

**Московский авиационный институт  
(Национальный исследовательский университет)**

Институт: №8 «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Операционные системы»

**Курсовой проект  
Тема: Текстовый редактор**

Студент: Бирюков В. В.

Группа: М80-207Б-19

Преподаватель: Миронов Е. С.

Дата:

Оценка:

Москва, 2020

## Постановка задачи

Целями курсового проекта является приобретение практических навыков в использовании знаний, полученных в течении курса, а также проведение исследования в выбранной предметной области.

Текстовый редактор должен обладать следующим функционалом:

1. Добавление текста.
2. Удаление текста.
3. Просмотр текста.
4. Отображать файл через *mtar*.
5. Возможность задания размера окна *mtar* с помощью ключей запуска программы.
6. Подсчет контрольной суммы по файлу, которая хранится рядом с ним. При несовпадении контрольной суммы вывод ошибки о том, что файл был изменен извне.

## Теоретическая часть

**Текстовый редактор** – самостоятельная компьютерная программа или компонент программного комплекса (например, редактор исходного кода интегрированной среды разработки или окно ввода в браузере), предназначенная для создания и изменения текстовых данных в общем и текстовых файлов в частности.

**Контрольная сумма** – некоторое значение, рассчитанное по набору данных путём применения определённого алгоритма и используемое для проверки целостности данных при их передаче или хранении.

## Алгоритм решения задачи

Для создания интерфейса используется библиотека *ncurses*. Интерфейс представляет из себя пустое окно, в которое выводится текст. Поддерживаются функции вставки и удаления текста по положению курсора, который позиционируется клавишами-стрелками.

Текст файла доступен частями, размер которых может задаваться ключом *-m*. Значение по умолчанию – размер страницы. Из-за ограничений задания сдвига для *mtar* пользовательское значение размера также округляется до ближайшего числа, кратного размеру страницы. В конкретный момент времени доступны редактирование и просмотр одной части файла, при выходе за границы этой части будет подгружена следующая или предыдущая, при этом все внесенные изменения запишутся в файл.

Обязательным аргументом при запуске является имя файла. Если файл не существует — он будет создан.

В качестве контрольной суммы используется 8-битный циклический избыточный код. Файл разбивается на блоки по 8 байт. Контрольной суммой блока считается 16-битная сумма байт, при этом старшие 8 бит добавляются к младшим побитовым XOR, а затем откидываются.

Контрольная сумма хранится в отдельном файле с расширением *.chksm*, проверяется при открытии файла (если существует) и пересчитывается при завершении работы программы.

## Листинг программы

```
// main.cpp
#include <utility>
#include <list>
#include <string>
#include <iostream>
#include <ncurses.h>
#include <ctype.h>
#include <string.h>
#include <stdlib.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>

#include "checksum.hpp"

#define ctrl(k) ((k) & 0x1f)
#define check(VALUE, MSG, BADVAL) if (VALUE == BADVAL) { perror(MSG); exit(1); }

using string_t = std::list<char>;
using text_t = std::list<string_t>;
using cursor_t = std::pair<text_t::iterator, string_t::iterator>;
const int ENTER = 10;
const int TAB_SIZE = 4;

void str2text(char *str, int size, text_t &text) {
    text.clear();
    text.push_back(string_t());
    for (int i = 0; i < size && str[i] != '\0'; ++i) {
        if (str[i] == '\n') {
            text.push_back(string_t());
        } else if (str[i] == '\t') {
            for (int i = 0; i < TAB_SIZE; ++i) {
                text.back().push_back(' ');
            }
        } else {
            text.back().push_back(str[i]);
        }
    }
}

void load(text_t &text, int mmap_size, int fd, int offset) {
    char *chunk = (char *)mmap(NULL, mmap_size, PROT_READ, MAP_PRIVATE, fd, offset);
    check(chunk, "mmap error", MAP_FAILED);
    str2text(chunk, mmap_size, text);
    check(munmap(chunk, mmap_size), "munmap error", -1);
}
```

```

void save(text_t &text, int mmap_size, int fd, int offset, struct stat &file_info) {
    FILE* tmp = tmpfile();
    check(tmp, "tmpfile error", NULL);
    char linebreak = '\n';
    int new_size = text.size()-1;
    for (auto line = text.begin(); line != text.end(); ++line) {
        new_size += line->size();
        for (char ch: *line) {
            fwrite(&ch, 1, 1, tmp);
        }
        if (line != --text.end()) {
            fwrite(&linebreak, 1, 1, tmp);
        }
    }
    if (offset + mmap_size < file_info.st_size) {
        check(lseek(fd, offset+mmap_size, SEEK_SET), "lseek error", -1);
        char c;
        while (read(fd, &c, 1) > 0) {
            fwrite(&c, 1, 1, tmp);
        }
    }
    check(ftruncate(fd, file_info.st_size + new_size - (file_info.st_size
        < mmap_size ? file_info.st_size : mmap_size)), "ftruncate error", -1);
    check(lseek(fd, offset, SEEK_SET), "lseek error", -1);
    check(fseek(tmp, 0, SEEK_SET), "fseek error", -1);
    char c;
    while (fread(&c, 1, 1, tmp) > 0) {
        write(fd, &c, 1);
    }
    fclose(tmp);
    check(lseek(fd, 0, SEEK_SET), "lseek error", -1);
    check(fstat(fd, &file_info), "fstat error", -1);
}

int main(int argc, char *argv[])
{
    std::string filename = "";
    int pagesize = sysconf(_SC_PAGESIZE);
    int mmap_size = pagesize;

    for (int i = 1; i < argc; ++i) {
        if (strcmp(argv[i], "-m") == 0) {
            ++i;
            if (i == argc) {
                std::cerr << "Invalid argument\n";
                exit(1);
            }
            mmap_size = strtol(argv[i], NULL, 10);
            if (mmap_size <= 0) {
                std::cerr << "Invalid argument\n";
                exit(1);
            }
            if (mmap_size % pagesize != 0) {
                mmap_size = pagesize * (mmap_size / pagesize + 1);
            }
        }
        else {
            filename = std::string(argv[i]);
        }
    }
}

