Project 1

Due: Friday, October 17th 11:59pm

Description

The project will be implemented as three separate programs. There will be a logger – responsible for logging all activity. There will be an encryption program – responsible for encrypting and decrypting strings. There will be a driver program that will interact with the user to use the encryption program. The entire system will be ran by running the driver program, which will launch the other programs and communicate with them through pipes. If you use C/C++ to code your project you must use the linux system calls fork,pipe, and dup2. If use Java to code your project you must use the Process class. If you use Python use the Subprocess module. Examples for all these approaches will be provided in class. Details of each of the programs are below.

Details

Logger

The logger will write log messages to a log file. The log messages are lines of text where the first sequence of non-whitespace characters is considered the action, and the rest of the line is considered the message. The log message will be recorded, with a time stamp in 24 hour notation, in the log file as a single line using the following format.

```
YYYY-MM-DD HH:MM [ACTION] MESSAGE
```

So, the log message "`START Logging Started.`", logged March 2nd, 2025 at 11:32 am would be recorded as below.

```
2025-03-02 11:32 [START] Logging Started.
```

The logger program should accept a single command line argument – the name of the log file. The logger program will then accept log messages via standard input until it receives "`QUIT`".

Encryption Program

The encryption program should accept *commands* given as lines via standard input. The first word (sequence of non-whitespace characters) should be treated as a command, and the rest of the line(after the first space) as the argument for that command. Output is printed to standard out as a line of text where the first word is a *response type*. The currently set key is remembered by the encryption program. The encryption program should handle the following commands.

PASS Sets the current passkey to use when encrypting or decrypting.

ENCRYPT Using a Vigenère cypher with the current passkey, encrypt the argument and output the result. If no passkey is set output an error.

DECRYPT Using a Vigenère cypher with the current passkey, decrypt the argument and output the result. If no passkey is set output an error.

QUIT Exit the program.

The encryption program has the following response types.

RESULT The preceding command succeeded. The rest of the line (after a space) is the result of the executed command.

ERROR The preceding command failed. The rest of the line (after a space) is the error message.

More information on the Vigenère cypher can be found at the following url.

https://en.wikipedia.org/wiki/Vigen%C3%A8re_cipher

The Vigenère cypher only works on letters, and is case insensitive. Since the encryption program is the backend, you may assume correct input of only one case. For instance, you may assume that it always receive input in uppercase.

Example Interaction

Below is a single run of the encryption program.

- Input: ENCRYPT HELLO
  Output: ERROR Password not set
- Input: PASSKEY HELLO
  Output: RESULT
- Input: ENCRYPT HELLO
  Output: RESULT OIWWC

Driver Program

The driver program should accept a single command line argument – the name of the log file. Upon start, the driver program will create two new processes, executing the logger (giving the log file name as a command line argument) and the encryption program. Pipes should be used to connect to their standard input and standard output. Python and Java provide streams to communicate over these pipes. C/C++ will need to use the read and write functions to communicate.

Once set up, the driver program should print a menu of commands and prompt the user for commands – looping until the quit command is received. Each command should be logged, and the result of each command should also be logged. The start and exit of the driver program should be logged. All strings entered to be encrypted or decrypted should be saved in a history that lasts only for this run. The driver program should accept the following commands.

password Provide the user with the option of using a string in the history or entering a new string. If a new string will be used, prompt the user for a password, and then set it as the current passkey in the encryption program. If the history will be used, provide the user with a menu where a number can be used to select a string stored in the history. The entered password is not stored in the history. *Notice that password is used by the driver and passkey is used by the encryption program.*

encrypt Provide the user with the option of using a string in the history or entering a new string. If a new string will be used, prompt the user for a string, record it in the history, and send an encrypt command to the encryption program. If the history will be used, provide the user with a menu where a number can be used to select a string stored in the history. The results should be printed to standard output and saved in the history.

decrypt Provide the user with the option of using a string in the history or entering a new string. If a new string will be used, prompt the user for a string, record it in the history, and send an decrypt command to the encryption program. If the history will be used, provide the user with a menu where a number can be used to select a string stored in the history. The results should be printed to standard output and saved in the history.

history Show the history.

quit Send QUIT to the encryption program and logger, and then exit the program.

In the above commands whenever the history is used, provide the user with a means to exit the history and enter a new string. Passwords are never directly logged. Notice that commands to the encryption program are all on one line, but the driver program is more interactive. The command is expected to be inputted separately, and a separate prompt will ask for more information, such as the string to encrypt.

Make sure to provide error checking. There should be only letters in the input to the encrypt, decrypt, and password commands. If the user tries to encrypt "Hello World!", this should result in an error message.

The input should be case insensitive. So, encrypting "Hello" and "hello" should have the same result.