

**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 №5**

«Консоль управления моделью Simple Computer. Клавиатура.  
Обработка нажатия клавиш. Неканонический режим работы  
терминала»

Выполнили студенты группы ИА-831:

Зарубин Максим Евгеньевич

Дорощук Никита Андреевич

Проверил преподаватель:

Токмашева Елизавета Ивановна

Новосибирск

2020

## **Задание к лабораторной работе.**

1. Прочитайте главу 5 пособия по курсу «ЭВМ и периферийные устройства». Обратите особое внимание на параграф 5.1. Изучите страницу man для команд `infocmp` и `read`, базы `terminfo`.
2. Используя оболочку `bash` и команду `read`, определите последовательности, формируемые нажатием на буквенно-цифровые, функциональные клавиши и клавиши управления курсором. Используя команду `infocmp`, убедитесь, что получены правильные последовательности символов, генерируемые функциональными клавишами «F5» и «F6».
3. Разработайте функции:  
`int rk_readkey (enum keys *)` анализирует последовательность символов (возвращаемых функцией `read` при чтении с терминала) и возвращает первую клавишу, которую нажал пользователь. В качестве параметра в функцию передаётся адрес переменной, в которую возвращается номер нажатой клавиши (`enum keys` – перечисление распознаваемых клавиш);  
`int rk_mytermstore (void)` сохраняет текущие параметры терминала;  
`int rk_mytermrestore (void)` восстанавливает сохранённые параметры терминала;  
`int rk_mytermregime (int regime, int vtime, int vmin, int echo, int sigint)` переключает терминал между режимами. Для неканонического режима используются значения второго и последующего параметров.
4. Оформите разработанные функции как статическую библиотеку `myReadkey`. Подготовьте заголовочный файл для неё.
5. Для защиты лабораторной работы необходимо подготовить программу, демонстрирующую использование созданной библиотеки функций (сборка программы с библиотекой, использование заголовочного файла, примеры вызовов каждой функции, проверка корректности работы функций при различных входных значениях). Необходимо доработать программу лабораторной работы № 3, выводящую на экран согласно рисунку П2.1 консоль управления Simple Computer так, чтобы можно было задавать значения ячейкам оперативной памяти, регистрам, и обрабатывалось нажатие клавиш «s», «l».

## **Описание реализованных функций.**

`int rk_readkey (enum keys *)` – Переключаем режим терминала с помощью `int rk_mytermregime`. Потом читаем данные из устройства с помощью `read`, передавая туда номер дескриптора, адрес буфера,

куда помещаем прочитанную информацию и максимальный размер этого буфера. После с помощью цикла switch перебираем варианты символов. Если выпадает '\E', то читаем ещё данные с устройства с помощью read 2 раза и с помощью цикла switch перебираем его варианты, такие как F5, F6 и стрелки. В конце функции снова переключаем режим терминала.

