Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 4

Управление файлами в ОС GNU/Linux
Тема

Преподаватель _____ А. С. Кузнецов
Подпись, дата Инициалы, Фамилия

Студент КИ19-17/1Б, №031939174 _____ А. К. Никитин
Номер группы, зачетной книжки Подпись, дата Инициалы, Фамилия

Красноярск 2021

1 Цель

Программная реализация обработки текстовой информации, хранящейся во внешней памяти, с использованием системных вызовов низкоуровневого ввода-вывода.

2 Задачи

1. Ознакомиться с краткими теоретическими сведениями по управлению файлами в ОС GNU/Linux.

2. Разработать программу, позволяющую выполнять операции над файлами с записями фиксированной длины. Обеспечить сборку программы с использованием утилиты GNU *make*

3. Вывести результат программы на консоль.

Описание варианта:

Структура данных: учебная группа; количество студентов; количество девушек. Создать два запроса, позволяющих подсчитать количество групп, состоящих только из юношей, и определить, имеются ли группы, где их количества девушек и юношей равны.

3 Описание использованных функций

1) open() – открытие файла и получение дескриптора для работы с ним;

2) close() – закрытие файла после работы с ним;

3) write() – запись данных в файл;

4) read() – чтение данных из файла;

5) lseek() – перемещение указателя внутри файла;

6) fstat() – получение информации о файле с помощью файлового дескриптора;

7) remove() – удаление файла;

8) rename() – переименование файла.

4 Исходные тексты программ

На листинге 1 представлен код программы fileFunctions.c.

