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:

* to work on a Unix based system
* to explore the impact of the memory accesses pattern on the performance

What you need to do:

* to work on a Unix based system
* Review a few Unix commands
* Analyze the time complexity of two algorithms that perform the same number of instructions
* Explore the impact of the memory accesses pattern on the execution time

**Important:**

* *One submission per team.* ***You must complete all tasks on an Engineering Unix Tux machine.***
* *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.***

Task 1: Basic Unix Commands (20 points)

(Well written short answers are acceptable for this assignment)

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

**In order to save space, for this assignment and future ones, clip out the screenshots to contain only the relevant information. Make sure that the screenshots[[2]](#footnote-2) are easily readable.**

1) (4 points) The shell allows *redirection* of the output. By default, the result of your commands is displayed by default on the terminal. You can redirect the "displaying" onto a file. For example, execute these commands in your nnnn directory you created for this course (nnnn is your course #):

- ls

- ls **>** dump

- cat dump

**Explanation**

After logging onto the TUX machine, I entered cd 3330 to change from the current directory to the desired location of “3330”. Then using the ls command to list the files and other sub directories in the current directory which is “3330”. Now with the use of “>” the output of the ls command will be saved to the file that follows, which is called “dump”. After executing “ls > dump” there will not be anything displayed directly to the terminal. To verify that the command worked I then enter “cat dump” to show the contents of this file. The “cat” command will show the contents of the dump file. Below is a screenshot showing these actions.

A computer screen shot of a computer code

Description automatically generated

2) (4 points) What does the command wc do?

Execute these commands under the same directory where the file *dump* is:

- wc dump

- wc -l dump

**Explanation**

Firstly the “wc” command stands for “word count”. This command can be used to count the number of lines, characters, and words in a file or even simply an output stream of some sort. The “wc dump” command when entered and executed will show the output of all three of these. So it will display the output of the total number of lines, characters, and words in the given file called “dump”. Once I have added the parameter “-l” with the complete command being “wc -l dump”, this will now only show the total number of lines present in the file “dump”. The totals for both the characters and words will not be included in this version of the command.

A screen shot of a computer

Description automatically generated

3) (4 points) Download the files poem1.tex and poem2.tex. These are slightly different translations of Rimbaud's poem "*Le Dormeur Du Val*". How do they differ? Here is a command you could use:

- cmp poem1.tex poem2.tex

- cmp -b poem1.tex poem2.tex

**Explanation**

The “cmp poem1.tex poem2.tex” command compares the contents of the two text files. It highlights only differences between the two text files. The screenshot below confirms there is a difference between the files, specifically byte 673, on line 17. The “cmp -b poem1.tex poem2.tex” will highlight the same difference between the two text files, but will also show the bits that differ between the files. Specifically, the difference is Poem1.txt has a value of “i” and Poem2.txt has a value of “o” at byte 673.

A screen shot of a computer code

Description automatically generated

4) (4 points) What does the command grep do?

**Explanation**

The “grep” command can be used to search for certain text patterns that can be found within a given file or with using the output of other commands when executed. This command provides a great way to search for specific patterns or strings within a given file or command output like previous described.

* The “grep his poem1.tex” command will search the file “poem1.tex” and display any lines that include the exact string “his”.
* The “echo po\*.tex” command will display the names of any file within the current directory that start with the character string “po”. The asterisk symbol matches any sequence of characters that follow “po”. After executing this command, the files that are returned to be returned and displayed are “poem1.tex poem2.tex”.
* The command “grep his po\*.tex” is a combination of commands similar to the two I described above. The “grep his” part of the command will search for the pattern “his” in all of the files in the current directory that start with “po” and then end with the version type “.tex”. Below are 3 screenshots showing each of these commands.

Execute these commands under the same directory where the files poem1.tex and poem2.tex are.

- grep his poem1.tex // no space around "."

- echo po\*.tex // no space around "."

- grep his po\*.tex // no space around "."

