## horizontal line



Text Editor

21.11.2022

**─**

Atara Rushi

Roll No: 22

Division: E

Branch: Computer Engineering

# Definition

Create a Text Editor With Basic Functionalities.

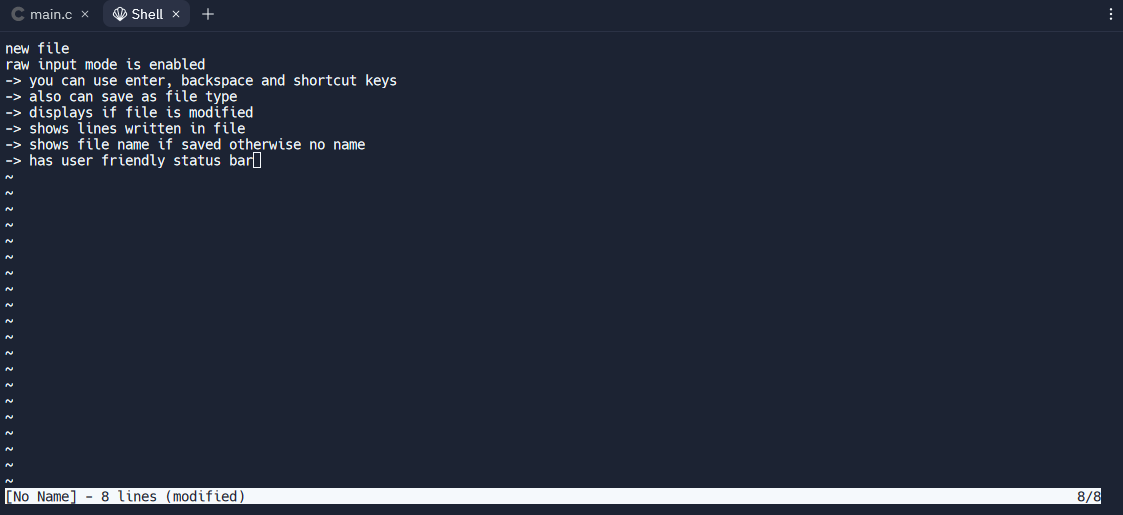
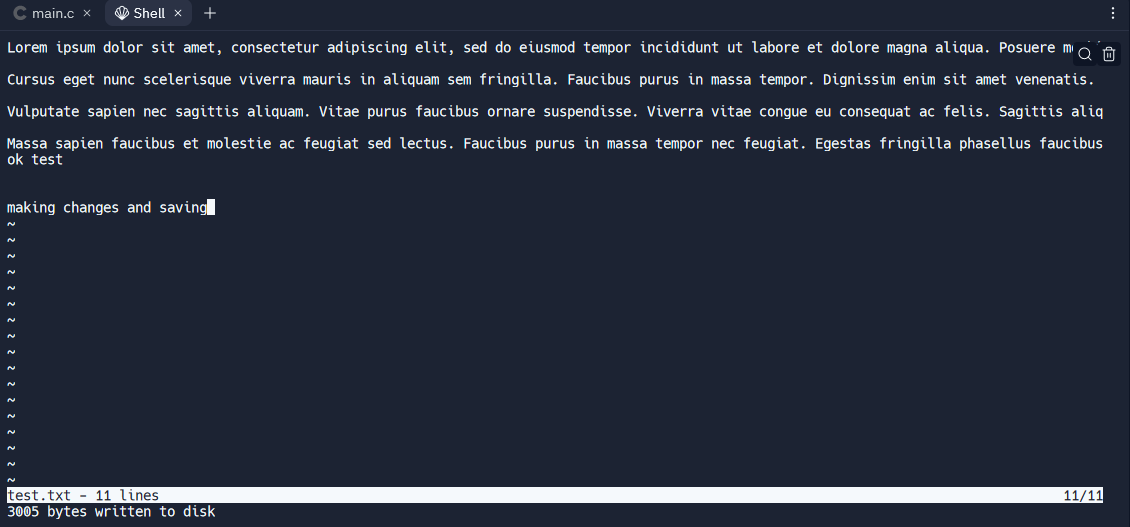
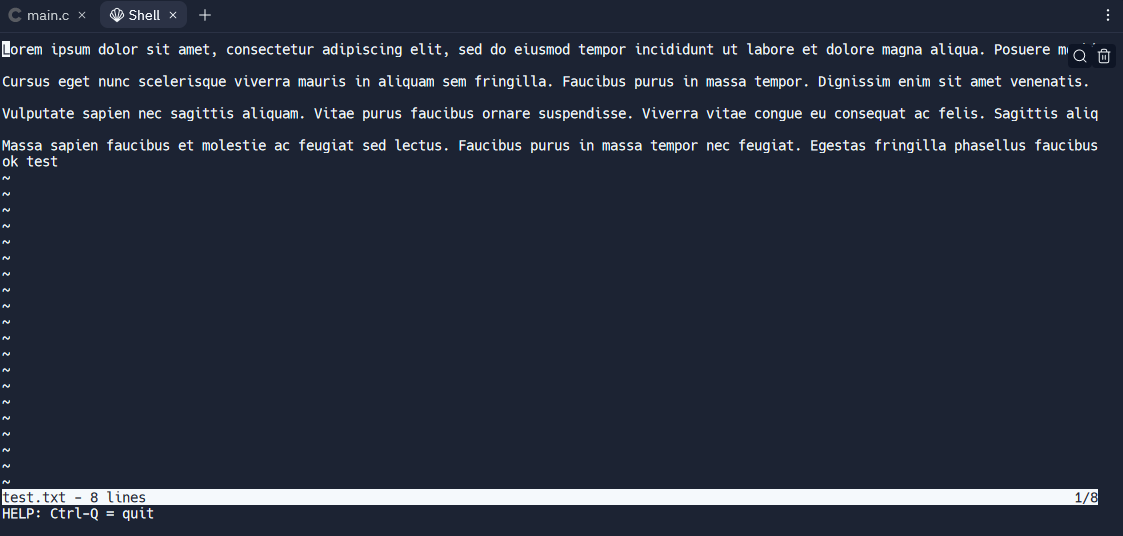
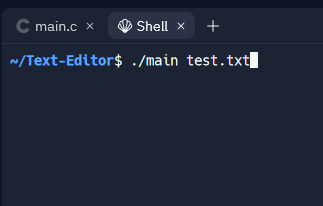
# Functionalities

