Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и Информационных технологий

Кафедра Информатики

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3**

Преподаватель \_\_\_\_\_\_\_\_ Пересунько П. В.

подпись, дата инициалы, фамилия

Студент КИ15–17Б, 031510065 \_\_\_\_\_\_\_\_\_\_\_ Радионов Т.В.

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2016

**1 Цель работы**

**2 Исходные тексты программ**

/\*! \file file.c

\* \brief Файл обработки записи и чтения файлов

\* \details Файл содержит функции, предназначенные для чтения,

\* записи, а также конвертирования в необходимые форматы для работы

\*/

#include <fcntl.h>

#include <stdio.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <unistd.h>

#include <syscall.h>

#include <string.h>

#include <stdlib.h>

/\*! \brief Структура Human

\* \details Структура для хранения данных в необходимом формате

\* \param surname - фамилия

\* \param name - имя

\* \param height - рост

\*/

typedef struct Human

{

char surname[32];

char name[32];

int height;

} Human;

/\*! \brief Конвертация в структурный тип Human

\* \details Функция перевода массива типа char в массив данных типа Human.

\* \param line - строка данных

\* \param \*humans - указатель на массив данных

\* \param index - порядковый индекс перевода в i-ую структуру

\*/

void\* ConvertToStruct(char\* line, struct Human \*humans, int index)

{

strcpy(humans[index].surname, strtok(line, ","));

strcpy(humans[index].name, strtok(NULL, ","));

int height;

height = atoi(strtok(NULL, ";"));

humans[index].height = height;

}

/\*! \brief Конвертация в строку

\* \details Функция перевода массива структур в массив символов типа char

\* \param \*humans - указатель на массив данных

\* \param numOfLines - количество строк в файле

\* \return line - возвращает массив типа char

\*/

char\* ConvertToLine(struct Human \*humans, int numOfLines)

{

char\* line;

line = (char\*)malloc(numOfLines\*(sizeof(struct Human)));

for (int i = 0; i < numOfLines; i++)

{

strcat(line, humans[i].surname);

strcat(line, ",");

strcat(line, humans[i].name);

strcat(line, ",");

char height[16];

snprintf(height, 16, "%d", humans[i].height);

strcat(line, height);

strcat(line, ";\n");

printf("%s %s %d\n", humans[i].surname, humans[i].name, humans[i].height);

}

return line;

}

/\*! \brief Конвертация в структурный тип Human

\* \details Функция перевода массива типа char в массив данных типа Human.

\* \param \*humans - указатель на массив данных

\* \param \*numOfLines - указатель на количество строк в файле

\* \return humans - возвращает массив типа Human

\* NULL - если возникли проблемы при чтении файла

\*/

struct Human\* ReadFile(struct Human \*humans, int \*numOfLines)

{

int index = 0;

int end;

ssize\_t read\_bytes;

size\_t length;

struct stat fileInfo;

char\* buffer;

char\* line;

// Создание файла

int fileDescriptor = open("out.txt", O\_RDWR | O\_APPEND | O\_CREAT, 0666);

// Если произошла ошибка при создании

if (fileDescriptor < 0)

{

fprintf(stderr, "Cann't open file\n");

return;

}

// Выясняем размер файла и выделяем память для считывания

fstat(fileDescriptor, &fileInfo);

length = fileInfo.st\_size;

buffer = (char\*)malloc(length);

// Считываем содержимое файла

read\_bytes = read(fileDescriptor, buffer, length);

if (read\_bytes < 0)

{

fprintf(stderr, "Cannot read file\n");

return NULL;

}

// Подсчитываем количество строк и находим индекс конца файла

for (int i = 0; i < strlen(buffer); i++)

{

if (buffer[i] == '\n')

{

(\*numOfLines)++;

end = i;

}

}

// Выделяем память для массива структур

humans = (struct Human\*)malloc((\*numOfLines)\*sizeof(struct Human));

// Разбиваем на строки

line = strtok(buffer, "\n");

char arr\_lines[\*numOfLines][128];

while (line != NULL)

{

strcpy(arr\_lines[index], line);

line = strtok(NULL, "\n");

index++;

}

// Выполняем структуризацию строк

for (int i = 0; i < (\*numOfLines); i++)

{

printf("%s\n", arr\_lines[i]);

ConvertToStruct(arr\_lines[i], humans, i);

}

close(fileDescriptor);

return humans;