Листинг 1 – Код программы с операциями по манипулированию файлами

```
#include <unistd.h>
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <sys/stat.h>
#include <fcntl.h>
#include "fileFunctions.h"

#define GROUP_SIZE 50

/*! \brief Checks whether the file from the path exists or not.
 *
 * \param path Path to the file. The file can be local
 *
 *  \return 0 if exists else -1
 */
int checkFileExists(char* path)
{
    if (access(path, F_OK) != 0)
    {
        puts("The file must exist!");
        return -1;
    }

    return 0;
}

/*! \brief Checks whether the file from the path empty or not.
 *
 * \param fd File descriptor
 *
 *  \return 0 if not empty else -1
 */
int checkFileEmpty(int fd)
{
    if (getFileSize(fd) == 0)
    {
```

```
        puts("The file must not be empty!");
        return -1;
    }

    return 0;
}

/*! \brief Calculates the string size (length) that will be loaded into file
 *
 * \param groupSize The size of one group in a file
 *
 *  \return String size
 */
int getStringSize(int groupSize)
{
    return groupSize - 2 * sizeof(int);
}

/*! \brief Calculates the file size
 *
 * \param fd File descriptor
 *
 *  \return File size
 */
int getFileSize(int fd)
{
    struct stat info;
    int fstatSuccess = fstat(fd, &info);
    if (fstatSuccess)
    {
        perror("Getting file info went wrong.");
    }

    return info.st_size;
}

/*! \brief Writes information in bytes from buffer into file. Similar to the
function write() from <unistd.h>
 * but also outputs errors
 *
 * \param fd File descriptor
 * \param info Variable with information that will be loaded into file
```

```
 * \param size The number of bytes that will be loaded into file
 *
 *  \return -1 if something went wrong. Else the number of bytes that have been
successfully loaded into file
 */
int writeInfo(int fd, void* info, size_t size)
{
    int writeReturn = write(fd, info, size);

    if (writeReturn == -1)
    {
        perror("Write into file went wrong.");
        return -1;
    }

    if (writeReturn == 0)
    {
        perror("Nothing was written.");
        return -1;
    }

    return writeReturn;
}

/*! \brief Writes meta information (size of one group) into beginning of the file
(if the file is empty)
 *
 * \param fd File descriptor
 *
 *  \return Nothing
 */
void writeMeta(int fd)
{
    lseek(fd, 0, SEEK_SET);

    int size = GROUP_SIZE;

    if (getFileSize(fd) == 0)
    {
        writeInfo(fd, &size, sizeof(int));
    }
}
```

```
/*! \brief Reads information from the file. Similar to the function read() from
<unistd.h>
 * but also outputs errors
 *
 * \param fd File descriptor
 * \param info Variable with information that will be loaded into file
 * \param size The number of bytes that will be loaded into file
 *
 *  \return -1 if something went wrong. Else the number of bytes that have been
successfully read from file
 */
int readInfo(int fd, void* info, size_t size)
{
    int readReturn = read(fd, info, size);

    if (readReturn == -1)
    {
        perror("Read error");
        return -1;
    }

    if (readReturn == 0)
    {
        perror("Nothing has been read.");
        return -1;
    }

    return readReturn;
}

/*! \brief Reads meta information (size of one group) from the file
 *
 * \param fd File descriptor
 *
 *  \return Nothing
 */
int readMeta(int fd)
{
    lseek(fd, 0, SEEK_SET);

    int metaSize;
```

```
    readInfo(fd, &metaSize, sizeof(int));
    return metaSize;
}

/*! \brief Deletes the file from the working directory
 *
 * \param filename The name of file
 *
 *  \return Nothing
 */
void deleteFile(char* filename)
{
    int removeReturn = remove(filename);
    if (removeReturn != 0)
    {
        perror("File delete error");
        puts("");
        return;
    }
}

/*! \brief Deletes information about one group from the file. The group is
identified by name
 *
 * \param fd File descriptor
 * \param groupName The name of a group
 *
 *  \return Is there such a group (0) or not (-1)
 */
int deleteGroup(int fd, char* groupName)
{
    int fd1 = open("groups1", O_CREAT | O_WRONLY);
    writeMeta(fd1);

    int meta = readMeta(fd);
    char currentGroupName[getStringSize(meta)];
    int currentStudentsNumber;
    int currentFemalesNumber;

    int isFound = -1;
    while (lseek(fd, 0, SEEK_CUR) < getFileSize(fd))
```

```
    {
        readInfo(fd, &currentGroupName, sizeof(char) * getStringSize(meta));
        readInfo(fd, &currentStudentsNumber, sizeof(int));
        readInfo(fd, &currentFemalesNumber, sizeof(int));

        if (strcmp(groupName, currentGroupName) == 0)
        {
            isFound = 0;
            continue;
        }

        writeInfo(fd1, &currentGroupName, sizeof(char) * getStringSize(meta));
