Joshua Dunne
Assignment 6

Abstract

We're back together with our old friend, the caesar cipher. We're doing the same thing as we did with bash, but this time in C. We want our program to read from a file, and, as specified by parameters, shift it n characters, reverse the direction if wanted, and number the lines if specified. The program should then output this to the command line, allowing us to pipe output if wanted.

Introduction

As per specifications, we're developing using our command line and command line utilities. Necessary for our undertaking then is an editor, for which I chose vim. We need gcc and relevant source installed, and for this build-essentials was installed using apt. Because I chose to separate the source, also included and used was make. Later we use ps, grep, and head to gather statistics and process information.

Summary of Results

First to demonstrate usage, let's undertake much the same process as we did with our bash script.We need to build it.

```
jdizzle@jdizzle-vbuntu:~/CS497/CS497/Assignment-6$ make all
cc    -c -o caesar_cipher.o caesar_cipher.c
gcc -g -c caesar_cipher.c -o caesar_cipher.o
cc    -c -o main.o main.c
gcc -g -c main.c -o main.o
gcc -g caesar_cipher.o main.o -o main
```

We then create a sample text file which we can run through the Caesar cipher to demonstrate correctness.

```
jdizzle@jdizzle-vbuntu:~/CS497/CS497/Assignment-6$ cat aFile
abcdefg
hijklmnop
qrstuvwxyz
these are some lines
I have in a file
when we run our program,
and then in reverse
we should return our original text
jdizzle@jdizzle-vbuntu:~/CS497/CS497/Assignment-6$
```

We're then going to run our command twice, the first time to shift characters in a positive direction, we do this by not including the *-r* flag when calling our command, we pipe this into a file. From that file we then run our command with the *-r* flag to shift in the negative direction.

```
jdizzle@jdizzle-vbuntu:~/CS497/CS497/Assignment-6$ ./main -s 3 aFile > anotherFile
jdizzle@jdizzle-vbuntu:~/CS497/CS497/Assignment-6$ ./main -r -s 3 anotherFile
abcdefg
hijklmnop
qrstuvwxyz
these are some lines
I have in a file
when we run our program,
and then in reverse
we should return our original text
```

As we were presented with the original text, we know that our program works both ways. To demonstrate that we can number lines as specified, we can include the *-n* flag when calling our program.

```
jdizzle@jdizzle-vbuntu:~/CS497/CS497/Assignment-6$ ./main -s 5 -n aFile
1 fghijkl
2 mnopqrstu
3 vwxyzabcde
4 ymjxj fwj xtrj qnsjx
5 N mfaj ns f knqj
6 bmjs bj wzs tzw uwtlwfr,
7 fsi ymjs ns wjajwxj
8 bj xmtzqi wjyzws tzw twnlnsfq yjcy
```

We were wanted to gather process information, so, let's do that now. To do so I am going to run two commands, the first will allow us to label the columns so that the second commands output makes more sense.

```
jdizzle@jdizzle-vbuntu:~/CS497/CS497/Assignment-6$ ps aux | head -1
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME (
jdizzle@jdizzle-vbuntu:~/CS497/CS497/Assignment-6$ ps aux | grep ma
jdizzle     9157  4.8  7.9 1192095556 319148 ?   Sl   20:42   0:54 ./
```

As we were asked for the processID, we can see that it correlates to be 9157, we used 4.8% of the CPU, 7.9% of available memory, also, because we've ran this several times, we've taken up a total of 54 seconds of run time.

Conclusion

As we've demonstrated, the fruits of our labor is a simple c program that reads from a file. Depending on parameters it then shifts alphabetic characters in either a positive direction, given a shift to do so. We also gathered and showed statistics of the process, such as its PID, its CPU usage, and its total run time.