}

/\*! \brief Запись в файл

\* \details Функция записи массива структур в файл

\* \param \*humans - указатель на массив данных

\* \param \*numOfLines - указатель на количество строк в файле

\*/

void\* RecordFile(struct Human \*humans, int \*numOfLines)

{

char \*text;

// Создание файла

int fileDescriptor = open("out.txt", O\_WRONLY | O\_TRUNC | O\_APPEND | O\_CREAT, 0666);

// Если произошла ошибка при создании

if (-1 == fileDescriptor)

{

fprintf(stderr, "Cann't open file\n"); // Вывод ошибки на экран

}

text = ConvertToLine(humans, \*numOfLines);

printf("text: %s\n", text);

write(fileDescriptor, text, strlen(text));

close(fileDescriptor);

// Освобождение памяти

free(humans);

}

/\* Вариант 18.

Структура данных: фамилия; имя; рост. Создать два запроса,

позволяющих найти максимальный рост, начиная с элемента списка с заданным

номером.\*/

#include <files.c>

/\*! \brief Очистка экрана

\* \details Функция очистки экрана терминала

\*/

void\* ClearScreen()

{

printf("\033[2J");

printf("\033[0;0f");

}

/// Вывод содержимого файла на экран

/\*! \brief Вывод содержимого файла на экран

\* \details Функция отображения массива структур на экран терминала

\* \param \*humans - указатель на массив данных

\* \param numOfLines - количество строк в файле

\*/

void\* PrintOnConsole(struct Human \*humans, int numOfLines)

{

printf("number surname name height\n");

for (int i = 0; i < numOfLines; i++)

{

printf("#%d %s %s %d\n",

i, humans[i].surname, humans[i].name, humans[i].height);

}

}

/\*! \brief Добавление элемента структуры

\* \details Добавление новой структуры в массив структур путем ввода в терминал полей

\* и запись в файл

\* \param \*humans - указатель на массив данных

\* \param \*numOfLines - указатель на количество строк в файле

\*/

void\* AddOneStructure(struct Human \*humans, int \*numOfLines)

{

//char\* newStructure;

//char newStructure[128];

struct Human \*buf\_humans;

//printf("Введите новую единицу структуры (surname,name,height;): ");

//gets(newStructure);

//scanf("%s", newStructure);

//ClearScreen();

//printf("%s\n", newStructure);

// Увеличиваем список структур на одну единицу

buf\_humans = (struct Human\*)malloc((\*numOfLines)\*sizeof(struct Human));

for (int i = 0; i < (\*numOfLines); i++)

{

buf\_humans[i] = humans[i];

}

(\*numOfLines)++;

free(humans);

humans = (struct Human\*)malloc((\*numOfLines)\*sizeof(struct Human));

for (int i = 0; i < (\*numOfLines) - 1; i++)

{

humans[i] = buf\_humans[i];

}

free(buf\_humans);

printf("Введите информацию:\n", humans[\*numOfLines - 1].surname);

printf("фамилия: ");

scanf("%s", humans[\*numOfLines - 1].surname);

printf("имя: ");

scanf("%s", humans[\*numOfLines - 1].name);

printf("рост: ");

scanf("%d", &humans[\*numOfLines - 1].height);

ClearScreen();

// Конвертируем строку и добавляем к списку структур

//ConvertToStruct(newStructure, humans, (\*numOfLines) - 1);

// Записываем в файл

RecordFile(humans, numOfLines);

}

/\*! \brief Удаление элемента структуры

\* \details Удаление элемента структуры из массива структур путем ввода в терминале

\* индекса элемента и запись в файл

\* \param \*humans - указатель на массив данных

\* \param \*numOfLines - указатель на количество строк в файле

\*/

void\* DeleteOneStructure(struct Human \*humans, int \*numOfLines)

{

int index;

struct Human \*buf\_humans;

printf("Введите номер строки для удаления: ");

scanf("%d", &index);

ClearScreen();

buf\_humans = (struct Human\*)malloc((\*numOfLines)\*sizeof(struct Human));

for (int i = 0; i < (\*numOfLines) - 1; i++)

{

if (i < index)

{

buf\_humans[i] = humans[i];

}

else

{

buf\_humans[i] = humans[i+1];

}

}

(\*numOfLines)--;

free(humans);

humans = (struct Human\*)malloc((\*numOfLines)\*sizeof(struct Human));

for (int i = 0; i < (\*numOfLines); i++)

