

## Abstract

This assignment delves into the development of a Bash shell script and GCC to compile a C script on a Linux terminal. This is to help us with facilitating basic encryption, similar to that of assignment 5. It was also to help the user print out and identify key computer resources by outputting them alongside the output of the specific C script command.

## Introduction

Ubuntu will be used in a VirtualBox Virtual machine.

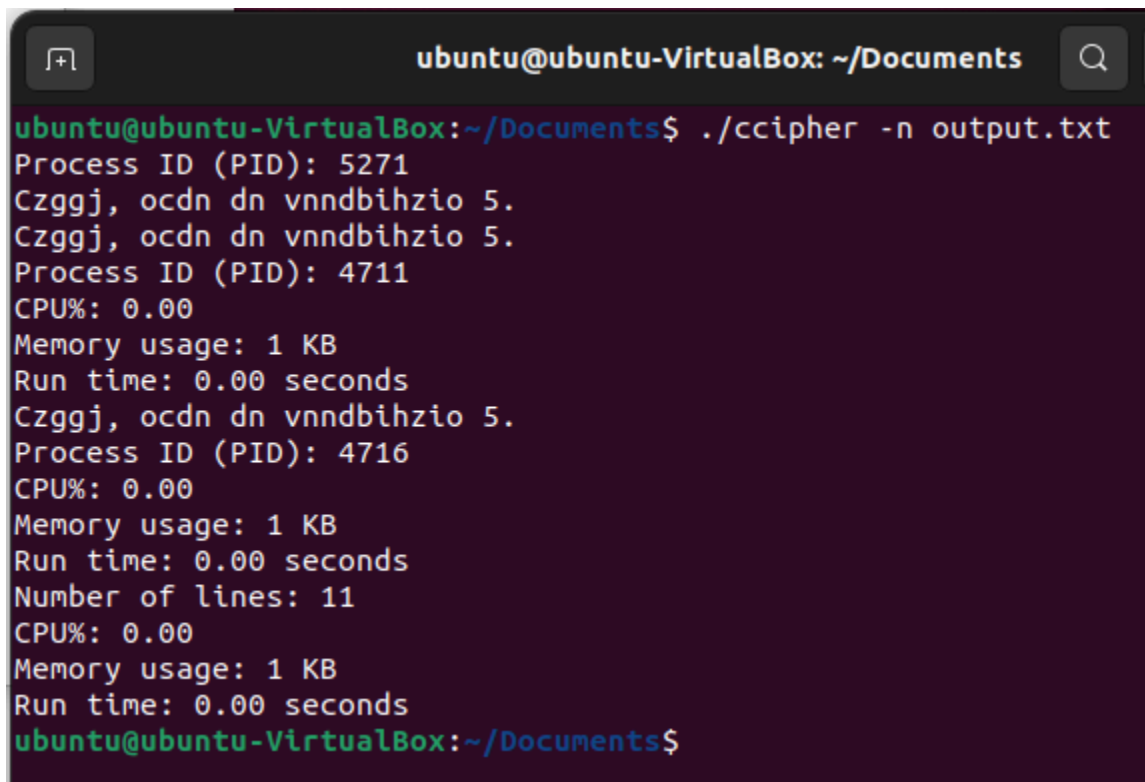
### Changing File permissions to run as an executable

```
chmod +x ccipher.c
```

### Running a file as a script

```
./ccipher
```

## Summary of Results

A screenshot of a terminal window titled 'ubuntu@ubuntu-VirtualBox: ~/Documents'. The terminal shows the execution of the command './ccipher -n output.txt'. The output of the script is as follows:

```
ubuntu@ubuntu-VirtualBox:~/Documents$ ./ccipher -n output.txt
Process ID (PID): 5271
Czggj, ocdn dn vnndbihzio 5.
Czggj, ocdn dn vnndbihzio 5.
Process ID (PID): 4711
CPU%: 0.00
Memory usage: 1 KB
Run time: 0.00 seconds
Czggj, ocdn dn vnndbihzio 5.
Process ID (PID): 4716
CPU%: 0.00
Memory usage: 1 KB
Run time: 0.00 seconds
Number of lines: 11
CPU%: 0.00
Memory usage: 1 KB
Run time: 0.00 seconds
ubuntu@ubuntu-VirtualBox:~/Documents$
```

```

Memory usage: 1 KB
Run time: 0.00 seconds
Number of lines: 11
CPU%: 0.00
Memory usage: 1 KB
Run time: 0.00 seconds
ubuntu@ubuntu-VirtualBox:~/Documents$ ./ccipher -s 5 input.txt
Process ID (PID): 5293
Mjqqt, ymnx nx fxxnlrjsy 5.
CPU%: 0.00
Memory usage: 1 KB
Run time: 0.00 seconds
ubuntu@ubuntu-VirtualBox:~/Documents$ ./ccipher -r -s 5 input.txt >> output.txt
ubuntu@ubuntu-VirtualBox:~/Documents$
1 Czggj, ocdn dn vnndbihzio 5.
2 Czggj, ocdn dn vnndbihzio 5.
3 Process ID (PID): 4711
4 CPU%: 0.00
5 Memory usage: 1 KB
6 Run time: 0.00 seconds
7 Czggj, ocdn dn vnndbihzio 5.
8 Process ID (PID): 4716
9 CPU%: 0.00
10 Memory usage: 1 KB
11 Run time: 0.00 seconds

```

In this assignment, we begin something similar to Assignment 5 but now we involve GCC and compiling in the C language. We execute the program and use 1 of 3 options given to us (-r for reversing a shift, -n for outputting the number of lines in the file, and -s to encrypt/shift over an inputted number of lines). We also output things involving the program when it is running such as Process ID, CPU%, memory usage, and run time to help us identify how much of the computer's resources it is using and help the user identify better with what is going on in the foreground and background. I initially had problems with the instructions because I automatically assumed you wanted a running program that gave out a menu for the user to understand the script when running but also initially accepted a terminal command line such as `ccipher -r -s 5 input_file.txt >> output_file.txt`. This was all for nothing as I reread and did the entire assignment all over again when I realized I overthink it. I struggled as well with the challenge as it just refused to print out the PIDs and computer resources due to being unable to grab them until I fixed it after 2 days of debugging. Overall, it has been some time since I last coded in C, so it took me some time to relearn what I coded in back in middle school and understand the semantics of how to code the encryption method for the application, but I was able to succeed and see how useful this could be in future projects if I see the need to.

## Conclusion

In conclusion, this assignment has shown the implementation of a Caesar cipher application in C, showing system calls and file manipulation techniques. By utilizing `open()`, `read()`, and `close()` system calls, we made a program capable of encrypting and decrypting text files based on user-specified shift values. Through the assignment, we experience in handling command-line arguments, reading files, and performing cryptographic operations. Also, the challenge to identify the process ID and resource usage of the running program enhanced our understanding of system-level interactions and resource management. By exploring CPU usage, memory utilization, and runtime statistics, we looked into the performance characteristics and resource demands of our application. Overall, this assignment helped our proficiency in C programming, system calls, and understanding of cryptographic techniques and their practical application in software development, showcasing the importance of robust encryption methods in

safeguarding sensitive information and highlighting the need for documentation and presentation in conveying technical concepts effectively.