

Unix Project 2: VIM Clone

Texas State University
CS4350
Professor Gu
03/27/18

By
James Bellian, Derek Sittenauer, Osamuyi Edomwande, Robert Sarvis, and John
Barulich

1. Introduction

The group was coordinated by assigning different operations and key aspects of the project to each of the members. First all tasks were determined from the assignment. Then we took these tasks and grouped them into five groups of responsibilities. The responsibility groups were then each given to a member at random. Individual header and implementation files from each member and the main project file were uploaded to github where everyone could have access to the current state of the program. Individual responsibilities are as follows:

- i. James Bellian was responsible for creating the main window and basic UI in "Editor.c". James also wrote the operation to edit text in the console, which can also be found in "Editor.c" under the function "type()". The Makefile for the project was also created by James. The introduction paragraph was written by James with responsibilities written by their respective team members. James also helped write Task II describing how to use the editor.
- ii. Derek Sittenauer was responsible for creating the copy/paste and save functions of the text editor. Also Derek worked on section 4 (task III) part b of the paper.
- iii. Osamuyi Edomwande was responsible for writing foo.c, foo's make file and the delete function. As well as working on section 2 of the report.
- iv. John Barulich was responsible for writing the add line and fillBuffer functions. John also contributed to the search function. John worked on section IV part c, to obtain screenshots of the successful new line addition.
- v. Robert Sarvis was responsible for creating the function that would search for text and if the user wanted to, replace the text with new text. This code was placed in the file search.c and search.h. Robert worked on Section IV (Task III) of the paper, part D, which requested screenshots of the searching and replacing functions.

2. Task I

Foo.c

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <ncurses.h>
```

```
int main(int argc, char **argv) {
```

```
    //If not given run command and file name
    if(argc != 2)
    {
```

```

        printf("Usage: ./foo <text-file>\n");
        exit(1);
    }
    //If file cannot be found
    FILE *file = fopen(argv[1], "r");
    if(file == NULL)
    {
        printf("Cannot find file\n");
        exit(1);
    }
    //create character array buffer
    char buffer[10000];
    //declare window vars
    int row, col, y, x, MAX_ROW;

    //Create the screen
    initscr();
    //Store max rows and column
    getmaxyx(stdscr, MAX_ROW, col);

    //Begin at the top of the window
    row = 0;

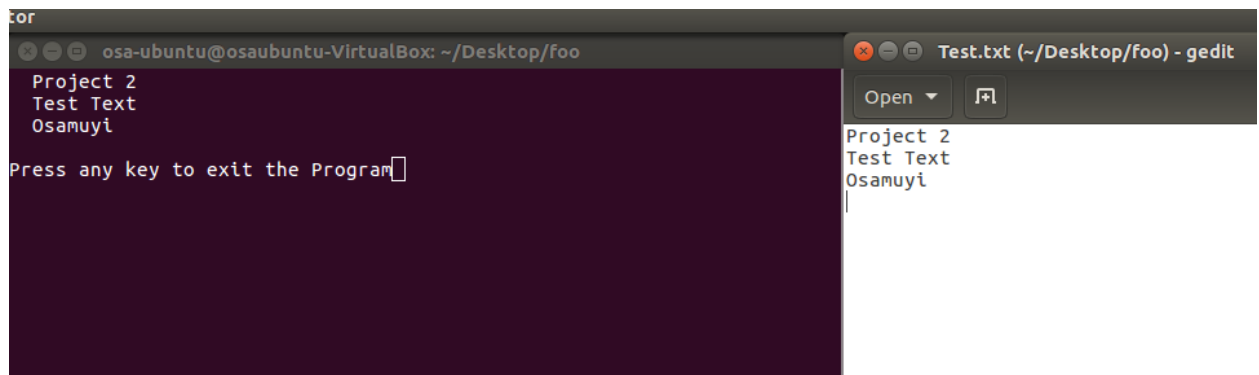
    //read file
    while(fgets(buffer, sizeof buffer, file) != NULL)
    {
        //get x and y position
        getyx(stdscr, y, x);
        //print message to screen
        mvprintw(row, (col-strlen(buffer))/25, "%s", buffer);
        //increment row
        row = row + 1;
        refresh();
    }
    //Closes window
    printw("Press any key to exit the Program");
    getch();
    endwin();
    exit(0);
}

```

b.