{

humans[i] = buf\_humans[i];

}

free(buf\_humans);

// Записываем в файл

RecordFile(humans, numOfLines);

}

/\*! \brief Поиск максимального роста с i-ой позиции

\* \details Обход массива структур с указанной в терминале позиции и нахождение

структуры с наибольшим значением поля height

\* \param \*humans - указатель на массив данных

\* \param numOfLines - количество строк в файле

\*/

void\* FindMaxHeight(struct Human \*humans, int numOfLines)

{

int index;

int max\_index;

printf("Введите номер строки откуда следует искать: ");

scanf("%d", &index);

ClearScreen();

// Алгоритм поиска максимального роста

int max = humans[0].height;

for (int i = index; i < numOfLines; i++)

{

if (humans[i].height > max)

{

max = humans[i].height;

max\_index = i;

}

}

printf("Самый высокий рост: ");

printf("#%d %s %s %d\n",

max\_index, humans[max\_index].surname, humans[max\_index].name,

humans[max\_index].height);

}

/\*! \brief Главная функция

\* \param argc количество аргументов

\* \param argv массив системных переменных

\* \return 0 - успешное выполнение программы,

\* EXIT\_FAILURE - какая либо ошибка

\*/

int main(int argc, char \*argv[])

{

struct Human \*humans;

int numOfLines = 0;

int menu;

humans = ReadFile(humans, &numOfLines);

do

{

printf("\n1. Вывести записи на экран\n");

printf("2. Добавить запись\n");

printf("3. Удалить запись\n");

printf("4. Найти запись с максимальным ростом\n");

printf("5. Выход\n");

printf("\n>> Введите пункт меню: ");

scanf("%d", &menu);

switch (menu)

{

case 1:

{

ClearScreen();

PrintOnConsole(humans, numOfLines);

break;

}

case 2:

{

ClearScreen();

PrintOnConsole(humans, numOfLines);

AddOneStructure(humans, &numOfLines);

printf("#%d %s %s %d\n",

numOfLines-1, humans[numOfLines-1].surname,

humans[numOfLines-1].name,

humans[numOfLines-1].height);

break;

}

case 3:

{

ClearScreen();

PrintOnConsole(humans, numOfLines);

DeleteOneStructure(humans, &numOfLines);

break;

}

case 4:

{

ClearScreen();

PrintOnConsole(humans, numOfLines);

FindMaxHeight(humans, numOfLines);

break;

}

}

} while (menu != 5);

free(humans);

return 0;