A screenshot of a computer screen

Description automatically generated

5) (4 points) The ampersand character **&** allows to use the shell (command prompt) while a command or a program is being executed. Execute these commands to see the effect of & (emacs is a popular editor):

- emacs&

- ps

-emacs&

- ps

Kill the two emacs processes

Take a screenshot (must show starting from the first command emacs&) and insert it here.

A computer screen shot of a black screen

Description automatically generated

Task 2 (80 points)

The objective of this task is to explore the impact of the pattern of the memory accesses on the execution time of a program. In order to complete this exploration, you must measure the execution time of two algorithms that execute the same job by performing exactly the same number of instructions/operations. After measuring, you must draw conclusions and explain potential differences between the execution time of the two algorithms. You must work these hands-on laboratory exercise on a Tux machine.

# Exercise: For this exercise, you must work with the lab6.c program. This program will allow you to collect data.

1) (5 points) Consider the following algorithm:

**InitializeMatrix0(M,n) // Version 0**

inputs: Matrix of characters, , the dimension of the nxn M matrix

output: none. Each element of Matrix M is initialized with '0'

for i = 1 to n

for j = 1 to n

Matrix[i][j] = '0';

What is the time complexity of the above algorithm (Version 0) when you count the assignments "Matrix[i][j]='0';". How many of these assignments will be executed? Express this number as a function of .

Answer here ... (Brief justification is expected. I trust that you know now how to analyze such algorithm)

**Explanation**

This is a nested for loop, controlled by an inner loop “i” and outer loop “j”. The total amount of assignments is determined by the product of the loop iterations. Therefore loop “j” will execute completely for each iteration of “i”. The equation for assignments A(n) = n x n, where n is number of iterations. Therefore, the time complexity of counting assignments will be O(n2).

2) (5 points) Consider the following algorithm:

**InitializeMatrix1(M,n) // Version 1**

inputs: Matrix of characters, , the dimension of the nxn M matrix

output: none. Each element of Matrix M is initialized with '0'

for j = 1 to n

for i = 1 to n

Matrix[i][j] = '0';

What is the time complexity of the above algorithm (Version 1) when you count the assignments " Matrix[i][j]='0';". How many of these assignments will be executed? Express this number as a function of .

Answer here ... (Brief justification is expected. I trust that you know now how to analyze such algorithm)

**Explanation**

This is a nested for loop, controlled by an inner loop “j” and outer loop “i”. The total amount of assignments is determined by the product of the loop iterations. Therefore loop “i” will execute completely for each iteration of “j”. The equation for assignments A(n) = n x n, where n is number of iterations. Therefore, the time complexity of counting assignments will be O(n2).

3) (10 points) Check within the source program lab6.c whether the C code implements "faithfully" the pseudocode for both version to initialize the matrix (Examine the methods InitializeMatrix0 and InitializeMatrix1). Compare the two algorithms: do they perform the same job? do they execute the same number of operations? what is the key difference between the two algorithms? If these two algorithms execute under the exactly same conditions (same hardware, OS, load...), **should** them perform similarly (same execution time)? Answer/Justify/Explain

**Explanation**

Both algorithms do not do the exact same job. Yes, they are both nested for loops, however, the key difference is in the order the loops “i” and “j” execute. The InitializeMatrix0 method uses “i” as the outer loop then “j” as the nested loop whereas InitializeMatrix1 method uses “j” as the outer loop then “i” as the inner loop. This small but key difference changes the order of the how the matrix initializes values. InitializeMatrix0 method sets elements in the matrix by moving across each row before moving to the next row. InitializeMatrix1 method sets elements in the matrix by moving across each column before moving to the next column. Both methods will execute the same number of operations and share the same time complexity as they both accept the same parameter named “Dimension” which determines the ”n” number of iterations.

