Assignment 7 – XXD

Sebastian Avila

CSE 13S – Fall 2023

Purpose

The purpose of this program is to make a simple version of the `xxd` command. This program takes input and prints out the index value of the first byte, the hexadecimal values of each character, and the input read. It does this with a buffer size of 16 bytes.

How to Use the Program

To use this program you will need to have two files in the same directory: `xd.c` and `Makefile`. With these files in the same directory, use command `make clean` to remove any preexisting object files. Then, use command `make format` to format all the header and source files. Then use command `make` to compile the program. You may also have an input file in the directory or in a sub-folder. Then run the program using `./xd -i inputfile` or `./xd`. The latter uses `stdin` as input. The program only either accepts one command line argument or none. If more than one is given, the program terminates. The program uses several optional compiler flags:

- `-Wall`: This flag enables all warning messages.
- `-Werror`: This flag turns all warnings into errors.
- `-Wextra`: This flag enables extra warning flags that are not enabled by `-Wall`.
- `-Wstrict-prototypes`: This flag warns if a function is declared or defined without specifying the argument types.
- `-pedantic`: This flag issues all the warnings demanded by strict ISO C and ISO C++.
- `-lm`: This flag links the math.h library. This allows the program to access and use the functions from the math.h library.

Program Design

The program begins by checking the command line arguments. If their are more than 2 (the program executable and an input file), then the program exits with a nonzero value. Otherwise, it creates a integer variable to hold the input file descriptor value. It sets this value to `stdin`. It then checks if the number of arguments is equal to 2; if it is, it attempts to open a file with the name of the second argument. If the file does not open successfully, the program terminates. If the number of command line arguments is not 2, then `stdin` remains as the input file. The program then initializes an empty character string of length 17 for the buffer. It also initializes an integer value to hold the current index of the first byte. It then reads from the input file and store the number of bytes read in a variable `res`. It then begins a loop which continues until exited. In this loop, it first checks if 0 bytes were read. If this is the case, the program terminates. Otherwise, it checks if 16 bytes were read. If 16 bytes were read, the program first prints the index as a hex value, then it prints the hex values of the characters in the buffer using a function, then it prints the buffer. It then sets all the bytes in the buffer string to the null character, sets the number of bytes read

to 0 and increases the index by 16. It then checks whether `res` is equal to -1. This means an error occurs while reading input. If this happens, the program terminates. Otherwise, it reads input again and stores the number of bytes into a variable `temp`. If 0 bytes are read, the program breaks out of the loop. If an error occurs while reading, the program terminates. Otherwise, it adds the number of bytes read in `temp` to `res`. It then restarts the loop. Once the loop ends, the program checks if the number of bytes read is greater than 0. If it is, it prints the index as a hex value, then it prints the hex values of the characters in the buffer using a function, then it prints the buffer. It then returns 0 and terminates.

Function Descriptions

void print_hex(char *buffer, int len)

This function takes in two parameters: a pointer to a character string `buffer` and an integer `len`. It does not return anything. Its purpose is to print each of the characters as their corresponding hex value with two digits. The values are printed in groups of 4 digits with a space in between. If the buffer does not contain 16 bytes, spaces are printed instead of the hex value for any empty bytes.

```
void print_hex(char *buffer, int len)
    for i, 1 to i <= 16
        if i > len then
            print " "
        else
            print hex value with 2 digits

        if i % 2 = 0 then
            print " "

    print " "
```

Psuedocode

```
buffered reading
    char buffer[17] = ""
    res = read(fdin, &buffer, 16)

    while true
        if res = 0 then
            return 0

        if res == 16 then
            print program output
            set all bytes in buffer to "\0"

        if res = -1 then
            return 1

        temp = read(fdin, buffer + res, 16-res);
        if temp = 0 then
            break
        if temp == -1 then
            return 0

        res = res + temp

    if res > 0
        print program output
```

```
main
    if argc > 2 then
        return 1

    fdin = 0;

    if argc == 2 then
        fdin = open(argv[1], O_RDONLY, 0)
        if fdin = -1 then
            return 1

    char buffer[17] = ""
    res = read(fdin, &buffer, 16)
    index = 0

    while true
        if res = 0 then
            return 0

        if res == 16 then
            print index as hex
            print_hex(buffer,16)
            for i, 0 to i < 16
                if 32 > buffer[i] > 126 then
                    buffer[i] = "."
            print buffer
            set all bytes in buffer to "\0"
            index = index + 16

        if res = -1 then
            return 1

        temp = read(fdin, buffer + res, 16-res);
        if temp = 0 then
            break
        if temp == -1 then
            return 0

        res = res + temp

    if res > 0
        print index as hex
        print_hex(buffer,16)
        for i, 0 to i < 16
            if 32 > buffer[i] > 126 then
                buffer[i] = "."
        print buffer

    return 0
```

Error Handling

- Too many command line arguments : If too many command line arguments are given, the program will be terminated with a nonzero exit code.
- Invalid input file: If the program cannot open the input file, the program will be terminated with a nonzero exit code.

- Errors will reading program : If the program encounters any errors while reading input, the program will be terminated with a nonzero exit code.

Results

References