.

Assignment 4 – XD Report Template

Airi Kokuryo

CSE 13S – Winter 24

Purpose

The purpose of this report is to understand how to read files using the C programming, as well as change them into different forms such as binary, hexadecimal, or decimals. This will be the explanation to creating the program which will read from other files and write them out using buffers of 16 bytes long. It will include the psudo code for the program, as well as the explanation on the functions it will be using.

Questions

Please answer the following questions before you start coding. They will help guide you through the assignment. To make the grader's life easier, please do not remove the questions, and simply put your answers below the text of each question.

- What is a buffer (talk about the data type and the purpose)? Why use one?

  Buffer is a temporary storage for input or output variables. This is used for increasing performance by allowing synchronous operations and not needing to access a physical disk subsystems.

- What is the return value of `read()`? What are the inputs?

  The return value of read() is the number of bytes read, and will return -1 if it is an error. The inputs are the file descriptor from which to read data, buffer where the data will be stored, and the number of bytes to read.

- What is a file no. ? What are the file numbers of `stdin`, `stdout`, and `stderr`?

  File no. is a non-negative integer that represents an open file in a process. It is used to perform I/O operations. The file numbers of stdin is 0, stdout is 1, and stderr is 2.

- What are the cases in which `read(0,buffer,16)` will return 16? When will it *not* return 16?

  It will successfully return 16 when there are at least 16 bytes of data available in the input stream. It will not return 16 when the input is fewer than 16 bytes.

- Give at least 2 (very different) cases in which a file can not be read all at once

1. One example is when the file size is larger than the available memory.
2. Another example is when the using a network stream or reading data from a remote resource where there may be limitations in amount of data to be fetched in a single request.

- What is the range of char? A byte? The ASCII table? Which range should your program accept (if any of those)

  The range of a signed char is from 0 to 255. However, if it is an unsigned char, it would be -128

to 127. The size of a byte can vary between different systems, but typically it is 8 bits. The range of an ASCII table is from 0 to 127 since it uses 7 bits to represent each character.

- What is the decimal equivalent of the 8 bit integer 0b1001 0110? [1]

  The decimal of 0b1001 0110 would be 150 if it is unsigned, and -105 if it is signed. There is only positive numbers for unsigned, however for signed binary, you will know it is a positive or a negative by checking the most left number. If it is 0, it would be positive, and if it is 1, it would be negative.

- Convert the 8 bit integer above to a 32 bit integer. [2]

  It would be 11111111 11111111 11111111 10010110 for a 32 bit integer.

- What does the `%X` format specifier mean? What type of data does it expect? [3]

  The %X is a format specifier to format and interpret hexadecimal values for printf or scanf. It expects the hexadecimal data type.

Testing

List what you will do to test your code. Make sure this is comprehensive. [4] Be sure to test inputs with delays and a wide range of files/characters.

It will be for testing the following cases:

- The program will keep on reading the file even though there is an delay
- The program will check if the outputs are the same as expected

How to Use the Program

Audience: Write this section for the user of your program. You are answering the basic question, "How do I use this thing?". Don't copy the assignment exactly; explain this in your own words. This section will be longer for a more complicated program and shorter for a less complicated program. You should show how to compile and run your program. You should also describe any optional flags or inputs that your program uses, and what they do.

To show "code font" text within a paragraph, you can use `\lstinline{}`, which will look like this: `text`.

For a code block, use `\begin{lstlisting}` and `\end{lstlisting}`, which will look like this:

```
Here is some code in lstlisting.
```

And if you want a box around the code text, then use `\begin{lstlisting}[frame=single]` and `\end{lstlisting}`

which will look like this:

```
Here is some framed code (lstlisting) text.
```

Want to make a footnote? Here's how.[5]

Do you need to cite a reference? You do that by putting the reference in the file `bibtex.bib`, and then you cite your reference like this[1][2][3].

[1] is it positive or negative? how do you know?

[2] remember that negative numbers in 2s compliment convert things differently from positive numbers

[3] it is not the same as the `%x` format specifier

[4] This question is a whole lot more vague than it has been the last few assignments. Continue to answer it with the same level of detail and thought.

[5] This is my footnote.

Program Design

Audience: Write this section for someone who will maintain your program. In industry you maintain your own programs, and so your audience could be future you! List the main data structures and the main algorithms. You are answering the basic question, "How is this thing organized so that I can have a chance of fixing it?". This section will be longer for a more complicated program and shorter for a less complicated program.

Pseudocode

Give the reader a top down description of your code! How will you break it down? What features will your code have? How will you implement each function.

- Open connection with the files
- Read files
- Printout in the given format using the data read from the file
- close the connection with files and free the memories used

Function Descriptions

For each function in your program, you will need to explain your thought process. This means doing the following

- The inputs of every function (even if it's not a parameter)
- The outputs of every function (even if it's not the return value)
- The purpose of each function, a brief description about a sentence long.
- For more complicated functions, include pseudocode that describes how the function works
- For more complicated functions, also include a description of your decision making process; why you chose to use any data structures or control flows that you did.

Do not simply use your code to describe this. This section should be readable to a person with little to no code knowledge. **DO NOT JUST PUT THE FUNCTION SIGNATURES HERE. MORE EXPLANATION IS REQUIRED.**

1. write_char(char* text, int index)

- Input: char* text, int index
- Output: void type
- Purpose of function: to write out the given data 16 bytes at a time in a given format. It will be using the char pointer called "text" to get the temporary data.

2. clean_char(char* text)

- Input: char* text
- Output: void type
- Purpose: The purpose of this function is to clear the data inside the given char pointer by putting a null inside.

3. read_file(int file_descriptor, char* text)

- Input: int file_descriptor, char* text

- Output: void
- Purpose of function: This function will read from the file using the given file_descriptor, and put the data inside the char pointer called "text" to be written out in the given format.
- Pseudocode: It will input each char inside the text string until it is 16 bytes. Once it becomes 16 bytes or encounters an EOF, it will run the write_char function to use that data to write out the given data in a specific format. It will then run the function clear_char to clear the data inside the text string, and start the loop again to input the next data inside the string.

Optimizations

This section is optional, but is required if you do the extra credit. It due `only` on your final design. You do not need it on your initial.

In what way did you make your code shorter. List everything you did! I made my code shorter by using the %X instead of creating a new function for changing decimal to hexadecimal. Also, I used a buffer that only lives inside the read_file function to temporarily get all the data, while using the char pointer which has a defined length using malloc, so the strng does not have to be witten out using a forloop which you would have to when using an array. This was useful because you only have to give the pointer to the function.

References

[1] Wikipedia contributors. C (programming language) — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/C_(programming_language), 2023. [Online; accessed 20-April-2023].

[2] Robert Mecklenburg. *Managing Projects with GNU Make, 3rd ed.* O'Reilly, Cambridge, Mass., 2005.

[3] Walter R. Tschinkel. Just scoring points. *The Chronicle of Higher Education*, 53(32):B13, 2007.