1. When you have logged in to a shell, how (i.e., using what commands?) do you first find out information about the environment?
   1. The version of your Linux distribution and the version of your Linux kernel.
      1. there's multiple commands to figure out the version of the distribution and kernels. The command I used was hostnamectl.
   2. The name (binary path) of the current shell.
      1. using echo $0 command line i was able to find out the name of the shell was -bash. Using echo $SHELL i got the binary path of the shell (/bin/bash)
   3. RAM, disk space, and CPU.
      1. cat /proc/cpuinfo for the processor.
      2. cat /proc/meminfo for ram
      3. df for disk space
2. Using the man command, find out what the following commands do: which, pwd, who, whoami, env and whereis. Try using each of them.
   1. Which – located a command
   2. Pwd – print name of current/working directory
   3. Who – show who is logged on
   4. Whoami – print effective userid
   5. Env – run a program in a modified environment
   6. Whereis – locate the binary, source and manual page files for a command.
3. Linux commands can be classified as internal (built into the shell) and external (separate program binaries). How can you tell if a specific command (e.g., cd) is internal or external? Figure out where at least three external commands reside on the system.
   1. The commands that are directly executed by the shell are known as internal commands. No separate process is there to run these commands.
   2. The commands that are executed by the kernel are known as external commands. Each command has its unique process id.
   3. External commands that reside in : found their path by using which <command>
      1. cat is in (/usr/bin/cat)
      2. man is in (/usr/bin/man)
      3. file is /usr/bin/file
4. Making your own commands: the PATH environment variable lists the directories the shell uses to search for external commands. Where can you find documentation on it? How can you add the current directory (whichever directory you are currently in) to PATH? Then, how to make that change permanent? Try to identify multiple ways
   1. export PATH=”$HOME/student:$PATH” to add your current directory, to find current directory use pwd.
   2. Need to edit ~/.bashrc file using nano command and append export PATH=”$HOME/student:$PATH” at the last line. Once done editing use source command to load the file in current shell and double check with echo $PATH. Next time you load in the directory has been added to the path.
5. Look at the permissions of the program binaries of the external commands you have just found above. Who owns them? What group are they in?
   1. Can use ls -ld path/to/command to find owner and group. Which is root for both for nano
6. For those same program binaries, figure out what the permission bits mean by reading the man page of chmod (this is the command you could use to change those permission bits)
   1. r is read, w is write, x is execute, X is execute only if the file is a directory or already has execute permission for some user, s set user or group ID on execution, t is restricted deletion flag or sticky bit
7. What are the owner, group, and permissions of /etc/passwd and /etc/shadow? What are these files used for?
   1. Owner and group is root, permissions for user is read and write, group and o are read only
   2. Owner is root, group is shadow, owner has read and write, group has read only
   3. /etc/shadow contains the password information for the systems accounts and optional aging information
   4. /etc/passwd changes passwords for user accounts
8. What does it mean to have execute permission on a directory?
   1. Means you’re able to enter the directory or execute a files or programs.
9. The ls command can be used to get a listing of the files in a directory. What options are passed to ls to see: the permission bits above; all the files within a directory (including hidden files)? How to make a file hidden?
   1. I use -ld flag to view permission bits
   2. I use -a to view all files including .hidden files
   3. Use mv command to make a file hidden and use a dot in front of the new hidden file(mv file.py .file.py)
10. Compile and run [csimpleshell.c](http://people.scs.carleton.ca/~soma/os-2019w/code/csimpleshell.c). How does its functionality compare to that of bash? List at least 3 differences.
    1. After every command entered it outputs child process ids
    2. Will not cd into a made directory
    3. Cannot check history of previous code using arrow keys