}

**3 Примеры работы программ**

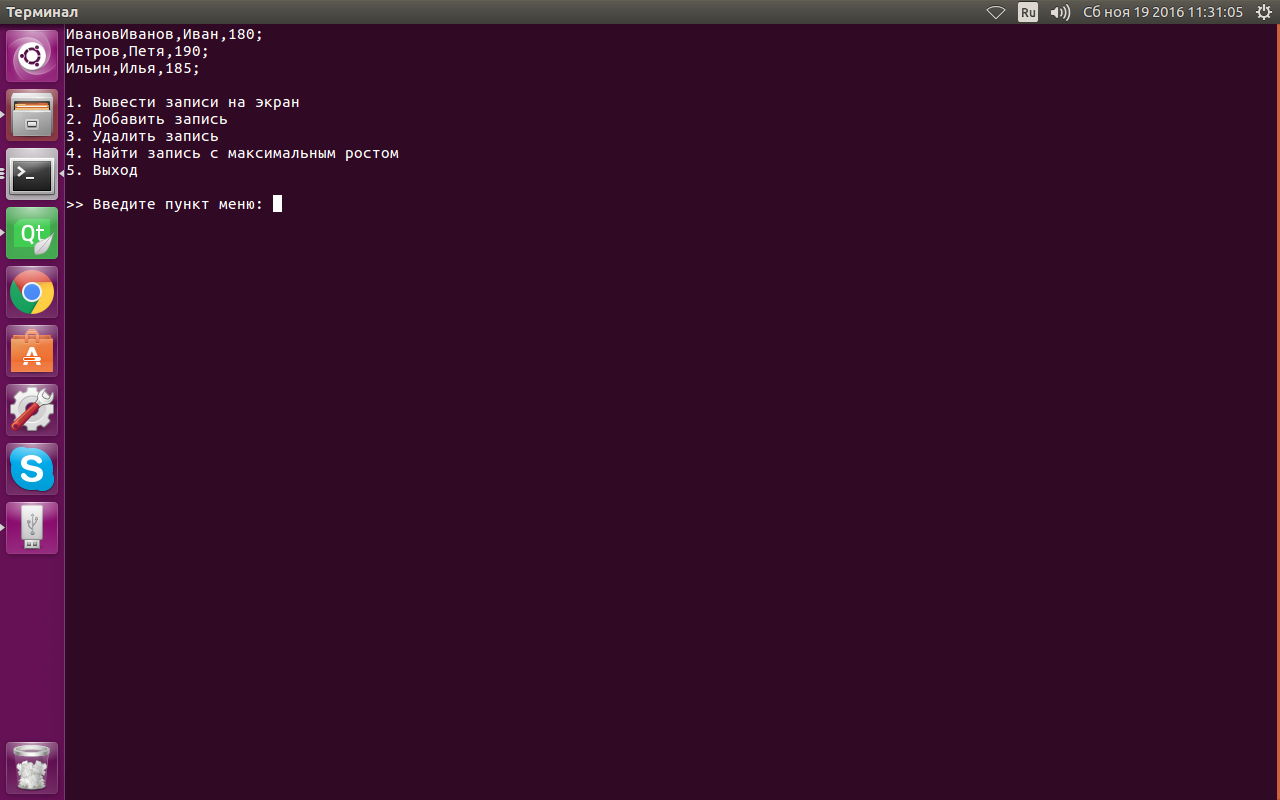


