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. after submitting, download your submission to check whether it is the right version and it is complete. A wrong version will get at least a 25 points penalty that cannot be claimed back by the *Leniency Policy for Borderline Incidents*. So, check your submissions.

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

* USE THIS **EDITABLE** FILE AS THE STARTING DOCUMENT YOU WILL TURN IN. **KEEP IN THE QUESTIONS** AND **INSERT** YOUR ANSWERS **RIGHT AFTER THE QUESTIONS/PROMPTS**. **DO NOT DELETE ANYTHING** 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:

* to work on a Unix based system
* to become familiar with the notion of a *process*
* to assess/evaluate the *fairness* and *reliability* of the CPU scheduler on Tux machines

What you need to do:

* Examine and understand the pseudocode of a provided C source program *lab3*. (You do not need to dig into the C code)
* Run experiments of *lab3*
* Collect and process data about the experiments
* Build plots
* Interpret and discuss the plots
* Conclude about the fairness and reliability of the CPU scheduler.

**Important:**

* *One submission per team.* ***You must complete all tasks on an Engineering Unix Tux machine, otherwise a 0 will be assigned***
* *Writing and presentation of your report are considered to grade your hands-on lab. Your conclusions* ***must be supported*** *by the data/measurements you collect. Your conclusions must be correct.*
* ***Questions about this lab must be posted on Piazza if you need timely answers****.*
* ***Work ahead. Do not wait until the last minute.***

(Reminder) **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*). A 25 points penalty will be applied when any required information is missing. Make sure that the screenshots are easily readable.**

Exercise (100 points)

This hands-on lab has multiple parts: 1) Compile, execute, skim, and understand the pseudocode of a provided C *program*, 2) run experiments, 4) collect data, and 5) analyze and report your results.

The instructor designed and implemented in C the provided C program lab3.c.

**The Provided C Program**

The help you understand the C program, the instructor provides here its pseudocode:

**Inputs**:

**np**: number of **concurrent** processes to execute

**nm**: number of minutes each process must run

**Output**:

Displays when each process starts executing its task

Displays the index achieved by the task executed by each process. The more a process "computes" (uses the CPU), the higher is c.

Collects (writes to a file) (done by each child process)

Displays the amount of time the process "ran" on the CPU (done by each child process)

Process the data collected by each process. (Done by Parent process)

Displays mean, standard deviation (std) and percentage std/mean of (Done by Parent process)

**Code (Parent)**:

for i=1 to ***np***

create (fork) a process Pi

for i=1 to ***np***

"green light" Pi to start its task

Compute/displays mean and standard deviation/mean of collected by child processes.

**Code of Process Pi (Child Process):**

wait for green light

Display start time

Execute A computing task for ***nm*** minutes (increases )

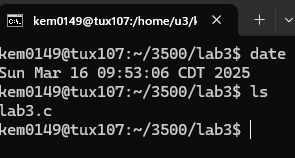
Displays the amount of time the process competed for the CPU (should be ***nm*** minutes)

Collects (writes to a file "file.dat")

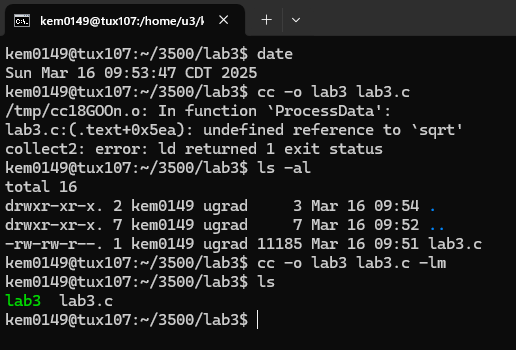
Do not hesitate to ask questions or seek help on Piazza. Start early to avoid getting stuck at the last time when your classmates and instructor are not available.

