2022 3rd International Conference for Emerging Technology (INCET)
Belgaum, India. May 27-29, 2022

Terminal Based Text Editor

Sumitkumar Shrivastav
Dept. of Information Technology and MCA
Vishwakarma Institute of Technology
(Affiliated to Savitribai Phule Pune University)
Pune, India
shrivastavsumit.99@gmail.com

Harsh Attri
D*ept. of Electronics & Telecommunication*
Vishwakarma Institute of Technology
(Affiliated to Savitribai Phule Pune University)
Pune, India
shattriharsh@gmail.com

2022 3rd International Conference for Emerging Technology (INCET) | 978-1-6654-9499-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/INCET54531.2022.9824597

***Abstract*— Text editors have been functionally used to provide ease in editing any particular document. Shortcuts have played the most important role in providing the same. Test editors which are terminal based are very technical for any general user to understand. The proposed text editor helps deal with this problem by providing shortcuts in terminal based editors which are present in general editors. The proposed editor has also got some enhanced performance in editing when compared to its counterparts. Use of efficient and customized data structures have been done to ensure this. Several other important features have been included to provide a rich experience. This provides the user with a complete package at a single platform.**

Index Terms – Ncurses Library, text editors, data structures, linked lists, terminal, linux.

I. INTRODUCTION

Text editors are one among the key requirements of a programmer. Text editors acts sort of a tool which brings any idea to existence. Hence, the necessity arises for an editor, that's not only freed from any errors or bugs but also suits the programmer‘s needs and desires. Preferences vary from person to person and hence the wide selection of editors available within the software world. This editor focusses on these needs of the programmer and guarantees a user-friendly experience to make sure quick learning and long-term usage by the user. This editor also incorporates compiler-like features which include auto-complete, display of undeclared & unused variables and other features. Building a terminal based editor is linked with clever usage of macros and code snippets to display items and essentially control the terminal output. Like all software it consists of a front and rear where front is handled by the ncurses library and backend by effective C code.

A. Ncurses

Ncurses or new curses is a programming library providing an application programming interface (API) that permits the programmer to write down text-based user interfaces in an exceedingly terminal-independent manner. it's a toolkit for developing "GUI-like" application software that runs under a terminal emulator. It also optimizes screen changes, so as to scale back the latency experienced when using remote shells. It provides functions to make windows etc. Its sister libraries panel, menu and form provide an extension to the essential curses library. These libraries usually come along side curses. One can create applications that contain multiple windows, menus, panels and forms. Windows are often managed independently, can provide 'scroll ability' and even may be hidden.

This allows the task of manipulation of the output on the terminal. The terminal within the normal mode essentially prints regardless of what the user types onto the screen. However, for an editor to be built, this had to be disabled. Thus, to access the terminal within the RAW mode the ncurses library was used.

Ncurses incorporates several features which will be utilised to create an editor. The noecho() as well as raw() feature are often used to bring the terminal into RAW mode. Further, the library also consists of several macros where every key on the keyboard is mapped to a macro. These are often utilized within the rear to code each and every key ranging from the essential A-Z’s and 0-9’s followed by complicated combinations like backspace, delete, arrow left-right-up-down, shift as well as control keys combined with other basic keys and tab.

B. Data Structures

A data structure could be seen as a particular way of organizing data in a computer so it may be used effectively. Various arrangements are studied so as to possess a particular and customised data structure which may allow the efficiency to be maximised and therefore the inefficiencies to be delivered to the smallest amount. There are many data structures which have been utilised earlier to create a text editor which include arrays, linked lists, doubly linked lists.

An array can be called as a collection of things stored at contiguous memory locations. The thought is to store multiple items of an equivalent type together. This makes it easier to calculate the position of every element by simply adding an offset to a base value and this finally results in a far better direct access to all or any the info that has been stored. But the matter that this arrangement faces is that it's static and when the info exceeds its limit, operating system has to reallocate all its memory to a replacement location and replica all its existing data alongside new data to the new location. Also inserting or deleting a data item within the middle are often time consuming.

A linked list could be called as a linear collection of similar elements whose order isn't given by their physical placement in memory. Instead, each element points to a subsequent memory pointer. It's a data structure consisting of a set of nodes which together represent a sequence. It's dynamic in nature. But the memory consumption for storing each item during a node is extremely high and therefore the reverse traversal between nodes are often a drag.

