Programming project#0 – IT383 Spring 2016

January 14 2016

Due: Jan 22 2016 at 11:55PM

Total points: 40 programming project points

Programs containing compilation errors will receive failing grades. Those containing run-time errors will incur a substantial penalty. You should make a serious effort to complete all programs on time. You will lose 10% per day (up to 3 days) if the submission is made after the deadline. Programming assignments are individual. You should complete them with your own effort. If you need help, you should come to my office hours or contact me by email. You may discuss concepts with others in the class, but not specific program code**.** The total points of this programming project are 30 points. However, the 100 points are NOT same as 100 points in exams, quizzes, or quizzes. **If you are not sure about the grading policy, refer to the course syllabus.**

NOTE: Each program should be placed in its own subdirectory. Also, you need to include a Makefile in the same directory as the source code to allow the instructor to compile your program automatically. You will submit a single "ZIP" file that includes all the subdirectories where your programs are located. The name of ZIP file should be program0_*YourLastName_YourFirstName*.zip. You will need to submit your SINGLE zip file to ReggieNet. (NOTE: You will automatically lose 5 points if you do not follow the submission instruction correctly.)

- You might want to use the following Linux command to create a single zip file:

```
% zip –r program0_yourlastname.zip ./1 ./2 ./3
```

[1] (10 points) C program for File I/O - version 1

Write a C program called "my_copy.c". This program makes a duplicate copy of a given *text* or *binary* file. An example usage of the program is as follows:

```
$ my_copy source_file destination_file
```

Requirements:

1) You need to use ALL of the following C APIs (all of them) in your program to read and write data from/into files.

fopen(), fclose(), fread(), fwrite()

DO NOT use fgets() or fputs() in this program since your program should be able to handle both binary and text files.

2) Your program should check the error codes returned by calls to fopen/fread/fwrite functions.

3) The size of the buffer (i.e., array) for file input and output operations should not be larger than 10240 bytes. In your program, you will need to use a loop where:

   a) a block of data is read from the source file to the buffer(i.e., array)
   b) the block of data in the buffer is written into the destination file

Please note that the size of a source file can be very large (as large as hundreds Mega bytes). Therefore, you will need to use a loop

4) You might want to try the following sample test cases. Note that "diff" is a Linux command to check if two files are exactly the same in terms of size and contents.

```
% my_copy  /etc/passwd ./passwd
% diff /etc/passwd ./passwd

% my_copy /bin/ls   ./ls
 % diff /bin/ls   ./ls
```

Note: Create a subdirectory called "1". Change to the subdirectory. "my_copy.c" and Makefile should be placed under the subdirectory "1".

[2] **(10 points) C program for File I/O – version 2**

The description of the second program is basically same as the first one except that you need to use different set of C APIs as presented in the following:

Requirements:

1) You need to use ALL of the following C APIs (all of them) in your program to read and write data from/into files.

open(), close(), read(), write()

2) Your program should check the error codes returned by calls to open/read/write functions.

3) The size of the buffer (i.e., array) for file input and output operations should not be larger than 10240 bytes. In your program, you will need to use a loop where:

   a) a block of data is read from the source file to the buffer(i.e., array)
   b) the block of data in the buffer is written into the destination file

Please note that the size of a source file can be very large (as large as hundreds Mega bytes). Therefore, you will need to use a loop

4) You might want to try the following sample test cases. Note that "diff" is a Linux command to check if two files are exactly the same in terms of size and contents.

```
% my_copy  /etc/passwd ./passwd
% diff /etc/passwd ./passwd

% my_copy /bin/ls   ./ls
% diff /bin/ls   ./ls
```

Note: Create a subdirectory called “2”. Change to the subdirectory. “my_copy.c” and Makefile should be placed under the subdirectory “2”.

[3] (20 points) [Pointers and Arrays] C program for Stack

In this problem, you will implement a basic data structures using array: stack. You can assume the elements are of type ***int***. A stack is usually called last in first out (LIFO) list. It supports two important operations, push and pop. Here's the list of all operations that you need to implement.

- void stack_init(stack *s, int capacity)
  Initializes the stack that stores integers with maximum size capacity. (Hint: you will need to call malloc(sizeof(int)*capacity) inside stack_init())
- int stack_size(stack *s)
  Returns the size of the stack, i.e., the number of elements in the array.
- int stack_pop(stack *s)
  Returns the integer element on top of the stack. If the stack is empty, the return value is undefined.
- void stack_push(stack *s, int e)
  If the stack is not full, push the item on top of the stack. Otherwise, do nothing.
- void stack_deallocate(stack *s)
  Frees this stack.

The push and pop operations of stack only operates on one end of the array. Figure 1 shows an example.

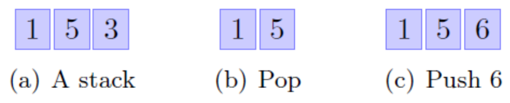

Figure 1: Illustration of Stack Operations

Create header “dslib.h", which contains declarations of stack, and implement them in “stack.c”. Test your program in “test.c", where the main function is. Also, create a “Makefile" to manage your project. For your information, you will need to use *malloc(..)* API in this program.

Note: Create a subdirectory called “3”. Change to the subdirectory. “dslib.h”, “stack.c”, “test.c” and Makefile should be saved under the subdirectory “3”.

Hint: You will be able to get information on C language by reading the lecture slides on the following website: http://www.cs.cornell.edu/courses/cs2022/2011sp/