**What You Need To Do**:

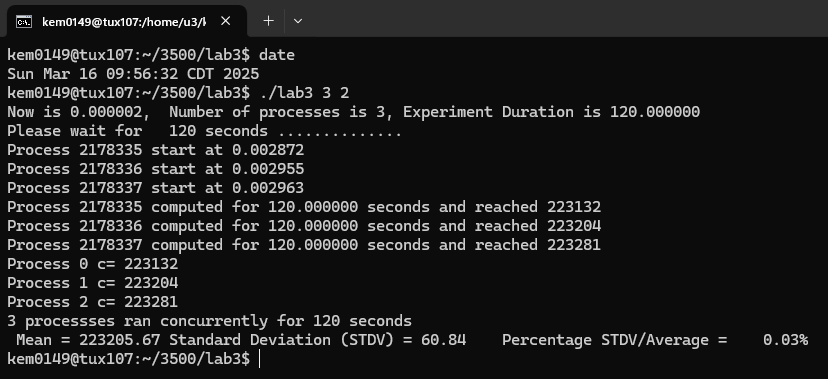
1. Download the C program lab3.c from Canvas
2. Store lab3.c to your Auburn home directory lab3 that you created in M1: hands-on laboratory exercise. This directory's path is ~/nnnn/lab3 where ~ is your Auburn home directory path.
3. **Log on** any Tux machine
4. **Go** to your lab3 directory at ~/nnnn/lab3 (See M1: Hands-on Lab..)
5. **Type** **ls** to insure the file *lab3.c* is in the current directory (i.e., ~/nnnn/lab3)



1. **Type** the command **cc -o lab3 lab3.c -lm** (This will compile lab3.c and produce executable lab3)
2. **Insure** that the compilation is successful. Check whether the executable lab3 is in the current directory (use ls -al). If successful, complete the following actions and answer the questions:



1. **Test** the program lab3 for running three child processes for 2 minutes
2. To test lab3, **type** the command ./lab3 3 2



1. **Observe** the outputs produced by lab3 execution
2. After you become familiar with lab3, you will need to **check** whether lab3 is a good measurement tool meeting some requirements described below.
3. After checking the requirements, you will run lab3, collect data, plot your results, and discuss your results

**Requirements (for lab3 to be a good measurement tool)**

1) **Requirement I:** If we run one process multiple times for the same number of **nm** minutes, the number should be the same each time within

2) **Requirement 2:** If we run **np** processes multiple times for ***k\* nm*** minutes, the number displayed should be times larger than the number displayed for running **nm** minutes within . For example, if I run 20 processes for minutes, the number displayed should be times larger than the number displayed for running 20 processes for minutes within .

3) **Requirement 3:** If we run **100**processes multiple times for ***2*** minutes, the number displayed should be times larger than the number displayed for running **1000** processes for **2**minutes within . For this requirement, your code will be tested with these values **100** and **1000**.

**CPU Scheduling Evaluation:**

The objectives are:

a) to check whether lab3 program meets the requirements

b) to evaluate the **fairness** and the **reliability.**

We describe below the experiments you must conduct to achieve the objectives.

a) Checking whether your task meets the requirements

**Requirement 1** (10 points):

- Run lab3 five times during 2 minutes with np = 1 (one process). For each run of lab3, collect and report here the five values you collected for .

