Questions and Exercises to work out and turn in:

Grading Guidelines (See Appendix):

In general, a right answer will get full credit when:

1. It is right (worth 25%)
2. It is right **AND** neatly presented making it easy and pleasant to read. (worth an **extra** 15%)
3. There is an **obvious and clear link[[1]](#footnote-1)** between 1) the information provided in the exercise and in class and 2) the final answer. A clear link is built by properly writing, justifying, and documenting an answer (worth an **extra** 60%).
4. Calculation mistakes will be minimally penalized (2 to 5% of full credit) while errors on units will be more heavily penalized.

**Late Submission** : as specified in the syllabus. Day counting starts one minute after the deadline.

**Check Your Submission:**  after submitting, download your submission to check whether it is the right version and it is complete.

You are welcome/encouraged to discuss exercises with other groups or the instructor. But, ultimately, **personal** writing is expected.

* USE THIS FILE AS THE STARTING DOCUMENT YOU WILL TURN IN. **KEEP IN THE QUESTIONS** AND **INSERT** YOUR ANSWERS **RIGHT AFTER THE QUESTIONS**.
* IF USING HAND WRITING (STRONGLY DISCOURAGED), REWRITE THE QUESTIONS.
* FAILING TO FOLLOW TURN IN DIRECTIONS /GUIDELINES WILL COST A 30% PENALTY.

Objectives of this assignment:

* Resume introduction to basic Unix commands.
* Explore commands related to processes

What you need to do:

* Execute and understand basic Unix commands on Engineering Tux machines.
* Always document your work by taking/inserting screenshots. To save space, clip out the screenshots to contain only relevant information. See Appendix for the required information.
* Insert answers as indicated (right after the question/directions)

Exercise 1: Basic Unix Commands (30 points)

(Well written short answers are acceptable for this assignment)

The objective of this exercise is to get familiar with basic frequently used commands.

**Task**:

**In order to save space, for this assignment and future ones, clip out the screenshots to contain only the relevant information (*the date, the Tux machine you are using for the question, the Auburn username, and material related to the question/task*). Make sure that the screenshots are easily readable.**

Consider these basic frequently used Unix commands:

who, history, pwd, cp, mv, kill, date, cat, ps, top

For each of the above commands,

1) Provide a brief description (you may use the **man** command, but use your own words ultimately)

2) Execute the command on a Tux machine

3) Briefly comment the results/outcomes of the execution

4) Report any unexpected behavior.

No need to provide a screenshot

**who** – The who command displays a list of users to see who is logged into the terminal system with their login times and in some cases will include their IP addresses if they’re remotely logged in.

**history** – The history command displays a list (your history) of commands that you’ve executed into your terminal. It is useful for finding previous commands that you may not remember and may need for debugging.

**pwd** – Print working directory; the pwd command prints the current directory that is working for the user. You should see the absolute path, or location, of the directory that you are in.

**cp** – The cp command will copy files or directories from one location to another. For example, if you run “cp p1.txt p2.txt”, it will copy the “p1.txt” file to the “p2.txt” file. If the file or directory doesn’t exist, then there will be errors that pop up.

**mv** – The mv command will move files or directories to another. It can also be used to rename files. For example, if you run “mv p1.txt location1/”, it will move the “p1.txt” file to the “location1/” directory. Similar to the cp command, errors will pop up if a file with the same name exists in the target destination, or if you lack permissions to the target directory.

**kill** – The kill command sends a signal to terminate, or kill, a running process. It will no longer appear in your ps or top outputs. If the process can’t be terminated or you don’t have permission to terminate it, nothing will happen and/or system errors will occur.

**date** – The date command displays the current date and time. For example, if you run “date”, it will give us today’s date (ex: Tue Sep 10 08:30:40 CDT 2024). If the system’s clock isn’t right, it will display the wrong time.

**cat** – Concatenate; the cat command will read and display the content of files. For example, if you run “cat p1.txt”, it will read and display the content of the file “p1.txt” in the terminal. If the file doesn’t exist, then there will be an error message.

**ps** – Process status; the ps command will display a list of active processes running in your current session.

**top** – Table of processes; the top command will display a real-time view of running system processes (ex: CPU).

[Error! Hyperlink reference not valid.]( ) Exercise 2: Process Management Tools/Commands (70 points)

The objective of this exercise is to observe the "life" of processes using two commonly used Unix commands: ps and top.