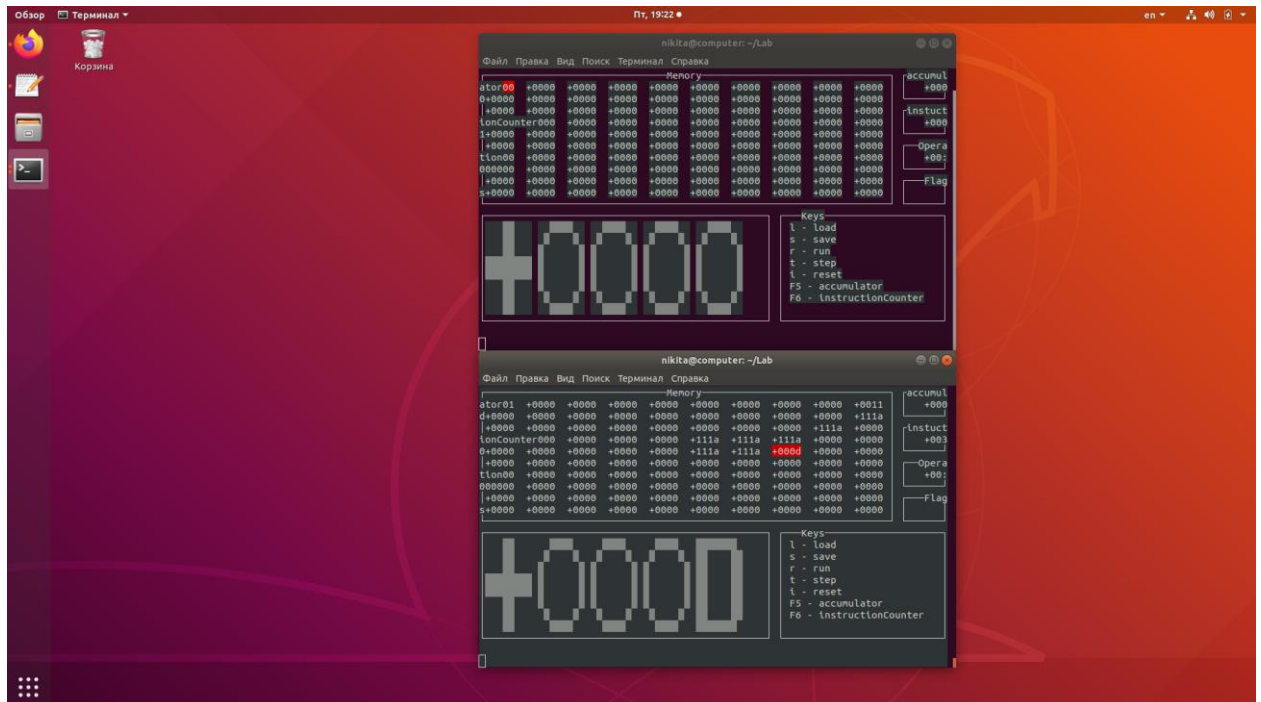
int rk\_mytermsave (void) – чтобы сохранить текущие параметры терминала, используем вызов tcsetattr, с переданными туда номером дескриптора файла и адресом памяти, куда поместить структуру, описывающую режимы работы терминала. Если возникнет ошибка, функция вернёт -1, иначе 0.

int rk\_mytermrestore (void) – чтобы восстановить сохраненные параметры терминала, используем вызов tcsetattr, с переданными туда номером дескриптора файла, правила их замены, в нашем случае TCSADRAIN и параметры, которые хотим восстановить.