```

```

}

if (filename == "") {
    std::cerr << "File name expected\n";
    exit(1);
}

if (checkcs(filename) == 0) {
    std::cerr << "Checksum does not match\n";
    exit(1);
}

text_t text;
int offset = 0;
int fd = open(filename.c_str(), O_RDWR | O_CREAT, S_IRWXU);
check(fd, "open error", -1);
struct stat file_info;
check(fstat(fd, &file_info), "fstat error", -1);

if (file_info.st_size == 0) {
    check(ftruncate(fd, mmap_size), "ftruncate error", -1);
}
load(text, mmap_size, fd, offset);

cursor_t cursor(text.begin(), text.begin()->begin());
int cursor_x = 0, cursor_y = 0;
text_t::iterator start_line = text.begin();

initscr();
raw();
keypad(stdscr, TRUE);
noecho();
int width, height;
getmaxyx(stdscr, height, width);

int key;
while (true) {
    clear();
    move(0, 0);
    int i = 0;
    for (auto line = start_line; line != text.end() && i < height; ++line, ++i) {
        int j = 0;
        for (auto ch = line->begin(); ch != line->end() && j < width-1; ++ch, ++j) {
            addch(*ch);
        }
        printw("\n\r");
    }
    move(cursor_y, cursor_x);
    refresh();

    key = getch();

    if (key == ctrl('q')) {
        break;
    }

    if (key == ctrl('s')) {
        save(text, mmap_size, fd, offset, file_info);
    }
}

```

```

else if (key == KEY_UP) {
    if (cursor.first != text.begin()) {
        --cursor.first;
        cursor.second = cursor.first->begin();
        int i;
        for (i = 0; cursor.second != cursor.first->end()
            && i < cursor_x; ++i, ++cursor.second);
        if (i != cursor_x) {
            cursor_x = i;
        }
        if (cursor_y > 0) {
            --cursor_y;
        } else if (start_line != text.begin()) {
            --start_line;
        }
    } else if (offset > 0) {
        save(text, mmap_size, fd, offset, file_info);
        offset -= mmap_size;
        load(text, mmap_size, fd, offset);
        cursor.first = --text.end();
        cursor.second = cursor.first->begin();
        int i;
        for (i = 0; cursor.second != cursor.first->end()
            && i < cursor_x; ++i, ++cursor.second);
        if (i != cursor_x) {
            cursor_x = i;
        }
        start_line = --text.end();
        for (cursor_y = 0; start_line != text.begin()
            && cursor_y < height-1; ++cursor_y, --start_line);
    }
}

else if (key == KEY_DOWN) {
    if (cursor.first != --text.end()) {
        ++cursor.first;
        cursor.second = cursor.first->begin();
        int i;
        for (i = 0; cursor.second != cursor.first->end()
            && i < cursor_x; ++i, ++cursor.second);
        if (i != cursor_x) {
            cursor_x = i;
        }
        if (cursor_y < height-1) {
            ++cursor_y;
        } else if (start_line != --text.end()) {
            ++start_line;
        }
    } else if (offset + mmap_size < file_info.st_size) {
        save(text, mmap_size, fd, offset, file_info);
        offset += mmap_size;
        load(text, mmap_size, fd, offset);
        cursor.first = text.begin();
        cursor.second = cursor.first->begin();
        int i;
        for (i = 0; cursor.second != cursor.first->end()
            && i < cursor_x; ++i, ++cursor.second);
        if (i != cursor_x) {
            cursor_x = i;
        }
        cursor_y = 0;
    }
}