Рисунок 1

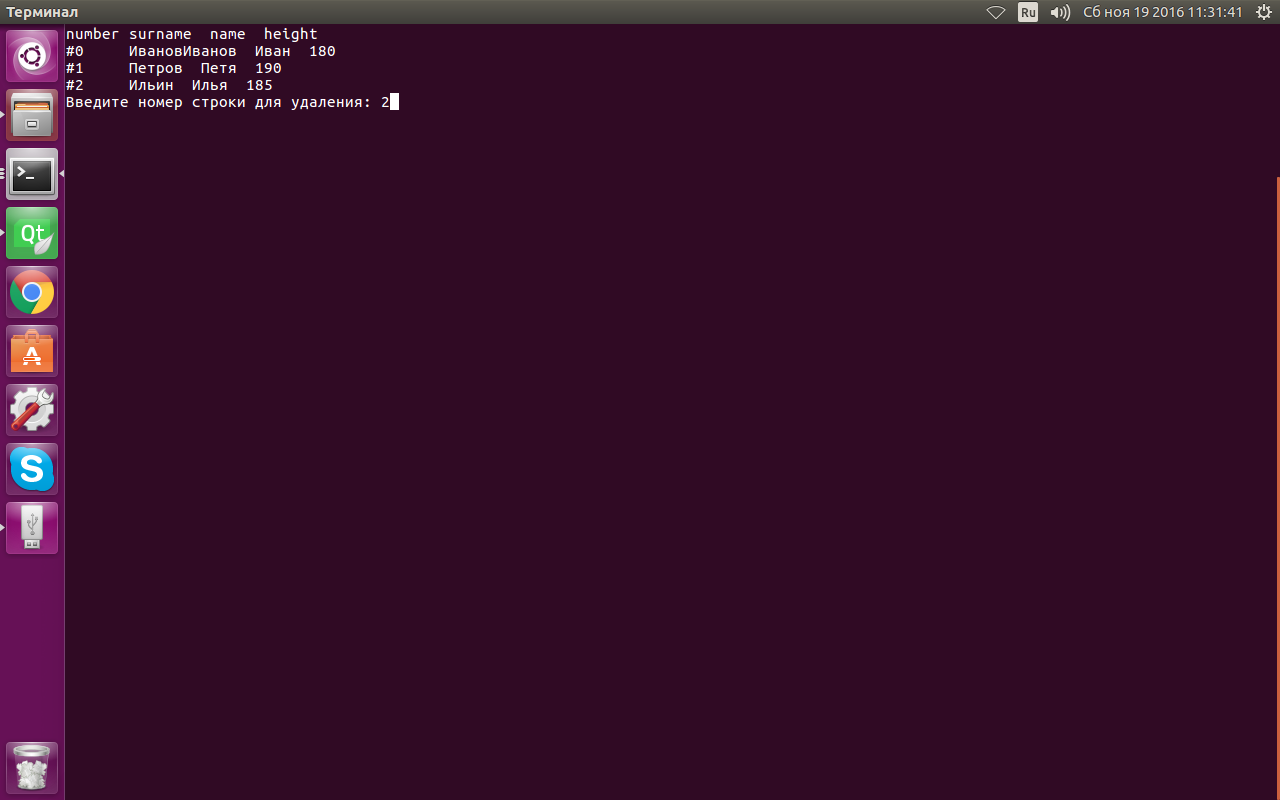


Рисунок 2

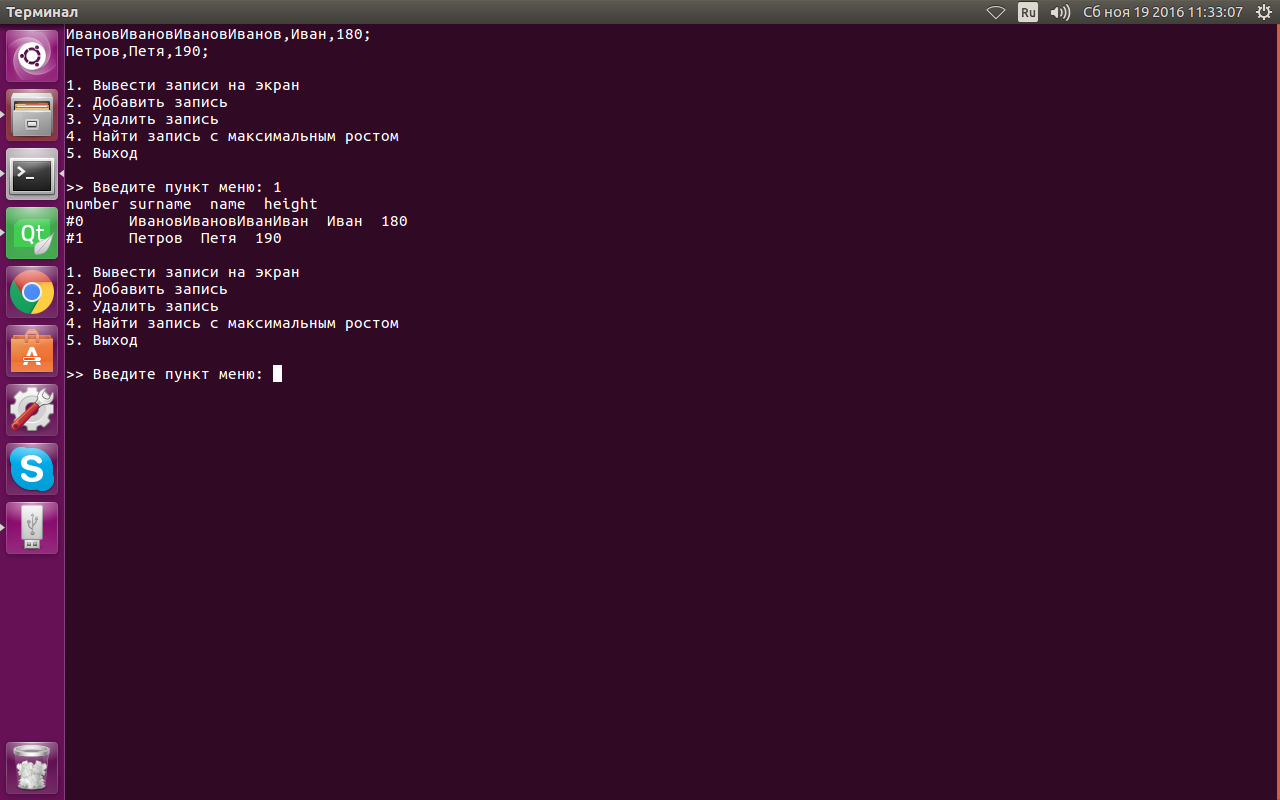