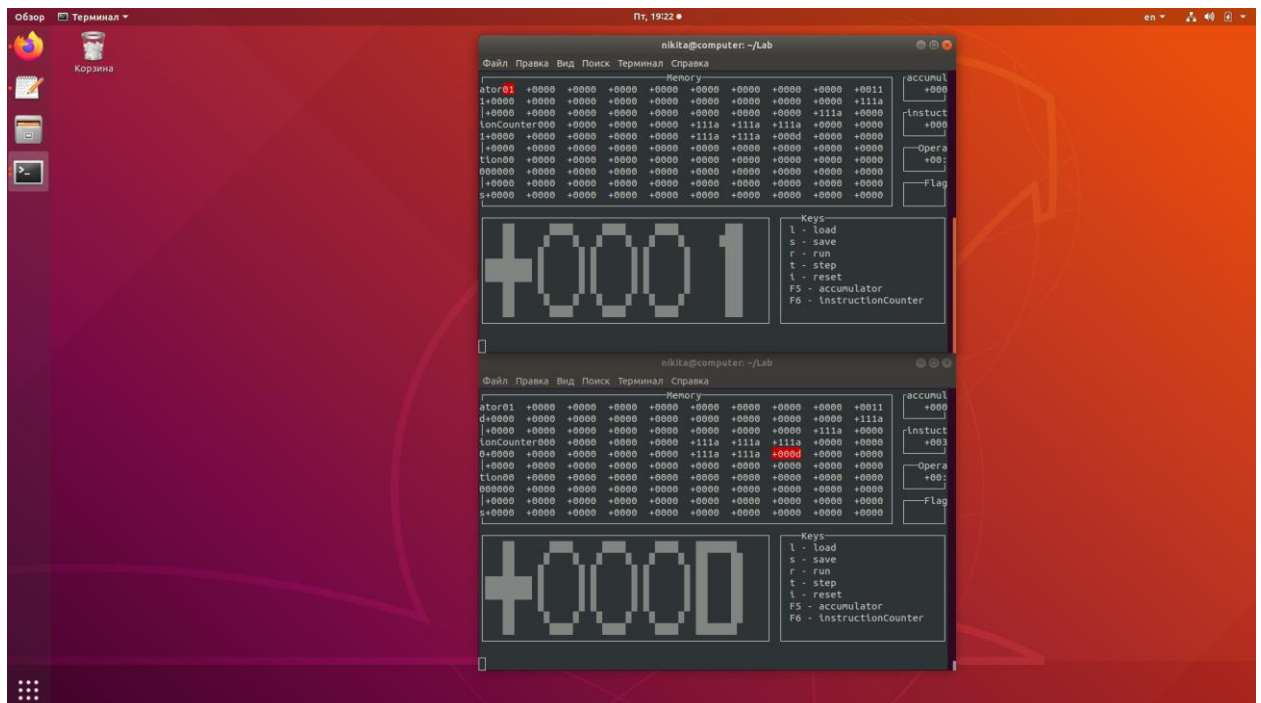
int rk\_mytermregime (int regime, int vtime, int vmin, int echo, int sigint) – Сначала проверяем на правильность переданных значений параметров regime, echo и sight, если они не входят в границы, то функция завершается с ошибкой. Далее создадим переменную типа struct termios, в которую присвоим текущие параметры терминала. Потом проверяем параметры regime, echo и sight, переданные в функцию. Если единица, то устанавливаем значения флага в единицу, если нуль, то устанавливаем значения флага в нуль. А vtime и vmin просто заменяем вместо предыдущих их значений. И устанавливаем новые параметры с помощью tcsetattr, только теперь в правила их замены пишем TCSANOW.

## Скриншоты проверки работы функций.

В нижнем терминале меняем значения ячеек и нажимаем 's'(save)



Далее в верхнем терминале нажимаем клавишу l(load) и в нем загружается сохраненная конфигурация терминала.



## Листинг программы.

### **myReadkey.h**

```
struct termios term;

enum keys {undefined=0, l=1, s=2, r=3, t=4, i=5, f5=6,
f6=7, down=8, up=9, k_left=10, k_right=11, enter=12,
esc=13 };

int rk_readkey(enum keys *k);

int rk_mytermsave(void);

int rk_mytermregime(int regime,int vtime,int vmin,int
echo,int sigint);
```

### **myReadkey.c**

```
#include <stdio.h>
#include <sys/types.h>
#include <string.h>
#include <stdlib.h>
#include <termios.h>
#include <unistd.h>
#include <myReadkey.h>

struct termios term;

enum keys {undefined=0, l=1, s=2, r=3, t=4, i=5, f5=6,
f6=7, down=8, up=9, k_left=10, k_right=11, enter=12,
esc=13 };

int rk_readkey(enum keys *k){
    char c[2];
    *k=undefined;
    int j=0,f=2;
    while(1) {
        read(1,&c,2);
        switch(j) {
            case 0: switch(c[0]) {
```

```

        case 'l': *k=l; break;
        case 's': *k=s; break;
        case 'r': *k=r; break;
        case 't': *k=t; break;
        case 'i': *k=i; break;
        case '\n': *k=enter; break;
    }
    if(c[1]=='[') {j++; continue;}
    break;
case 1: switch(c[0]) {
        case 'A': *k=up; break;
        case 'B': *k=down; break;
        case 'C': *k=k_right; break;
        case 'D': *k=k_left; break;
        case '1': if(c[1]=='5') *k=f5;
                  if(c[1]=='7') *k=f6;
                  read(0,&c,1); break;
    }
    break;
    }
break;
}
return 0;
}

```

```

int rk_mytermsave(void){
    if((tcgetattr(1, &term))== -1)
    {

```

```

        perror("Error: tcgetattr");
        return -1;
    }
    return 0;
}

```

```

int rk_mytermload(void) {
    if((tcsetattr(1, TCSANOW, &term)) < 0)
    {
        perror("Error: tcsetattr");
        return -1;
    }
    return 0;
}