A doubly linked list may be a modified version of linked list which provides ease in reverse traversal with the introduction of an additional pointer. But the matter of memory consumption isn't solved during this.

This creates the necessity for a new customized arrangement which may provide the advantages of those data structures and at an equivalent time help to scale back its inefficiencies. The data structure should be able to provide fast access like an array and ease in traversal like a doubly linked list.

II. LITERATURE SURVEY

Data Structure is the primary focus for the creation of an efficient text editor. Several types of data structures have been utilized over time in order to have improves efficiencies. Each data structure specializes in some tasks but will also carry some challenges to be overcome. Below is a table which depicts the usage of various data structures for the creation of a text editor. It also contains the features and challenges that comes with it.

TABLE I. LITERATURE SURVEY

Author[citation]	Methodology	Features and Challenges
Data Structures in the Andrew Text Editor, Wilfred J. Hansen(1986)	Use of Array Data Structure in implementation of Text Editor	Useful for small files, allows faster access. Modifying the file is costly.
Data Structures for Text Sequences, Charles Crowley(1998)	Comparison of Array and Linked List Data structures	Array is useful in Fast access, linked lists allow easy insertion and deletion. Editing is very costly for arrays, memory Consumption for LL is high.
Design and Implementation of an Advanced Text Editor(ZED), Craig Bruce(1995)	Design and Implementation of a Text Editor using Doubly Linked Lists	Ease in insertion and deletion operations. Increased memory consumption.
Data Structures used in Text Editors, Ethan Z. Booker(2004)	Use of Trie data structure for the implementation of Text Editor	The feature of Search, auto-complete works best in this data structure. Memory Consumption is very high.

All the various data structures used in the above editors have been effective in some aspects and inefficient in some other aspects. The proposed editor makes a good attempt in addressing this problem by bringing a customized data structure which can have all the benefits and at the same time it can reduce the inefficiencies.

III. ALGORITHMS

A. Insertion Algorithm

Insertion and deletion are the two most important features in any text editor. These features build the foundation for almost all the other advanced features. Below mentioned algorithm precisely mentions the technique which can be brought to use for insertion of any character in the text editor at any particular position. It also explains about the data structure which has been used to build the editor.

Insertion algorithm:

```
typedef struct node {
        int *p;
        struct node *next, *prev;
}node;

typedef struct buffer {
        node *head, *tail, *start, *end;
        int bufRows;
}buffer;
```

The data structure which has been used in the construction of this text editor is a much customised one which looks into all the needs of a modern text editor. The data structure which has been used is a combination of doubly linked lists with arrays. We consider every row as a Node and the column constituted within it as an array (*p). Following is the procedure which has been applied to insert any line or character in the editor.

1) Initialize *start* pointer to the *temp* pointer.

2) If it is the first character in editor, then a new node is created and *start* pointer is pointed to this node. *next* and *prev* pointer are made null. *tail* and *head* pointer are pointed to current node.

3) Else traverse to the node at which character is to be inserted.

4)

Case 1: If character is '0/'

i) Move cursor to the next line.

Case 2: If character is '/n'

i) If position is at the end of line then simply move cursor to the next line.

ii) If position is in the middle:
if next line does not exist then create new node and copy all the characters which are at right side from current position to the next line. Move cursor to the next line.

Case 3: Any other character

i) If position is at the end of line then store character at the position and move cursor to the next line.

ii) If position is in the middle then insert character at that position and move all characters (at the right side) towards right by single column.

iii) If position is neither at the end of the line nor middle, then just insert character at that position and move cursor to the next column.

Case 4: Inserting new line in between two lines.

i) Create new node and point prev pointer to the prev node.
ii) Point next pointer of new node to the next pointer of previous node. Point next pointer of prev node to the new node.
iii) Point prev pointer of node just next to the new node to the new node.

All these operations lead to a successful insertion of any new character or line at any desired location. Similar set of operations with an opposite sense can be used to implement the delete operation which is one of the two basic features of any text editor.

