Федеральное агентство связи Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

Факультет: Информатики и вычислительной техники

Кафедра телекоммуникационных сетей и вычислительных средств

Дисциплина: Архитектура ЭВМ

Отчёт по лабораторной работе №3

«Консоль управления моделью Simple Computer. Текстовая часть»

Выполнили студенты группы ИА-831:

Зарубин Максим Евгеньевич

Дорощук Никита Андреевич

Проверил преподаватель:

Токмашева Елизавета Ивановна

Новосибирск

2020

Цель работы

Изучить принципы работы терминалов ЭВМ в текстовом режиме. Понять, каким образом кодируется текстовая информация и как с помощью неѐ можно управлять работой терминалов. Разработать библиотеку функций myTerm, включающую базовые функции по управлению текстовым терминалом (очистка экрана, позиционирование курсора, управления цветом). Начать разрабатывать консоль управления Simple Computer (вывести на экран текстовую часть).

Задание на лабораторную работу

1. Прочитайте главу 5 практикума по курсу «Организация ЭВМ и систем». Обратите особое внимание на параграфы 5.4 и 5.5. Изучите страницу man для команды infocmp, базы terminfo, функции ioctl.

2. Откройте текстовый терминал и запустите оболочку bash (оболочка запускается автоматически). Используя команду infocmp, определите (и перепишите их себе) escapeпоследовательности для терминала, выполняющие следующие действия: очистка экрана и перемещение курсора в левый верхний угол (clear_screen); перемещение курсора в заданную позицию экрана (cursor_address); задание цвета последующих выводимых символов (set_a_background); определение цвета фона для последующих выводимых символов (set_a_foreground); скрытие и восстановление курсора (cursor_invisible, cursor_visible).

3. Используя оболочку bash, команду echo –e и скрипт2, проверьте работу полученных последовательностей. Символ escape задается как \033 или \E. Например – echo -e "\033[m". Для проверки сформируйте последовательность escape-команд, выполняющую следующие действия: очищает экран; выводит в пятой строке, начиная с 10 символа Ваше имя красными буквами на черном фоне; в шестой строке, начиная с 8 символа Вашу группу зеленым цветом на белом фоне; перемещает курсор в 10 строку, 1 символ и возвращает настройки цвета в значения «по умолчанию».

4. Разработать следующие функции:

int mt_clrscr (void)- производит очистку и перемещение курсора в левый верхний угол экрана;

int mt_gotoXY (int, int) - перемещает курсор в указанную позицию. Первый параметр номер строки, второй - номер столбца;

int mt_getscreensize (int * rows, int * cols) - определяет размер экрана терминала (количество строк и столбцов);

int mt_setfgcolor (enum colors) - устанавливает цвет последующих

выводимых символов. В качестве параметра передаѐтся константа из созданного Вами перечислимого типа colors, описывающего цвета терминала;

int mt_setbgcolor (enum colors) - устанавливает цвет фона последующих выводимых символов. В качестве параметра передаѐтся константа из созданного Вами перечислимого типа colors, описывающего цвета терминала. Все функции возвращают 0 в случае успешного выполнения и -1 в случае ошибки. В качестве терминала используется стандартный поток вывода.

5. Оформите разработанные функции как статическую библиотеку myTerm. Подготовьте заголовочный файл для неѐ.