1. Create File
2. Edit File
3. Open File
4. Save File

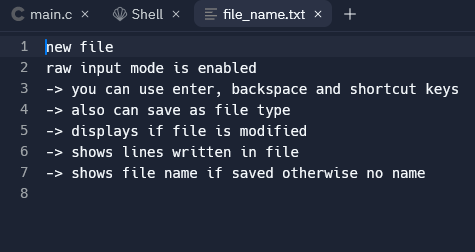
# Program

| **#define \_DEFAULT\_SOURCE;** **#define \_BSD\_SOURCE;** **#define \_GNU\_SOURCE;**  **#include <ctype.h>** **#include <errno.h>** **#include <fcntl.h>** **#include <stdarg.h>** **#include <stdlib.h>** **#include <stdio.h>** **#include <stdlib.h>** **#include <string.h>** **#include <sys/ioctl.h>** **#include <sys/types.h>** **#include <termios.h>** **#include <time.h>** **#include <unistd.h>**  */\* Defines \*/*  **#define CTRL\_KEY(k) ((k) & 0x1f)** **#define MEOW\_VERSION "0.0.1b"** **#define MEOW\_TAB\_STOP 8** **#define MEOW\_QUIT\_CONFIRM\_TIMES 2**  **enum** editor\_keys {  BACKSPACE = 127,  ARROW\_LEFT = 1000,  ARROW\_RIGHT,  ARROW\_UP,  ARROW\_DOWN,  DELETE\_KEY,  HOME\_KEY,  END\_KEY,  PAGE\_UP,  PAGE\_DOWN };  */\* Data \*/*  **typedef** **struct** **editor\_row** {  **int** size;  **int** rsize;  **char** \*characters;  **char** \*render; } editor\_row;  **typedef** **struct** **editor\_configuration\_t** {  **int** x\_coordinate;  **int** y\_coordinate;  **int** rx\_coordinate;  **int** row\_offset;  **int** col\_offset;  **int** screen\_rows;  **int** screen\_cols;  **int** number\_of\_rows;  **int** dirty\_flag;   **char** \*file\_name;  **char** status\_message[80];  **time\_t** status\_message\_time;  editor\_row \*row;   **struct** **termios** **original\_termios**; } **editor\_configuration\_t**; **editor\_configuration\_t** editor\_config;  */\*  Function Prototypes \*/*  **void** **editor\_set\_status\_message**(**const** **char** \*fmt, ...); **void** **editor\_refresh\_screen**(); **char** \***editor\_prompt**(**char** \*prompt);  */\*  Terminal \*/*  **void** **die**(**const** **char** \*s) {  write(STDOUT\_FILENO, "\x1b[2J", 4);  write(STDOUT\_FILENO, "\x1b[H", 3);  perror(s);  exit(1); }  **void** **disable\_raw\_mode**(**void**) {  tcsetattr(STDIN\_FILENO, TCSAFLUSH, &editor\_config.original\_termios);  die("tcsetattr");  **return**; }  **void** **enable\_raw\_mode**(**void**) {  **if**(tcgetattr(STDIN\_FILENO, &editor\_config.original\_termios) == -1) die("tcgetattr");  atexit(disable\_raw\_mode);    **struct** **termios** **raw** = **editor\_config**.**original\_termios**;  raw.c\_iflag &= ~(BRKINT, ICRNL | INPCK | ISTRIP | IXON);  raw.c\_oflag &= ~(OPOST);  raw.c\_cflag |= (CS8);  raw.c\_lflag &= ~(ECHO | ICANON | IEXTEN | ISIG);  raw.c\_cc[VMIN] = 0;  raw.c\_cc[VTIME] = 1;    **if**(tcsetattr(STDIN\_FILENO, TCSAFLUSH, &raw) == -1) die("tcsetattr");  **return**; }  **int** **read\_key**(**void**) {  **int** nread;  **char** c;  **while**((nread = read(STDIN\_FILENO, &c, 1)) != 1) {  **if**(nread == -1 && errno != EAGAIN) die("read");  }   **if**(c == '\x1b') {  **char** sequence[3];   **if**(read(STDIN\_FILENO, &sequence[0], 1) != 1) {  **return** '\x1b';  }  **if**(read(STDIN\_FILENO, &sequence[1], 1) != 1) {  **return** '\x1b';  }   **if**(sequence[0] == '[') {  **if**(sequence[1] >= '0' && sequence[1] <= '9') {  **if**(read(STDIN\_FILENO, &sequence[2], 1) != 1) {  **return** '\x1b';  }  **if**(sequence[2] == '~') {  **switch**(sequence[1]) {  **case** '1': **return** HOME\_KEY;  **case** '3': **return** DELETE\_KEY;  **case** '4': **return** END\_KEY;  **case** '5': **return** PAGE\_UP;  **case** '6': **return** PAGE\_DOWN;  **case** '7': **return** HOME\_KEY;  **case** '8': **return** END\_KEY;  }  }  } **else** {  **switch**(sequence[1]) {  **case** 'A': **return** ARROW\_UP;  **case** 'B': **return** ARROW\_DOWN;  **case** 'C': **return** ARROW\_RIGHT;  **case** 'D': **return** ARROW\_LEFT;  **case** 'H': **return** HOME\_KEY;  **case** 'F': **return** END\_KEY;  }   }  } **else** **if**(sequence[0] == '0') {  **switch**(sequence[1]) {  **case** 'H': **return** HOME\_KEY;  **case** 'F': **return** END\_KEY;  }  }   **return** '\x1b';  } **else** {  **return** c;  } }  **int** **get\_cursor\_position**(**int** \*rows, **int** \*cols) {  **char** buffer[32];  **unsigned** **int** i = 0;    **if**(write(STDOUT\_FILENO, "\x1b[6n", 4) != 4) {  **return** -1;  }   **while**(i < **sizeof**(buffer) - 1) {  **if**(read(STDIN\_FILENO, &buffer[i], 1) != 1) {  **break**;  }  **if**(buffer[i] == 'R') {  **break**;  }  i++;  }  buffer[i] = '\0';    **if**(buffer[0] != '\x1b' || buffer[1] != '[') {  **return** -1;  }  **if**(sscanf(&buffer[2], "%d;%d", rows, cols) != 2) {  **return** -1;  }    **return** 0; }  **int** **get\_window\_size**(**int** \*rows, **int** \*cols) {  **struct** **winsize** **ws**;  **if**(ioctl(STDOUT\_FILENO, TIOCGWINSZ, &ws) == -1 || ws.ws\_col == 0) {  **if**(write(STDOUT\_FILENO, "\x1b[999C\x1b[999B", 12) != 12) {  **return** -1;  }    **return** get\_cursor\_position(rows, cols);  } **else** {  \*cols = ws.ws\_col;  \*rows = ws.ws\_row;  **return** 0;  }  **return** -1; }  */\*  ROW Operations \*/*  **int** **editor\_row\_char\_idx\_to\_render\_idx**(editor\_row \*row, **int** x\_coordinate) {  **int** rx\_coordinate = 0;  **for**(**int** j = 0; j < x\_coordinate; j++) {  **if**(row -> characters[j] == '\t') {  rx\_coordinate += (MEOW\_TAB\_STOP - 1) - (rx\_coordinate % MEOW\_TAB\_STOP);  }  rx\_coordinate++;  }  **return** rx\_coordinate; }  **void** **editor\_update\_row**(editor\_row \*row) {  **int** tabs = 0;  **for**(**int** j = 0;j < row -> size; j++) {  **if**(row -> characters[j] == '\t') {  tabs++;  }  }    free(row -> render);  row -> render = malloc(row -> size + tabs \* (MEOW\_TAB\_STOP - 1) + 1);   **int** idx = 0;  **for**(**int** j = 0; j < row -> size; j++) {  **if**(row -> characters[j] == '\t') {  row -> render[idx++] = ' ';  **while**(idx % MEOW\_TAB\_STOP != 0) {  row -> render[idx++] = ' ';   }  } **else** {  row -> render[idx++] = row -> characters[j];  }   }    row -> render[idx] = '\0';  row -> rsize = idx;   **return**; }  **void** **editor\_insert\_row**(**int** at, **char** \*string, **size\_t** length) {  **if**(at < 0 || at > editor\_config.number\_of\_rows) {  **return**;  }  editor\_config.row = realloc(editor\_config.row, **sizeof**(editor\_row) \* (editor\_config.number\_of\_rows + 1));  memmove(&editor\_config.row[at + 1], &editor\_config.row[at], **sizeof**(editor\_row) \* (editor\_config.number\_of\_rows - at));    editor\_config.row[at].size = length;  editor\_config.row[at].characters = malloc(length + 1);  memcpy(editor\_config.row[at].characters, string, length);  editor\_config.row[at].characters[length] = '\0';    editor\_config.row[at].rsize = 0;  editor\_config.row[at].render = NULL;   editor\_update\_row(&editor\_config.row[at]);   editor\_config.number\_of\_rows++;  editor\_config.dirty\_flag++;  **return**; }  **void** **editor\_free\_row**(editor\_row \*row) {  free(row -> render);  free(row -> characters);  **return**; }  **void** **editor\_delete\_row**(**int** at) {  **if**(at < 0 || at >= editor\_config.number\_of\_rows) {  **return**;  }   editor\_free\_row(&editor\_config.row[at]);  memmove(&editor\_config.row[at], &editor\_config.row[at + 1], **sizeof**(editor\_row) \* (editor\_config.number\_of\_rows - at - 1));  editor\_config.number\_of\_rows--;  editor\_config.dirty\_flag++;   **return**; }  **void** **editor\_row\_insert\_char**(editor\_row \*row, **int** at, **int** c) {  **if**(at < 0 || at > row -> size) {  at = row -> size;  }   row -> characters = realloc(row -> characters, row -> size + 2);  memmove(&row -> characters[at + 1], &row -> characters[at], row -> size - at + 1);  row -> size++;  row -> characters[at] = c;  editor\_update\_row(row);  editor\_config.dirty\_flag++;    **return**; }  **void** **editor\_row\_append\_string**(editor\_row \*row, **char** \*string, **size\_t** length) {  row -> characters = realloc(row -> characters, row -> size + length + 1);  memcpy(&row -> characters[row -> size], string, length);  row -> size += length;  row -> characters[row -> size] = '\0';  editor\_update\_row(row);  editor\_config.dirty\_flag++;   **return**; }  **void** **editor\_row\_delete\_char**(editor\_row \*row, **int** at) {  **if**(at < 0 || at >= row -> size) {  **return**;  }   memmove(&row -> characters[at], &row -> characters[at + 1], row -> size - at);  row -> size--;  editor\_update\_row(row);   editor\_config.dirty\_flag++;    **return**; }  */\*  EDITOR OPERATIONS \*/*  **void** **editor\_insert\_char**(**int** c) {  **if**(editor\_config.y\_coordinate == editor\_config.number\_of\_rows) {  editor\_insert\_row(editor\_config.number\_of\_rows, "", 0);  }  editor\_row\_insert\_char(&editor\_config.row[editor\_config.y\_coordinate], editor\_config.x\_coordinate, c);  editor\_config.x\_coordinate++;   **return**; }  **void** **editor\_insert\_new\_line**(**void**) {  **if**(editor\_config.x\_coordinate == 0) {  editor\_insert\_row(editor\_config.y\_coordinate, "", 0);  } **else** {  editor\_row \*row = &editor\_config.row[editor\_config.y\_coordinate];  editor\_insert\_row(editor\_config.y\_coordinate + 1, &row -> characters[editor\_config.x\_coordinate], row -> size - editor\_config.x\_coordinate);  row = &editor\_config.row[editor\_config.y\_coordinate];  row -> size = editor\_config.x\_coordinate;  row -> characters[row -> size] = '\0';  editor\_update\_row(row);  }   editor\_config.y\_coordinate++;  editor\_config.x\_coordinate = 0; }  **void** **editor\_delete\_char**(**void**) {  **if**(editor\_config.y\_coordinate == editor\_config.number\_of\_rows) {  **return**;  }   **if**(editor\_config.x\_coordinate == 0 && editor\_config.y\_coordinate == 0) {  **return**;  }   editor\_row \*row = &editor\_config.row[editor\_config.y\_coordinate];  **if**(editor\_config.x\_coordinate > 0) {  editor\_row\_delete\_char(row, editor\_config.x\_coordinate - 1);  editor\_config.x\_coordinate--;  } **else** {  editor\_config.x\_coordinate = editor\_config.row[editor\_config.y\_coordinate - 1].size;  editor\_row\_append\_string(&editor\_config.row[editor\_config.y\_coordinate - 1], row -> characters, row -> size);  