CSC 3320: System-Level Programming — Week of 11/18/2024

Lab 12

© Instructor: Dr. Md. Mahfuzur Rahman — **Lab Quiz: 30 Points, Problem Solving: 70 Points**

Objectives:

Today we will be covering the following topics:

1. Practice `File Management` in a C program.
2. Practice "system calls" (`read()`, `write()`) in a C program.
3. Practice passing "command-line arguments" when you run a C program.

Instructions:

- Attendance is mandatory.
- Labs must be completed individually.
- If you have any questions, please do not hesitate to ask TA.
- Follow submission instructions in the deliverable section.
- There will be a lab quiz of 30 points arranged by Lab TA.
- Visit the broader grading criteria after the deliverable section. (last page)
- Lab assignments are due at midnight on the day of your lab (i.e., by 11:59 PM)

1. Write a program to create a file named `DATA.txt` to store decimal integers (32 to 126). These numbers are valid ASCII codes for printable characters. That is, the `DATA.txt` file will store `32, 33, 34, ... 126`. Next, open the `DATA.txt` file you created in `read-only` mode and read the integers using system call `read()`. Remember, while reading integers as raw bytes using `read()` system call, read as characters (one character at a time) and build an integer based on delimiters (newline or spaces) (Hint: only valid integers may contain characters 0 to 9). Write the two versions of the integer you formed: 1) print the integer directly using `write()` system call (you will need to convert the number to its equivalent number string first) 2) Print the binary encoding of the number (side-by-side with spaces in between these two versions) on standard output using the same system call `write()`. Use command line argument to specify the file name.

   You might find the following functions useful for this lab:

   ```
   ssize_t bytes_read = read(fd, buffer, 1); // to read a single raw byte from a file (fd)
   num = atoi(number_buffer); // to form a number from digits (saved as characters in number_buffer)
   int length = snprintf(num_str, sizeof(num_str), "%d", num); // for num to string conversion
   write(STDOUT_FILENO, num_str, length) // To write the number as string saved in num_str
   write(STDOUT_FILENO, &num, sizeof(num)) // To write the raw binary-encoded value of num
   ```

   Use the `vi` editor to create your program and save it as `lab12.c`. After compilation, you must run your program like the following:

   `./prog DATA.txt` (you are not allowed to store the filename `DATA.txt` in your code)

   Your program's output on the terminal may look like the following:

   ```
   32
   33 !
   34 "
   ```

```
35 #
36 $
37 %
.. .
.. .
.. .
```

2. Now, make sure you completed the following tasks:
   (a) (05 points) Make sure you pass your filename `DATA.txt` as a command-line argument.
   (b) (10 points) Make sure you created `DATA.txt` file and store integers `32, 33, ..., 126` using standard C library functions (e.g. `fopen(), fprintf()` etc.).
   (c) (05 points) Make sure you reopen the the file `DATA.txt` and read the file using `read()` system call.
   (d) (10 points) Make sure you were able to read character by character and could make a whole integer before printing to the terminal. (Hint: When you know you got your integer, null-terminate your number string).
   (e) (10 points) Make sure you are printing the integer as a string using `write()` system call.
   (f) (10 points) Make sure you are printing the integer as a number (binary encoded representation) using `write()` system call.
   (g) (05 points) Make sure you explained your code to the TA or give enough documentation in your submission.
   (h) (03 points) Start recording your session using the `script` utility.
   (i) (03 points) Show the contents of lab12.c using the `cat` command.
   (j) (03 points) Compile lab12.c with required flags for the object file name `[use -o]` and C version `[-std=c99]`.
   (k) (03 points) Run your program using appropriate command.
   (l) (03 points) Finish your recording (use the `exit` command).

Deliverables
For today's lab, clean the text file (.txt) you recorded during your terminal session, if there are unwanted control characters. In other words, make it as you observed during your terminal session. Please name your text file as **lastname_firstname_lab12.txt**. You will need to submit the text file (terminal session record) and your C file (lab12.c) to the **Lab 12** dropbox in iCollege.

Broader Grading Criteria

- If no C (`.c`) file is submitted (regardless if `.txt` file submitted or not), a student will receive only 40% for attendance. Submission will not be graded.
- If C file is given but no `.txt` file (terminal session) is given, a submission will receive maximum 70% (will vary between 40% to 70% based on the correctness of the C program).
- If a `.txt` file is given along with the `.c` file, but the `.txt` file is not clean and not comprehensible to the TA, a submission will receive maximum 80% (will vary between 40% to 80% based on the correctness of the C program).
- If both clean `.txt` file and the `.c` file are given, your submission will be normally evaluated based on the tasks and the corresponding point distributions.
- Screenshots will not satisfy the requirements for code and/or the .txt files submission.
- There should be compatibility between lab quiz performance and problem-solving (programming) performance. Otherwise, you may be called for an interview with Lab TA.