        writeInfo(fd1, &currentStudentsNumber, sizeof(int));
        writeInfo(fd1, &currentFemalesNumber, sizeof(int));
    }

    close(fd);
    close(fd1);
    deleteFile("groups");
    int renameReturn = rename("groups1", "groups");
    if (renameReturn != 0)
    {
        perror("Rename error");
        puts("");
    }
    return isFound;
}

/*! \brief Moves the pointer in the file to one group
 *
 * \param fd File descriptor
 * \param groupName The name of a group
 *
 *  \return Is there such a group (0) or not (-1)
 */
int seekToGroup(int fd, char* groupName)
{
    lseek(fd, 0, SEEK_SET);

    int meta = readMeta(fd);
    char currentGroupName[getStringSize(meta)];
```

```
    while (lseek(fd, 0, SEEK_CUR) < getFileSize(fd))
    {
        readInfo(fd, &currentGroupName, sizeof(char) * getStringSize(meta));
        if (strcmp(groupName, currentGroupName) == 0)
        {
            lseek(fd, -sizeof(char) * getStringSize(meta), SEEK_CUR);
            return 0;
        }
        lseek(fd, sizeof(int) * 2, SEEK_CUR);
    }

    return -1;
}

/*! \brief Changes the name of one group in the file
 *
 * \param fd File descriptor
 * \param groupName The name of a group where a new name will be
 * \param newGroupName A new name of the group
 *
 *  \return Is there such a group (0) or not (-1)
 */
int changeName(int fd, char* groupName, char* newGroupName)
{
    int meta = readMeta(fd);
    int existError = seekToGroup(fd, groupName);
    if (existError != 0)
        return -1;

    writeInfo(fd, newGroupName, sizeof(char) * getStringSize(meta));

    return 0;
}

/*! \brief Changes the students number of one group in the file
 *
 * \param fd File descriptor
 * \param groupName The name of a group where a new students number will be
 * \param newStudentsNumber A new students number of the group
 *
 *  \return Is there such a group (0) or not (-1)
 */
```

```
int changeStudentsNumber(int fd, char* groupName, int newStudentsNumber)
{
    int meta = readMeta(fd);
    int existError = seekToGroup(fd, groupName);
    if (existError != 0)
        return -1;

    lseek(fd, sizeof(char) * getStringSize(meta), SEEK_CUR);
    writeInfo(fd, &newStudentsNumber, sizeof(int));

    return 0;
}

/*! \brief Changes the females number of one group in the file
 *
 * \param fd File descriptor
 * \param groupName The name of a group where a new females number will be
 * \param newFemalesNumber A new females number of the group
 *
 *  \return Is there such a group (0) or not (-1)
 */
int changeFemalesNumber(int fd, char* groupName, int newFemalesNumber)
{
    int meta = readMeta(fd);
    int existError = seekToGroup(fd, groupName);
    if (existError != 0)
        return -1;

    lseek(fd, sizeof(char) * getStringSize(meta) + sizeof(int), SEEK_CUR);
    writeInfo(fd, &newFemalesNumber, sizeof(int));

    return 0;
}

/*! \brief Returns a students number from the group in file
 *
 * \param fd File descriptor
 * \param groupName The name of a group with the students number
 *
 *  \return Student number of the group or -1 if there is no such group in the
file
 */
```

```
int getStudentsNumber(int fd, char* groupName)
{
    lseek(fd, 0, SEEK_SET);

    int meta = readMeta(fd);
    char currentGroupName[getStringSize(meta)];

    int studentsNumber;
    while (lseek(fd, 0, SEEK_CUR) < getFileSize(fd))
    {
        readInfo(fd, &currentGroupName, sizeof(char) * getStringSize(meta));
        if (strcmp(groupName, currentGroupName) == 0)
        {
            readInfo(fd, &studentsNumber, sizeof(int));
            return studentsNumber;
        }
        lseek(fd, sizeof(int) * 2, SEEK_CUR);
    }
    return -1;
}

/*! \brief Returns a students number from the group in file
 *
 * \param fdDest File descriptor of the file where the data will be loaded
 * \param fdSource ile descriptor of the file where the data is kept
 *
 *  \return Nothing
 */
void saveFile(int fdDest, int fdSource)
{
    lseek(fdDest, 0, SEEK_SET);
    lseek(fdSource, 0, SEEK_SET);

    int meta = readMeta(fdSource);
    writeMeta(fdDest);

    char groupName[getStringSize(meta)];
    int studentsNumber;
    int femalesNumber;
    while (lseek(fdSource, 0, SEEK_CUR) < getFileSize(fdSource))
    {
        readInfo(fdSource, &groupName, sizeof(char) * getStringSize(meta));
```

```
        readInfo(fdSource, &studentsNumber, sizeof(int));
        readInfo(fdSource, &femalesNumber, sizeof(int));

        writeInfo(fdDest, &groupName, sizeof(char) * getStringSize(meta));
        writeInfo(fdDest, &studentsNumber, sizeof(int));
        writeInfo(fdDest, &femalesNumber, sizeof(int));
    }

    close(fdSource);
    close(fdDest);
}
```