editor\_delete\_row(editor\_config.y\_coordinate);  editor\_config.y\_coordinate--;  }  **return**; }  */\*  FILE I/O \*/*  **char** \***editor\_rows\_to\_string**(**int** \*buffer\_length) {  **int** total\_length = 0;  **for**(**int** j = 0; j < editor\_config.number\_of\_rows; j++) {  total\_length += editor\_config.row[j].size + 1;  }  \*buffer\_length = total\_length;   **char** \*buffer = malloc(total\_length);  **char** \*p = buffer;   **for**(**int** j = 0; j < editor\_config.number\_of\_rows; j++) {  memcpy(p, editor\_config.row[j].characters, editor\_config.row[j].size);  p += editor\_config.row[j].size;  \*p = '\n';  p++;  }   **return** buffer; }  **void** **editor\_open\_file**(**char** \*file\_name) {  free(editor\_config.file\_name);  editor\_config.file\_name = strdup(file\_name);    FILE \*fp = fopen(file\_name, "r");  **if**(!fp) {  die("open");  }   **char** \*line = NULL;  **size\_t** line\_cap = 0;  **ssize\_t** line\_length;   **while**((line\_length = getline(&line, &line\_cap, fp)) != -1) {  **while**(line\_length > 0 &&  (line[line\_length - 1] == '\n' ||  line[line\_length - 1] == '\r')) {  line\_length--;  }  editor\_insert\_row(editor\_config.number\_of\_rows, line, line\_length);  }   free(line);  fclose(fp);   editor\_config.dirty\_flag = 0;    **return**; }  **void** **editor\_save\_file**(**void**) {  **if**(editor\_config.file\_name == NULL) {  editor\_config.file\_name = editor\_prompt("Save as: %s (ESC to Cancel)");  **if**(editor\_config.file\_name == NULL) {  editor\_set\_status\_message("Save Aborted");  **return**;  }  }   **int** length;  **char** \*buffer = editor\_rows\_to\_string(&length);   **int** fd = open(editor\_config.file\_name, O\_RDWR | O\_CREAT, 0644);   **if**(fd != -1) {  **if**(ftruncate(fd, length) != -1) {  **if**(write(fd, buffer, length) == length) {  close(fd);  free(buffer);  editor\_config.dirty\_flag = 0;  editor\_set\_status\_message("%d bytes written to disk", length);  **return**;  }  }  close(fd);  }    free(buffer);  editor\_set\_status\_message("Can't save! I/O Error: %s", strerror(errno));  **return**; }  */\*  Append Buffer \*/*  **typedef** **struct** **append\_buffer\_t** {  **char** \*buffer;  **int** length; } **append\_buffer\_t**;   **#define APPEND\_BUFFER\_INIT { NULL, 0 }**  **void** **append\_buffer\_append**(**append\_buffer\_t** \*append\_buffer, **const** **char** \*string, **int** length) {  **char** \***new** = realloc(append\_buffer -> buffer, append\_buffer -> length + length);   **if**(**new** == NULL) {  **return**;  }   memcpy(&**new**[append\_buffer -> length], string, length);  append\_buffer -> buffer = **new**;  append\_buffer -> length += length;   **return**; }  **void** **append\_buffer\_free**(**append\_buffer\_t** \*append\_buffer) {  free(append\_buffer -> buffer);  **return**; }   */\*  Input \*/*  **char** \***editor\_prompt**(**char** \*prompt) {  **size\_t** buffer\_size = 128;  **char** \*buffer = malloc(buffer\_size);   **size\_t** buffer\_length = 0;  buffer[0] = '\0';   **while**(1) {  editor\_set\_status\_message(prompt, buffer);  editor\_refresh\_screen();   **int** c = read\_key();  **if**(c == DELETE\_KEY || c == CTRL\_KEY('h') || c == BACKSPACE) {  **if**(buffer\_length != 0) {  buffer[--buffer\_length] = '\0';  }  } **else** **if**(c == '\x1b') {  