```

```

int rk_mytermregime(int regime,int vtime,int vmin,int
echo,int sigint){
    struct termios myterm;

    tcgetattr(1,&myterm);

    if(regime) myterm.c_lflag |= ICANON;
    else myterm.c_lflag &= ~ICANON;

    if(echo) myterm.c_lflag |= ECHO;
    else myterm.c_lflag &= ~ECHO;

    if(sigint) myterm.c_lflag |= ISIG;
    else myterm.c_lflag &= ~ISIG;
}

```

```

    myterm.c_cc[VMIN] = vmin;
    myterm.c_cc[VTIME] = vtime;

    return tcsetattr(1, TCSANOW, &myterm);
}

```

### **Main.c**

```

#include <sys/types.h>
#include <stdio.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <termios.h>
#include <unistd.h>
#include "myTerminal.h"
#include "my_bigchar.h"
#include "sc_memory.h"
#include "myReadkey.h"

#define P 0
#define O 1
#define M 2
#define T 3
#define E 4

int verifMemory(int t){
    int op, com;

    sc_commandDecode(t, &com, &op);

    int mass[40] =
{0x10,0x11,0x20,0x21,0x30,0x31,0x32,0x33,0x40,0x41,0x42
,0x43,0x51,0x52,0x53,0x54,0x55,0x56,0x57,0x58,0x59,0x60
,0x61,0x62,

```

```
0x63,0x64,0x65,0x66,0x67,0x68,0x69,0x70,0x71,0x72,0x73,  
0x74,0x75,0x76};
```

```
    int status = -1;  
  
    int i;  
    for(i = 0; i < 40; i++){  
        if(2*mass[i] == com){  
            status = 0;  
            break;  
        }  
    }  
    return status;  
}
```

```
int main(){  
    int iC = 1;  
    int status = 1,t;  
    int statusAc = 1, t1;  
    int statusIc = 1;  
    char temp[4], temp1[4];  
    sc_regInit();  
    mt_setfgcolor(WHITE);  
  
    int r, c;  
    mt_getscreensize(&r,&c);  
    sc_memoryInit();  
    metka:  
    mt_clrscr();  
    bc_box(0,0,13,72);  
    bc_box(0,73,4,75);
```

```

bc_box(4,73,3,75);
bc_box(7,73,3,75);
bc_box(10,73,3,75);
bc_box(13,52,10,75);
bc_box(13,0,10,51);
mt_gotoXY(1,4*7+5);
write(1,"Memory",6);
int i,j = 0,value,k=2;
    for(i = 1; i <= 100; i++,j++){
        sc_memoryGet(i,&value);
        if(i == iC && statusIc == 1){
            mt_setbcolor(RED);
        }
        mt_gotoXY(k,j*7+2);
        char buf1[8];
        int l1 = sprintf(buf1,"+%.4x",value);
        write(1,buf1,5);
        if(i%10==0){
            k++;
            j = -1;
        }
        if(i == iC && statusIc == 1){
            mt_setbcolor(BLACK);
        }
    }
mt_gotoXY(1,11*7-3);
write(1,"accumulator",11);
mt_gotoXY(2,11*7);
if(statusAc == 1){

```

```

        sc_memoryGet(iC, &value);
        char buf3[8];
        int l3 = sprintf(buf3, "+%.4x", value);
        write(1, buf3, l3);
    } else if(statusAc == 0){
        char buf3[8];
        int l3 = sprintf(buf3, "+%.4x", t1);
        write(1, buf3, l3);
    }
    mt_gotoXY(4, 11*7-3);
    write(1, "instuctionCounter", 17);
    mt_gotoXY(5, 11*7);
    sc_memoryGet(iC, &value);
    char buf2[8];
    int l2 = sprintf(buf2, "+%.4x", iC);
    write(1, buf2, l2);
    mt_gotoXY(7, 11*7-1);
    write(1, "Operation", 9);
    mt_gotoXY(8, 11*7);
    write(1, "+00:00", 6);
    mt_gotoXY(10, 11*7);
    write(1, "Flags", 5);
    int reg;
    sc_regGet(P, &reg);
    if (reg){
        mt_gotoXY(10, 11*7);
        write(1, "P", 1);
    }
    sc_regGet(O, &reg);