```

```

        start_line = text.begin();
    }
}

else if (key == KEY_LEFT) {
    if (cursor.second == cursor.first->begin()
        && cursor.first != text.begin()) {
        --cursor.first;
        cursor.second = cursor.first->end();
    } else if (cursor.second != cursor.first->begin()) {
        --cursor.second;
    } else if (offset > 0) {
        save(text, mmap_size, fd, offset, file_info);
        offset -= mmap_size;
        load(text, mmap_size, fd, offset);
        cursor.first = --text.end();
        cursor.second = cursor.first->end();
        cursor_x = cursor.first->size();
        start_line = --text.end();
        for (cursor_y = 0; start_line != text.begin()
            && cursor_y < height-1; ++cursor_y, --start_line);
        continue;
    } else {
        continue;
    }

    if (cursor_x == 0 && cursor_y > 0) {
        --cursor_y;
        cursor_x = cursor.first->size();
    } else if (cursor_x > 0) {
        --cursor_x;
    } else if (start_line != text.begin()) {
        --start_line;
    }
}

else if (key == KEY_RIGHT) {
    if (cursor.second == cursor.first->end()
        && cursor.first != --text.end()) {
        ++cursor.first;
        cursor.second = cursor.first->begin();
    } else if (cursor.second != cursor.first->end()) {
        ++cursor.second;
    } else if (offset + mmap_size < file_info.st_size) {
        save(text, mmap_size, fd, offset, file_info);
        offset += mmap_size;
        load(text, mmap_size, fd, offset);
        cursor.first = text.begin();
        cursor.second = cursor.first->begin();
        start_line = text.begin();
        cursor_y = 0;
        cursor_x = 0;
        continue;
    } else {
        continue;
    }
}

if ((cursor_x == width-1 || (cursor.second == cursor.first->begin()
    && cursor.first != text.begin())) && cursor_y < height-1) {
    ++cursor_y;
    cursor_x = 0;
}

```

```

    } else if (cursor_x < width-1) {
        ++cursor_x;
    } else if (start_line != --text.end()) {
        ++start_line;
    }
}

else if (key == KEY_BACKSPACE) {
    if (cursor.second == cursor.first->begin()
        && cursor.first != text.begin()) {
        auto prev_line = cursor.first;
        int size = cursor.first->size();
        --prev_line;
        int prev_size = prev_line->size();
        prev_line->insert(prev_line->end(), cursor.first->begin(),
            cursor.first->end());

        text.erase(cursor.first);
        cursor.first = prev_line;
        cursor.second = cursor.first->end();

        if (cursor_x == 0 && cursor_y > 0) {
            --cursor_y;
            cursor_x = prev_size;
        } else if (start_line != text.begin()) {
            --start_line;
        }

        while (size--) {
            --cursor.second;
        }
    } else if (cursor.second != cursor.first->begin()) {
        auto col = cursor.second;
        --cursor.second;
        cursor.first->erase(cursor.second);
        cursor.second = col;
        --cursor_x;
    } else {
        continue;
    }
}

else if (key == ENTER) {
    cursor_x = 0;
    if (cursor_y < height-1) {
        ++cursor_y;
    } else if (start_line != --text.end()) {
        ++start_line;
    }
    auto end = cursor.first->end();
    ++cursor.first;
    text.insert(cursor.first, string_t(cursor.second, end));
    --cursor.first;
    --cursor.first;
    cursor.first->erase(cursor.second, cursor.first->end());
    ++cursor.first;
    cursor.second = cursor.first->begin();
}

else if (key == KEY_RESIZE) {
    getmaxyx(stdscr, height, width);
    cursor_x = 0;