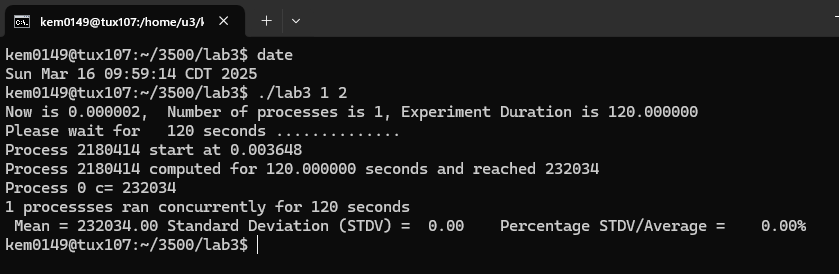
c = 232034

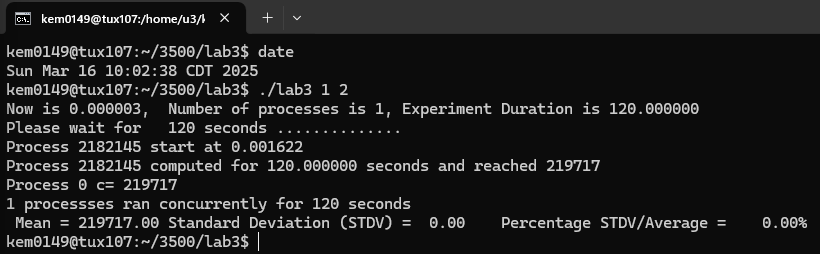
c = 219717

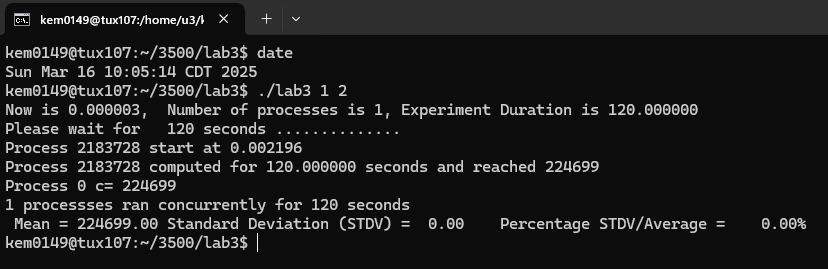
c = 224699

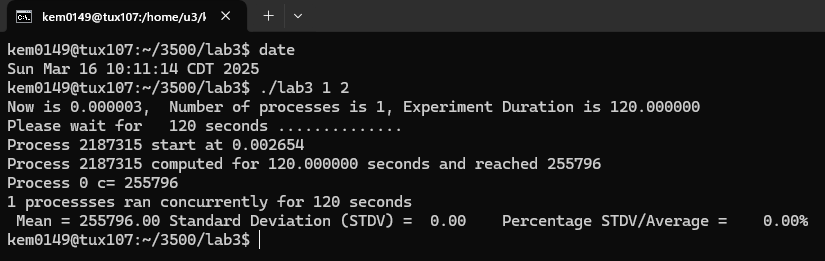
c = 255796

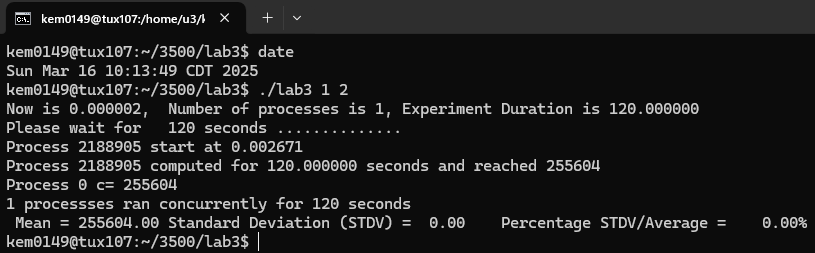
c = 255604











Are the five values within of the mean of the 5 values of c?

The mean of the values is 237570. The c values have to be within the range of 213813 – 261327. With checking the values of c it shows that yes all five values are +-10% of the mean of the 5 values of c.

**Requirement 1I** (10 points):

**First**, Run lab3 once during 2 minutes with np = 20 (20 processes).

Collect the mean (for all 20 processes) .

