**cgpic**

**STREAM EDITOR SOFTWARE**

High Level Design & Low-Level Design

**Document Control :**

| **Project Revision History** | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  |  |  |
| **Date** | **Version** | **Author** | **Brief Description of Changes** | | | | **Approver Signature** | |
| 07/12/2022 | 0.1 | BHARGAVI.K | Basic design and documentation | | | |  | |
| 08/12/2022 | 0.1 | KANCHANA | Basic design and documentation | | | |  | |

| **Team Members** |
| --- |

| **Employee ID** | **Name** |
| --- | --- |
| 46289167 | Preethi. S |
| 46289231 | Bhargavi. K |
| 46289160 | Kanchana |
| 46289232 | Sudha |
| 46289261 | Mounika |

**S. No. TABLE OF CONTENT Page.no.**

**1. INTRODUCTION**

1.1. INTENDED AUDIENCE 5

1.2. ACRONYMS/ABBREVIATIONS 5

1.3. PROJECT PURPOSE 5

1.4. KEY PROJECT OBJECTIVES 6

1.5. PROJECT LIMITATION 6

1.6. FUNCTIONAL OVERVIEW 6

1.7. RISKS 7

**2. DESIGN OVERVIEW**

2.1. DESIGN OBJECTIVES 7

2.1.1. RECOMMENDED ARCHITECTURE 7

2.2. PATTERN SPACE 7

2.3. SPRINT FILES 8

2.4. EXTRACTING CONTENTS OF A FILE 9

**3. SYSTEM ARCHITECTURE**

3.1. ARCHITECTURE OF LINUX 9

3.2. CASE INSENSITIVE OF STREAM EDITOR 10

3.2.1. IMPORTANCE OF CASE INSENSITIVE 10

3.3. CASE SENSITIVE 10

3.3.1. CHANGE THE CASE SENSITIVE OF FILES AND DIRECTORIES 11

3.3.2. MODIFY CASE SENSITIVE 11

3.4. USE OF SED IN LINUX 11

3.5. VALGRIND 11

3.6. DETECT MEMORY ACROSS ERRORS AND LEAKS 11

3.7. PSEUDOCODE 12

3.8. GCC 12

3.9. STREAM EDITOR DRAWBACKS 12

3.10. HANDLING PROBLEMS 13

3.11. MAKEFILE 13

3.12. ADVANTAGES OF MAKEFILE 14

3.13. STREAM EDITOR STRINGS 14

**4. DETAILED SYSTEM DESIGN**

4.1. MANAGING PATTERNS IN STREAM EDITOR 14

4.2. PERFORMANCE 15

4.3. SET OF LINE ADDRESS 15

4.4. ADDRESS SPECIFIERS 15

4.5. ACTION SPECIFIERS 15

4.6. STREAM EDITOR-WORKFLOW 16

4.7. INPUTS 17

4.8. OUTPUTS 18

4.9. CONCLUSION 19

**1.INTRODUCTION**

A stream editor is used to perform basic text transformation on an input stream. While in some ways similar to an editor which permits scripted edits. SED was developed by during 1973-74 by Lee E. McMahon of Bells Labs. Today, it runs on all major operating systems.

Since its beginning, it has support for regular expressions. SED accepts inputs from files as well as pipes. Additional, it can also accept inputs from standard input streams.

SED is written and maintained by free software foundation (FSF) and it is distributed by GNU/Linux. Hence it is often preferred to as GNU SED. It can also solve many complex tasks with a few lines of SED script.

