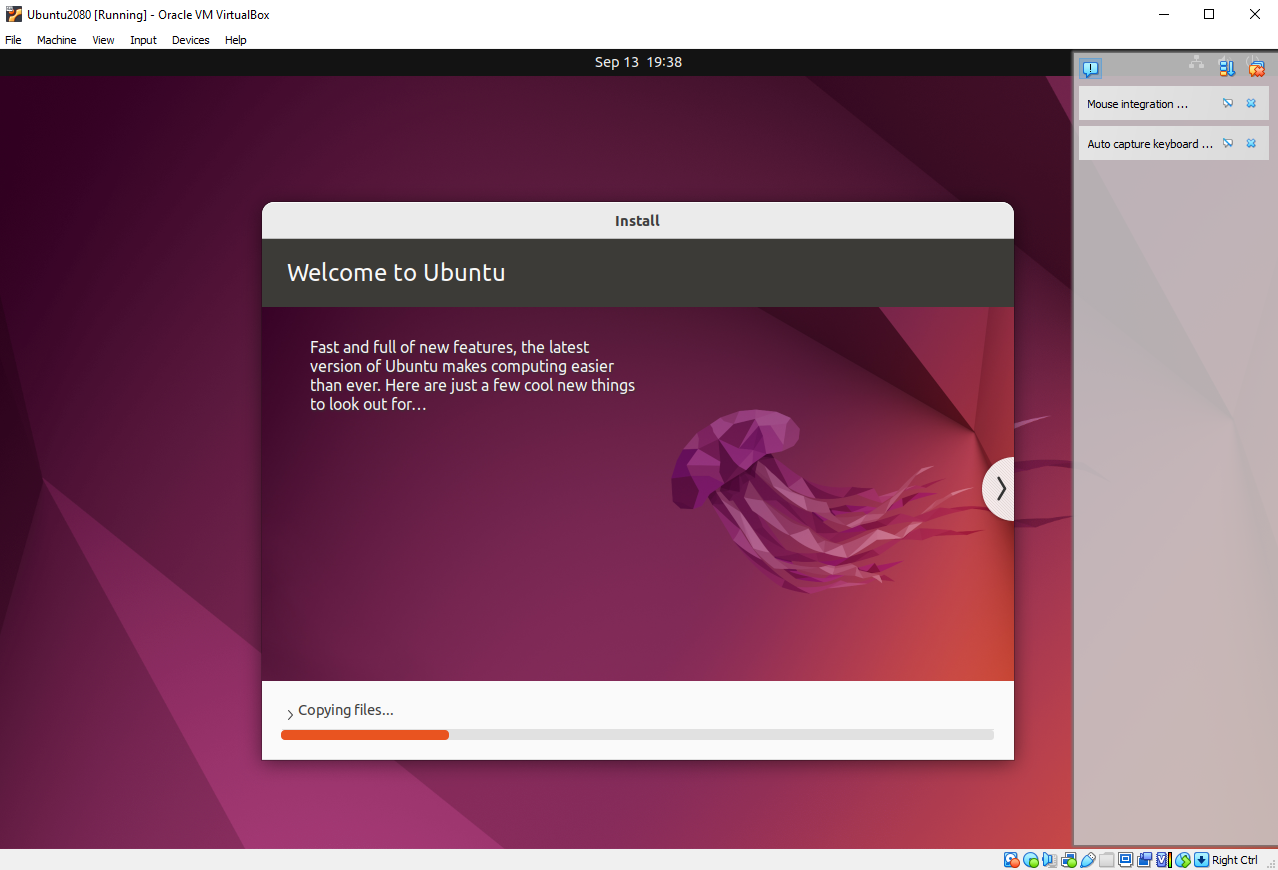
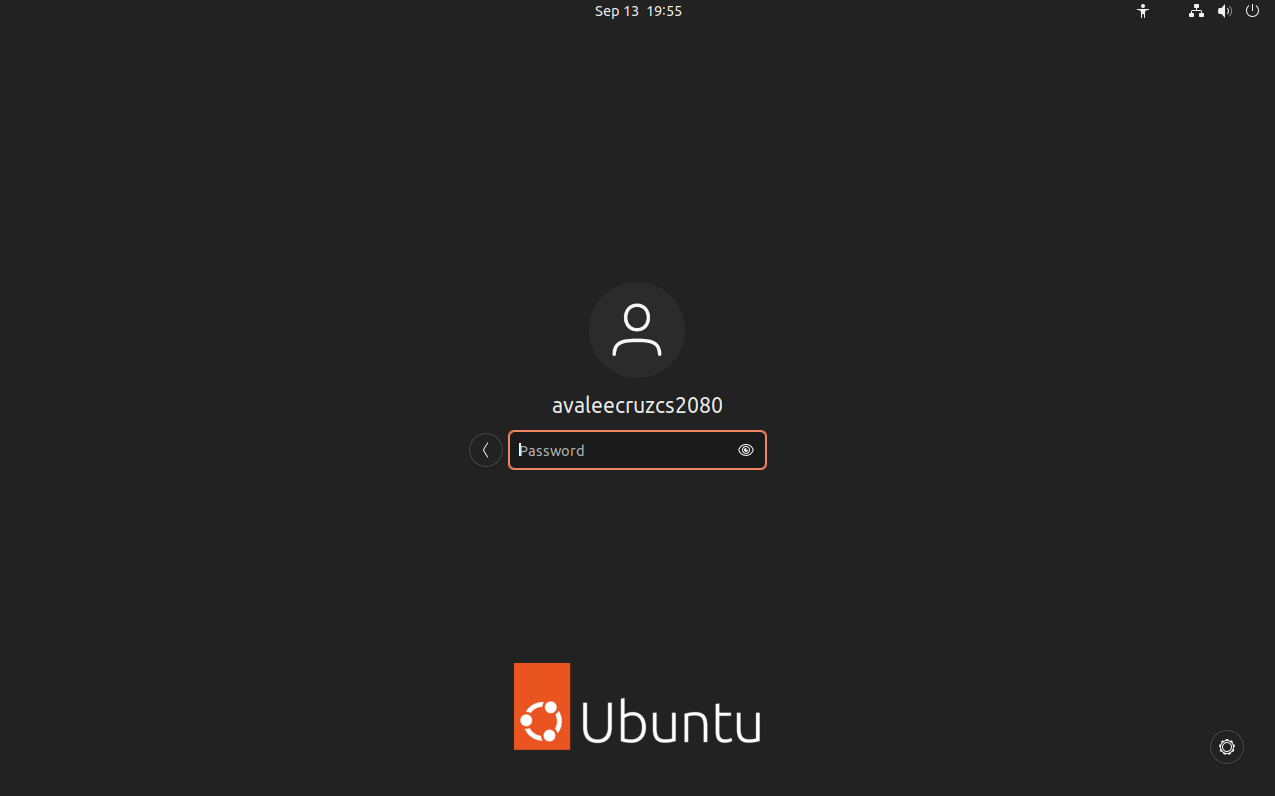
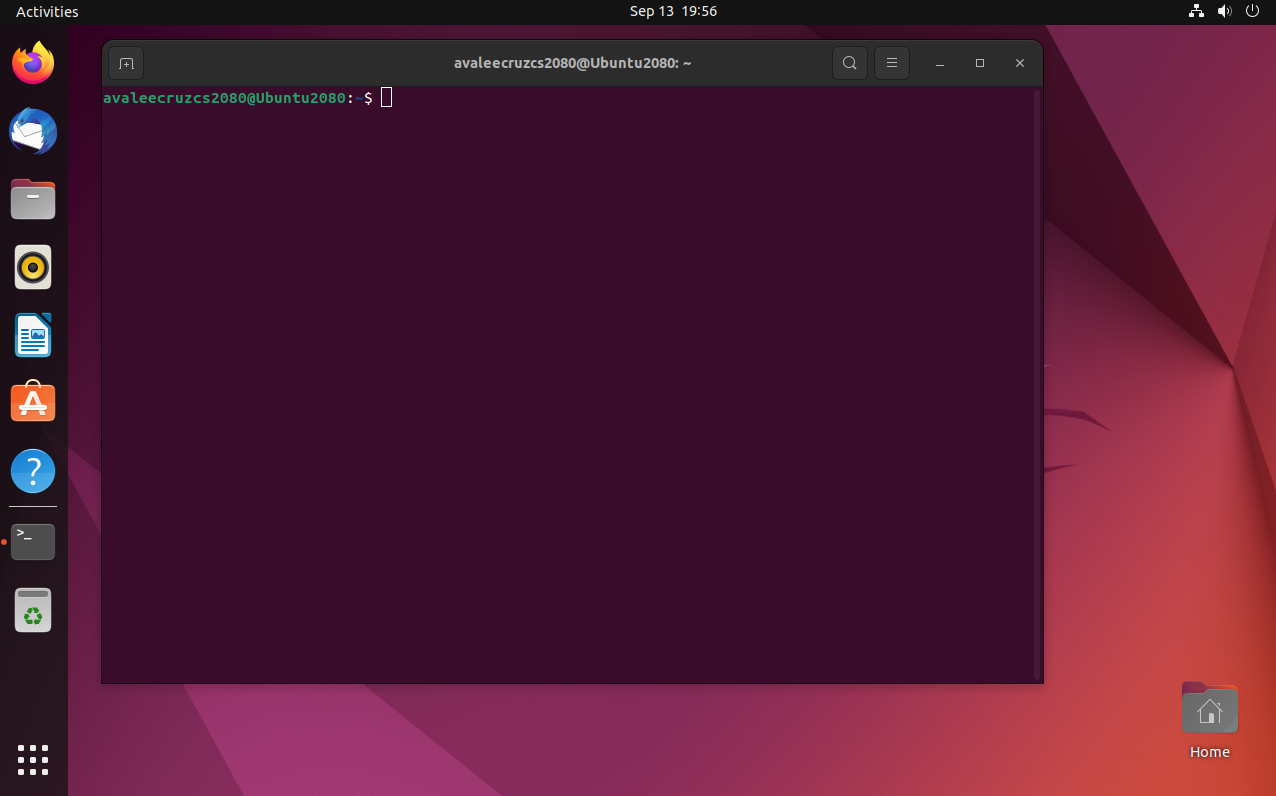
* *docker pull <registry\_url>/<image\_name.:tag* - Download from registry
* *docker build --t image\_name .* - Build the image
* *docker create --name <container\_name> <image\_name>* - Phase 1: Creation
* *docker start <container\_name or container\_id>* - Phase 2: Start
* *docker run --name <container\_name> <image\_name>* - Phase 1 and 2
* *docker pause container <container\_name or container\_id>* - Pause
* *docker unpause <container\_name or container\_id>* - Unpause
* *docker stop <container\_name or container\_id>* - Stop
* *docker kill <container\_name or container\_id>*  - Kill
* *docker rm <container\_name or container\_id>* - Removes the specific container
* *docker --version* or *docker --v*- Check the version
* *docker-compose --v* - Check the version of docker-compose
* *sudo systemctl status docker* - Check the status of docker
* *sudo systemctl docker enable --now* - Docker to enable docker
* *sudo systemctl docker disable --now* - Docker to disable docker
* *docker ps* - Check all the running containers
* *docker ps -a* - Check all container including the stopped ones
* *docker images* - Check all docker images
* *docker info* - See detailed info about Docker config, version indo, images, containers, etc.
* *docker pull image\_name:version\_number* - Download docker images of a specific version
* *docker swarm init --advertise-addr ip\_address* - Initialize docker swarm
* *docker swarm join-token manager* - Join a manager to the swarm
* *docker node ls* - Check the status of the nodes in docker swarm cluster
* *docker network ls* - Check the swarm network
* *docker inspect node\_number* - To check all info on a particular node (in manager node)