Process 0 c= 65114

Process 1 c= 69625

Process 2 c= 63005

Process 3 c= 64369

Process 4 c= 62048

Process 5 c= 68016

Process 6 c= 67210

Process 7 c= 66164

Process 8 c= 66214

Process 9 c= 61939

Process 10 c= 63303

Process 11 c= 66889

Process 12 c= 65220

Process 13 c= 71691

Process 14 c= 64422

Process 15 c= 65043

Process 16 c= 69036

Process 17 c= 70620

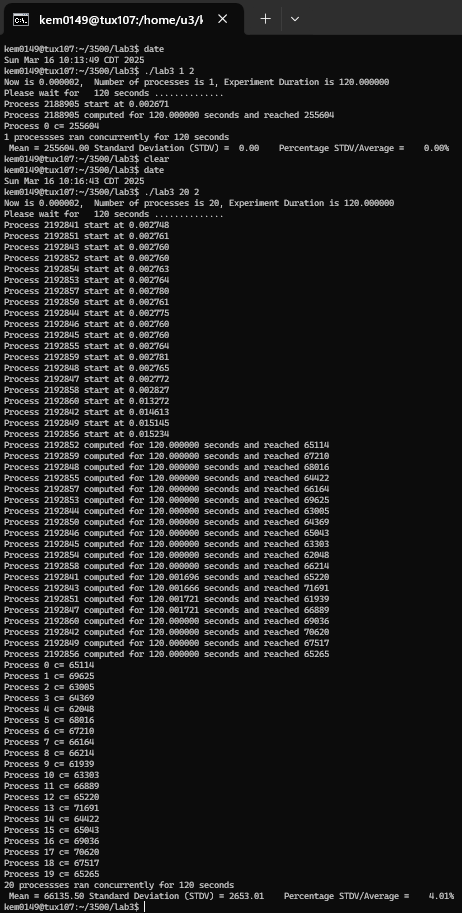
Process 18 c= 67517

Process 19 c= 65265

Mean = 66135.50

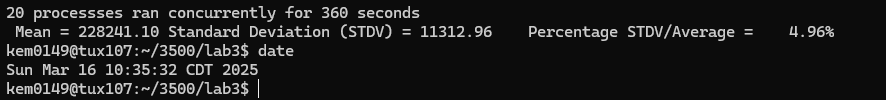
Standard Deviation = 2653.01

Percentage STDV/Average = 4.01%



**Second**, Run lab3 once during 3\*2 (6) minutes with np = 20 (20 processes).

Collect the mean (for all 20 processes)



Mean 228241.10

Standard Deviation = 11312.96

Percentage STDV/Average = 4.96%

Is within? 22824.11

C6 = 228241.10

+-10% = 205416.99 – 251065.21

C2 = 66135.50

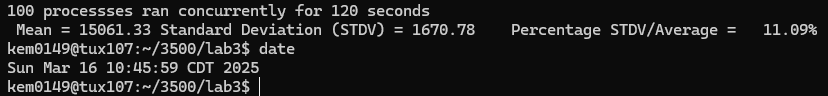
3 \* c2 = 198406.5

This shows that it is not within +-10% of each other.

**Requirement 1I1** (10 points):

**First**, Run lab3 once during 2 minutes with np = 100 (100 processes).

Collect the mean (for all 100 processes)



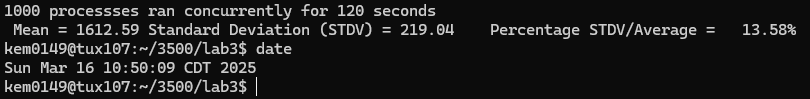
Mean = 15061.33

Standard Deviation = 1670.78

Percentage STDV/Average = 11.09%

**Second**, Run lab3 once during 2 minutes with np = 1,000 (1000 processes).

Collect the mean (for all 1,000 processes)



Mean = 1612.59

Standard Deviation = 219.04

Percentage STDV/Average = 13.58%

Is within?

C100 = 15061.33

+-10% = 13555.197 – 16567.463

C1000 = 1612.59

10 \* c1000 = 16125.9

10 \* C1000 is within +-10% of c100

a) (30 points) **Fairness**: If the CPU scheduler is **fair**, the measure **c** (number displayed by lab3) must be close for all child processes you run within the **same** experiment. We call an *experiment* the fact to run the program *lab3* **once**. We measure how the values c produced by each child process are close by computing the percentage of the standard deviation/mean: the lower is this value, the closer are the c values of the processes, and the fairer is the CPU scheduler. Ideally, all processes must use the CPU for the same duration and therefore reach the same value c.

In order to evaluate fairness, you must conduct one experiment during 2 minutes for each of the following values of **np** (number of processes): 2, 5, 10, 20, 40, 60, 80, 100, 200, 400, 500, 1000. For each experiment with value , compute the percentage of the standard deviation/mean of the measure c your task displays for each process. Plot the percentage of the standard deviation/mean of c versus the number of processes (put the np values 2, 5, 10, 20, 40, 60, 80, 100, 200, 400, 500, 1000 on the x-axis and the percentage of the standard deviation/mean of c on the y-axis) Discuss this plot and draw reasonable conclusions regarding fairness. In other words, is the CPU scheduler of the Tux machines fair based on the data you collected?