**1.1. Intended audience**

This document could be shared or view across the following members the CG Employees, BU SME’s, Internal SME’s

| CG Employees |  |
| --- | --- |
| BU SME’s |  |
| Internal SME’s |  |

**1.2. Acronyms/Abbreviations**

| SES | Stream Editor Software |
| --- | --- |
| GCC | GNU Compiler Collection |
| GDB | GNU Debugger |

**1.3. Project Purpose**

The main purpose of this stream is a logical entity that represents a file or device, that can accept input or output. All input and output functions in standard C, operate on data streams. Streams can be divided into text, streams and binary streams Stream editor for filtering and transforming text. The tool searches through, replaces, adds, and deletes lines in a text file without opening the file in a text editor. Stream Editor which parses text files and used for making textual transformations to a file. In the part of project a source code is invoked with SED commands. A stream is a popular concept for how to do input/output. Basically, a stream is a sequence of characters with functions to take characters out of one end, and put characters into the other end. In the case of input/output streams, one end of the stream is connected to a physical I/O device such as a keyboard or display. If it is a console output stream, your program puts characters into one end of the stream, and the display system takes characters out of the other and puts them on the screen. If it is a console input stream, the keyboard puts characters into one end of the stream, and your program takes characters out of the other and store the results in variables in the program. If no characters are waiting in the input stream, your program must wait until you supply some by typing on the keyboard. File streams follow the same principle, except that the file system is attached to the other end of the stream.

The Standard I/O streams: stdin and stdout

Two streams exist to allow you to communicate with your “console”- the screen and keyboard setup that makes up the traditional computer user interface. These are the input stream stdin(console input), which is connected to the keyboard, and the output stream stdout (console output), which is connected to your display. These two streams are created when the system starts your program. To use stdin and stdout and the input and output functions for them, you have to tell the compiler about them by including the relevant.

**1.4. Key Project Objectives**

* Manage everything from a single place
* Keep an eye on all tasks and provide reports
* Highly configured overflow

**1.5. Project Limitations**

For those we want to write portable SED scripts be aware that some implementations have been known to limit line length(for the pattern and hold spaces) to be know more than 4000 bytes. GNU SED has no built-in line length, as long as it can use a malloc(), more virtual memory you can feed or construct lines as long as you like.

**1.6. Functional Overview**

The command SED is an abbreviation for stream editor. It is an wildly popular tool on linux system. SED is not a text editor by itself. However, it can perform various modifications to manipulate the given text. The text input is sent as a stream. SED, then perform the instructed actions on the stream. This guide gives an overview of SED command and how to operate it in order to successfully manipulate text in linux. SED command will print the content of the text file by performing the operations described within the single quotes and printing the output, SED will simply perform a blank operation and print the entire content of the file. It will also accepts the output from a different command as the input stream.

**1.7. Risks**

In Stream Editor the sed command is very useful for getting sed to stop processing any more input once you have done what you want. However, you need to be very careful not to use q in any sed script that writes its edits back to the original files. After q is executed, no further output is produced. It should not be changed in any case where you want to edit the front of the files and pause the remainder through unchanged. In this case q is a dangerous beginners mistake.

**2. DESIGN OVERVIEW**

**2.1. Design Objectives**

In stream editors the file is treated as continues flow are sequence of characters instead of line numbers, which means here you can type paragraph. Example: SED editor in Unix.

**2.1.1. Recommended Architecture**

****

**Fig:1 Stream Editor**

**2.2. Pattern Space**

SED maintains a pattern space, a work space or temporary buffer where a single line of input is held while the editing commands are applied. The transformation of pattern space by a two line script. By the line script is shown in fig:2. It changes the Unix system to the Unix’s operating system.

Initially, the pattern space contains a copy of single input line. The normal form through the script is to execute each command on that line until the end of the script is reached. The first command in the script is applied to that line. Some SED commands change the flow through the script. The N command reads another line into the pattern space without removing the current line, so you can test for patterns across multiple lines. Other commands tell SED to exit before reaching the bottom of the script or to go to a labelled command. SED also maintains a second temporary buffer called hold scope. We can copy the contents of the pattern space to the hold space and retrieve them later.

Diagram

Description automatically generated

**Fig:2 The command in the script**

**2.3. Sprint Files**

It is not practical to enter longer editing scripts on the command line. That is why it is usually best to create script file that contains the editing instructions. The editing scripts is simply a list of SED commands that are executed in the order in which they appear. This form, -f option, requires that you specifies the name of the command line.