На листинге 2 представлен код программы userInput.c.

Листинг 2 – Код программы с операциями по взаимодействию с пользователем через консоль

```
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/stat.h>
#include <fcntl.h>
#include "userInput.h"
#include "fileFunctions.h"

#define GROUP_SIZE 50

/*! \brief Input validation for natural number
 *
 * \param number Int pointer where the appropriate number will be loaded
 *
 *  \return Nothing
 */
void inputNat(int* number)
{
    while (!scanf("%d", number))
    {
        fflush(stdin);
        printf("Please, input correct information!\n");
    }
    if (*number <= 0)
    {
```

```
        printf("Please, enter natural number!\n");
        inputNat(number);
    }
}

/*! \brief Input validation for natural number with zero
 *
 * \param number Int pointer where the appropriate number will be loaded
 *
 *  \return Nothing
 */
void inputNat0(int* number)
{
    while (!scanf("%d", number))
    {
        fflush(stdin);
        printf("Please, input correct information!\n");
    }
    if (*number < 0)
    {
        printf("Please, enter natural number or 0!\n");
        inputNat(number);
    }
}

/*! \brief User string input of arbitrary length
 *
 * \param word The destination word where information will be putted
 *
 *  \return Nothing
 */
void inputString(char** word)
{
    int count = 0;
    char inputChar = 0;

    fflush(stdin);

    *word = NULL;
    while (1)
    {
        inputChar = getchar();
```

```
        if (inputChar == '\n')
            if (count != 0)  // ќбработка считывани¤ единичного \n
                break;
            else
                continue;
        else
        {
            *word = realloc(*word, count + 1);
            if (*word == NULL)
                perror("Realloc has returned a NULL pointer");

            (*word)[count] = inputChar;
            count++;
        }
    }
    (*word)[count] = '\0';
}

/*! \brief Checks if the string is a natural number
 *
 * \param string String that may be a number or not
 *
 *  \return true or false
 */
int checkNat(char* string)
{
    for (int i = 0; string[i] != '\0'; i++)
        if (!isdigit(string[i]))
            return -1;

    if (strtol(string, NULL, 10) <= 0)
        return -1;
    return 0;
}

/*! \brief Checks if the string is a natural number or zero
 *
 * \param string String that may be a number or not
 *
 *  \return true or false
 */
int checkNat0(char* string)
```

```
{
    for (int i = 0; string[i] != '\0'; i++)
        if (!isdigit(string[i]))
            return -1;

    if (strtol(string, NULL, 10) < 0)
        return -1;

    return 0;
}

/*! \brief Inputs the group characteristics (name, students number and females
number in group into file) and writes it
 * into the file
 *
 * \param fd File descriptor
 *
 *  \return Nothing
 */
void groupWrite(int fd)
{
    int studentsNumber;
    int femalesNumber;
    char* name;

    int string_size = getStringSize(GROUP_SIZE);
    char cutName[string_size];

    writeMeta(fd);

    puts("Input the group name:");
    inputString(&name);
    strncpy(cutName, name, string_size);

    puts("Input the number of students in group:");
    inputNat(&studentsNumber);

    puts("Input the number of females in group:");
    inputNat0(&femalesNumber);

    if (femalesNumber > studentsNumber)
    {
```

```
        puts("Number of females cannot be larger than number of students!");
        return;
    }

    writeInfo(fd, cutName, sizeof(char) * string_size);
    writeInfo(fd, &studentsNumber, sizeof(int));
    writeInfo(fd, &femalesNumber, sizeof(int));

    printf("Writing structure to file has been completed!\n\n");
    close(fd);
}

/*! \brief Inputs the name of group and deletes this group from the file
 *
 * \param fd File descriptor
 *
 *  \return Nothing
 */
void groupDelete(int fd)
{
    char* searchName = NULL;

    puts("Input the group name:");
    inputString(&searchName);

    int deleteGroupReturn = deleteGroup(fd, searchName);
    if (deleteGroupReturn != 0)
    {
        puts("This group does not exist!");
        return;
    }

    puts("The group has been deleted.\n");
}

/*! \brief Inputs the name of group and new characteristics ("-" if remain) and
changes the information for this group
 *
 * \param fd File descriptor
 *
 *  \return Nothing
 */
```

```
void groupEdit(int fd)
{
    char* searchName = NULL;