```
osa-ubuntu@osaubuntu-VirtualBox:~/Desktop/foo$ make clean
rm -rf foo
osa-ubuntu@osaubuntu-VirtualBox:~/Desktop/foo$ make
gcc -g -Wall foo.c -o foo -lncurses
foo.c: In function 'main':
foo.c:25:23: warning: variable 'MAX_ROW' set but not used [-Wunused-but-set-variable]
    int row, col, y, x, MAX_ROW;
                        ^
foo.c:25:20: warning: variable 'x' set but not used [-Wunused-but-set-variable]
    int row, col, y, x, MAX_ROW;
                    ^
foo.c:25:17: warning: variable 'y' set but not used [-Wunused-but-set-variable]
    int row, col, y, x, MAX_ROW;
                ^
osa-ubuntu@osaubuntu-VirtualBox:~/Desktop/foo$
```

c.



3. Task II

- a) The design of the program uses two basic modes, Edit Mode and Command Mode similar to VIM. A key difference is that in our editor, 'Esc' switches between Command and Edit Mode depending on which is currently set. The following chart describes the Command mode editing operations and the files they can be found in.

Key	Operation	Source File
O	Insert line	textBuffer.c
S	Save to text file	save.c
F	Find and Replace	search.c
D	Delete line	delete.c

Y	Start/End Copy. Select a starting index and ending index to copy text.	copy.c
P	Paste the text saved to clipboard	copy.c
Q	Quits the program when in Command Mode	Editor.c
(Type while in Edit Mode)	Enters text to the screen, storing the characters in the text array.	Editor.c
(Command Line Argument)	Read Text file to console. File is supplied as an argument to the program, reading done on initialization.	textBuffer.c

b) The data structure for the Editor is declared in Editor.h. It is a struct that uses an array of char pointers to hold lines of text. These lines are manipulated directly and read from the screen to be re-written when the file is saved. Editor.h is as follows:

```
// Editor.h
// The header for the Text Editor
// Programmed by James Bellian 03/24/18

#ifndef Editor_h
#define Editor_h

#include <ncurses.h>
#include <stdlib.h>
#include <string.h>

struct TextEditor;

#define MAXLENGTH 9999

// Editor is the main structure of the program, creates the ncurses window,
// switches between edit and command mode,
typedef struct TextEditor
{
    char ** text; // the body text of the editor
    int x, y; // x and y coordinates of cursor
```

```

    int w, h; // width and height of the terminal window
    int scroll; // the number of lines scrolled down
    char mode; // 'e' for edit mode, 'c' for command, 'x' for exit
    int input; // user input from keyboard
    int num_lines; // number of lines in the text field
} Editor;

// Initializes ncurses and sets variables within the struct 'e'
void init( Editor *e, char ** lines, int nl, char * fn);

// begins the main loop of the Editor (DOES NOT NEED CALLED, CALLED AFTER INIT)
void start(Editor *e);

// Draws text from the string buffer to the screen
void update_window(Editor *e);

// Prints blank spaces to clear out the screen
void clear_window( Editor *e);