Assignment 1

1. What is the CPU instruction cycle? How does the CPU handle each of the stages, and what registers are used during each stage? 10 points  
   The CPU instruction cycle consists of the stages fetch, decode, and execute that the central processing unit iterates in order to process instructions. The CPU handles the fetch stage by getting the memory address from the program counter and copying it into the memory address register to be loaded in the memory data register. From here, the loaded instruction is placed in the instruction register to be decoded. Once it’s decoded, the control unit will execute the instruction and copy it to the Arithmetic/Logic Unit if necessary.
2. What are some key differences between the memory and the registers (name at least 3)? Name two registers and briefly explain each of them. 5 points  
   The registers hold the instructions that the CPU is currently processing. Memory holds the instructions for the entire program that needs to be retrieved by the CPU. Registers are also faster to access but can only hold a small amount of data. Meanwhile, memory is more difficult to access but can hold terabytes of data. Two registers are the program counter and the Memory Address Register (MAR). The program counter holds the memory address of the next instruction to be decoded and executed by the control unit. The MAR holds the memory address that is currently being fetched or the address that data will be sent to.
3. What component always runs in the Operating System (OS) after bootstrapping? What component provides hardware component access privileges to the system programs? What is the central role of middleware in mobile OSs? 10 points  
   The component always running in the OS after bootstrapping is the kernel. The component that provides hardware components with access privileges to the system programs is the device driver. The central role of middleware in mobile OSs is to provide additional services to application developers. This can include supporting databases, multimedia, and graphics.
4. Assume you are on your computer, watching a YouTube video. Suddenly, you want to watch a YouTube video on a different topic, so you type the name of that topic into the search field while the previous YouTube video is still playing. What is the mechanism through which the CPU in your computer is aware of a keystroke? Name and describe the mechanism. 10 points

The mechanism through which the CPU in your computer is aware of a keystroke is the interrupts mechanism. The CPU gets an interrupt signal by the signal bus from the device controller and puts aside its current execution instructions. It dispatches it to the interrupt handler, waits for the handler to service the device and clear the interrupt, then continues the execution instructions that were interrupted.

1. How can the CPU increase its utilization in terms of handling processes? Name and describe two mechanisms. 5 points  
     
   The CPU can increase its utilization in terms of handling processes through multiprogramming and multitasking. Multiprogramming is a technique that increases CPU utilization by always having a job the CPU needs to execute. This means the OS can run multiple programs, as the name implies. Multitasking is when the CPU switches between multiple jobs to provide fast response times. As the CPU is waiting for the I/O response from the user, it continues with another job which allows jobs to be run concurrently.
2. Suppose you are a user of an OS with no superuser privilege. However, you write a program to modify the kernel. Will you be able to run the program successfully? If not, why? What will happen if you try to run the program? 10 points  
     
   You will not be able to run the program successfully. When you try to run a privileged instruction to modify the kernel outside of kernel mode, the hardware does not execute the instruction and rather traps it to the operating system. If you try to run the program, it will give you an abnormal termination and an error-appropriate message.
3. Based on your understanding of the OS concepts, differentiate between a Virtual Machine (virtualization) and a container (containerization)? 10 points  
     
   Virtualization is the ability to designate hardware to run in different environments and give the illusion that multiple operating systems are running on one computer. This also includes emulation which is a method for simulating a different hardware that a target program runs on so it can be executed. Containerization is similar, but on a smaller level. It runs multiple user spaces to isolate processes on a single operating system. This means that only one application (instance) of an OS is run which increases accessibility.
4. Explain the purpose of the following directories in a Linux system: a) /bin, b) /boot, c) /dev, d) /etc, e) /home, f) /lib, g) /media, h) /mnt, i) /opt, j) /sbin, k) /srv, l) /tmp, m) /usr, n) /proc, o) /var, p) /root. 10 points  
   a) /bin - Hold binary or executable programs  
   b) /boot - Contains files used in booting the operating system  
   c) /dev - Location of the device files  
   d) /etc - Contains system configuration files  
   e) /home - Home directory (default)  
   f) /lib - Has kernel modules and a shared library  
   g) /media - Contains removal media devices  
   h) /mnt - Temporary mount directories for mounting the file system  
   i) /opt - Optional or third-party software  
   j) /sbin - Binary executable programs for admin  
   k) /srv - Holds server-specific and server-related file  
   l) /tmp - Temporary space  
   m) /usr - Holds user related programs  
   n) /proc - Info about running processes with a specific process ID  
   o) /var - Holds log files  
   p) /root - Stores all directories in Linux
5. Download Ubuntu 22.04 LTS and install it via VirtualBox. Provide a screenshot of each step and discuss what you have done in each step. Create a username following this rule: *firstnamelastnamecs2080*, where firstname is your first name and last name is your last name. After successfully installing Ubuntu, take a screenshot of the terminal. 15 points