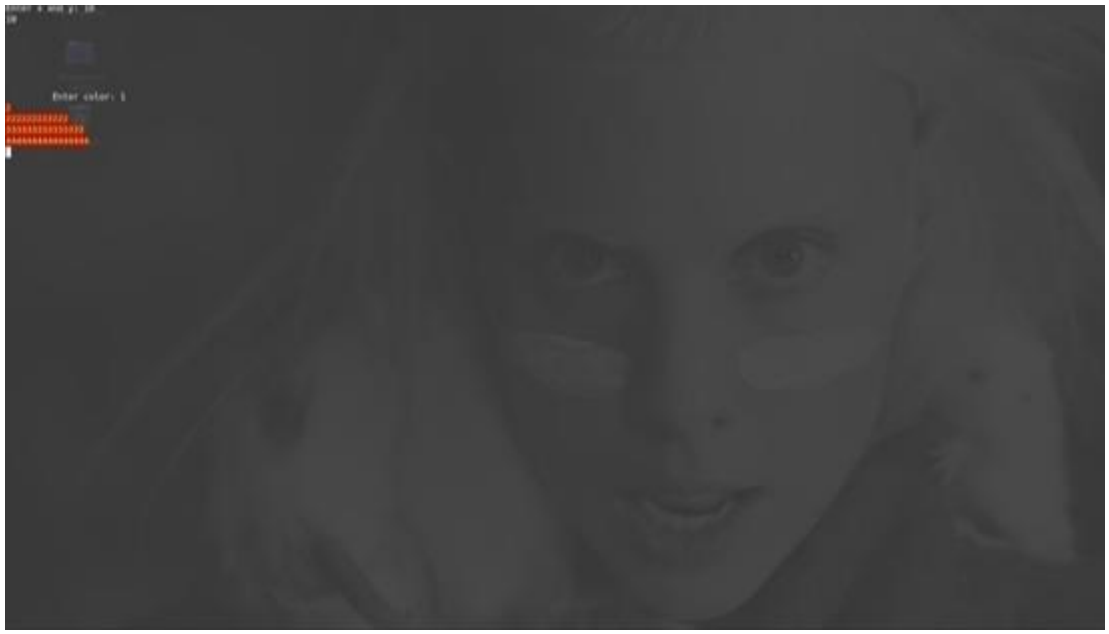
Описание реализованных функций.

1. int mt_clrscr (void) – С помощью printf выводим заранее заготовленную константную последовательность из 6 символов, благодаря которой происходит очистка экрана и перемещение курсора в левый верхний угол экрана.
2. int mt_gotoXY (int, int) - С помощью printf выводим заранее заготовленную константную последовательность и двух переданных в него значений, благодаря которым курсор перемещается в позицию этих значений.
3. int mt_getscreensize (int *rows, int *cols) – Создаём переменную типа struct winsize. С помощью ioctl помещаем значения размера экрана в эту переменную и потом присваиваем эти значения нашим переданным x и y. Если поместить значения не получилось, то функция завершается с ошибкой.
4. int mt_setfgcolor (enum colors) – Сначала проверяем входит ли интовое значение в основной промежуток текстовых терминалов(они содержат в основном до 8 цветов), а после с помощью printf выводим заранее заготовленную константную последовательность с номером переданного значения цвета, благодаря которым устанавливается цвет последующих выводимых символов. Иначе функция завершается с ошибкой.
5. int mt_setbgcolor (enum colors) - Сначала проверяем входит ли интовое значение в основной промежуток текстовых терминалов(они содержат в основном до 8 цветов), а после с помощью printf выводим заранее заготовленную константную последовательность с номером переданного значения цвета, благодаря которым устанавливается цвет