Рисунок 3

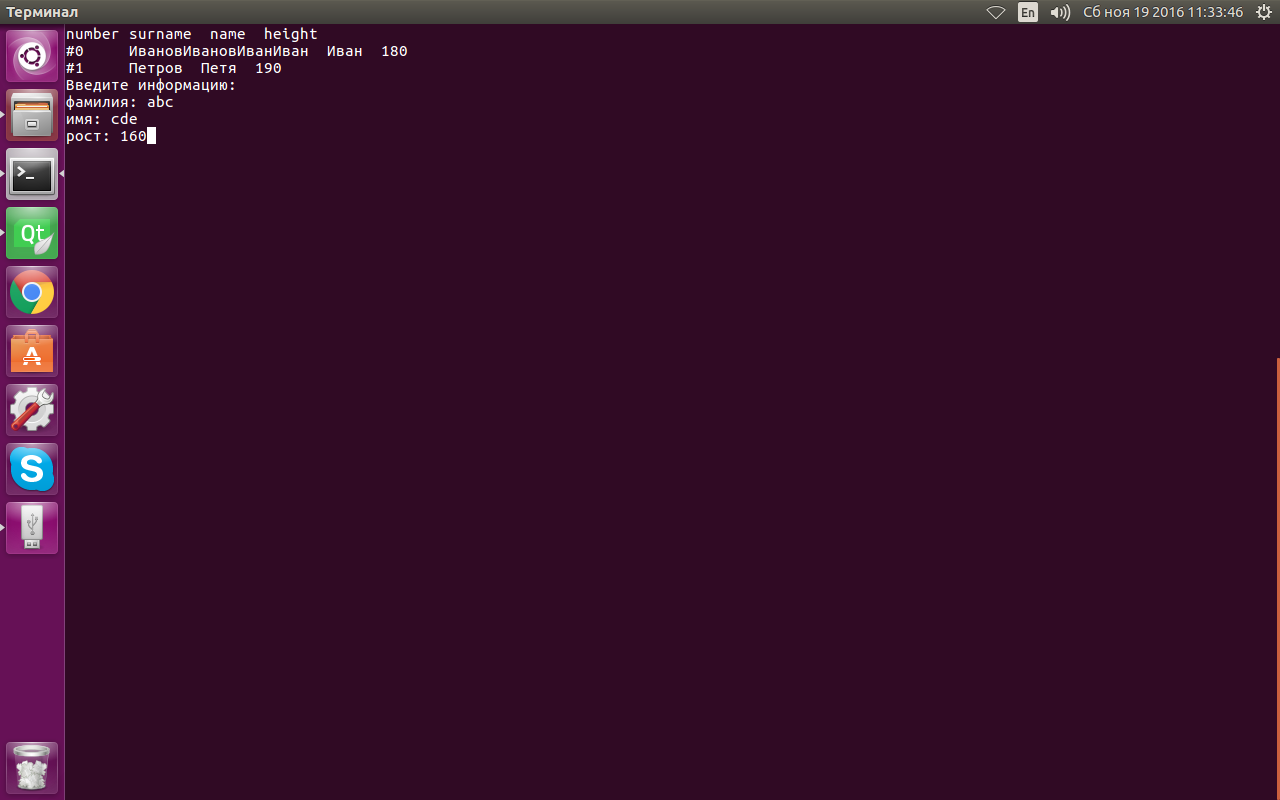


Рисунок 4

