CS311 Operating Systems
Assignment 01: C Revisions
Due : 10h00am Monday 2nd Octobre, 2023.

FCSE
GIKI

Fall 2023

1 Objectives

(a) Test C programming file IO.

(b) Test C programming dynamic memory usage.

(c) Test ability to write clean readable code.

(d) Test ability do error handling in code.

(e) Test ability to follow written instructions.

You will be using the C programming language to do this assignment. You will need to use C functions related to string manipulation, file i/o, and memory management.

2 Tasks

2.1 Matrix Multiplication

Q1. Make a program that does matrix multiplication. [1]

You'll write a program that reads two matrices into the RAM and performs a multiplication to produce the product matrix:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1p} \\ c_{21} & c_{22} & \cdots & c_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mp} \end{bmatrix}$$

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in} + b_{nj} = \sum_{k=1}^{n} a_{ik}b_{kj}$$

(a) The input matrices will be stored in text files whose names will provided via command line arguments along with the output file name in which your program will store the product matrix. The command may be run like this:

[1] read about the working of `fopen()`, `fclose()`, `fscanf()`, `fprintf()`, `exit()`, `malloc()`, `free()`, `strlen()`, `strcpy()` and such families of functions. Read also about command line arguments. Use linux man pages.

```
prompt> ./matmul matfile1.txt matfile2.txt [outfile.txt]
```

The square brackets around the `outfile.txt` mean that this argument is optional.

(b) In case, the user fails to provide a name for either of the two input files, your program should display a helpful error message, and exit with the code -1. If the output file is given, your program should write the product matrix into that file and, when it's not, your program should write the product matrix to `stdout`.

(c) In case, an input file supplied by the user does not exist,your program should display a helpful error message, and exit with the code -2.

(d) If the dimensions of the input matrices are not favorable to multiplication, your program should output a helpful message and exit with the code -4.

(e) In case, your program fails to allocate required amount of memory, your program should display a helpful error message, and exit with the code -3.

(f) Upon any exit, your program should always de-allocate any memory it had allocated and close any files that it had opened.

2.1.1 Input file format

The input files will be text files and contain matrices; there format is:
2 3
9 6 -4
3 7 8

The first line will contain two positive integers representing the number of rows (R) and columns (C) of the matrix.

The next R rows will contain C integers each. Each such row represents one row of the matrix and the integers are the values at each column position for that row.

In the above example, the first row tells us that our matrix has 2 rows and 3 columns. The next two lines define the matrix whose first row contains the integers 9, 6, -4, and the second row comprises integers 3, 7, and 8.

You can assume that the input files will always have the data in correct format.

2.1.2 Output file format

In case the output filename is given, you will write your matrix in the output file using the same format as that of the input files. In case it's not, you'll display the matrix on `stdout` using the same format as that of the input files.

2.2 lcat

Q2: Make a command that concatenates the respective lines of the files given as its command line arguments.

The bash command `cat` takes multiple filenames as input and concatenates their content one after the other and displays it on `stdout`. [2]

Here you'll make a command called `lcat` which can take multiple filenames as command line arguments and it would display the lines of all these files concatenated, i.e., the first line of the

[2]You can read up on it with `man cat`

output should be the concatenation of the first lines of all the files, the second line of the output should be the concatenation of the second lines all the files, ..., and so on.

The number of lines will of course not be the same in all the input files.In case a a file ends early, the program should skip this file and concatenate the lines for the rest of the files at that position. Assuming `file1.txt`, `file2.txt`, and `file3.txt` have the contents below respectively:

The quick and the lazy dog jump.	Brown fox running fast.	Is rolling really dolling? when you lolling.

Running the command may be run like this:

```
prompt> ./lcat file1.txt file2.txt file3.txt
```

This should produce:

The quick Brown fox Is rolling
and the lazy dog running fast really dolling?
jump. when you lolling.

Your program should follow the instructions about proper coding and error checking and resource freeeing mentioned in this document.

3 Error Checking and Clean Code instructions

When working with Linux System calls, it is extremely important that you always check the return values of the system calls to verify whether the call was successful or not. The function documentation would usually contain all the information you will need. You should always read the function documentation.

Your programs should guard against invalid user input and in case of an invalid input, it should print a helpful error message.

You would lose marks if your program crashes during use.

It is the programmer's responsibility to free any system resources i.e. memory , file descriptors, etc., they have acquired from the system. Your program should always free system resources once it is done using them.

Code should be properly indented, readable and commented.

You should always your compile your code using the `-Wall` option to check for warnings.

When displaying their valid output on the screen your programs should write to `stdout` and when displaying error messages, they should write to `stderr`.

4 Submission Instructions

1. Submission will be on MS Teams.

2. You submission should consist of two .c files only. Do not compress.

3. You can name your submission as u2021xxx_a1q1.c and u2021xxx_a1q2.c where u2021xxx is your registration number.

4. Missing submission deadline on MS Teams will cost you 40 marks. Submissions received more than 24 hours after submission deadline will get a 0.

5 Rubric

This is an individual assignment. Any form of collaboration, cheating, plagiarism will get you a 0. Giving your code to somebody else, even if it is for their understanding only, is not allowed.

You may be called for a viva; if you are unable to explain **any line** of your submitted code, you'll get a 0 even if it's working perfectly (we live in the age of chatgpt!).

Category	Marks
Q1 working properly	50 marks
Q2 working properly	50 marks
Missing deadline	-40 marks
Not following instructions	upto -60 marks
Max marks	100 marks