**In order to produce answers consistent with the answer key, you must execute ONLY the commands you are asked to perform in Exactly the order given.**

In order to complete this hands-on lab, you must complete the following actions:

- **Download** the C program lab2.c from Canvas

- **Move** lab2.c to your Auburn home directory lab2 that you created in M1: hands-on laboratory exercise. This directory's path is ~/nnnn/lab2 where ~ is your Auburn home directory path (watch if necessary the video about how to move files from your local machine to your Auburn home directory). Observe that you do not need to know the C programming language to complete this hands-on laboratory exercise. When executed, the program lab2.c creates (forks) three child processes. The parent process and children run infinite loops. The only way to stop them is to kill them.

- **Log on** any Tux machine

- **Go** to your lab2 directory at ~/nnnn/lab2 (See M1: Hands-on Lab..)

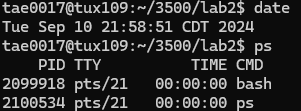
- **Type** **ls** to insure that the file *lab2.c* is in the current directory (i.e., ~/nnnn/lab2)

- **Type** the command **cc -o lab2 lab2.c** (This will compile lab2.c and produce executable lab2)

- **Insure** that the compilation is successful. Check whether the executable lab2 is in the current directory (use ls -al). If successful, complete the following actions and answer the questions:

1) (7 points) Type the **ps** command (consider a man ps to learn about ps)

What are the processes IDs and names listed by the command ps? You can insert a screenshot here to answer the question



2) (7 points) **Type** the command **./lab2&** followed by return and then **type** the command **ps**. Do not forget the '**&**' character. If you do not add the '**&**' character, you will not be able to get the prompt for the next commands.

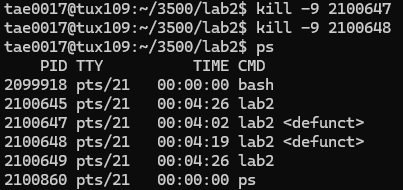
What are the processes IDs and names listed by the command **ps**? You can insert a screenshot here to answer the question

A screenshot of a computer

Description automatically generated

The program lab2 creates 4 processes: a parent process and 3 child processes. The parent has a smaller process ID than its children's.

3) (7 points) In order to kill a process with process ID *n*, you must type the command kill -9 n. For this question, kill two **child** processes. Type the command **ps** and take a screenshot



Do the killed processes show up on the screenshot?

Yes, the killed processes still show up on the screenshot; however, they show up as <defunct> because the parent process has not destroyed them properly.

If the two child processes still show up, how are they different from the alive child process?

The two child processes still show up and they are different from the alive child process because the killed processes are defunct, or dead processes that haven’t been destroyed properly.

4) (7 points) Kill the parent process and type the **ps** command

Take a screenshot

A screen shot of a computer

Description automatically generated

Do the killed processes show up?

No, the killed processes don’t show up because the parent process has also been killed, which will only leave the one child process that is still alive.

Does the alive child process still show up?

Yes, the alive child process still shows up in the directory.

If yes, kill it. Check with **ps** that all processes created by lab2 were killed.

5) (7 points) Let us call ***Terminal A*** the terminal you used so far for all the previous questions you answered so far. Keep Terminal A open and open a new ***terminal B***. Use Terminal B to log on the **SAME** Tux machine you used under Terminal A. Execute on Terminal A the command **./lab2&** followed by return and then the command **ps.**

Do you see (on Terminal A) the processes created by the program lab2?

Yes, I see the 4 processes (one parent process and 3 child processes) created by the program lab2 on Terminal A.

Type on Terminal B the command **ps**. Do you see (on Terminal B) the processes created by the program lab2?

No, I don’t see the 4 processes created by the program lab2 on Terminal B because the ps command only works in one current terminal, and since the processes were started on Terminal A, they belong to only that terminal. If you type “ps” in Terminal B, it will only show the processes that are running in that specific terminal.

If no, can you explain why the processes created by the program lab2 do not appear on Terminal B when executing **ps**? Do a **man** **ps** or Google **ps** to understand and document your answer.

6) (7 points) On Terminal B, type the command **top -U username** (where username is your Auburn username).

Do you see on Terminal B the processes created by the program **lab2**?