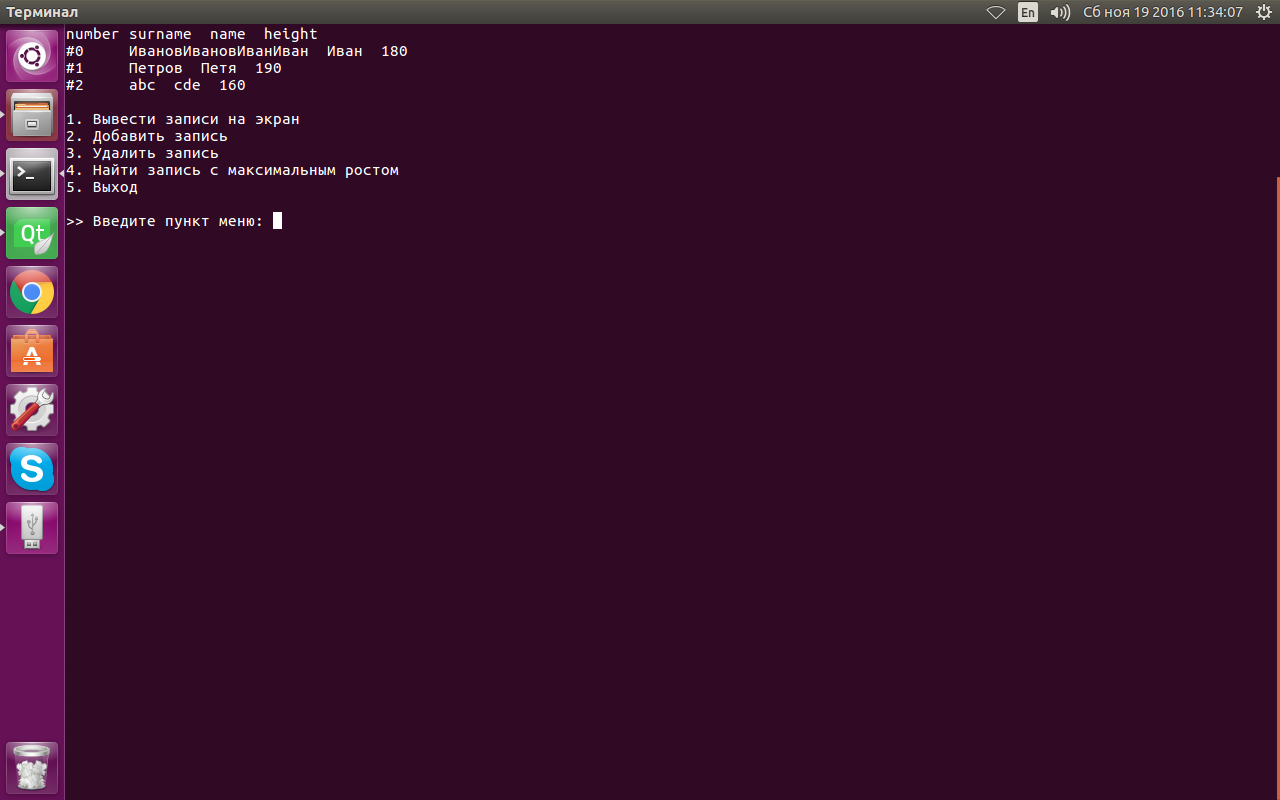


Рисунок 5

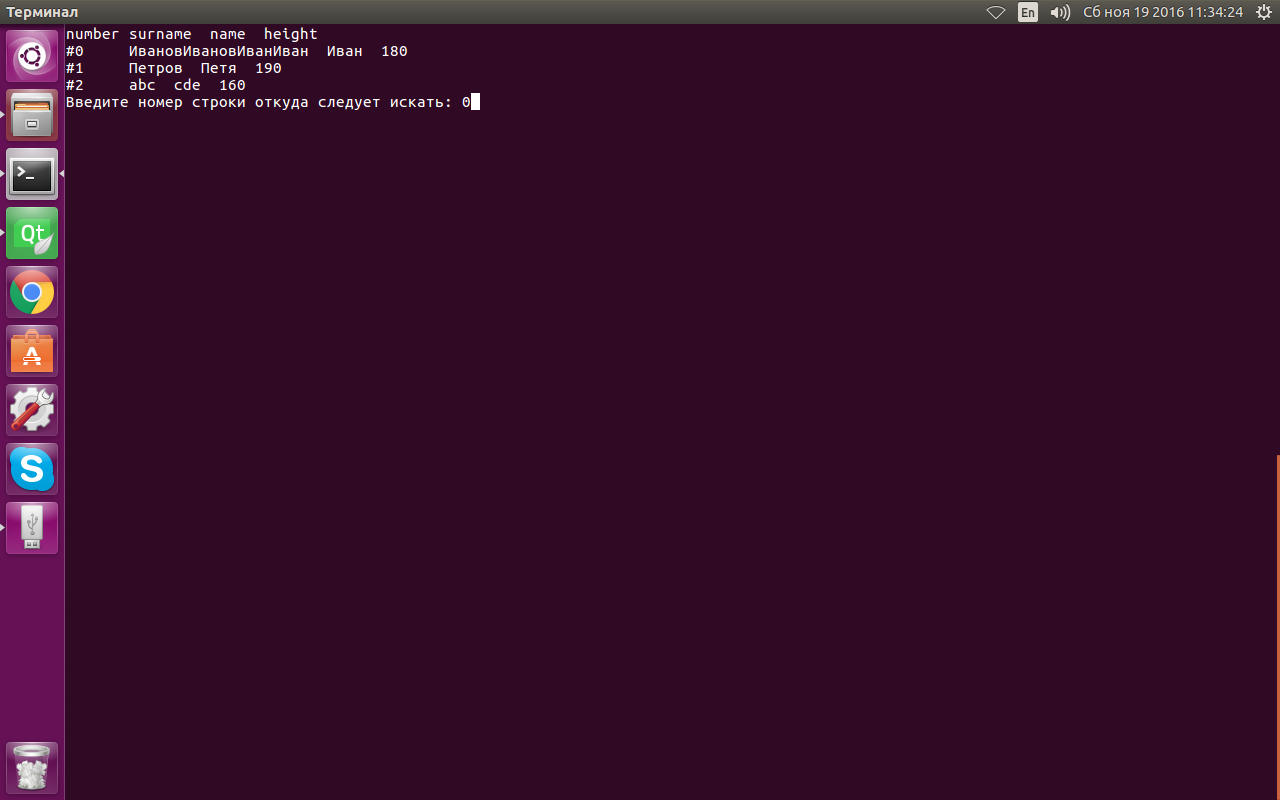