2 (mean: 245217, standard deviation: 40) – 0.02%

5 (mean: 94725.6, standard deviation: 4459.26) – 4.71%

10 (mean: 82528.8, standard deviation: 14005.92) – 16.97%

20 (mean: 65067.9, standard deviation: 5645.17) – 8.68%

40 (mean: 33688.03, standard deviation: 1410.16) – 4.19%

60 (mean: 23320.13, standard deviation: 2583.27) – 11.08%

80 (mean: 17519.9, standard deviation: 1863.72) – 10.64%

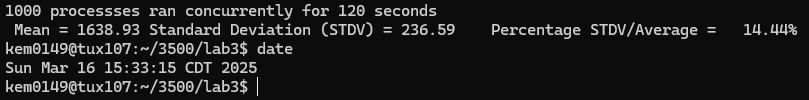
100 (mean: 14243.63, standard deviation: 1552.11) – 10.90%

200 (mean: 7733.88, standard deviation: 985.92) – 12.75%

400 (mean: 3921.63, standard deviation: 499.40) – 12.73%

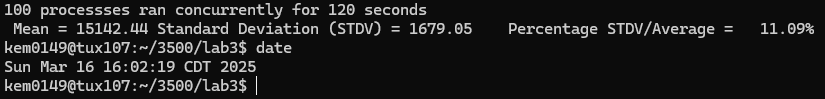
500 (mean: 3153.34, standard deviation: 393.35) – 12.47%

1000 (mean: 1638.93, standard deviation: 236.59) – 14.44%



The plot illustrates that at the beginning at 2 and 5 the fairness percentage was low and closer to the c value. As the number increases the fairness percentage grows. This shows that the CPU scheduler struggles at higher loads.

b) (20 points) **Reliable**: If the CPU scheduler is **reliable**, the average measure **c** (number displayed by lab3) for **np** processes must remain the same (within .) for successive experiments running for the same number of processes **np** each time. In other words, if you execute the same process on a machine, the execution time should not widely vary: the execution time should not be one time 2 minutes, then 2 hours, then 1 minute, then 30 minutes.....

In order to evaluate reliability, run 10 times lab3 with np=100 (100 process) during 2 minutes (./lab3 100 2). 

1 – 15142.44

2 – 14228.24

3 – 16118.9

4 – 15759.76

5 – 15721.64

6 – 15596.34

7 – 15679.15

8 – 14546.3

9 – 15569.99

10 – 15395.37

Plot here the mean versus the run # (put the values 1, 2, ....10 on the x-axis) and the mean on the y-axis. (no tables)

Discuss here this plot and draw reasonable conclusions regarding reliability. In other words, is the CPU scheduler of the Tux machines reliable based on the data you collected?

Yes, the CPU scheduler of the Tux machines are reliable based on the data. Each of the points of the data is within +-10% of each other. 10% of the 1st run is 1514.244 and each one of the following runs are within that.

**Do not hesitate to ask questions on Piazza if you have any doubt.**

**Getting Started**

1. compile the code I provided you by typing:

cc -o lab3 lab3.c -lm

1. Execute the code (evaluate 1 process during 2 minutes): ./lab3 1 2
2. Observe what is displayed
3. Compile and execute to check whether your task meets the requirements (see above)
4. Make sure to complete all required tasks.
5. Do not hesitate to ask questions on Piazza or during office hours.

**Common mistakes**

1) Finish the code at the last minute: you will not have enough time to collect data and write a good report

2) Finish the lab/report at the last minute without polishing may result in a low score. You must have time to reflect about the meaning of these experiments and the conclusions you may draw.

**What to turn in?**

One file:

**Electronic copy** of **this** file that includes your answers. I repeat: you must insert your answers and plots in **this** file. Do not delete anything from this file. This file with your plots and answers must be put posted **separately** on Canvas (not in a zipped folder). **A penalty of 30 points will be applied if these instructions are not followed.**

Good writing and presentation are expected.