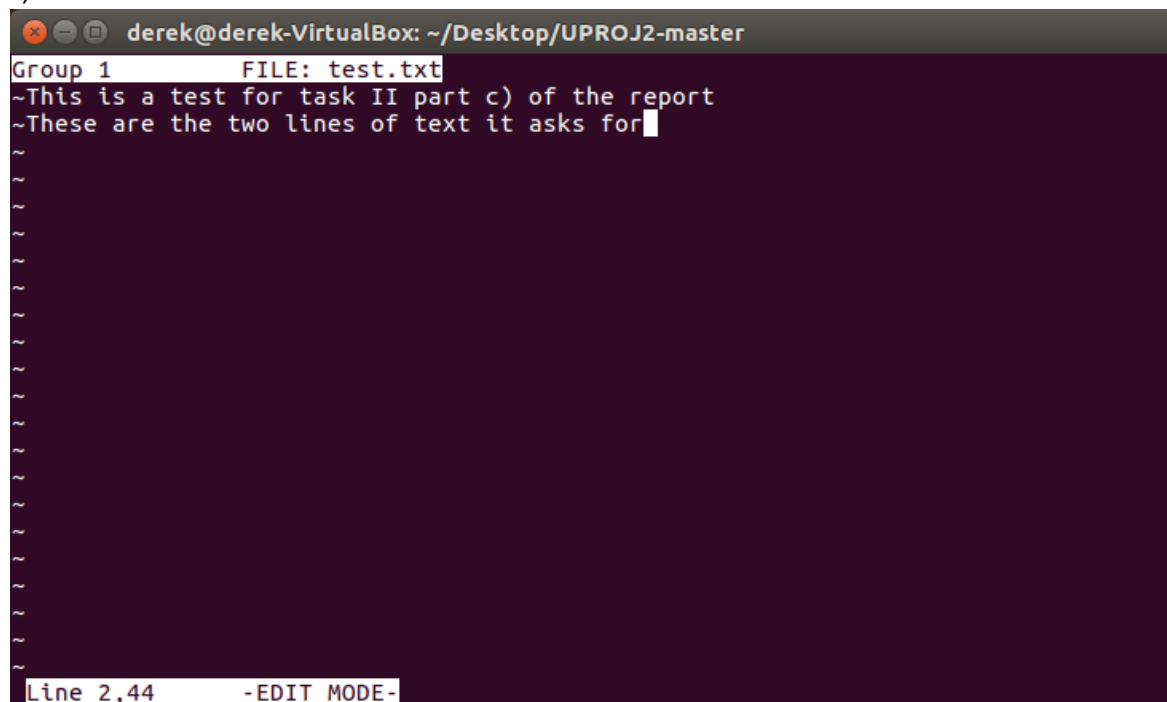
bool type(Editor *e, char letter);

int strLen(char * str);

