МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

## Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных систем и технологий

Отчет по лабораторной работе № 7

по дисциплине:” Системное программирование”

на тему: ”Концепция «Файл» и операции ввода-вывода ”

Вариант 6

Выполнили**:**  студент группы 10701222 Дяденькина Т.М.

Приняла**:** пр. Давыденко Н.В.

Минск 2024

**Цель работы**: Изучить фундаментальные основы и концепции файлового ввода-вывода в Linux, системные вызовы низкоуровневого ввода-вывода, основные функции ввода-вывода стандартной библиотеки языка программирования С, а также закрепить навыки применения данной библиотеки при написании системных программ. Изучить встроенный инструментарий для разработки приложений под семейство ОС Linux и фундаментальные основы системного программирования с использованием компиляторов gcc/g++, отладчика gdb и других для проектирования, компиляции, отладки и запуска приложений на языке программирования С/C++.

**Задание**

Уровень 1

Перестроить программу из упражнений 1-3, проделав то же самое, но с использованием механизма буферизации. Проведите компиляцию однофайлового проекта. Исходные файлы программ дополните комментариями. Исходные коды измененных программ, команды и результат компиляции программ, а также результаты работы программ представьте в отчете.

Уровень 2

Выполните задание уровня 1.

С помощью очень полезной утилиты strace, запустите переданную ей программы (листинга 6 и 9) и выведите в стандартный поток ошибок отчет об использованных системных вызовах. Полученный результат представьте в отчете.

Уровень 3

Выполните задание уровня 2.

Утилитой diff или другой подобной утилитой сравните полученные отчеты. Используя утилиту diff или другую подобную утилиту, сравните текст программ листинга 6 и 9. Разберитесь в кодах программ, запустите программы и изучите их поведение.

**Выполнение**

Уровень 1

Abook1

#include <sys/uio.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/stat.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#define ABOOK\_FNAME "example1.txt"

#define NAME\_LENGTH 50

#define PHONE\_LENGTH 30

#define EMAIL\_LENGTH 30

struct iovec ab\_entry[3];

char name\_buffer[NAME\_LENGTH];

char phone\_buffer[PHONE\_LENGTH];

char email\_buffer[EMAIL\_LENGTH];

void abook\_failed(int retcode) {

fprintf(stderr, "Cannot open address book\n");

exit(retcode);

}

void abook\_add(void) {

printf("Name: ");

scanf("%s", ab\_entry[0].iov\_base);

printf("Phone number: ");

scanf("%s", ab\_entry[1].iov\_base);

printf("E-mail: ");

scanf("%s", ab\_entry[2].iov\_base);

int fd = open(ABOOK\_FNAME, O\_WRONLY | O\_CREAT | O\_APPEND,

S\_IRUSR | S\_IWUSR | S\_IRGRP);

if (fd == -1) abook\_failed(1);

if (writev(fd, ab\_entry, 3) <= 0) {

fprintf(stderr, "Cannot write to address book\n");

exit(1);

}

close(fd);

}

void abook\_find(void) {

char find\_buffer[NAME\_LENGTH];

printf("Name: ");

scanf("%s", find\_buffer);

int fd = open(ABOOK\_FNAME, O\_RDONLY);

if (fd == -1) abook\_failed(1);

while (readv(fd, ab\_entry, 3) > 0) {

if (!strcmp(find\_buffer, ab\_entry[0].iov\_base)) {

printf("Found: %s, %s, %s\n", ab\_entry[0].iov\_base, ab\_entry[1].iov\_base, ab\_entry[2].iov\_base);

close(fd);

return;

}

}

printf("Name '%s' hasn't found\n", find\_buffer);

close(fd);

}

void abook\_delete(void) {

char find\_buffer[NAME\_LENGTH];

char tfile[] = "/tmp/abook-XXXXXX";

char ch;

int tfd = mkstemp(tfile);

if (tfd == -1) {

fprintf(stderr, "mkstemp() error\n");

exit(1);

}

unlink(tfile);

printf("Name: ");

scanf("%s", find\_buffer);

int fd = open(ABOOK\_FNAME, O\_RDONLY);

if (fd == -1) abook\_failed(1);

while (readv(fd, ab\_entry, 3) > 0) {

if (!strcmp(find\_buffer, (char \*)ab\_entry[0].iov\_base)) {

printf("Delete: %s\n", (char \*)ab\_entry[0].iov\_base);

continue;

}

writev(tfd, ab\_entry, 3);

}

close(fd);

close(tfd);

fd = open(ABOOK\_FNAME, O\_WRONLY | O\_TRUNC);

if (fd == -1) abook\_failed(1);

tfd = open(tfile, O\_RDONLY);

if (tfd == -1) abook\_failed(1);

while (readv(tfd, ab\_entry, 3) > 0) {

writev(fd, ab\_entry, 3);

}

close(fd);

close(tfd);

}

int main(int argc, char \*\*argv) {

if (argc < 2) {

fprintf(stderr, "Usage: abook { add , find, delete }\n");

return 1;

}

ab\_entry[0].iov\_base = name\_buffer;

ab\_entry[0].iov\_len = NAME\_LENGTH;

ab\_entry[1].iov\_base = phone\_buffer;

ab\_entry[1].iov\_len = PHONE\_LENGTH;

ab\_entry[2].iov\_base = email\_buffer;

ab\_entry[2].iov\_len = EMAIL\_LENGTH;

if (!strcmp(argv[1], "add")) {

abook\_add();

} else if (!strcmp(argv[1], "find")) {

abook\_find();

} else if (!strcmp(argv[1], "delete")) {

abook\_delete();

} else {

fprintf(stderr, "%s: unknown command\nUsage: abook { add , find, delete }\n", argv[1]);

return 1;

}

return 0;