  
  
  
Download and Install Ubuntu and VirtualBox  
  
Open the VirtualBox and Press New  
  
  
Enter a Username and Password with the Format *firstnamelastnamecs2080*  
  
  
Modify the VirtualBox’s Hardware  
  
  
Create a Virtual Hard Disk  
  
  
Finish Creating VirtualBox  
  
  
Wait for It to Load  
  
  
Sign Into Ubuntu  
  
  
Show Terminal

1. Who are your teammates for the final project? What is going to be the main topic, contributions, and deliverables for your final project? Are you going to extend any existing open-source projects? Are you going to release your code as open-source software? If yes, under what license and why? 15 points  
   My teammates for the final project are Ryan Montgomery and Matthew Fuller. Our main topic is going to be recreating Conway's Game of Life in a terminal. We will extend the open source project on GitHub by Jerrit Glasker given at this link <https://github.com/jerr-it/cellmaker>. Our contributions will consist of programming the game to work in a UNIX terminal, and given time, potentially programming other games as well to create an arcade style format. The deliverables will be visual evidence of the terminal Game of Life itself and the code we created based on the original open source project. Upon completion, we will release our code as open-source software under the MIT License since it was also the license used for the original open-source software.

Assignment 2

Part 1:

1. What is free (libre) open-source software? List the four basic freedoms it must protect.  
   Free (libre) open-source software is software protected with the freedoms of use. These freedoms include the freedom to use the program as you wish, the freedom to study how the program works and to change it as you wish, the freedom to redistribute copies, and the freedom to distribute copies of your modified version.
2. Is open-source software always free of charge? Explain it.  
   No, open-source software isn’t always free of charge. The idea of “free” in open-source software refers to the freedoms that protect the usage and modification of the program. This does not mean that the software is also free in the monetary sense.
3. What are some benefits of using open-source software over proprietary software? Name four.  
   One benefit of using open-source software (OSS) is that it is easier to develop since proprietary software has regulations on its use. OSS is also more “community-driven” according to Andrew Park who wrote the article “Open Source vs. Proprietary: Development, Licensing, Business Models, Security, and More.” This means that people can collaborate to improve the code with little worry about restrictions on sharing or modifying. OSS also tends to be lower in price or even free while proprietary software often requires a purchase or subscription. Finally, OSS is overall more flexible since it doesn’t require contracts and is built to be used by anyone, not just professionals.
4. Does free-of-charge software (freeware) mean it is “free” (libre) open-source software? Explain it.  
   No, just because it is free-of-charge does not mean that it is libre open source. As stated by Vikram Gupta, a writer for *Built In*, in his article “What is freeware?”, “Freeware is *cost-free* software but free software is *copyright-free* software.” This means that although the product is free, it is not protected by the freedoms of free software and therefore can be licensed or have restrictions on its use.
5. List the two categories of open-source licenses. What is their difference? Give two examples for each category and name one software product that is distributed under that license.  
   The two categories of open-source licenses are copyleft and permissive licenses. Copyleft licenses mean that others have the right to use, modify, and share the code as long as that same agreement is held with derivatives of the original software. Permissive licenses differ because they allow freedom to use, modify, and share the code without the requirement that others allow the same freedoms for what they create from the original software. Two examples of a copyleft license include the GNU General Public License and the Eclipse Public License. Two examples of a permissive license include the MIT license and the Berkeley Software Distribution license. A software product under the copyleft license is Linux while a software product under a permissive license is the game 2048.