#endif

```

c)



The screenshot shows a terminal window titled "derek@derek-VirtualBox: ~/Desktop/UPROJ2-master". Inside the terminal, a text editor is open with the file "test.txt". The editor shows two lines of text: "~This is a test for task II part c) of the report" and "~These are the two lines of text it asks for". The cursor is at the end of the second line. The editor interface includes a status bar at the bottom showing "Line 2,44" and "-EDIT MODE-".

Char text[V, e, r, t, i, c, a, l, , t, e, x, t][H, o, r, i, z, o, n, t, a, l, , t, e, x, t];

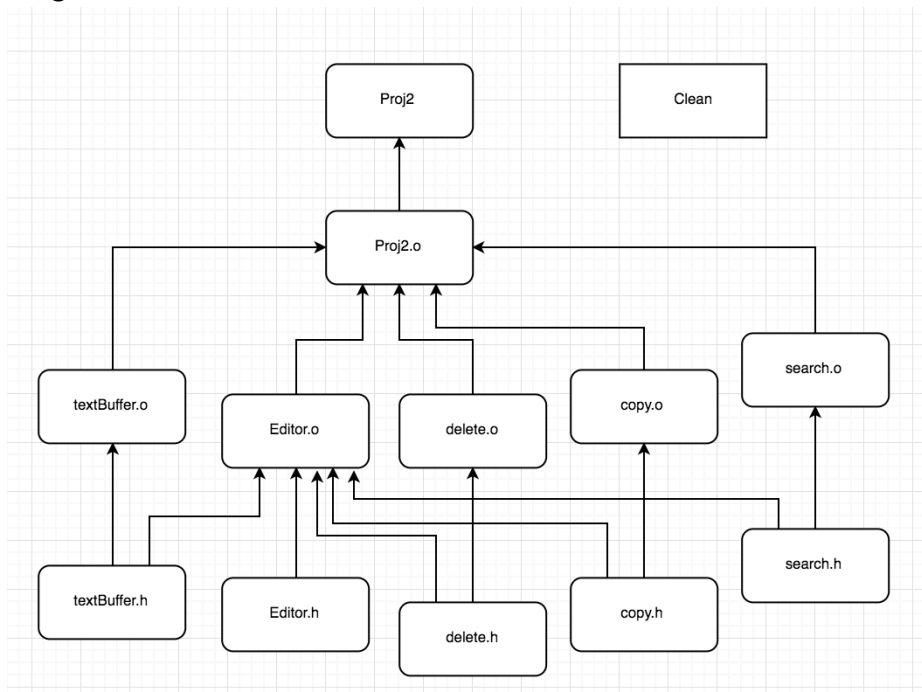
Horizontal text

V
e
r
t
i
c
a
l

t
e
x
t

4. Task III:

a) Diagram for Makefile



b)

Below are the attached screenshots using the “ESC” key to switch between editing and command modes.

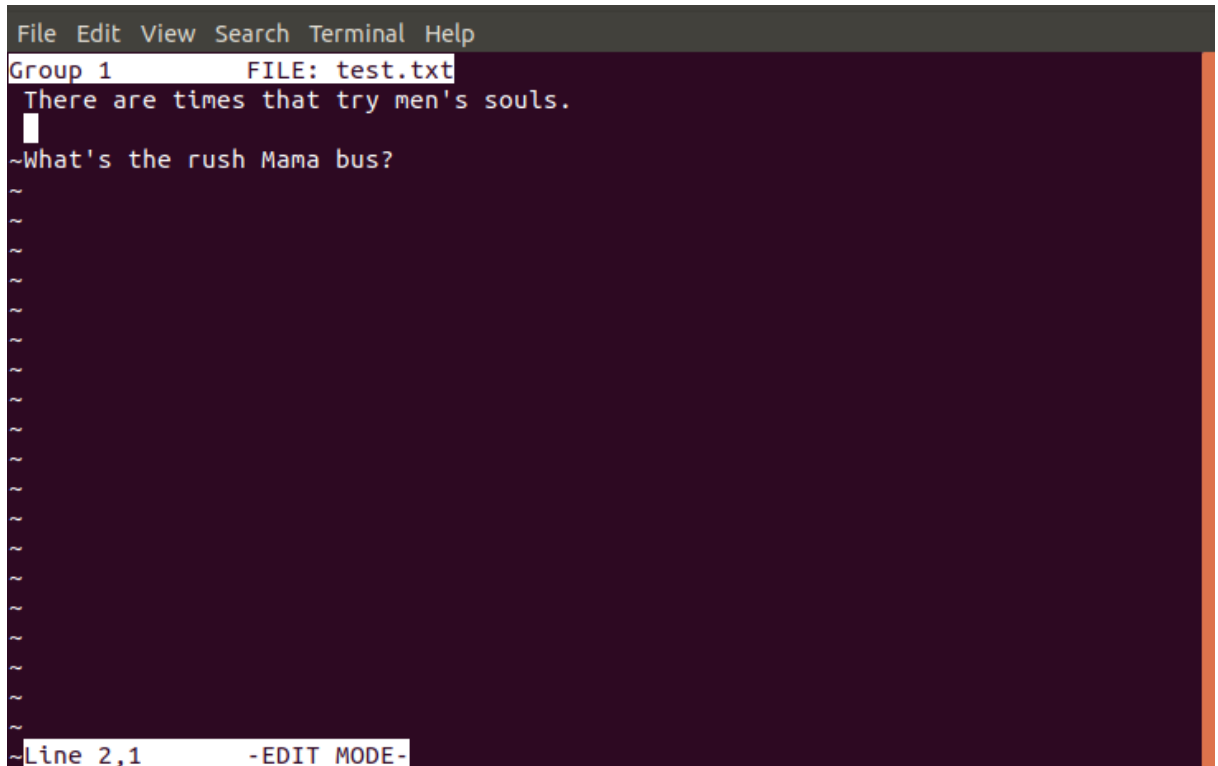
[illegible]

Below is the switch into command mode.

[illegible]

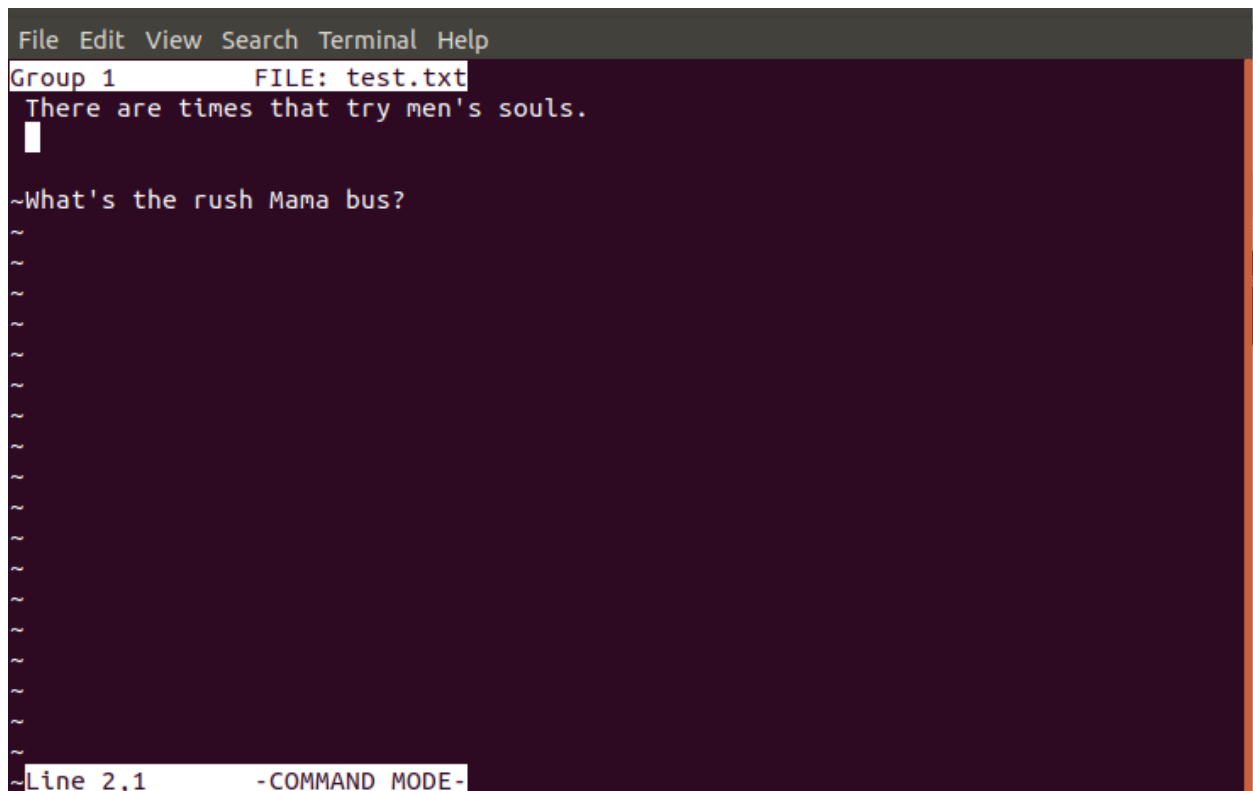
c)

Below, we show screenshots of the before and after of adding another line of text before a line already in the editor.



This screenshot shows the vi editor in insert mode. The top menu bar includes 'File Edit View Search Terminal Help'. The status bar at the top indicates 'Group 1' and 'FILE: test.txt'. The text 'There are times that try men's souls.' is on the first line, and '~What's the rush Mama bus?' is on the second line. The cursor is positioned at the start of the second line. The status bar at the bottom shows '~Line 2,1' and '-EDIT MODE-'. The editor window has a dark purple background with a vertical orange bar on the right side.

The screenshot below shows the editor after the successful addition of an extra line.



This screenshot shows the vi editor in command mode after adding a new line. The top menu bar and status bar are the same as in the previous screenshot. The text 'There are times that try men's souls.' is on the first line, and '~What's the rush Mama bus?' is on the second line. A new empty line has been added below the second line. The status bar at the bottom shows '~Line 2,1' and '-COMMAND MODE-'. The editor window has a dark purple background with a vertical orange bar on the right side.

When the Search and Replace function is called it asks the user to input a string to find, once the string is entered, it looks for the first string that matches and then moves the cursor to this location. Once there, another prompt asks if the user wants to replace the text, if No is selected then the program exits, if Yes is selected then it asks for the string you want to input there and then copies it over the existing information.

```
coffeeahoy@coffeeahoy: ~/Downloads/UPROJ2-master
File Edit View Search Terminal Help
Group 1 FILE: test.txt
~Tis is a test of the search and replace function.
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
You Entered: search
~
Do you want to replace the string located here with another? y/n:
Replacement: 
Line 1,1 -COMMAND MODE-
```

Below is the outcome of the search and replace function.

[illegible]