Assignment 0 Design

Prepared by: Miranda Eng & Shirley Phuong

CSE130: Principles of Computer Systems Design

October 19, 2020

Part 1: Main Function

The main function takes in the number of arguments and a String array of the file names if given. It contains a for loop to go through all of the files listed and copy the data from the files to standard output. If no files are given or if a file name is given as "-", then the program will use standard input to write to standard output.

Check if there are any files listed. If there aren't, then write STDIN to STDOUT.

```
if (argc < 2)
     print(STDIN, STDOUT, buffer)
```

Use a for loop to go through all files listed on the command line for the rest of the main function.

Check if the name given is "-"

```
if (file is "-")
     print(STDIN, STDOUT, buffer)
     go back to beginning of for loop
```

This will read the standard input and repeat it to standard output.

Open the file and check if there are any errors.

```
if (open file gives an error)
     give warning
     go back to beginning of for loop
```

If there are no errors, call the print function to read the file and write to standard output.

```
print(file descriptor, STDOUT, buffer)
```

Part 2: Print Function

The print function takes in two integer variables that signify where we are reading from and where we are writing to. It also takes in a pointer to the char buffer array.

A char array is used as a buffer for reading/writing. It is set with 30KiB of memory.

```
char buffer[BUFFER_SIZE];
```

Use a while loop to read/write one byte at a time.

```
while (read)
     write to STDOUT
```

Re-initialize and flush the buffer.

```
memset(buffer, 0, BUFFER_SIZE)
```

Assignment Question

How does the code for handling a file differ from that for handling standard input? Describe how this complexity of handling two types of inputs can be solved by one of the four ways of managing complexity described in Section 1.3.

When the code is handling a file, we have to first open the file before we can read and write to standard output. This helps to define a file descriptor for our file. The main difference is when we are reading the file or the standard input. The first argument in the read function for files is the integer file descriptor. For standard input, we use the preprocessor symbol STDIN_FILENO. In order to handle these two types of inputs, the code uses abstraction, by the robustness principle. The robustness principle: be tolerant of inputs and strict on outputs. This means you can enter in two types of inputs, but the code will handle it because of the liberal design. If it is still able to interpret the values, the function will accept them. If not, a new file will be created. We reuse the code of printing out file contents since the same code pattern for printing is repeated. We abstract this and allow different in and out file description numbers. Also since the effect of the robustness principle is to suppress, we construct our outputs more strictly to reduce noise that may show up in the interface, thus displaying more elegant warnings and errors.