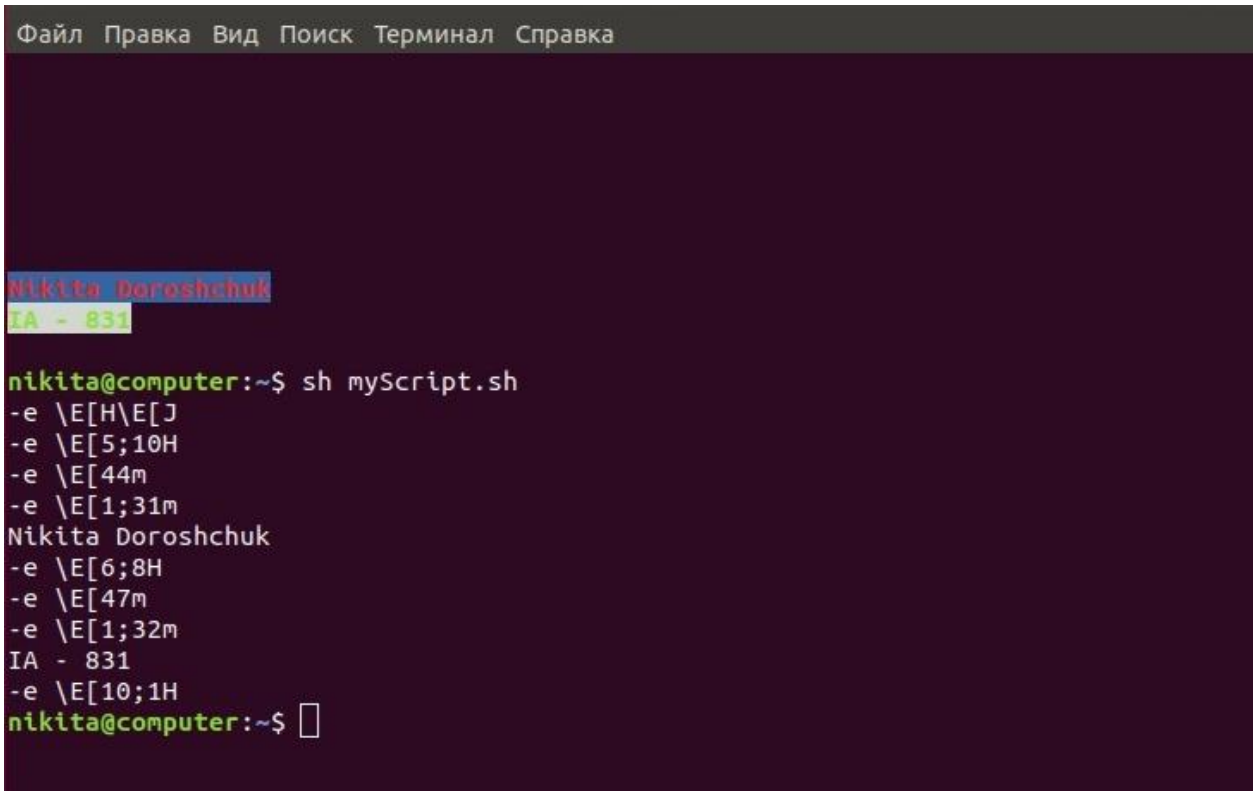
фона последующих выводимых символов. Иначе функция завершается с ошибкой.

- Enum - это тип который может содержать значения указанные программистом. Его ещё называют типом перечисления. Для него характерно перечисление целочисленных именованных констант.

Скриншоты выполнения программы:



Выполнив пункт 3, получим:



Файл MyTerm.h:

```
#ifndef MYTERM_H
#define MYTERM_H
#include <stdio.h>
#include <sys/ioctl.h>
#define CLR_SCR "\E[H\E[J"
#define GOTO_XY "\E[%d;%dH"
#define BG_COLOR "\E[4%dm"
#define FG_COLOR "\E[1;3%dm"
#define TERM "/dev/tty"
enum COLOR {RED = 1, GREEN = 2, YELLOW = 3, DARK_BLUE = 4, PURPLE = 5, BLUE = 6, WHITE = 7, DEFAULT = 9};
int mt_clrscr ();
int mt_gotoXY (int , int);
int mt_setbgColor (enum COLOR);
int mt_setfgColor (enum COLOR);
int mt_getScreenSize (int *, int *);
#endif
```

Файл MyTerm.c:

```
#include "MyTerm.h"
int mt_clrscr () {
        printf(CLR_SCR);
        return 0;
}
int mt_gotoXY (int x, int y) {
        printf(GOTO_XY, x, y);
```

```
        return 0;
}
int mt_setbgColor (enum COLOR color) {
        if ((int)color >= 1 || (int)color <= 7) {
                printf(BG_COLOR, color);
                return 0;
        }
        else return -1;
}
int mt_setfgColor (enum COLOR color) {
        if ((int)color >= 1 || (int)color <= 7) {
                printf(FG_COLOR, color);
                return 0;
        }
        else return -1;
}
int mt_getScreenSize (int *x, int *y) {
        struct winsize ws;
if (!ioctl (1, TIOCGWINSZ, &ws)) {
                *x = ws.ws_row;
                *y = ws.ws_col;
                return 0;
        }
        return -1;
}
```

Файл main.c:

```
#include <stdio.h>
#include "MyTerm.h"
int main()
{
	int test;
	printf("Enter x and y: ");
	int x, y;
	scanf("%d%d", &x, &y);
	int* x1 = &x;
	int* y1 = &y;
	mt_gotoXY(x, y);
	int color;
	printf("Enter color: ");
	scanf("%d", &color);
	mt_setbgColor(color);
	mt_setfgColor(color + 1);
	mt_getScreenSize(x1, y1);
	scanf("%d", &test);
	while(test != 0)
	{
		scanf("%d", &test);
	}
	mt_clrscr();
	return 0;
```

}