```



```

        cursor_y = 0;
        start_line = text.begin();
        cursor.first = text.begin();
        cursor.second = cursor.first->begin();
    }

    else if (key == '\t') {
        for (int i = 0; i < TAB_SIZE; ++i) {
            cursor.first->insert(cursor.second, ' ');
            if (cursor_x == width-1 && cursor_y < height-1) {
                // ++cursor_y;
                // cursor_x = 0;
            } else {
                ++cursor_x;
            }
        }
    }

    else if (isprint(key)) {
        cursor.first->insert(cursor.second, key);
        if (cursor_x == width-1 && cursor_y < height-1) {
            // ++cursor_y;
            // cursor_x = 0;
        } else {
            ++cursor_x;
        }
    }
}
endwin();

save(text, mmap_size, fd, offset, file_info);
close(fd);
makecs(filename);

return 0;
}

// checksum.hpp
#include <string>
#include <iostream>
#include <stdio.h>
#include <stdint.h>
#include <string.h>
#include <sys/stat.h>

#define check(VALUE, MSG, BADVAL) if (VALUE == BADVAL) { perror(MSG); exit(1); }

uint8_t checksum(uint8_t block[8]) {
    uint16_t res = 0;
    for (int i = 0; i < 8; ++i) {
        res += block[i];
    }
    res = (res ^ (res >> 8)) & ((1u << 8) - 1);
    return static_cast<uint8_t>(res);
}

void makecs(std::string filename) {
    FILE *file = fopen(filename.c_str(), "rb");
    check(file, "fopen error", NULL);
    FILE *cs_file = fopen((filename + ".chksm").c_str(), "wb");
    check(cs_file, "fopen error", NULL);
}