```

```
if (reg){
    mt_gotoXY(10,11*7-2);
    write(1,"O",1);
}
sc_regGet(M,&reg);
if (reg){
    mt_gotoXY(10,11*7+2);
    write(1,"M",1);
}
sc_regGet(T,&reg);
if (reg){
    mt_gotoXY(10,11*7+4);
    write(1,"T",1);
}
sc_regGet(E,&reg);
if (reg){
    mt_gotoXY(10,11*7+6);
    write(1,"E",1);
}
mt_gotoXY(13,8*7);
write(1,"Keys",4);
mt_gotoXY(14,8*7-2);
write(1,"l - load",8);
mt_gotoXY(15,8*7-2);
write(1,"s - save",8);
mt_gotoXY(16,8*7-2);
write(1,"r - run",7);
mt_gotoXY(17,8*7-2);
write(1,"t - step",8);
```

```

mt_gotoXY(18,8*7-2);
write(1,"i - reset",9);
mt_gotoXY(19,8*7-2);
write(1,"F5 - accumulator",17);
mt_gotoXY(20,8*7-2);
write(1,"F6 - instructionCounter",23);
sc_memoryGet(iC,&value);
char buf1[8];
int l1 = sprintf(buf1,"+%.4x",value);
int mass[8];
for(i = 0; i < 5; i++){
    switch(buf1[i]){
        case 'a': mass[i] = 10; break;
        case 'b': mass[i] = 11; break;
        case 'c': mass[i] = 12; break;
        case 'd': mass[i] = 13; break;
        case 'e': mass[i] = 14; break;
        case 'f': mass[i] = 15; break;
        case '+': mass[i] = 16; break;
        default: mass[i] = buf1[i] - '0'; break;
    }
}

bc_printbigchar(big[mass[0]],14,2,BLACK,WHITE);
bc_printbigchar(big[mass[1]],14,11,BLACK,WHITE);
bc_printbigchar(big[mass[2]],14,20,BLACK,WHITE);
bc_printbigchar(big[mass[3]],14,29,BLACK,WHITE);
bc_printbigchar(big[mass[4]],14,38,BLACK,WHITE);
/mt_gotoXY(24,1);
rk_mytermregime(0,50,0,0,1);

```

```

enum keys key;
while(1) {
    rk_readkey(&key);
    switch(key){
        case 2: sc_memorySave("lab5.memory");
break;

        case 1: sc_memoryLoad("lab5.memory"); goto
metka; break;

        case 9: if(iC > 10) iC -= 10; goto metka;
break;

        case 8: if(iC <= 90) iC += 10; goto metka;
break;

        case 10: if(iC > 1) iC--; goto metka;
break;

        case 11: if(iC < 100) iC++; goto metka;
break;

        case 12: {
            if(status == 1){
                write(1,"Input: ",7);
                rk_mytermregime(1,0,50,1,1);
                read(1,&temp,4);
                sscanf(temp,"%x",&t);
                status = 0;
                mt_clrscr();
                goto metka;
            } else if(status == 0){
                if(verifMemory(t) == 0)
                    sc_memorySet(iC,t);
                rk_mytermregime(0,50,0,0,1);
                status = 1;
                mt_clrscr();
            }
        }
    }
}

```

```

        goto metka;
    }
} break;
case 6: {
    if(statusAc == 1){
        sc_memoryGet(iC,&t1);
        statusAc = 0;
    } else if(statusAc == 0){
        sc_memorySet(iC,t1);
        statusAc = 1;
        goto metka;
    }
} break;
case 7: {
    if(statusIc == 1){
        write(1,"Input: ",7);
        rk_mytermregime(1,0,50,1,1);
        read(1,&templ,4);
        sscanf(templ,"%x",&t);
        iC = t;
        mt_clrscr();
        goto metka;
    }

} break;

}

}

rk_mytermregime(1,0,50,1,1);
return 0;}

```