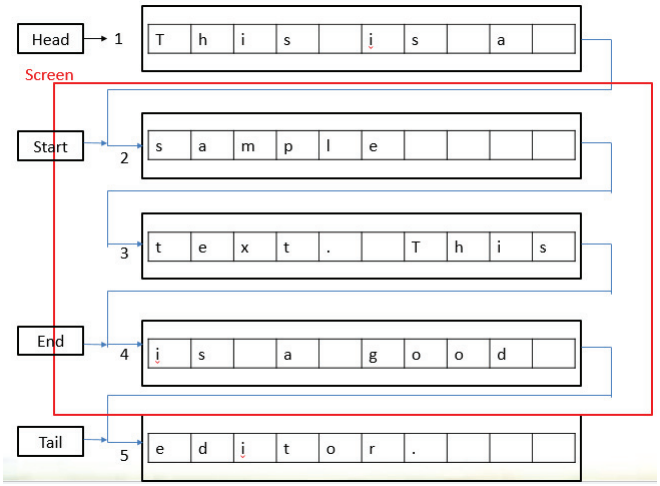

Fig. 1. Insertion Example

IV. DIAGRAMS

A. Flow Diagram

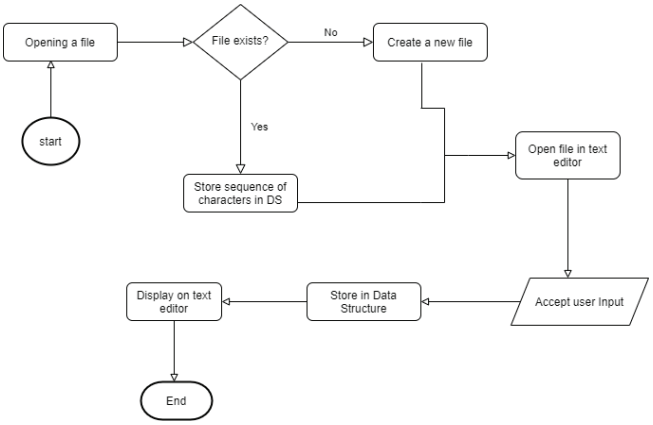

Fig. 2. Flow Diagram

Above flow diagram depicts the overall execution of the project. Initially, we need to open a file. Then, existence of the file will be checked. If the file does not exists then a new file be automatically created and will be opened in a Linux terminal. After opening a file, sequence of characters would be stored in the proposed data structure and the file will now be opened in a Text Editor. Now, user can write text data in the editor, the stored data will now be displayed on the screen and user can perform several operations in the editor itself e.g. cut, copy, paste etc.

V. INTERFACE

A. New File Page

When a new file is opened in the terminal, then it is initially empty as shown in the below image. Once we start to fill the lines, the placeholders will automatically get removed.

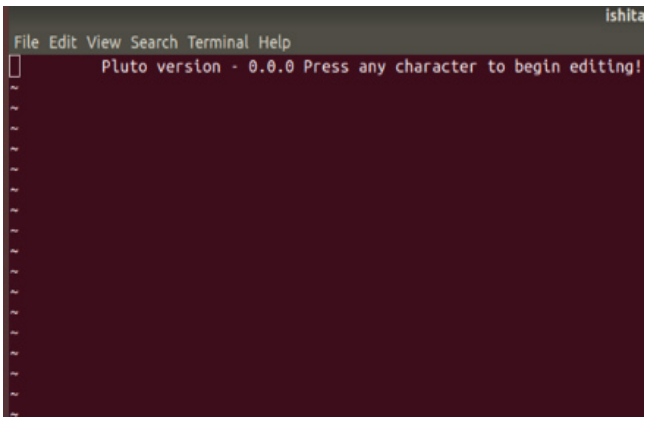

Fig. 3. New File page

B. Programming page

This page depicts that this editor allows full-fledged programs to be written and it can perform all the basic tasks of a text editor along with some advanced compiler operations available in the editor.