Рисунок 6

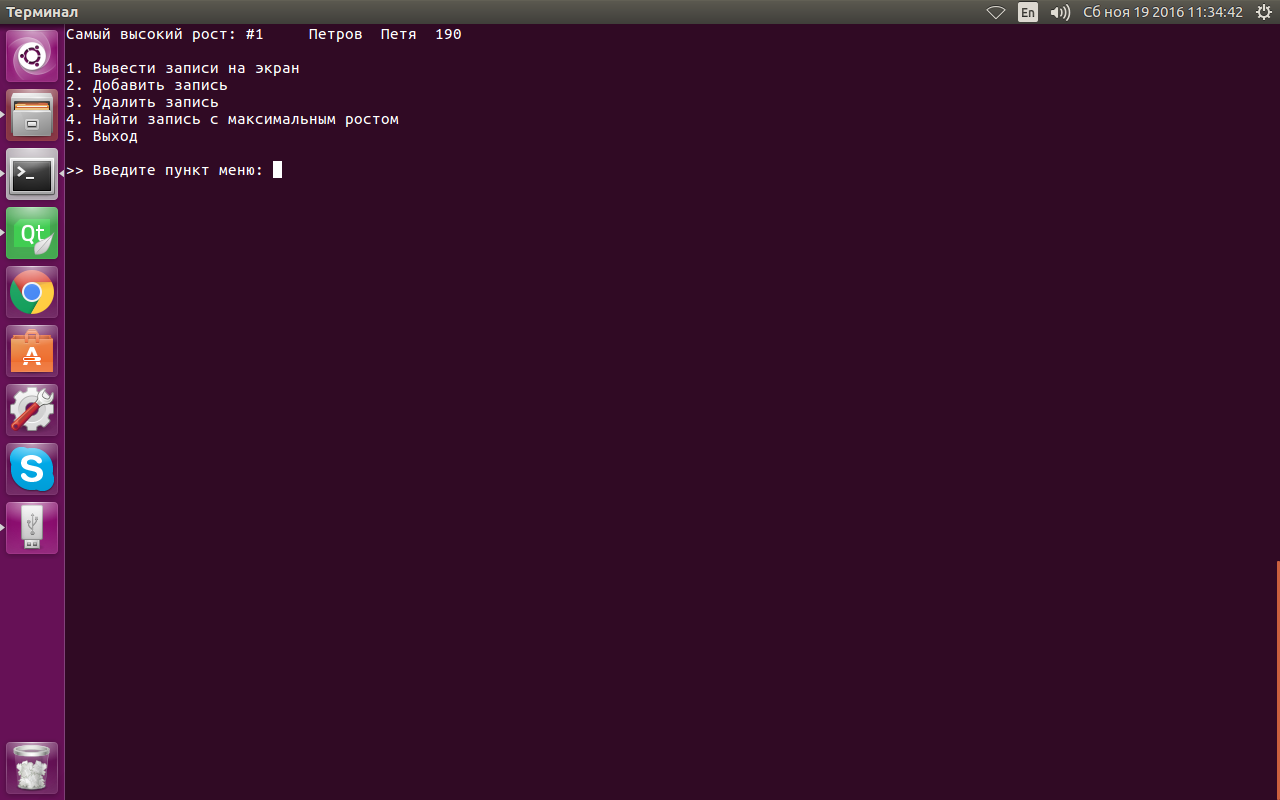


Рисунок 7