Yes, I see the processes created by the program lab2 in Terminal B.

How many "zombies" appear on Terminal B? (Search for "Zombie" on Terminal B...)

There are 2 “zombies” that appear on Terminal B.

7) (7 points) On Terminal A, kill two child processes.

Do you see on Terminal B the processes created by the program lab2?

Yes, I see the processes created by the program lab2 in Terminal B.

How many "zombies" appear on Terminal B?

There are now 4 “zombies” that appear on Terminal B because after the two child processes were killed, two more zombies appeared after the other two zombies.

Google "zombie Unix processes". **Based on your experience** in this hands-on laboratory exercise, what is a "Zombie" process?

A “zombie” process is a process that has already finished execution, but is still listed in the process table because its parent process has not destroyed them properly.

8) (7 points) On Terminal A, kill the parent process.

Do you see on Terminal B the processes created by the program **lab2**?

Yes, I see the processes created by the program lab2 in Terminal B.

How many "zombies" appear on Terminal B?

There are 2 “zombies” that appear on Terminal B after the parent process has been killed.

9) (7 points) Type on Terminal A the command ps

Does the information provided by **ps** on Terminal A match the information provided by top on Terminal B?

No, the information provided by “ps” on Terminal A does not match the information provided by “top” on Terminal B.

Based on your experience with **ps** and **top**, what is (are) the key difference(s) between them?

A key difference between “ps” and “top” is that “ps” is a one-time command that displays the status of a process while they’re running, and “top” is a real-time command that shows a continuously updating table of processes.

10) (6 points) On Terminal A, kill the last child process.

Do you see on Terminal B the processes created by the program lab2?

Yes, I see the processes created by the program lab2 in Terminal B, but there aren’t as many processes on there as there were before.

How many "zombies" appear on Terminal B?

There are still 2 “zombies” that appear on Terminal B.

Type **ps** on Terminal A.

All the processes that were recently created have been killed already, according to Terminal A.

Hit on Terminal B the key 'q' to leave the command **top**.

All the above questions are an **opportunity** to discover more about Unix OS commands. Use man and the Internet to learn more. Doing just the minimal will make you proficient. **Knowing well Unix may help you in an interview for your dream job**.

Do not hesitate to ask questions or seek help on Piazza. Start early to avoid getting stuck at the last time when your classmates and instructor are not available.

What you need to turn in:

* Electronic copy of this file (including your answers) (standalone). Submit the file as a Microsoft Word or PDF file.
* Recall that answers must be well written, documented, justified, and presented to get full credit.
* How this assignment will be graded:
* A right answer will get full credit when:
* It is right (worth 25%)
* It is right AND neatly presented making it easy and pleasant to read. (worth 15%)
* There is an obvious and clear link between 1) the information provided in the exercise and in class and 2) the final answer. A clear link is built by properly writing, justifying, and documenting an answer (worth 60%).
* Calculation mistakes will be minimally penalized (2 to 5% of full credit) while errors on units will be more heavily penalized.
* You are welcome/encouraged to discuss exercises with other students or the instructor. But, ultimately, personal writing is expected.

**Appendix**: Grading: What is an OBVIOUS and CLEAR LINK?

Here is an example to explain what an **obvious and clear link** is and how we grade your work.

Consider the following problem:

"(100 points) John travels from Auburn to Atlanta in his car at a speed of 50 mph. Leaving at 8am, at what time will John reach Atlanta".

Here are the answers of three students and their scores:

**Student 1** answers: "10am". Student 1 will get 25 points.

**Student 2**answers : "John will reach Atlanta at 10am". Student 2 will get 25+15 = 40 points

**Student 3** answers: "The time t to travel a distance d at speed v is equal to d/v = d/50mph. The problem does not provide the distance d from Auburn to Atlanta. Based on Google, the distance from Auburn to Atlanta is approximately 100 miles (**document is here**). Therefore, the time t = 100 miles/50mph = 2 hours. Since John left at 8am, he will then reach Atlanta at 8am + 2 hours = 10 am".