    char* groupName;
    char* stringStudentsNumber;
    char* stringFemaleNumber;

    puts("Input the group name:");
    inputString(&searchName);

    if (seekToGroup(fd, searchName) != 0)
    {
        puts("This group does not exist!");
        return;
    }

    puts("Input new group name ('-' if remain old):");
    inputString(&groupName);
    if (strcmp(groupName, "-") != 0)
    {
        changeName(fd, searchName, groupName);
        strcpy(searchName, groupName);
    }

    puts("Input new students number ('-' if remain old):");
    inputString(&stringStudentsNumber);
    if (strcmp(stringStudentsNumber, "-") != 0 && checkNat(stringStudentsNumber)
== 0)
    {
        int studentsNumber = strtol(stringStudentsNumber, NULL, 10);
        changeStudentsNumber(fd, searchName, studentsNumber);
    }

    puts("Input new females number ('-' if remain old):");
    inputString(&stringFemaleNumber);
    if (strcmp(stringFemaleNumber, "-") != 0 && checkNat0(stringFemaleNumber) ==
0)
    {
        int femaleNumber = strtol(stringFemaleNumber, NULL, 10);
        if (femaleNumber > getStudentsNumber(fd, searchName))
        {
            puts("Number of females cannot be larger than number of students!");
```

```
            return;
        }

        changeFemalesNumber(fd, searchName, femaleNumber);
    }

    close(fd);
}

/*! \brief Outputs the information about all groups from the file
 *
 * \param fd File descriptor
 *
 *  \return Nothing
 */
void showAll(int fd)
{
    int meta;
    int studentsNumber;
    int femalesNumber;

    meta = readMeta(fd);
    char groupName[getStringSize(meta)];

    while (lseek(fd, 0, SEEK_CUR) < getFileSize(fd))
    {
        readInfo(fd, &groupName, sizeof(char) * getStringSize(meta));
        readInfo(fd, &studentsNumber, sizeof(int));
        readInfo(fd, &femalesNumber, sizeof(int));

        printf("The group %s has %d students and %d females\n\n", groupName,
studentsNumber, femalesNumber);
    }

    close(fd);
}

/*! \brief Outputs the information about groups only with males from the file
 *
 * \param fd File descriptor
 *
 *  \return Nothing
```

```
 */
void showOnlyMales(int fd)
{
    int meta;
    int studentsNumber;
    int femalesNumber;

    meta = readMeta(fd);
    char groupName[getStringSize(meta)];

    while (lseek(fd, 0, SEEK_CUR) < getFileSize(fd))
    {
        readInfo(fd, &groupName, sizeof(char) * getStringSize(meta));
        readInfo(fd, &studentsNumber, sizeof(int));
        readInfo(fd, &femalesNumber, sizeof(int));

        if (femalesNumber == 0)
            printf("The group %s has %d students and %d females\n\n", groupName,
studentsNumber, femalesNumber);
    }

    close(fd);
}

/*! \brief Outputs the information about groups only with an equal number of males
and females from the file
 *
 * \param fd File descriptor
 *
 *  \return Nothing
 */
void showMalesEqualFemales(int fd)
{
    int meta;
    int studentsNumber;
    int femalesNumber;

    meta = readMeta(fd);
    char groupName[getStringSize(meta)];

    while (lseek(fd, 0, SEEK_CUR) < getFileSize(fd))
    {
```

```
        readInfo(fd, &groupName, sizeof(char) * getStringSize(meta));
        readInfo(fd, &studentsNumber, sizeof(int));
        readInfo(fd, &femalesNumber, sizeof(int));

        if (femalesNumber == studentsNumber - femalesNumber)
            printf("The group %s has %d students and %d females\n\n", groupName,
studentsNumber, femalesNumber);
    }

    close(fd);
}

/*! \brief Inputs the path to the file and saves the data from the "groups" into
it. If the file doesn't exist, creates
 * it
 *
 *  \return Nothing
 */
void save()
{
    char* filePath;
    puts("Input the path where the file will be saved:");
    inputString(&filePath);

    int fd1 = open("groups", O_RDONLY);
    int fd2 = open(filePath, O_CREAT | O_WRONLY);

    if (checkFileExists("groups") == 0 && checkFileEmpty(fd1) == 0)
        saveFile(fd2, fd1);
}

/*! \brief Inputs the path to the file and loads the data from this file into the
"groups" file.
 *
 *  \return Nothing
 */
void load()
{
    char* filePath;
    puts("Input the path where the source file will is located:");
    inputString(&filePath);
```

```
    int fd1 = open("groups", O_CREAT | O_WRONLY);
    int fd2 = open(filePath, O_RDONLY);

    if (checkFileExists(filePath) == 0 && checkFileEmpty(fd2) == 0)
        saveFile(fd1, fd2);
}
```