SED -f scriptfile file

**2.4. Extracting Contents of a File**

One type of SED application is used for extracting relevant material from a file in this way, SED functions like grep, with the additional advantage that the input can be modified prior to output this type of script good candidate for a shell script

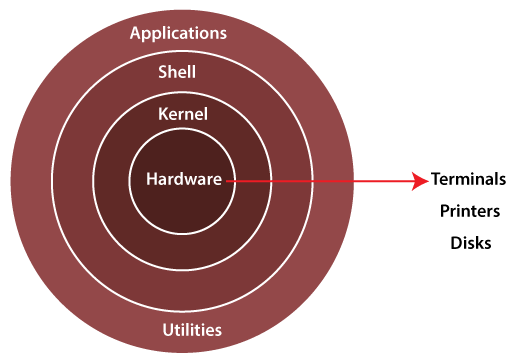
Example: Extracting a macro definition

Macros are defined in a macro package, often a single file that is located in a directory such as /usr/lib/macros. A macro definition always begins with the string and followed by an optional space and the one or two letter name of the macro. The definition ends with a line beginning with two dots(..). The script which shows in this project extracts a particular macro definition from a macro package.

**3. SYSTEM ARCHITECTURE**

**3.1. Architecture of Linux**

An operation system can be described as an interface among the computer hardware and the user of any computer. It is the group of software that handles the resources of the computer hardware and facilitates basic services for computer programs. On the other hand, Linux is one of the famous Unix OS. It is developed to provide a low-cost or free OS for several personal computer system uses. It is a complete OS including an X-Window system, Emacs editors, IP/TCP, GUI, etc.



**Fig:3 Architecture of Linux**

The Linux operating systems architecture mainly contains some library, hardware layer, system and shell utility.

**3.2. Case insensitive of Stream Editor**

A case insensitive search using SED under replacing or performing other options using SED commands. The value of this attribute will be 0 for case insensitive directories. Case insensitive is treating or interpreting upper and lower case letters are being the same. Often used in Stream Editor to indicate a comparison or equality test that does not distinguish between letters that only difference in case.

**3.2.1. Importance of Case Insensitive**

Case insensitive is important for past because they reduce the resource requirements on the earlier computer system. YOU did not need lower case letters, keyboard shift keys and the processing needed to handle them. Mostly, they were ingrind in the earlier days, there was not much point in changing them. Many programmers liked the idea that the language will catch capilatisation errors, entreat count, as the same symbols.

**3.3. Case sensitive**

Linux file system reads file and directory names as case sensitive.

**3.3.1. Change the Case Sensitive of files and directories**

To check if a directory is a case sensitive in the windows file system we use command to replace the file with file path.

**3.3.2. Modify Case Sensitive**

For pre directory case sensitive began in different windows, support was updated to include inspecting and modifying the case sensitive flag for a directory. Case sensitive is exposed using an extended attribute named. The value of this attribute will be 1 for case sensitive directories.

**3.4. Uses of SED in Linux**

In Unix-like operating systems, there is a text editing utility called sed, which stands for “stream editor”.

Basically, the sed is used to handle a text file and print the results on the standard output.

Replacing or substituting old characters with new characters is an essential requirement for any text editing tools. The advantage of SED is that it can leverage regular expressions perfectly.

**3.5. Valgrind**

The Valgrind framework is a powerful tool to debug your applications, and specially for memory allocation related bugs. We can get a detailed explanation of all Valgrind tools in the main site in our project and specially reading the Valgrind User manual. Valgrind can help to analyze any kind of software, in C language. Anyway, it is mostly used in C applications, mainly because the kind of errors found by Valgrind are usually more likely to happen in these language. This project will show you the memory leak detector available in the Memcheck tool. As you may already know, memory leaks can only be detected after the program execution. This project assumes that you already know and understand at least basically how the memory map in GNU/Linux system works, and specially the difference between memory statically allocated in the stack and memory dynamically allocated in the heap.

**3.6. Detect Memory Access Errors and Leaks**

**1.Compile the code with the -g flag:**

An optimisation error level of 1 is generally faster than level 0, also it can cause incorrect line numbers to be reported. An optimisation level higher than 1 can cause spurious unintialised -value errors to be report.

