Cipher-Project

Release 0.1.0

Group B Team [Cipher-Project]

Sep 06, 2021

CONTENTS

CHAPTER

ONE

ABOUT

Cipher is a very basic text encryption program.

It has two high level function.

- TO ENCRYPT

  INPUT ——> Processing -—> OUTPUT

  Text file(normal) -—> encryption -——> text file (encrypted)

- TO DECRYPT

  INPUT ——> Processing -—> OUTPUT

  Text file(encrypted) -—> decryption -——> text file (normal)

  TLDR: We open text file do sth to each alphabet and get unreadable file (but here the catch we can undo this process!)

CHAPTER

TWO

HOW USER WILL USE IT

It is command line program so a terminal is needed.

- By giving our program a input file

```
$ cipher input.txt
```

- Now cipher will generate output.txt in the same folder
- INTERACTIVE MODE!

```
$ cipher --interactive
```

Will launch a prompt

```
 Hello and welcome to cipher!

>>
```

You can type words into the prompt and it will get encrypted!

```
Hello and welcome to cipher!

>> hello
eadfj

>> raju
kdsj
```

CHAPTER

THREE

TASKS LIST

3.1 1. Writing welcome and help messages

Program launch garda ke dekhincha r kasari use garne information.

You will look at different command line programs to draw inspiration and techniques from and can be as creative as you can with the messages.

3.2 2. Validation logic and error messages

Galat option r command diyo bhane error dini texts.

You will look at various popular command line applications and learn about how they give error messages. How to guide user to use the program correctly and so on.

Note: After Task 1 and 2,

- You get familiarity of project codebase and how to contribute
- improve familiarity of command line application and how they operate

---

3.3 3. Reading text file

Simple text file read garne function.

3.3.1 C Concepts Used

- fopen, fclose, fread
- Loops

3.4 4. Writing out text file

Simple text file write out garne function.

3.4.1 C Concepts Used

- fopen, fclose, fread, fwrite
- Loops

Note: You probably already have done 3 and 4. Choosing this task will mean you wil integrate that code into the project which includes polishing and editing the code to fit into the uniformity of rest of project. You may have to learn and switch to standard fwrite and fread function if you havent already.

3.5 5. Writing a logger setup

It is similar to debugging using printf function. What we do is basically add printf statements in the parts of code to make sure every things is running correctly and write the state of program in a separate file.

You will have to write a funtion similar to printf that will write to a file called debug.log whenever we call this function.

If anything goes wrong in our program, we can view this file and instantly know what our program was doing and where it failed. This is called logging and it helps in debugging our code.

3.5.1 C Concepts Used

- fopen, fclose, fread, fwrite
- Loops
- structs (optionally)

Note: This task is extended version of read and writing files. If you want to strenthen your file handling skills this is a perfect fit.

3.6 6. Write Makefile

Currently Assigned

Pradip

Program compile and build logic.

You will develop custom "compile and run" function availabel in code editors and know what really goes when building an exe file out of a C source file.

Why tho?

For command line application like ours and in real world project management scenarios there are custom designed tools to build exe out fo source files. MakeFile will give us more power, customizability and understading. Eg. making compiler strict, enabling/disabling certain features, giving better warning and error message, switching C versions, customizing output exe name etc.

3.7 7. Parsing out the cli arguments to struct

User le deko command and option yeuta struct ma organize garne. You will get list of string and classify them as commands and options.

You will have to what options and commands the program offers the users with. Know how they are presented as a message to the users.

3.7.1 C Concepts Used

- Struct
- string functions
- functions and return values

3.7.2 Extra Concepts

- Basic knowledge of Command line programs
- Basic knowledge of designing error handling and validation

3.8 8. Core encryption logic

Text r key file ko content lini r encrypted text nikalni function.

