Assignment 3 – XD Report Template

Jason Maheru

CSE 13S – Winter 24

Purpose

Audience for this section: Pretend that you are working in industry, and write this paragraph for your boss. You are answering the basic question, "What does this thing do?". This section can be short. A single paragraph is okay.

- The purpose of this program is to be able to display binary files into a hexadecimal representation. The reason is so that you can see everything in a txt file because sometimes these files have invisible differences.

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? Why use one?
- What is the return value of `read()`? What are the inputs?
- What is a file no. ? What are the file numbers of `stdin`, `stdout`, and `stderr`?
- What are the cases in which `read(0,16)` will return 16? When will it *not* return 16?
- Give at least 2 (very differnt) cases in which a file can not be read all at once

1. A buffer is a memory where data is temporarily stored while it is being moved from one place to another.
2. The return value of read() is a ssizet value. There are three inputs. The first is "int fd" which is the file descriptor returned from the open function. The second is "void *buf", which is a pointer with data read that will be stored. The third is "sizet n" which determines the maximum number of bytes read.
3. The file number is also known as the file descriptor and it is used to identify or access a file within a process. `stdin` = 0, `stdout` = 1, and `stderr` = 2.
4. It will return 16 whenever there are 16 bytes of data to read. It will not return 16 when there are less than 16 bytes of data. Read will read up to as many available bytes and return the number of bytes read of 16.
5. One case is when the file is very large and is not readable in one go so it would have to be read multiple times for it to get all the data from the file. Another case could be that the machine might not be able to handle all the stuff in the file so you run a loop.

Testing

List what you will do to test your code. Make sure this is comprehensive. [1] Be sure to test inputs with delays.

- One way I will be testing the program out is by checking that the output of my file is the same as the output of the actual built-in program.
- I will also be checking if no input is provided.
- Another test will be checking that only 16 bytes of data are being read at a time.

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.[2]

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].

- This program is very straightforward to use. All you do is run "make" to make sure the program is ready to run. Then you run the program by using the command `./xxd <filename>`.

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.

```
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
```

[1] 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.

[2] This is my footnote.

```
#define BUFFER_SIZE 16

void print_hex

void print_ascii

int main (int argc, char **argv) {
    need to int fd to stdin

    check valid number of arguments are entered if not then return non zero error

    set buffer array size
    set bytes read
    set offset to 0

    run while loop for ((bytes = read(fd, buffer, BUFFER_SIZE)) > 0)
        print index of first byte
        run the hex value function for 16 byte buffer
        run the asciii function to print the ascii of the buffer
        set offset to the bytes read to keep track

    close the file

    return 0

create a function for printing the hexidecimals values (const unsigned char *buffer, ssize_t)
    iterate through the file to convert what is in the buffer array to hexidecimal
    make sure the values are together

    iterate for spaces to print spaces

now do the same thing for printing ascii
    print . for \n

    print \n
```

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. **MAIN FUNCTION**: It reads data from a file (if provided as a command-line argument) or from standard input. Prints the data in both hexadecimal and ASCII formats, showing 16 bytes per line. After reading and printing all data, it closes the file (if opened) and exits.

2. **PRINTHEX**: It prints the hexadecimal representation of a given buffer of data and each byte in the buffer is printed as a two-digit hexadecimal number. Lastly, spaces are added for formatting, and extra spaces are added if the buffer size is less than 16 bytes.

3. **PRINTASCII**: Prints the ASCII representation of a given buffer of data. Printable ASCII characters are printed directly and nonprintable characters are represented by dots.

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!

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.

@miscnoauthor_undated-zu:2005a, author = "Learn Linux", title = "stdin, stdout, stderr", year = "2005", howpublished = "`https://www.learnlinux.org.za/courses/build/shell-scripting/ch01s04.html`", url = "https://www.learnlinux.org.za/courses/build/shell-scripting/", note = "Accessed: 2024-2-6"