**In case of doubt, do not hesitate to ask questions on Piazza.**

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".

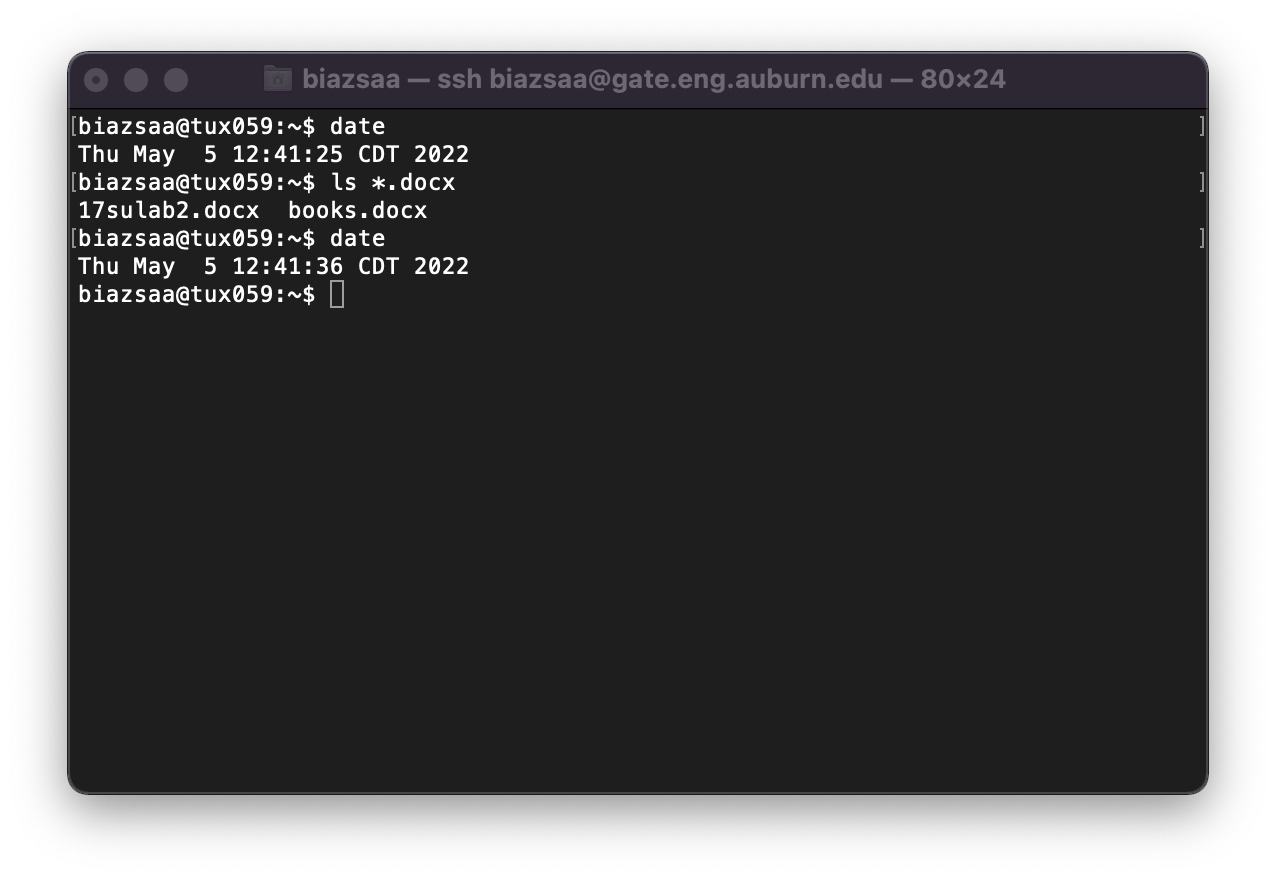
**Student 3** will get 25 + 15 + 60 = 100 points

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 Tux machine you are using for the exercise and the Auburn 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)