**2. Use of Valgrind as a wrapper for running the binary and performs stress testing:**

Memory access checking is unabled by default. The –leak-check option runs the memory like detector when the binary exits.

**3.7. Pseudocode**

projects use pseudocode for design.

The pseudocode version of the program use high-level style syntax,if…then, to describe the selections in the program. It has the advantage that no drawing is required, and the pseudocode can be entered directly into the text editor using the source code. It can be started as a brief outline and developed in stages, until it is ready to be translated into assembler syntax.

step1:According to the given requirements,for first requirement the input and pattern files created with the necessary text is opened in Read mode.

step2:The output file is opened in write mode to transfer the final output.

step3:Now verify whether the files are opened or Null.

step4:consider arrays as per requirement to store the strings and to modify.

step5:consider a string and search for it in pattern file.

step6:Replace it with the respective string.

step7:Transfer all the modified strings in the memory to the output file.

**3.8. GCC**

When you invoke GCC, it normally does preprocessing, compilation, assembly and linking. The overall options allow you to stop this process at any intermediate stage. For example the -c option says not to run the linker then the output consists of object files output by the assembler.

Most of the command line options that you can see with GCC are useful for c programs. When an option is only useful with another language. The GCC program accepts options and file names as operands. Many options have multi letter names. Therefore, multiple single letter options may not be grouped: -dr is very different from -d -r.

**3.9. Stream Editor Drawbacks**

* Hard to remember text from one line to another
* Not possible to go backward in the file
* No way to do forward references like /…. /+1
* No facilities to manipulate numbers
* Cumbersome syntax

**3.10. Handling Problems**

The main thing to keep in mind when dealing with SED is how it works. It works by reading one line in, performing all the tasks it known to perform on that one line, and then moving on to the next line. Each line is subjected to every editing command.

**3.11. Make file**

**Diagram

Description automatically generated**

Make file is a set of commands with variable names and targets to create object file and to remove them. In a single make file we can create multiple target to compile and to remove object, binary files. You can compile your project any number of times by using make files. A Make file can do a set of rules to determine which parts of a program need to be recompiled, and issues command to recompile them. Make file is a way of automating software building procedure and other complex tasks with dependencies. Make file contains dependency rules, macros and implicit rules. The make utility requires a file, which defines set of tasks to be executed. You may have make to compile a program from source code. Most open source project use make to compile a final executable binary. In make files we replace some names for automatic variables, our make file still have repeated content. If necessary modifying, we would have to upgrade our make file in several places. Thus, it becomes interest to assign variables for repeated teams and create functions, facilitating the keeping of the script. The variables used to come at the top of the make file because it is easier to find and modify them. Another option is to create a new file that contains the definition of the variables and then we can this file in our make file.

**3.12. Advantages of Make file**

It make codes more concise and clear to read and debug. No need to compile entire program every time whenever youn make a change to a functionality or a class. Make file will automatically compile only those files where change has occurred.

**3.13. Stream Editor Strings**

In Stream Editor the text substitution operations like “find and replace” are common in any text editor. In this project the SED performs text substitutions. Here in this file1 and file2 are the starting and ending addresses respectively, which are either line numbers or pattern strings. Both this files are optional parameters. The pattern is the text which we want to replace with the replacement string.

In string replacement the GNU SED provides some special escape sequences which can be used in the replacement string. The string replacement flags are GNU specific and may not work with variants of sed.

**4. DETAILED SYSTEM DESIGN**

**4.1. Managing Patterns in Stream Editor**

Many Linux users and Linux’s numerous distributions occasionally or regularly work at the command line level. Some tasks can be done faster or better in the terminal than in the graphical user interface. To do this, however, you need to know the various commands and how they work. One such command in Linux is “sed.”