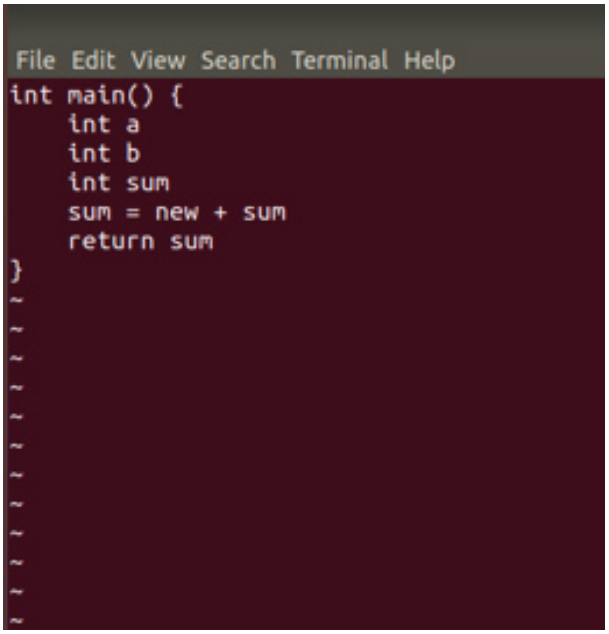

Fig. 4. Programming page

C. Search and Replace

The following image shows the menu of search and replace feature. It is very easy to understand its UI which makes it more appealing for a large audience.

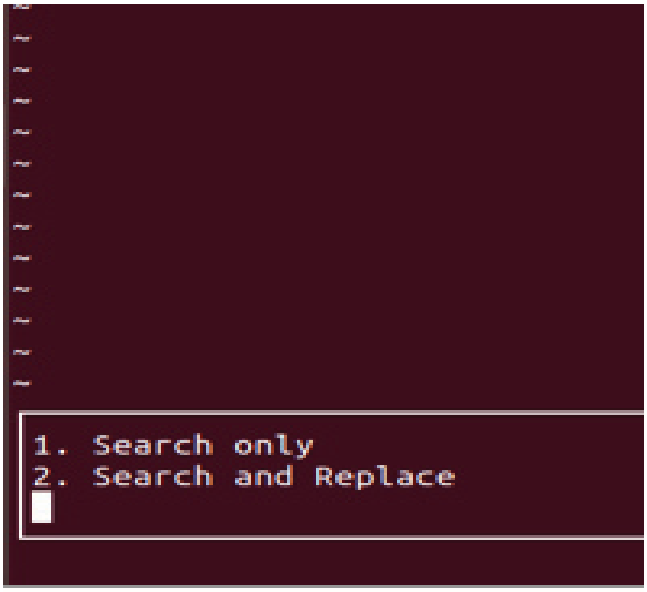

Fig. 5. Search and replace

VI. Conclusion

The need for a user friendly terminal based editor has been evident by the unavailability of general shortcuts and necessary features in the currently available terminal based text editor. This text editor brings in the use of a customised data structure which provides the best features of an array and doubly linked list and at the same time reduces its inefficiencies. Various shortcuts have been provided which are familiar to the users of any general text editor.

Some advanced features such as text auto completion, memory leak check, identification of unused variables have been incorporated. Search and replace is also functions very well and provides a good overall search speed. These features immensely help the usability of this editor. The editor can be used with the ease of a terminal along with all the features of a normal text editor. The user interface for all the features has been kept very easy to understand and use.

VII. Future Scope

Future work for this text editor can be:

- A voice based text editor can be built using voice recognition algorithms.
- A dedicated GUI can be made which has several aesthetic and animated features.
- Code generators can be added to its bank of features.
- Natural Language Processing can be included to better process and automate.
- Support for various languages can be included to increase the scope of its use.
- Support for several existing libraries and frameworks can be included.
- This editor can be provided in different platforms such as Windows and Mac apart from Linux.

References

[1] C. C. Charlton and P. H. Leng. Editors: two for the price of one. Software| Practice and Experience, 11:195-202, 1981.

[2] R. Pike. The text editor sam. Software| Practice and Experience, 17(11):813-845, November 1987.

[3] J. Gutknecht, Concepts of the text editor Lara, Comm. ACM, 28, (9), 942-960 (1985).

[4] C. W. Fraser, A generalized text editor, Comm. ACM, 23, (3), 154-158 (1980).

[5] Wilfred J. Hansen, Information Technology Center, Carnegie-Mellon University: Data Structures in the Andrew Text Editor, May 15, 1986

[6] https://www.averylaird.com/programming/the%20text%20editor/2017/09/30/the-piece-table/

[7] https://iq.opengenus.org/data-structures-used-in-text-editor/

[8] https://www.drdobbs.com/architecture-and-design/text-editors-algorithms-and-architecture/184408975?pgno=2

[9] http://texteditors.org/cgi-bin/wiki.pl?DesigningTextEditors

[10]] [An Adaptive and Intelligent Text Editor for Teacher's Support of Practical Exercises] Polina Atanasova, Stefan Stefanov, Irina Zheliazkova Rouse University, Bulgaria (2006).