

**LineEd – Your own line editor**

**INDEX**

**1.** **Introduction**  3-4

1.1 Intended Audience 03

1.2 Project Purpose 03

1.3 Key Project Objectives 03

1.4 Project Scope 03

1.5 Functional Overview 04

1.5.1 Header Files 04

1.5.2 Functions 04

1.6 Assumptions 04

1.7 Risks 04

**2.Design Overview** 05

2.1 Design Objective 05

2.2 Error Detection 05

2.2.1 Memory Management 05

2.2.2 Performance 05

**3.Detailed System Design** 6-7

3.1 Structure 06

**4.Flowchart** 08

**5.Environment Description** 9-10

5.1 Time Zone Support 09

5.2 Language Support 09

5.3 User Desktop Requirements 09

5.3.1 Integration Requirements 09

5.3.2 Network 09

5.3.3 Configuration 10

**6.References** 10

**1.INTRODUCTION**

A line editor is a basic type of computer-based text editor where one line of a file can be edited at a time. Here this line editor is used for reading, saving, appending the files and also adding text, finding text and deleting specific text in a file.

This line editor lets you access commands from the executable files that are user defined.

This is a text-based user interface used to run programs manage files and interact with the computer.

**1.1 Intended Audience:**

The target audience are the clients who access line editor “LineEd” to create text file line by line. This software can be used to edit one line of a file at a time.

**1.2 Project Purpose:**

Line editor accepts input from the user by using File handling and Data structures which gives output for every command as it works in actual editor.

**1.3 Key Project Objectives:**

1. The LineEd supports the following commands:

* read
* save
* append
* display
* add
* find
* delete
* quit
  1. **Project Scope:**

In this LineEd we can only edit one line at a time or an integral number of lines. It will take care of only one line. Here we develop our own line editor “LineEd” to create text file line by line. When LineEd starts it should display message ‘Enter a command’ and wait for user to enter LineEd command.

After user enters a command, LineEd would perform editing on current text buffer according to command and display appropriate output message in case of successful completion of command or error message in case of command fails.

**1.5 Functional Overview:**

**1.5.1 Header Files:**

* #include<stdio.h>
* #include<string.h>
* #include<stdlib>

**1.5.2 Functions:**

The functions that are used in line editor are:

* read\_file (): This function is used to reads file with given filename and load it contents in text buffer.
* save\_file (): This function is used to saves a text buffer as file.
* append\_file (): This function reads file with given filename and appends its contents to existing text buffer at the end.
* display\_file (): This function has various formats. It is used to display with no argument, display with noscroll and display 10-20 lines.
* add\_file (): This function is used to insert text as new line after the current line.
* find\_file (): This function is used to find lines containing word in text buffer.
* delete\_file (): This function is used to delete lines whose line numbers are from line no. to line no.
* quit\_file (): This function will quit the line editor.

**1.6 Assumptions:**

The file which we are opening to read, save, display, add and so on should be saved in text format only.

**1.7 Risks:**

If we delete the file, then we will not be able to create the file because this line editor does not have create command to create a file.

**2.DESIGN OVERVIEW**

**2.1 Design Objectives:**

* Reads file with a given filename.
* Saves text buffer as file.
* Appends content of a file to an existing text buffer.
* Displays entire content of text buffer.
* Inserts given text as new line after the current line.
* Finds lines containing given word in text buffer.
* Deletes lines from the given starting line no. to the given ending line no.
* Quits the line editor.

**2.2 Error Detection:**

We have used Valgrind for error detection where valgrind is a tool which is used to check the memory leaks during the execution. It can do many other things like cache profiling also. We have integrated the project again and again to resolve the errors. We used valgrind very efficiently in this project to detect errors.

**2.2.1 Memory Management:**

There are two ways in which the memory is getting managed:

1.Static

2.Dynamic

The variables which are assigned in the function occupies the size according to their datatype. While we used linked list to manage the notes so, according to the notes, the size is occupied dynamically.

**2.2.2 Performance:**

Performance is based on the configuration of the system. This application gets load fast. It doesn't take much time to complete its task.

**3. DETAILED SYSTEM DESIGN**

**3.1 Structure:-**

* This system consists of 7 functions:

**1.Read:**

This command reads file with given filename and loads its content in text buffer. If there is any text in text buffer it is overwritten. Current line number = 1.

**2.Save:**

This command saves text buffer as file. The command should display error if file already exist and should not overwrite file. Current line number does not change.

**3.Append:**

This command reads file with given filename and appends its content to existing text buffer at the end. If there is no text in text buffer then this command is same as read command. Current line number = 1.

**4.Display:**

Display command has various formats

1.display with no argument

Displays entire content of text buffer line by line with line numbers. If text buffer is large it may scroll on the screen.

2.display noscroll

Displays entire content of text buffer line by line with line numbers. If text buffer is large, it displays 12 lines and wait for user to enter any key. After user enters a key it displays next 12 lines and so on

3.display 10-20

Displays entire content of text buffer line by line with line numbers from line 10 to 20. If text buffer contains less than 10 lines then display nothing. If text buffer contains less than 20 lines then display from line 10 till end of text buffer. Current line number does not change.