Part 2:

1. What is Public-Key Cryptography? How does it differ from Symmetric-Key Cryptography?  
   Public-key cryptography is a form of message security that allows two parties to send information in disguise to each other while authenticating the identity of the sender and receiver. It works by giving each person two keys: a public key and a private key. When data is sent, it is encrypted using the receiver’s public key and the receiver can then decrypt the message using their private key. Public-key encryption differs from symmetric-key encryption because public-key uses two keys while symmetric-key uses only one key to do both the encrypting and decrypting.
2. How is the SSH public key different from the SSH private key?  
   The SSH public key is different from the SSH private key because the public key is published and available for everyone to see. Meanwhile, the private key is only known to you.
3. Where are the SSH keys stored in Linux?  
   The SSH keys in Linux are stored in the directory ~/.ssh/known\_hosts in the personal file directory.
4. How can we add the SSH public key of a user to a Linux server so that they may use it to log into the system?  
   To automatically copy the SSH key, we would use the following command (obtained from Abhishek Prakash’s article “How to Add SSH Public Key to Server” on *Linux Handbook*) and fill it in with the correct information for your user name and address of the server:  
     
   ssh-copy-id -i ~/.ssh/id\_rsa.pub USER\_NAME@IP\_ADDRESS\_OF\_SERVER
5. Explain the digital signature and its mechanism.  
   The digital signature is when you send out information encoded using your private key, meaning anyone with your public key can decode your message. This seems useless, but it actually can verify that the information sent out came from you because only you have access to your private key.

Part 3:

1. What is the UNIX philosophy? Name four principles.  
   The UNIX philosophy is an approach to developing good software. The principles are as follows:  
   1) Write programs that do one thing and do it well  
   2) Write programs that work together (the output of one program is the input of another)  
   3) Design and build software to be tried early  
   4) Write programs to handle text streams since that is a universal interface

Part 4:

1. How do you change the owner of a file or directory?  
   To change the owner of a file directory, use the chown command. The syntax and arguments for the command are as follows:  
    chown owner\_name file\_name
2. What do the Octal values 0, 1, 2, 3, 4, 5, 6, and 7 mean in the chmod command?  
   In the chmod command, we are given that 0 means no permissions, 1 means permission to execute, 2 means permission to write, and 4 means permission to read. By adding these permissions, we can get the other values. 3 means permission to execute and write, 5 means permission to read and execute, 6 means permission to write and read, and 7 means permission to read, write, and execute.
3. What do chmod 400, chmod 755, and chmod 700 mean?  
   Using the octal values from the question above, we can determine the permissions for the user, group, and others, in that order, according to the three digit number after chmod.  
   - For chmod 400, the first digit describes that the user can read the program, and the zeros afterwards indicate that the group and others have no permissions.   
   - chmod 755 means that the user can execute, write, and read to/from the program while the group and others (the fives) indicate that members of those groups can only read and execute the file.  
   - Finally, chmod 700 means that the user can read, write, and execute the file while others and people in the group have no permissions.
4. What does this access permission mean? -rw-r-x—  
   This access permission means that the user can read and write to the program, the group can read the file, and others can execute the file.
5. How should the chmod command look like, so that we can obtain this type of access permission for a file?  
   To obtain this type of access permissions for a file, the chmod command should look like:  
   chmod 641.

Part 5:

1. Use the grep command to show all files and directories in the current path that have “my” or “My” or “MY” or “mY” in their names.  
   ls | grep -i my
2. Use the grep command to show those parts of a text file, called file1.txt, that have numeric characters.  
   grep ‘[0-9]’ file1.txt

Part 6:

1. Kill all processes that are running on the Java Virtual Machine.  
   killall -9 java
2. Kill all processes that have “urgent” in their names.  
   killall -9 urgent

Part 7:

1. Append the content of a text file, my-txt1.txt, to another file, my-txt2.txt, by concatenating the files.  
   cat my-txt1.txt >> my-txt2.txt
2. Check the file size and the disk space usage for the concatenated file.  
   du -h my-txt1.txt  
   df -h my-txt1.txt