The command “sed” is part of the basic equipment of every Linux installation, as it belongs to the GNU Core Utilities. The tool is called a non-interactive text editor. This means that you don’t make a change directly to the file you are editing. Instead, you first create a temporary file whose contents are then passed to the source file. Linux SED proceeds line by line. Each line of a file is read individually, processed, and then output again. The most important function of SED is to search for certain strings in the file and then replace them with other characters.

**4.2. Performance:**

The SED is a stream editor that receives its input from standard input, changes that input as directed by commands in a command file, and writes the resulting stream to standard output.

If you do not provide a command file and do not use any flags with the Stream Editor command copies standard input to standard output without change.

When Stream Editor edits, it reads the input stream one line at a time into an area in memory called the pattern space. When a line of data is in the pattern space, Stream Editor reads the command file and tries to match the addresses in the command file with characters in the pattern space. If it finds an address that matches something in the pattern space, Stream Editor then performs the command associated with that address on the part of the pattern space that matched the address. The result of that command changes the contents of the pattern space.

When Stream Editor has tried to match all addresses in the command file with the contents of the pattern space, it writes the final contents of the pattern space to standard output. Then it reads a new input line from standard input and starts the process over at the start of the command file.

**4.3. Set of Line Address**

Use regular expressions to which lines

* Written between two slashes
* Process only lines that match
* May match several lines
* Lines may or may not be consecutives

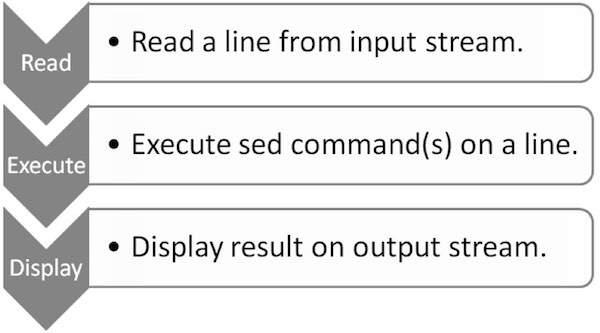
**4.4. Address Specifiers**

Address specifies either one line number to select a single line or a set of two lines, to select a group of contiguous lines.

**4.5. Action specifiers**

Action specifies print, insert, delete, substitute the text

**4.6. Stream Editor-Workflow**



**Read**: SED reads a line from the input stream (file, pipe, or stdin) and stores it in its internal buffer called pattern buffer.

**Execute**: All SED commands are applied sequentially on the pattern buffer. By default, SED commands are applied on all lines (globally) unless line addressing is specified.

**Display**: Send the (modified) contents to the output stream. After sending the data, the pattern buffer will be empty.

Pattern buffer is a private, in-memory, volatile storage area used by the SED.

By default, all SED commands are applied on the pattern buffer, hence the input file remains unchanged. GNU SED provides a way to modify the input file in-a-place. We will explore about it in later sections.

There is another memory area called hold buffer which is also private, in- memory, volatile storage area. Data can be stored in a hold buffer for later retrieval. At the end of each cycle, SED removes the contents of the pattern buffer but the contents of the hold buffer remains persistent between SED cycles.

However SED commands cannot be directly executed on hold buffer, hence SED allows data movement between the hold buffer and the pattern buffer.

Initially both pattern and hold buffers are empty.

If no input files are provided, then SED accepts input from the standard input stream (stdin).

If address range is not provided by default, then SED operates on each line.

**4.7. Inputs**

Text

Description automatically generated with medium confidence

**4.8. Output**

**Shape, rectangle

Description automatically generated**

**Shape, rectangle

Description automatically generated**

**Shape, rectangle

Description automatically generated**

**4.7. Conclusion**

The Stream Editor Software concludes that the SED command is printed specific lines from the file, searched for text, deleted lines, overwrote the original file, and used regular expressions to replace text in another file. You should be able to see already how you can quickly transfer text document using properly constructed SED commands. Stream Editor Software is a powerful tool and works very well for search and replace functions. The sed, so called stream editor, is an awesome tool to manipulate text in Linux. For some complex tasks, you probably need to do lots of steps or repetitive operations in a GUI or interactive text editing tool. But with the help of sed, all you need is one line of command.