**5.Add:**

This command will insert <text> as new line after the current line. Current line number = line number of newly inserted line.

**6.Find:**

This command will find lines containing <word> in text buffer. Current line number does not change.

**7.Delete:**

This command will delete lines whose line numbers are >= <FromLineNo> and >=<ToLineNo>. If only <FromLineNo> is given it will delete single line. Current line number= 1.

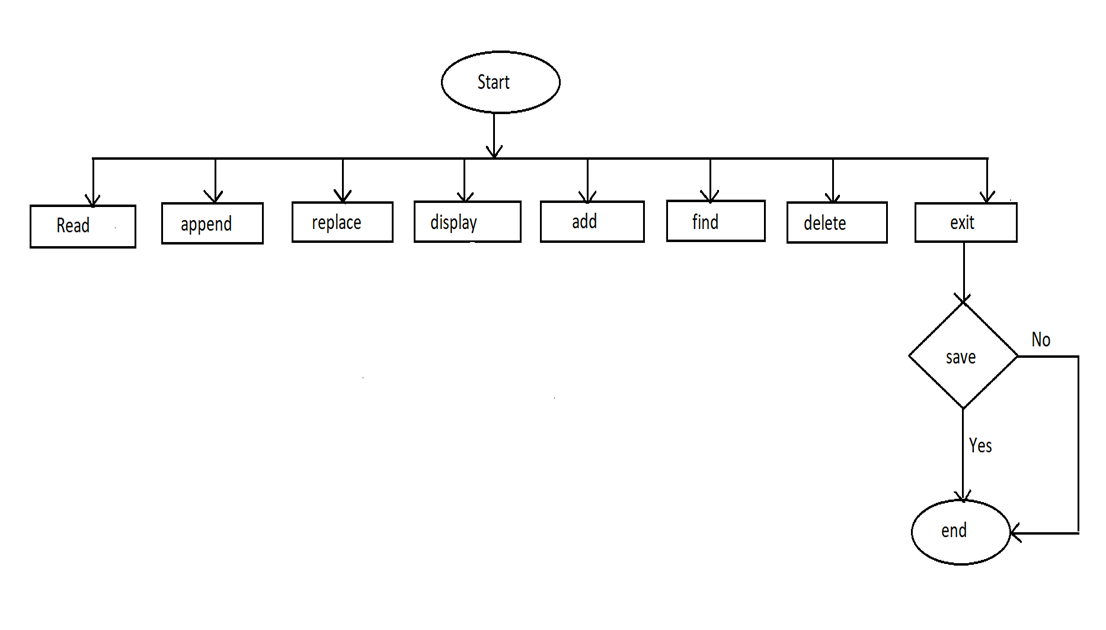
**8.Quit:**

This command will quit the LineEd. But before quitting it should check if current contents of text buffer is saved in file. If not, it should ask user if user wants to save the content of text buffer (yes/no), if users chooses ‘yes’ then it should ask filename to save to and save the content of text buffer to that file.

**3.2 Variables:**

**4.FLOWCHART**

The below flowchart depicts the workflow of line editor:



**5.ENVIRONMENT DESCRIPTION**

Linux is a Unix-like, open source and community-developed operating system (OS) for computers, servers, mainframes, mobile devices and embedded devices. It is supported on almost every major computer platform, making it one of the most widely supported operating systems. GCC compiler is used to compile and run a C program on the Linux operating system.

**5.1 Time Zone Support:** Supported on any given time (preferred Indian Standard Time).

**5.2 Language Support:**

C programming is a general-purpose, procedural, imperative computer programming language developed in 1972 by Dennis M. Ritchie at the Bell Telephone Laboratories to develop the UNIX operating system. C is easy to get started with, especially if you're running Linux. You can already run C code because Linux systems include the GNU C library. To write and build it, all you need to do is install a compiler, open a text editor, and start coding.

**5.3 User Desktop Requirements:**

* 64-bit processor, 1 GHz or faster
* At least 10 GB free hard drive space
* At least 1 GB RAM Server
* Windows system with minimum 4GB of RAM is required.

**5.3.1 Integration Requirements:**

* Language – C
* Tools – Valgrind, Makefile
* Compiler - gcc
* Linux Environment

**5.3.2 Network:**

* End to End

**5.3.3 Configuration:**

* **Operating System:**

Linux is an open-source operating system (OS). The OS sits between applications and hardware and makes the connections between all your software and the physical resources that do the work.

* **Database:**

File handling is used which refers to the method of storing data in the C program in the form of an output or input that might have been generated while running a C program in a data file.

* **Desktop:**

1GB for 32-bit Windows 7 or later desktops and 2GB for 64-bit Windows 7 or later desktops

**6.References:**

* All training classes conducted by trainer.
* All lab-sessions conducted during training.
* Google search