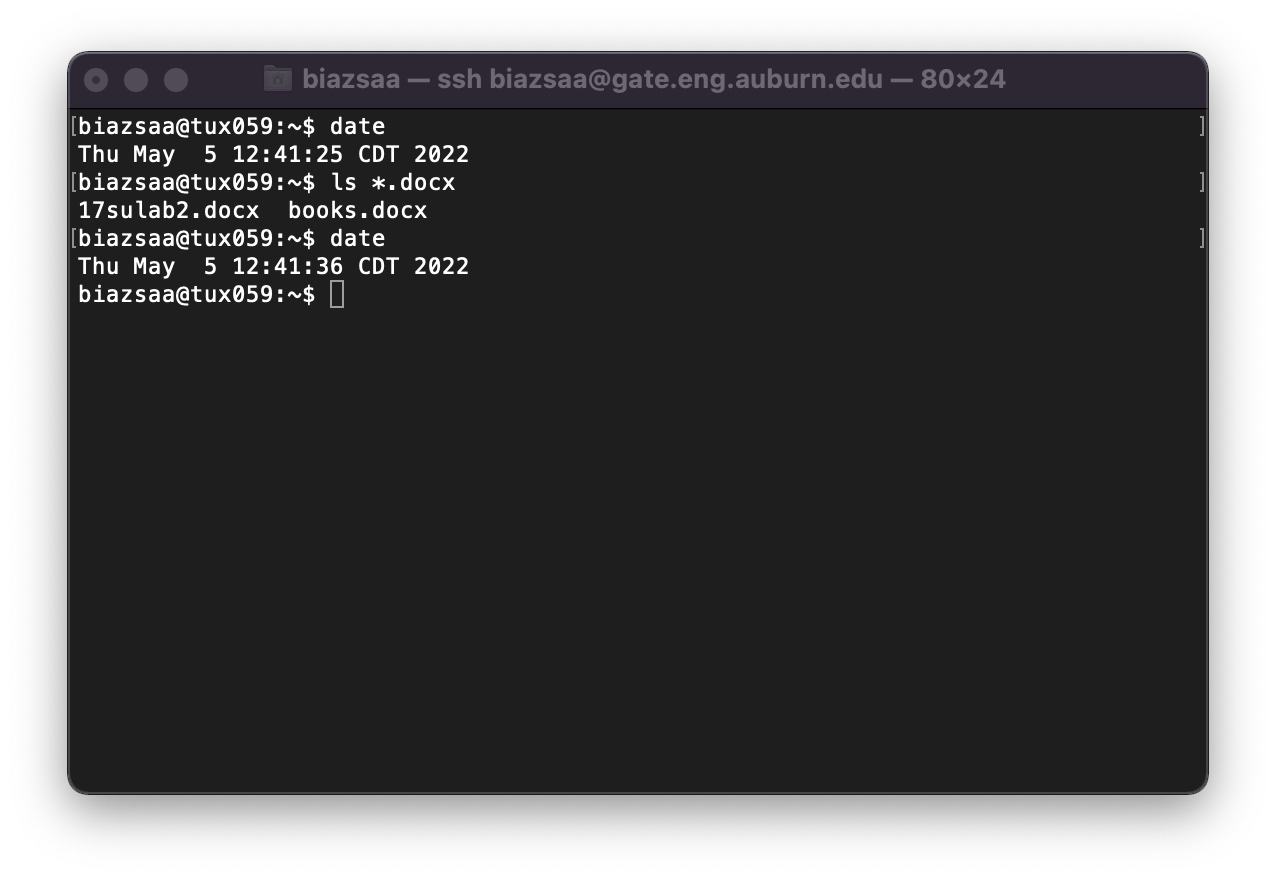
**Student 3** will get 25 + 15 + 60 = 100 points

Do you see the **direct** **link** going from the data provided in the question to the final answer, using general knowledge/formula and documents?.... Can you now solve the following problem and get 100 points?

"(100 points) Alice travels from Auburn to Atlanta in her car at a speed of 50 mph. Leaving at 8am, at what time will Alice reach Atlanta assuming that she had a flat tire that delayed her 30 minutes".

**Screenshot: Required Information**

**In order to save space, for this assignment and all *FUTURE* ones, clip out the screenshots to contain only the relevant information. *When applicable, ALL screenshots must show the date, the machine you are using for the exercise and the username of one of the team mates*. Make sure that the screenshots are easily readable. Below is template screenshot:**

****

1. See on the appendix what an obvious and clear link is. [↑](#footnote-ref-1)