IIT PALAKKAD

Indian Institute of Technology Palakkad
Department of Computer Science and Engineering
Operating Systems - Jul to Nov 2020
01 September 2020

Instructions:

- *Read the entire question, brain-storm with your team-mate and design the overall solution (break the solution to smaller sub-problems), and implement individual components of the solution separately (each team-member implements one or more components).*
- *Upload your submission on Moodle as a ROLLNO1_ROLLNO2.tar.gz*
- *Each student will be scored separately (individually) in the viva-voce.*
- *If you are borrowing code from other sources (internet or friends), acknowledge and give credit to the original author of the code.*

1. Write a simple image processing application in C to the following specifications:

    a. The input to the application should be a *ppm* image file. There are many free utilities to convert from *jpeg* images to *ppm* ones. Use any one of them for this.

    b. Your application must read this file and store the pixel information in a matrix.

    c. It must perform two transformations to the image, one after the other. Choose some simple transformations such as “RGB to grayscale”, “edge detection”, “image blur”, etc. These transformations should use matrix multiplication operation mandatorily. Let us say these two transformations are ***T1*** and ***T2***

    d. The resultant pixel matrix should be written to a new *ppm* file. If both the transformations are run, only one output file should be created.

    e. Organize your code into multiple files and folders so that it is logical and intuitive.

    f. You should create at least one library named ***matrixmath*** that has function definitions of all matrix operations used by your application. Additionally, it could also have function definitions for other matrix operations, but should not have functions that are not related to operations on matrices.

    g. The *matrixmath* library should have at least one function with the following prototype:

```
int multiply(matrix P, matrix A, matrix B);
returns:
```

```
        0 (zero)   - on success
        non-zero   - on failure
    where:
        P     - pointer to product matrix
        A, B  - pointers to input matrices
```

h. ***Hint:*** You will have to define a structure that completely defines a matrix (dimensions and values), and `typedef` it appropriately so that it matches the above function prototype. An example of the same is as follows:

```
struct matrix_t {
    uint32 m; uint32 n; float **A;
};
```

This example is only indicative. Your struct could be different.

2. Setup a build system for your application using the *make* utility with the following targets. Use Makefile variables, text-substitutions, pattern-matching, implicit rules, etc as much as possible.

    a. `build` - compile your application and produce an executable

    b. `run` - runs your application. Your application can assume default input and output files such as *input.ppm* and *output.ppm* located at default file-paths.

    c. `prepare` - takes input from user on input and output file names (paths). This followed by the target `run` should allow a user to run your application on an input file of his choice and produce a transformed image in a file of his choice. Additionally, this target can be used to take an image in a different format such as jpeg, and generate its corresponding ppm file.

    d. `tests` - runs all test cases on your application

    e. `test1/test2/test3/...` - runs only test case 1/2/3/... on your application

    f. `T1/T2` - Runs only transformation 1 or 2 on the input file

    g. `liba` - generates a statically-linked library (archive file)

    h. `libso` - generates a dynamically-linked library (shared object)

    i. `clean` - deletes all temporary files and folders (only header and source files should be retained)

3. Disassemble your final executable using objdump (-d option). Using your application as an example, explain the differences you notice in the executable when you use dynamically-linked libraries vs statically-linked libraries. Submit one document (strictly not more than 1000 words) per group.