На листинге 3 представлен код программы main.c.

Листинг 3 – Код главной программы

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>
#include "userInput.h"
#include "fileFunctions.h"

/*! \brief The main function.
 *
 *  \return Nothing
 */
int main()
{
    int userChoice;

    enum Case
    {
        addGroup = 1, editGroup, deleteGroup, fileDelete, viewGroups, onlyMales,
malesEqualFemales,
        saveFile, loadFile, exitProg
    };

    do
    {
        printf("Input your choice:\n"
               "1. Add a new group.\n"
               "2. Change group info.\n"
               "3. Delete group.\n"
               "4. Delete groups file.\n"
               "5. View all the groups.\n"
               "6. View only the groups with males.\n"
               "7. View only the groups with females.\n"
               "8. Save into file.\n"
```

```
       "9. Load from file.\n"
       "10. Exit\n");
inputNat(&userChoice);

switch (userChoice)
{

case (addGroup):
{
    int fd = open("groups", O_CREAT | O_APPEND | O_WRONLY);
    groupWrite(fd);
    break;
}

case (editGroup):
{
    int fd = open("groups", O_CREAT | O_RDWR);
    if (checkFileExists("groups") != 0 || checkFileEmpty(fd) != 0)
        break;

    groupEdit(fd);
    break;
}

case (deleteGroup):
{
    int fd = open("groups", O_RDONLY);
    if (checkFileExists("groups") != 0 || checkFileEmpty(fd) != 0)
        break;

    groupDelete(fd);
    break;
}

case (fileDelete):
{
    deleteFile("groups");
    puts("The file has been successfully deleted.\n");
    break;
}

case (viewGroups):
```

```
{
    int fd = open("groups", O_RDONLY);
    if (checkFileExists("groups") != 0 || checkFileEmpty(fd) != 0)
        break;

    showAll(fd);
    break;
}

case (onlyMales):
{
    int fd = open("groups", O_RDONLY);
    if (checkFileExists("groups") != 0 || checkFileEmpty(fd) != 0)
        break;

    showOnlyMales(fd);
    break;
}

case (malesEqualFemales):
{
    int fd = open("groups", O_RDONLY);
    if (checkFileExists("groups") != 0 || checkFileEmpty(fd) != 0)
        break;

    showMalesEqualFemales(fd);
    break;
}

case (saveFile):
{
    save();
    break;
}

case (loadFile):
{
    load();
    break;
}

case (exitProg):
```

```
        {
            exit(0);
            break;
        }

        default:
        {
            printf("Input the choice from 1 to 10!\n");
            continue;
        }

        }

    } while (1);
    return 0;
}
```

5 Содержимое файла Makefile

На листинге 4 представлено содержимое makefile.

Листинг 4 – Makefile

```
CC = gcc
CFLAGS = -std=gnu99
SOURCES = fileFunctions.c userInput.c main.c

all:
	$(CC) $(CFLAGS) $(SOURCES) -o start.o
```

6 Тестовые примеры работы программы

```
root@brain:/mnt/lab4# ./start.o
Input your choice:
1. Add a new group.
2. Change group info.
3. Delete group.
4. Delete groups file.
5. View all the groups.
6. View only the groups with males.
7. View only the groups with females.
8. Exit
1
Input the group name:
Group1
Input the number of students in group:
10
Input the number of females in group:
5
Writing structure to file has been completed!