```

```

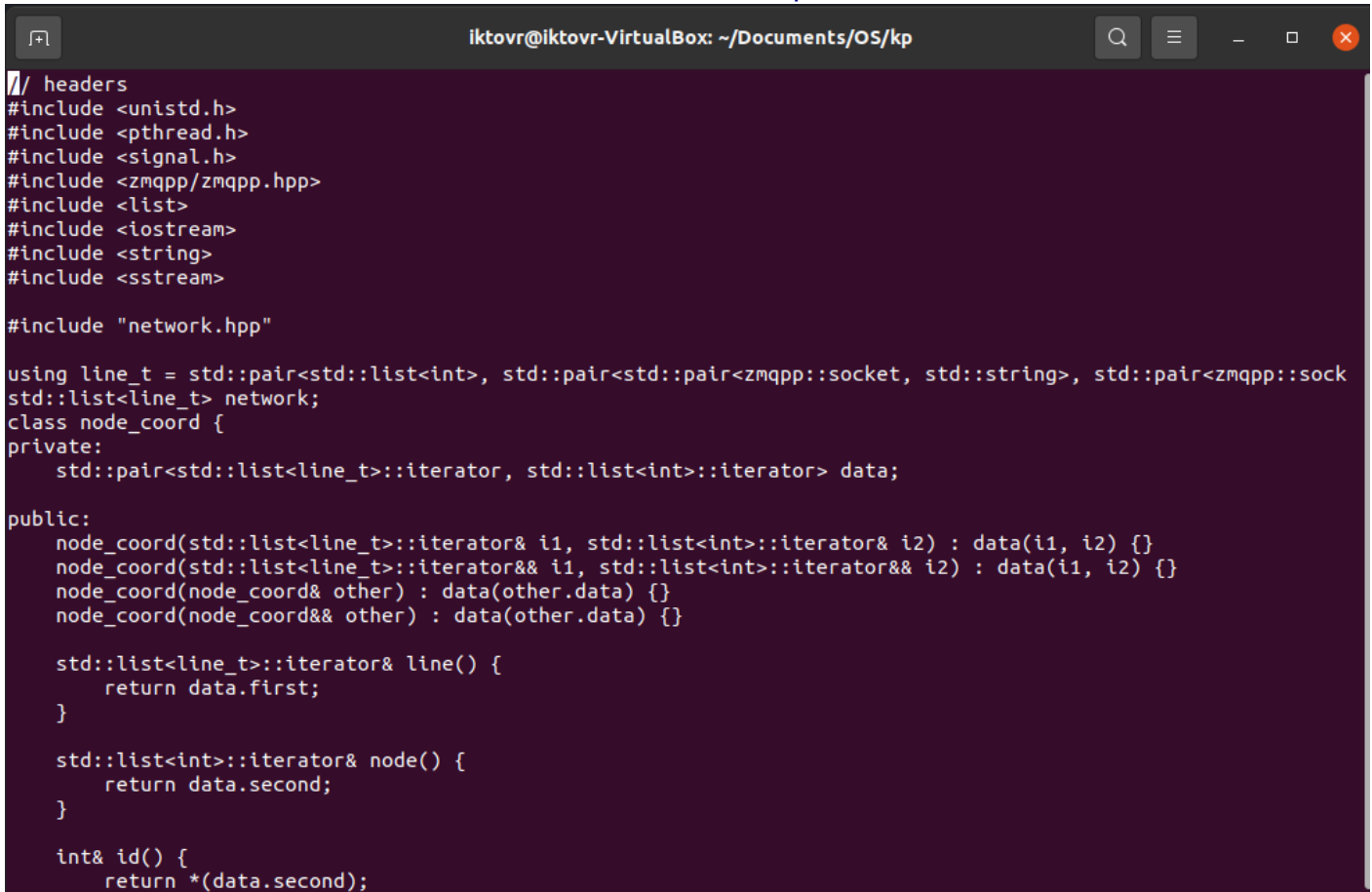
uint8_t block[8];
memset(block, 0, 8);
while (fread(block, 1, 8, file) > 0) {
    uint8_t cs = checksum(block);
    fwrite(&cs, 1, 1, cs_file);
    memset(block, 0, 8);
}
fclose(file);
fclose(cs_file);
}

int checkcs(std::string filename) {
    FILE *file = fopen(filename.c_str(), "rb");
    if (file == NULL) {
        return -1;
    }
    struct stat file_info;
    check(fstat(fileno(file), &file_info), "fstat error", -1);
    FILE *cs_file = fopen((filename + ".chksm").c_str(), "rb");
    if (cs_file == NULL) {
        return -1;
    }
    struct stat cs_info;
    check(fstat(fileno(cs_file), &cs_info), "fstat error", -1);
    if ((file_info.st_size / 8 + (file_info.st_size % 8 > 0 ? 1 : 0))
        != cs_info.st_size) {
        fclose(file);
        fclose(cs_file);
        return 0;
    }
    uint8_t block[8];
    memset(block, 0, 8);
    while (fread(block, 1, 8, file) > 0) {
        uint8_t cs = checksum(block);
        uint8_t cs2;
        if (fread(&cs2, 1, 1, cs_file) < 1) {
            fclose(file);
            fclose(cs_file);
            return 0;
        }
        if (cs != cs2) {
            fclose(file);
            fclose(cs_file);
            return 0;
        }
        memset(block, 0, 8);
    }
    fclose(file);
    fclose(cs_file);
    return 1;
}