editor\_set\_status\_message("");  free(buffer);  **return** NULL;  } **else** **if**(c == '\r') {  **if**(buffer\_length != 0) {  editor\_set\_status\_message("");  **return** buffer;  }  } **else** **if**(!iscntrl(c) && c < 128) {  **if**(buffer\_length == buffer\_size - 1) {  buffer\_size \*= 2;  buffer = realloc(buffer, buffer\_size);  }   buffer[buffer\_length++] = c;  buffer[buffer\_length] = '\0';  }  }  **return** buffer; }  **void** **move\_cursor**(**int** key) {  editor\_row \*row = (editor\_config.y\_coordinate >= editor\_config.number\_of\_rows) ? NULL : &editor\_config.row[editor\_config.y\_coordinate];    **switch**(key) {  **case** ARROW\_LEFT:  **if**(editor\_config.x\_coordinate != 0) {  editor\_config.x\_coordinate -= 1;  } **else** **if**(editor\_config.y\_coordinate > 0) {  editor\_config.y\_coordinate--;  editor\_config.x\_coordinate = editor\_config.row[editor\_config.y\_coordinate].size;  }  **break**;  **case** ARROW\_RIGHT:  **if**(row && editor\_config.x\_coordinate < row -> size) {  editor\_config.x\_coordinate += 1;   } **else** **if**(row && editor\_config.x\_coordinate == row -> size) {  editor\_config.y\_coordinate++;  editor\_config.x\_coordinate = 0;  }  **break**;  **case** ARROW\_DOWN:  **if**(editor\_config.y\_coordinate < editor\_config.number\_of\_rows) {  editor\_config.y\_coordinate += 1;  }  **break**;  **case** ARROW\_UP:  **if**(editor\_config.y\_coordinate != 0) {  editor\_config.y\_coordinate -= 1;  }  **break**;  }   row = (editor\_config.y\_coordinate >= editor\_config.number\_of\_rows) ? NULL : &editor\_config.row[editor\_config.y\_coordinate];  **int** row\_length = row ? row -> size : 0;  **if**(editor\_config.x\_coordinate > row\_length) {  editor\_config.x\_coordinate = row\_length;  }    **return**; }  **void** **process\_keypress**(**void**) {  **static** **int** quit\_confirm\_times = MEOW\_QUIT\_CONFIRM\_TIMES;    **int** c = read\_key();   **switch**(c) {  **case** '\r':  editor\_insert\_new\_line();  **break**;    **case** **CTRL\_KEY**('q'):  **if**(editor\_config.dirty\_flag && quit\_confirm\_times > 0) {  editor\_set\_status\_message("WARNING! File has unsaved changes. ""Press Ctrl-Q %d more times to quit.", quit\_confirm\_times);  quit\_confirm\_times--;  **return**;  }  write(STDOUT\_FILENO, "\x1b[2J", 4);  write(STDOUT\_FILENO, "\x1b[H", 3);  exit(0);  **break**;   **case** **CTRL\_KEY**('s'):  **editor\_save\_file**();  **break**;   **case** HOME\_KEY:  editor\_config.x\_coordinate = 0;  **break**;  **case** END\_KEY:  **if**(editor\_config.y\_coordinate < editor\_config.number\_of\_rows) {  editor\_config.x\_coordinate = editor\_config.row[editor\_config.y\_coordinate].size;  }  **break**;   **case** BACKSPACE:  **case** **CTRL\_KEY**('h'):  **case** DELETE\_KEY:  **if**(c == DELETE\_KEY) {  move\_cursor(ARROW\_RIGHT);  }  editor\_delete\_char();  **break**;    **case** PAGE\_UP:  **case** PAGE\_DOWN:  {  **if**(c == PAGE\_UP) {  editor\_config.y\_coordinate = editor\_config.row\_offset;  } **else** **if**(c == PAGE\_DOWN) {  editor\_config.y\_coordinate = editor\_config.row\_offset + editor\_config.screen\_rows - 1;  **if**(editor\_config.y\_coordinate > editor\_config.number\_of\_rows) {  editor\_config.y\_coordinate = editor\_config.number\_of\_rows;  }  }    **int** times = editor\_config.screen\_rows;  **while**(times--) {  move\_cursor(c == PAGE\_UP ? ARROW\_UP : ARROW\_DOWN);  }  }  **break**;    **case** ARROW\_LEFT:  **case** ARROW\_RIGHT:  **case** ARROW\_DOWN:  **case** ARROW\_UP:  move\_cursor(c);  **break**;   **case** **CTRL\_KEY**('l'):  **case** '\x1b':  **break**;    **default**:  editor\_insert\_char(c);  **break**;  }   quit\_confirm\_times = MEOW\_QUIT\_CONFIRM\_TIMES;    **return**; }  */\*  Output \*/*  **void** **editor\_scroll**(**void**) {  editor\_config.rx\_coordinate = 0;  **if**(editor\_config.y\_coordinate < editor\_config.number\_of\_rows) {  editor\_config.rx\_coordinate = editor\_row\_char\_idx\_to\_render\_idx(&editor\_config.row[editor\_config.y\_coordinate], editor\_config.x\_coordinate);  }     **if**(editor\_config.y\_coordinate < editor\_config.row\_offset) {  editor\_config.row\_offset = editor\_config.y\_coordinate;  }   **if**(editor\_config.y\_coordinate >= editor\_config.row\_offset + editor\_config.screen\_rows) {  editor\_config.row\_offset = editor\_config.y\_coordinate - editor\_config.screen\_rows + 1;  }   **if**(editor\_config.rx\_coordinate < editor\_config.col\_offset) {  editor\_config.col\_offset = editor\_config.rx\_coordinate;  }  **if**(editor\_config.rx\_coordinate >= editor\_config.col\_offset + editor\_config.screen\_cols) {  editor\_config.col\_offset = editor\_config.rx\_coordinate - editor\_config.screen\_cols + 1;  } }  **void** **editor\_draw\_rows**(**append\_buffer\_t** \*append\_buffer) {  **for**(**int** y = 0; y < editor\_config.screen\_rows; y++) {  **int** file\_row = y + editor\_config.row\_offset;  **if**(file\_row >= editor\_config.number\_of\_rows) {  **if**(editor\_config.number\_of\_rows == 0 && y == editor\_config.screen\_rows / 3) {  **char** welcome\_message[128];  **int** welcome\_message\_length = snprintf(welcome\_message, **sizeof**(welcome\_message), "Meow Text Editor -- version %s", MEOW\_VERSION);  **if**(welcome\_message\_length > editor\_config.screen\_cols) {  welcome\_message\_length = editor\_config.screen\_cols;  }    **int** padding = (editor\_config.screen\_cols - welcome\_message\_length) / 2;  **if**(padding) {  append\_buffer\_append(append\_buffer, "~", 1);  padding--;  }  **while**(padding--) {  append\_buffer\_append(append\_buffer, " ", 1);  }    append\_buffer\_append(append\_buffer, welcome\_message, welcome\_message\_length);  } **else** {  append\_buffer\_append(append\_buffer, "~", 1);   }  } **else** {  **int** length = editor\_config.row[file\_row].rsize - editor\_config.col\_offset;   **if**(length < 0) {  length = 0;  }  **if**(length > editor\_config.screen\_cols) {  length = editor\_config.screen\_cols;  }    append\_buffer\_append(append\_buffer, &editor\_config.row[file\_row].render[editor\_config.col\_offset], length);  }    append\_buffer\_append(append\_buffer, "\x1b[K", 3);  append\_buffer\_append(append\_buffer, "\r\n", 2);  }    **return**; }  **void** **editor\_draw\_status\_bar**(**append\_buffer\_t** \*append\_buffer) {  append\_buffer\_append(append\_buffer, "\x1b[7m", 4);  **char** status[80];  **char** r\_status[80];  **int** length = snprintf(status, **sizeof**(status), "%.20s - %d lines %s", editor\_config.file\_name ? editor\_config.file\_name : "[No Name]", editor\_config.number\_of\_rows, editor\_config.dirty\_flag ? "(modified)" : "");   **int** r\_length = snprintf(r\_status, **sizeof**(r\_status), "%d/%d", editor\_config.y\_coordinate + 1, editor\_config.number\_of\_rows);    **if**(length > editor\_config.screen\_cols) {  length = editor\_config.screen\_cols;  }   append\_buffer\_append(append\_buffer, status, length);    **while**(length < editor\_config.screen\_cols) {  **if**(editor\_config.screen\_cols - length == r\_length) {  append\_buffer\_append(append\_buffer, r\_status, r\_length);  **break**;  } **else** {  append\_buffer\_append(append\_buffer, " ", 1);  length++;  }  }  append\_buffer\_append(append\_buffer, "\x1b[m", 3);  append\_buffer\_append(append\_buffer, "\r\n", 2);    **return**; }  **void** **editor\_draw\_message\_bar**(**append\_buffer\_t** \*append\_buffer) {  append\_buffer\_append(append\_buffer, "\x1b[K", 3);    **int** message\_length = strlen(editor\_config.status\_message);  **if**(message\_length > editor\_config.screen\_cols) {  message\_length = editor\_config.screen\_cols;  }    **if**(message\_length && time(NULL) - editor\_config.status\_message\_time < 5) {  append\_buffer\_append(append\_buffer, editor\_config.status\_message, message\_length);  }   **return**; }  **void** **editor\_refresh\_screen**(**void**) {  editor\_scroll();    **append\_buffer\_t** append\_buffer = APPEND\_BUFFER\_INIT;   append\_buffer\_append(&append\_buffer, "\x1b[?25l", 6);  append\_buffer\_append(&append\_buffer, "\x1b[H", 3);   editor\_draw\_rows(&append\_buffer);  editor\_draw\_status\_bar(&append\_buffer);  editor\_draw\_message\_bar(&append\_buffer);    **char** buffer[32];  snprintf(buffer, **sizeof**(buffer), "\x1b[%d;%dH", (editor\_config.y\_coordinate - editor\_config.row\_offset) + 1, (editor\_config.rx\_coordinate - editor\_config.col\_offset) + 1);  append\_buffer\_append(&append\_buffer, buffer, strlen(buffer));    append\_buffer\_append(&append\_buffer, "\x1b[?25h", 6);    write(STDOUT\_FILENO, append\_buffer.buffer, append\_buffer.length);    append\_buffer\_free(&append\_buffer);    **return**; }  **void** **editor\_set\_status\_message**(**const** **char** \*fmt, ...) {  va\_list ap;    va\_start(ap, fmt);  vsnprintf(editor\_config.status\_message, **sizeof**(editor\_config.status\_message), fmt, ap);  va\_end(ap);    editor\_config.status\_message\_time = time(NULL);    **return**; }   */\*  Init \*/*  **void** **initialize\_editor**(**void**) {  editor\_config.x\_coordinate = 0;  editor\_config.y\_coordinate = 0;  editor\_config.rx\_coordinate = 0;  editor\_config.row\_offset = 0;  editor\_config.col\_offset = 0;  editor\_config.number\_of\_rows = 0;  editor\_config.dirty\_flag = 0;  editor\_config.row = NULL;  editor\_config.file\_name = NULL;  editor\_config.status\_message[0] = '\0';  editor\_config.status\_message\_time = 0;    **if**(get\_window\_size(&editor\_config.screen\_rows, &editor\_config.screen\_cols) == -1) {  die("get\_window\_size");   }  editor\_config.screen\_rows -= 2; }  **int** **main**(**int** argc, **char** \*argv[]) {  enable\_raw\_mode();  initialize\_editor();  **if**(argc >= 2) {  editor\_open\_file(argv[1]);  }   editor\_set\_status\_message("HELP: Ctrl-Q = quit");    **while**(1) {  editor\_refresh\_screen();  process\_keypress();   }  **return** 0; } |
| --- |

# Output







## 