3.8.1 C Concepts Used

- Arrays
- Binary operators
- String functions
- Extensive use of functions

3.8.2 Extra Concepts

- Researching and building out from theory
- Cyber security and mathematical understadings

Currently Assigned

Ashwin

Choosing this tasks will mean you have to co-ordinate with Ashwin. You will learn about the implementation and theory first and then implement some of functions used in encrypting text files like permutation and xor operations.

CHAPTER

FOUR

SAMPLE CODE LAYOUT

```
// ... c headers


int main(//......){
         // Recieve input as commands from terminal
         struct Command command = parse_user_input()


         //if no commands Greet the user and give him instructions
         greet_with_about_and_help()

         // if User says a wrong command give hime usage errors
         validate_input_struct()

         // Write our own key array
         int *keys = {};

         // Read text file
         get_encrypted_text(file)

         // ENcrypt
         get_encrypted_text(text, key)

         // write out the encrypted text
         get_encrypted_text(text, key)


         // if use says so launch interactive mode
         interactive_propmt_loop()
}
```

CHAPTER

FIVE

PROJECT SETUP GUIDE

Following video will help you to

- install C compiler and other necessary tools
- VsCode[OPTIONALLY],

Note: Watch only **FIRST 4 MINUTES** if you prefer other editor and **DO NOT WANT** to setup VsCode i.e it is not a requirement and any editor you are comfortable with will do just fine.

CHAPTER

SIX

TASKS GUIDES

Guides for respective task

Every task will have its own detail guide to follow, if you want to do a task and its detail guide is unavailable you can ask it to be created.

6.1 Task 6 : Detail Guide

- Download code zip from this link

6.2 Check Lists

- Output the program in build/cipher.exe
- use C11 version
- Show all warnings
- Show warnings as errors
- Show extra warnings as well
- Disable redefining variables (declaring variables twice)
- Enable better debugging features

CHAPTER

SEVEN

RESOURCE: ENCRYPTION

The project is about creating a simple Data Encryption Standard(DES). To be a little familiar with DES, you may want to learn about SDES from:

- G-SDES (Simpler DES)
- DES-Examples
- Example C implementation

The Breaking down of project has not yet been done. This is to be done after feedback from my fellow peers.

CHAPTER

EIGHT

EXTRAS: RESOURCES FOR GIT

8.1 Using Git

This project is managed using git. The primary location of git host is: https://codeberg.org/cipher-project/cipher/

1. setting up the worktree

   setting up the worktree for git is easy. just type these commands in either a terminal emulator or cmd.exe

   ```
   git config --global user.name "Name Here"
   git config --global user.email "your email"
   git clone 'https://codeberg.org/cipher-project/cipher/'
   cd c-project
   git checkout -b mybranch
   ```

   These will setup the name, email and then prepare a new branch for you to work on. Please note that working in a new branch is highly recommended instead of using the master branch directly.

2. Commiting

   If you did the above steps, you're on a new branch called `mybranch`. Do note that you should commit in a sense of completion of small tasks. this makes it easy to revert to certain period of history. After working, commit your changes with:

   ```
   git add "file you worked on" "file2" "file3" "etc"
   git commit
   ```

   This will ask you for a commit message and a commit body. Put that as asked.

3. Merging with upstream (Hasta la vista "pull request")

   After you've worked on your branch, it'd be nice if we could also look at it. To do that, simply prepare a patch and send an email containing the said patch. These commands create a patch which you can attach as a email attachment to ashwin (the at symbol) ashwink.com.np After sending the patch, I'll update that on the server but in your seperate branch. You can click on the branch link in project homepage to see your branch and commits that you've sent.

   You can use this commands to make .patch files which are to be attached in the email body.

   ```
   git format-patch origin
   ```

   Do note that git send-email is a lot better but requires some teeny tiny configuration.

1. Unrelated note (skip to next part)

   The reason for using email is because git was *made* to be used with email. Please note that terrible clients like (gmail) are terrible and for better email experience, you should think about using something like Thunderbird. For someone informed opinion: https://drewdevault.com/2018/07/02/Email-driven-git.html

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex
- search