```

## Тесты и протокол исполнения

```
iktovr@iktovr-VirtualBox:~/Documents/OS/kp$ ./main new.txt
iktovr@iktovr-VirtualBox:~/Documents/OS/kp$ cat new.txt
hello world
iktovr@iktovr-VirtualBox:~/Documents/OS/kp$ echo "!" >> new.txt
iktovr@iktovr-VirtualBox:~/Documents/OS/kp$ ./main new.txt
Checksum does not match
iktovr@iktovr-VirtualBox:~/Documents/OS/kp$ head -5 < demo.txt
#include <unistd.h>
#include <pthread.h>
#include <zmqpp/zmqpp.hpp>
#include <iostream>
#include <list>
iktovr@iktovr-VirtualBox:~/Documents/OS/kp$ ./main demo.txt
```



```
// headers
#include <unistd.h>
#include <pthread.h>
#include <signal.h>
#include <zmqpp/zmqpp.hpp>
#include <list>
#include <iostream>
#include <string>
#include <sstream>

#include "network.hpp"

using line_t = std::pair<std::list<int>, std::pair<std::pair<zmqpp::socket, std::string>, std::pair<zmqpp::sock
std::list<line_t> network;
class node_coord {
private:
    std::pair<std::list<line_t>::iterator, std::list<int>::iterator> data;

public:
    node_coord(std::list<line_t>::iterator& i1, std::list<int>::iterator& i2) : data(i1, i2) {}
    node_coord(std::list<line_t>::iterator&& i1, std::list<int>::iterator&& i2) : data(i1, i2) {}
    node_coord(node_coord& other) : data(other.data) {}
    node_coord(node_coord&& other) : data(other.data) {}

    std::list<line_t>::iterator& line() {
        return data.first;
    }

    std::list<int>::iterator& node() {
        return data.second;
    }

    int& id() {
        return *(data.second);
    }
};
```

```
iktovr@iktovr-VirtualBox:~/Documents/OS/kp$ head -5 < demo.txt
```

```
// headers
#include <unistd.h>
#include <pthread.h>
#include <signal.h>
#include <zmqpp/zmqpp.hpp>
iktovr@iktovr-VirtualBox:~/Documents/OS/kp$ ./main demo.txt -m 4096
```

```
iktovr@iktovr-VirtualBox: ~/Documents/OS/kp

        if (node.line()->first.empty()) {
            network.erase(node.line());
        }
    }

    default: {}
}

return NULL;
}

zmqpp::context context;

bool ping(zmqpp::socket& ping_sock, std::string& ping_port, int id) {
    int packet[2] = {id, 1};
    size_t length = 2 * sizeof(int);
    if (!ping_sock.send_raw(reinterpret_cast<char *>(packet), length, zmqpp::socket::dont_wait)) {
        return false;
    }
    if (!ping_sock.receive_raw(reinterpret_cast<char *>(packet), length)) {
        ping_sock.close();
        ping_sock = zmqpp::socket(context, zmqpp::socket_type::req);
        ping_sock.set(zmqpp::socket_option::receive_timeout, 1000);
        ping_sock.connect(host + ping_port);
        return false;
    }
    return (packet[1] == 1) ? true : false;
}

int main() {
    // std::cout << getpid() << std::endl;
}

c
```

# первый блок заканчивается в этом месте, на 4096-ом символе

iktovr@iktovr-VirtualBox:~/Documents/OS/kp\$ ./main demo.txt -m 8192

```
iktovr@iktovr-VirtualBox: ~/Documents/OS/kp

        ports >> node.out_port();
        node.out_sock().connect(host + node.out_port());

        ports >> node.ping_port();
        node.ping_sock().connect(host + node.ping_port());
        std::cout << -1 << std::endl;
    } else {
        std::cout << *unavailable.begin();
        for (auto node = ++unavailable.begin(); node != unavailable.end(); ++node) {
            std::cout << "; " << *node;
        }
        std::cout << std::endl;
    }
}

std::cout << "> ";
}

// for (auto& line: network) {
//     zmqpp::socket &out_node = line.second.first.first;
//     std::string &out_port = line.second.first.second;
//     zmqpp::socket &ping_node = line.second.second.first;
//     std::string &ping_port = line.second.second.second;
//     for (int& node: line.first) {
//         if (ping(ping_node, ping_port, node)) {
//             zmqpp::message
//         }
//     }
// }

// topology - 1 - lists
// commands - 2 - exec id name value (exec id name)
// ping - 1 - pingall 1 - p
```

# теперь он заканчивается позже, на 8192-ом

## Выводы

В ходе выполнения курсового проекта я применил знания, полученные в течении курса, а также приобрел новые. Я познакомился с библиотекой *ncurses*, которая позволяет создавать программы с текстовым псевдографическим интерфейсом. Также я узнал еще один способ применения *mmap* – получение части файла. Однако *mmap* не позволяет получить какую угодно часть, так задаваемый сдвиг от начала файла должен быть кратен размеру страницы. Я ближе познакомился с понятием контрольной суммы и реализовал один из простейших алгоритмов ее получения. Контрольная сумма является очень удобным способом проверки целостности файла.

## Список литературы

1. Memory Mapped Files – *Beej's Guide to Unix IPC*.  
URL: <http://beej.us/guide/bgipc/html/multi/mmap.html>
2. NCURSES Programming HOWTO.  
URL: <https://tldp.org/HOWTO/NCURSES-Programming-HOWTO/>
3. Простой расчет контрольной суммы – *Habr*.  
URL: <https://habr.com/ru/post/278171/>
4. Текстовый редактор – *Википедия*.  
URL: [https://ru.wikipedia.org/wiki/Текстовый\\_редактор](https://ru.wikipedia.org/wiki/Текстовый_редактор)
5. Контрольная сумма – *Википедия*.  
URL: [https://ru.wikipedia.org/wiki/Контрольная\\_сумма](https://ru.wikipedia.org/wiki/Контрольная_сумма)