Input your choice:
1. Add a new group.
2. Change group info.
3. Delete group.
4. Delete groups file.
5. View all the groups.
6. View only the groups with males.
7. View only the groups with females.
8. Exit
1
Input the group name:
Group2
Input the number of students in group:
12
Input the number of females in group:
0
Writing structure to file has been completed!
```

Рисунок 1 – Добавление новой группы в файл

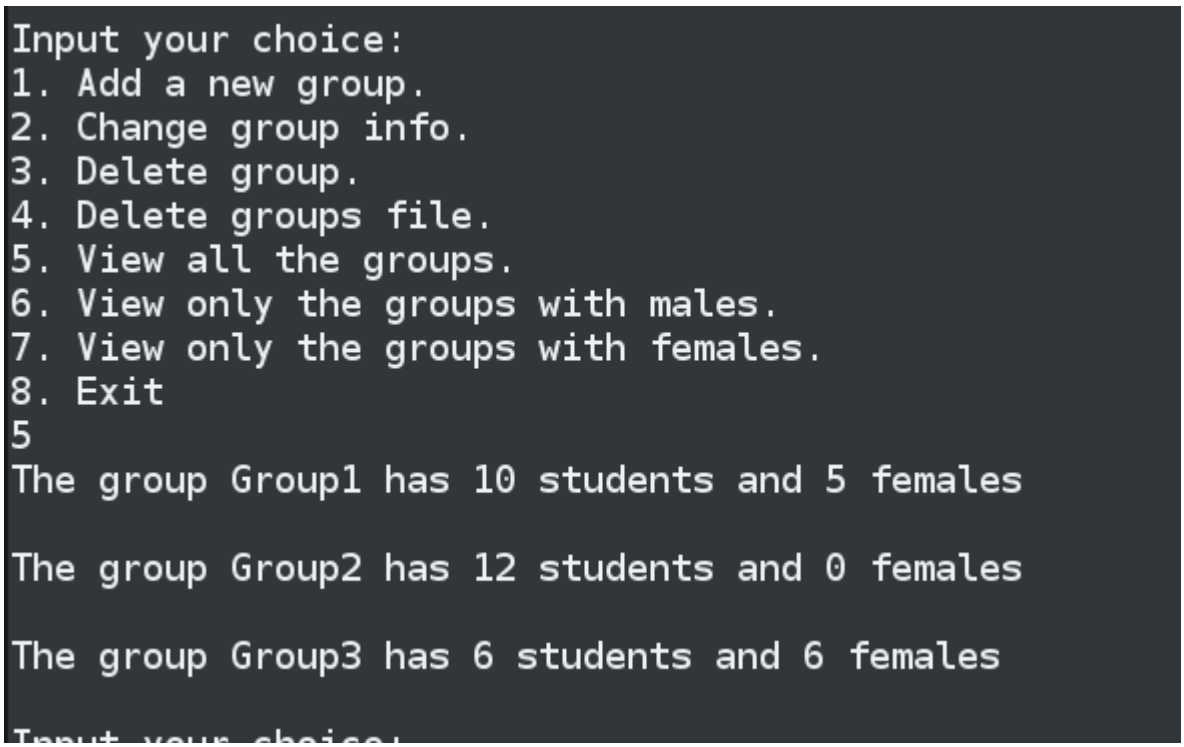

Рисунок 2 – Вывод всех групп из файла

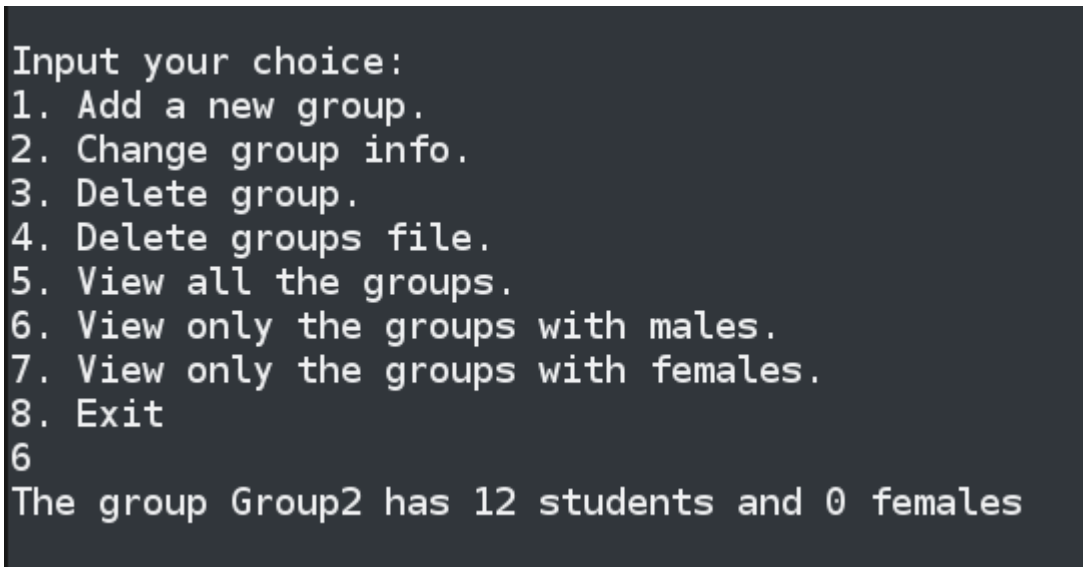

Рисунок 3 – Вывод групп, состоящих только из юношей

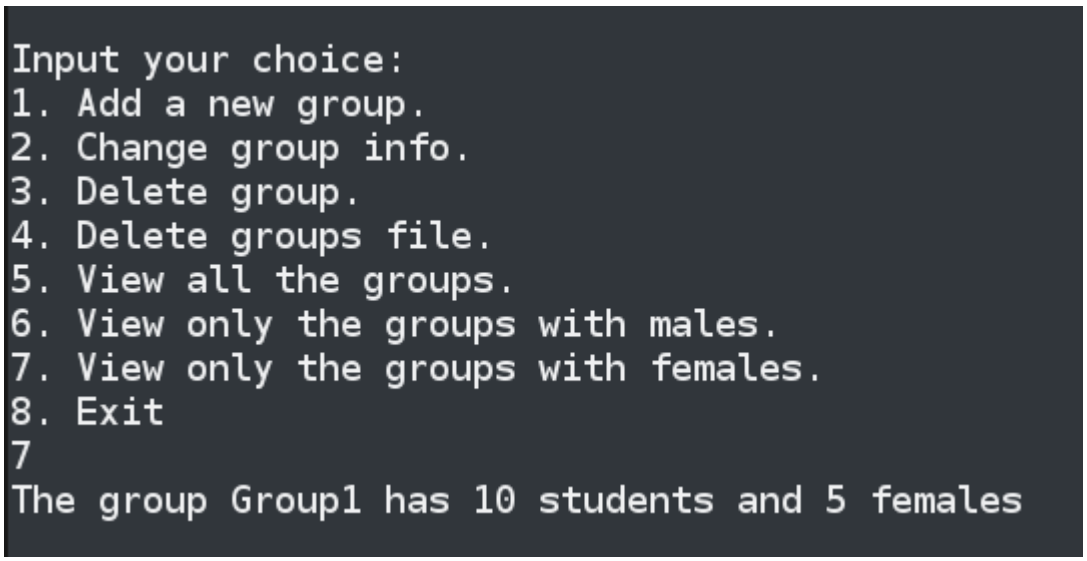

Рисунок 4 – Вывод групп, в которых количество юношей и девушек равны

