head.c

SWE decisions:

We must first find if the argument “-n” was given on the parameters.

If the parameter was given, we read the next parameter which should be a number. If the conversion between string and number was successful, we set the number of lines to be printed to be that number, otherwise we display an error message and exit the program.

If the parameter was not given, we display an error message because “-n” was provided without a number following it.

While reading the parameters if we read something that is not “-n” or the number following it we assume a file to read the information from. Otherwise, we read the information from standard input.

An assumption was made to only read the information from one file so if multiple files were provided, we only read the information from the last one.

After all parameters were read, we open the file using open with read only restriction or set the standard input as the file descriptor (fd).

Now we proceed to read and print the information while we haven’t passed the maximum number of lines to be printed or the fd hasn’t run out of lines.

We read from fd using a static buffer, if we don’t have any more on the file we exit the program, if read was unsuccessful we print an error message and exit the program, otherwise we proceed to print the line using write.

While writing the must iterate over the buffer to see if we have the new line character ‘\n’, if we do we make multiple printings from the same reading.

Another case is that if the buffer was fulfilled, we print everything that was read but don’t increase the number of lines printed because we haven’t finish with that line, so we pick it up were we left from the next reading.

Once we are done, we must close the file descriptor if we are reading from a file.

If closing was not successful, we print an error message.

Problems encounter:

The first problem encounter was to find if the parameter given was “-n”.

We solve it by creating a function that simulates strcmp. We pass the parameter read and “-n” to the function and if it returns 0, we know they are the same.

The second issue was to print an error message without using the fprintf function.

The way to solve was to create a function which concatenates two strings…

Another issue was that when reading from files we get multiple lines from one reading.

So, we decide to read the buffer until we hit a new line character or to the end of the buffer and return the number of bytes to print from the beginning to the number of bytes to finish the line or until the end of the buffer.

Another issue was that the buffer may sometimes get fill and cut a line in half, so we are unable to assume that the buffer always has entire line on it.

We proceeded to check if the buffer was fill and if the last character is not a new line character, we don’t increase the number of lines printed because the line would be cut in half.

Testing:

Command: ./head -n 2 nanpa

Output:

201200Jersey City NJ

201202Hackensack NJ

Command: ./head -n

Output:

head: option requires an argument -- n

usage: head [-n lines] [file ...]

Command: ./head -n 3

Output:

head: illegal line count -- -3