4) (5 points) Download the program lab6.c. Compile it by typing cc -o lab6 lab6.c. Execute the program lab6 by typing: ./lab6 maxdimension step Version where maxdimension is the dimension of the largest matrix M to initialize, step is the step of the data points, and Version is the version (0/1) of the algorithm. Below is the pseudocode of lab6.c:

**LaunchExperiment(M, n)**

inputs: Matrix of characters,

, the dimension of the largest matrix M to initialize,

step is step between data points (see pseudocode)

Version is the version of the algorithm to use

output: a **file** containing the execution time for a given matrix dimension. File is named **file-maxdimension-step-Version0/1.csv. This** file can be opened with Excel for plotting....

for dimension = step to maxdimension (increasing by step)

if Version == 0

InitializeMatrix0(M,dimension) // Version 0

else

InitializeMatrix1(M,dimension) // Version 1

write in file *dimension* and execution time of InitializeMatrix0/1(M,dimension)

***For example***, if you execute **./lab6 16384 2048 1**, *LaunchExperiment* will initialize the matrices with dimensions 2048, 4096, 6144, 8192, 10240, 12288, 14336, and 16384, respectively. For each initialized matrix, it will collect the execution time and will write the pairs (dimension, execution time) in a file named **file-16384-2048-Version1.csv** (recall **16384** is *maxdimension*, **2048** is the step, and **1** is the version). You will find the data in such a file to build plots. This file can be opened by Excel (csv format).

For this question Execute and time **./lab6 32768 2048 0**, report the time it takes to complete the execution time, take a screenshot of the execution (clip out the screenshot) and **insert it here..**

**Hint**: to time the execution, you can simply type ***time ./lab6 32768 2048 0***

The 3 times this output provided were the real at 0m19.614s, the user at 0m18.972s, and the sys at 0m0.639s. Below is the screenshot of this output.

A computer screen shot of a black screen

Description automatically generated

5) (5 points) Execute and time **./lab6 32768 2048 1**, report the time it takes to complete the execution time, take a screenshot of the execution (clip out the screenshot) and **insert it here..**

**Hint**: to time the execution, you can simply type ***time ./lab6 32768 2048 1***

The 3 times provided by this version of the output were the real at 6m14.361s, the user at 6m13.757s,

And the system at 0m0.632s.

**A screenshot of a computer

Description automatically generated**

6) (30 points)

a) Execute *time* ***./lab6 65536 4096 0*** and collect the file **file-65536-4096-Version0.csv**

b) Execute *time* ***./lab6 65536 4096* 1** and collect the file **file-65536-4096-Version1.csv**

c) (15 points) Build on the **same** graph the plots of execution time of the two versions of the algorithms (0 and 1) versus the matrix dimension: put the matrix dimension on the x-axis and the execution time on the y-axis.

A graph with a blue line

Description automatically generated

d) (15 points) Build the plot of as a function of matrix dimension: put matrix dimension on the x-axis and on the y-axis.

**A graph with a green line

Description automatically generated**

7) (5 points) Comment and discuss the two graphs built above. Highlight the potential differences (execution time) between Version 0 and Version 1 of the matrix initialization algorithm.

Both graphs show a clear trend of increasing execution time as matrix dimensions increase. In the first graph, where we plotted both Version 0 and Version 1 on the same graph, Version 1 consistently took longer than Version 0 for all matrix sizes. The second graph, which shows the ratio of Version 1 to Version 0, starts with a steep increase and then plateaus, indicating that as the matrix size gets larger, the relative difference in execution times between the versions becomes consistent. Version 1 is less efficient overall since the ratio is greater than 1, meaning Version 1 always takes more time than Version 0 for the given matrix dimensions.

8) (15 points) **Why?** Provide the **reason** of the potential difference in execution times between Version 0 and Version 1 of the matrix initialization algorithm.

The primary reason for the difference in execution times between Version 0 and Version 1 of the matrix initialization algorithm lies in the order of iteration over the matrix elements. In Version 0, the matrix is initialized in row-major order (outer loop is rows, inner loop is columns), which is consistent with the layout of the array in memory. This leads to better cache locality and less cache thrashing as contiguous memory locations are accessed in order. Version 1 initializes the matrix in column-major order (outer loop is columns, inner loop is rows), which leads to poorer memory access patterns since it's accessing memory in a less contiguous manner. The CPU cache is optimized for accessing data that's close in memory (spatial locality), which is what Version 0 does effectively, but Version 1 does not, hence the slower execution times for Version 1.

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

**Common mistake**

Starting the hands-on lab at the last minute.