```
The group Group2 has 12 students and 0 females

The group Group3 has 6 students and 6 females

Input your choice:
1. Add a new group.
2. Change group info.
3. Delete group.
4. Delete groups file.
5. View all the groups.
6. View only the groups with males.
7. View only the groups with females.
8. Exit
2
Input the group name:
Group2
Input new group name ('-' if remain old):
-
Input new students number ('-' if remain old):
15
Input new females number ('-' if remain old):
3
Input your choice:
1. Add a new group.
2. Change group info.
3. Delete group.
4. Delete groups file.
5. View all the groups.
6. View only the groups with males.
7. View only the groups with females.
8. Exit
5
The group Group1 has 10 students and 5 females

The group Group2 has 15 students and 3 females
```

Рисунок 5 – Изменение информации о группе в файле

```
Input your choice:
1. Add a new group.
2. Change group info.
3. Delete group.
4. Delete groups file.
5. View all the groups.
6. View only the groups with males.
7. View only the groups with females.
8. Exit
3
Input the group name:
Group3
The file has been successfully deleted.

The group has been deleted.

Input your choice:
1. Add a new group.
2. Change group info.
3. Delete group.
4. Delete groups file.
5. View all the groups.
6. View only the groups with males.
7. View only the groups with females.
8. Exit
5
The group Group1 has 10 students and 5 females

The group Group2 has 15 students and 3 females
```

Рисунок 6 – Удаление группы из файла