**What to turn in?**

One file

a) **Electronic copy** of **this** file that includes your answers. I repeat: you must insert your answers in **this** file. Do not delete anything from this file. This file with your answers must be put posted **separately** on Canvas (not in a zipped folder).

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".

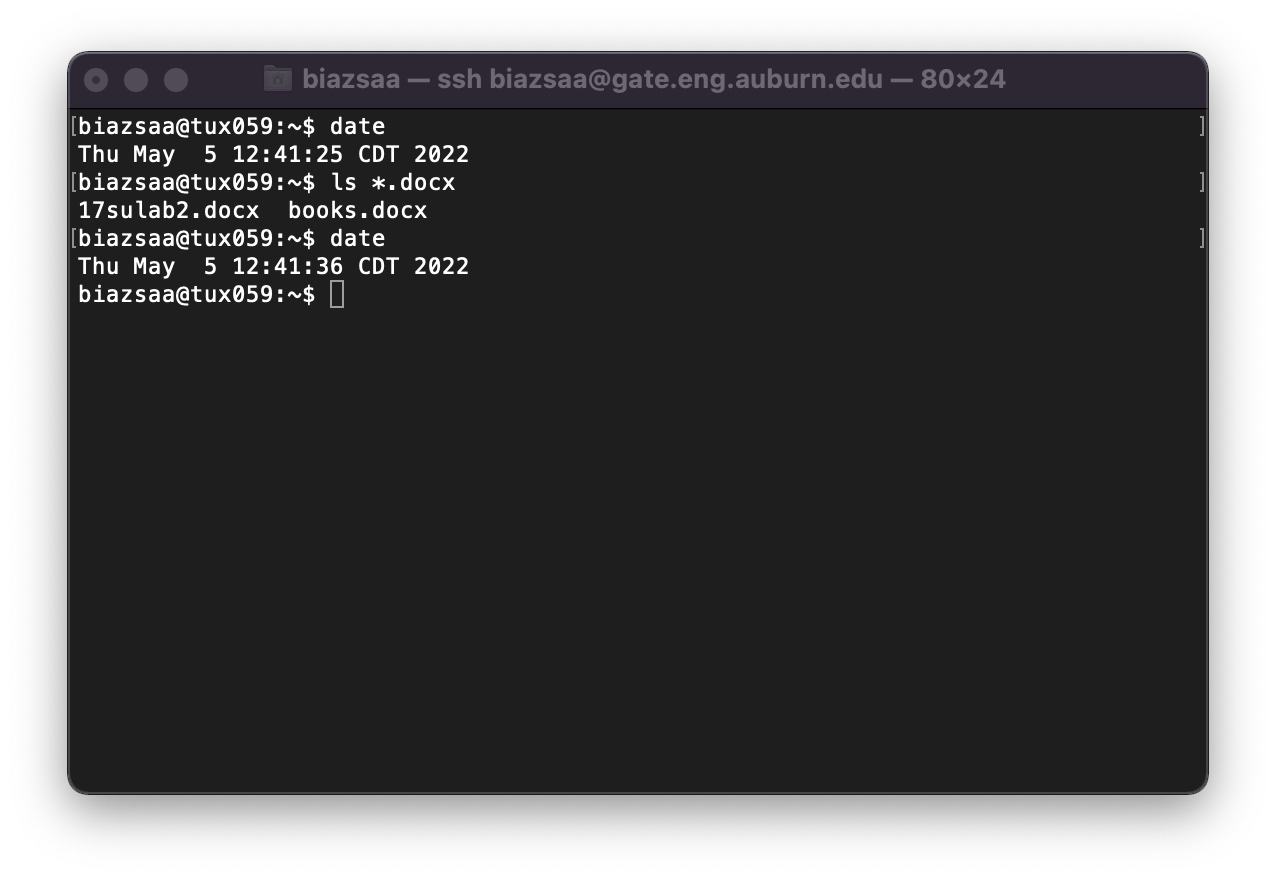
**Student 3** will get 25 + 15 + 60 = 100 points

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)
2. See Appendix About the REquired Information On All Screenshots [↑](#footnote-ref-2)