}

Abook

#include <sys/uio.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/stat.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#define ABOOK\_FNAME "example.txt"

#define NAME\_LENGTH 50

#define PHONE\_LENGTH 30

#define EMAIL\_LENGTH 30

struct iovec ab\_entry[3]; /\* 1) NAME; 2) PHONE and 3) E-MAIL \*/

char name\_buffer[NAME\_LENGTH];

char phone\_buffer[PHONE\_LENGTH];

char email\_buffer[EMAIL\_LENGTH];

void abook\_failed(int retcode) {

fprintf(stderr, "Cannot open address book\n");

exit(retcode);

}

void abook\_add(void) {

printf("Name: ");

scanf("%s", ab\_entry[0].iov\_base);

printf("Phone number: ");

scanf("%s", ab\_entry[1].iov\_base);

printf("E-mail: ");

scanf("%s", ab\_entry[2].iov\_base);

int fd = open(ABOOK\_FNAME, O\_WRONLY | O\_CREAT | O\_APPEND,

S\_IRUSR | S\_IWUSR | S\_IRGRP);

if (fd == -1) abook\_failed(1);

if (writev(fd, ab\_entry, 3) <= 0) {

fprintf(stderr, "Cannot write to address book\n");

exit(1);

}

close(fd);

}

void abook\_find(void) {

char find\_buffer[NAME\_LENGTH];

printf("Name: ");

scanf("%s", find\_buffer);

int fd = open(ABOOK\_FNAME, O\_RDONLY);

if (fd == -1) abook\_failed(1);

while (readv(fd, ab\_entry, 3) > 0) {

if (!strcmp(find\_buffer, ab\_entry[0].iov\_base)) {

printf("Found: %s, %s, %s\n", ab\_entry[0].iov\_base, ab\_entry[1].iov\_base, ab\_entry[2].iov\_base);

close(fd);

return;

}

}

printf("Name '%s' hasn't found\n", find\_buffer);

close(fd);

}

int main(int argc, char \*\*argv) {

if (argc < 2) {

fprintf(stderr, "Usage: abook { add , find }\n");

return 1;

}

ab\_entry[0].iov\_base = name\_buffer;

ab\_entry[0].iov\_len = NAME\_LENGTH;

ab\_entry[1].iov\_base = phone\_buffer;

ab\_entry[1].iov\_len = PHONE\_LENGTH;

ab\_entry[2].iov\_base = email\_buffer;

ab\_entry[2].iov\_len = EMAIL\_LENGTH;

if (!strcmp(argv[1], "add")) {

abook\_add();

} else if (!strcmp(argv[1], "find")) {

abook\_find();

} else {

fprintf(stderr, "%s: unknown command\nUsage: abook { add , find }\n", argv[1]);

return 1;

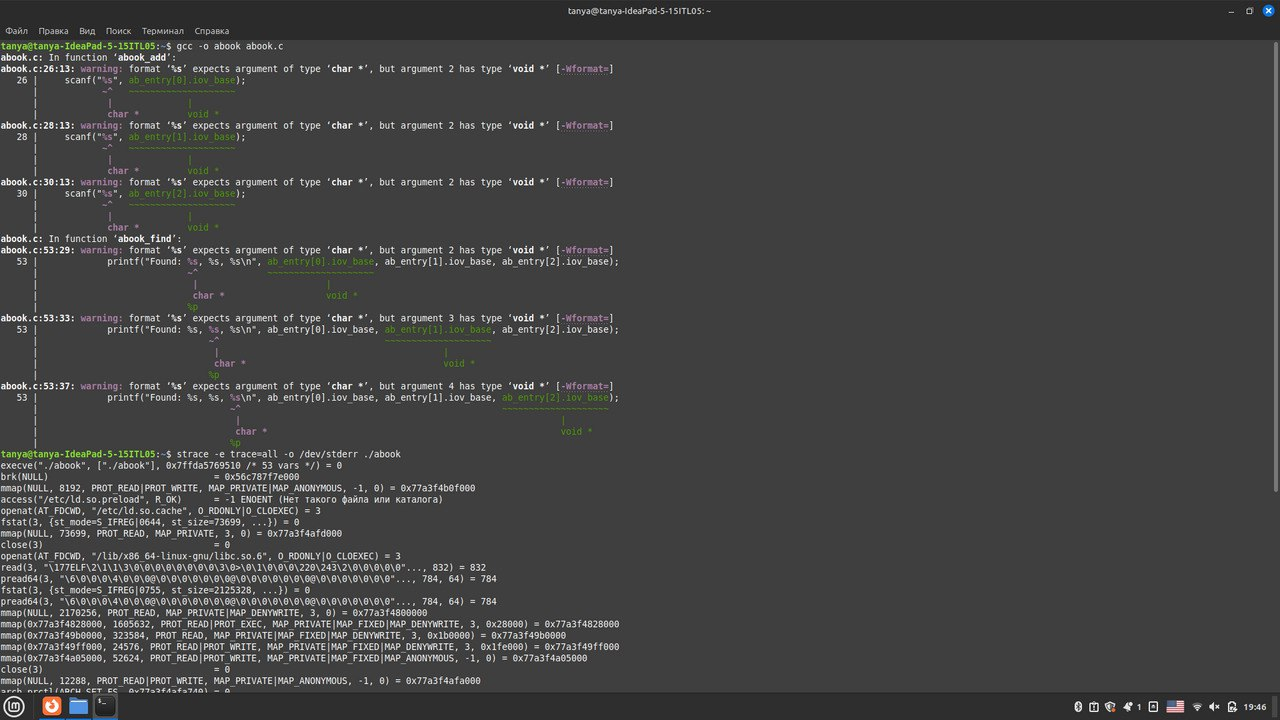
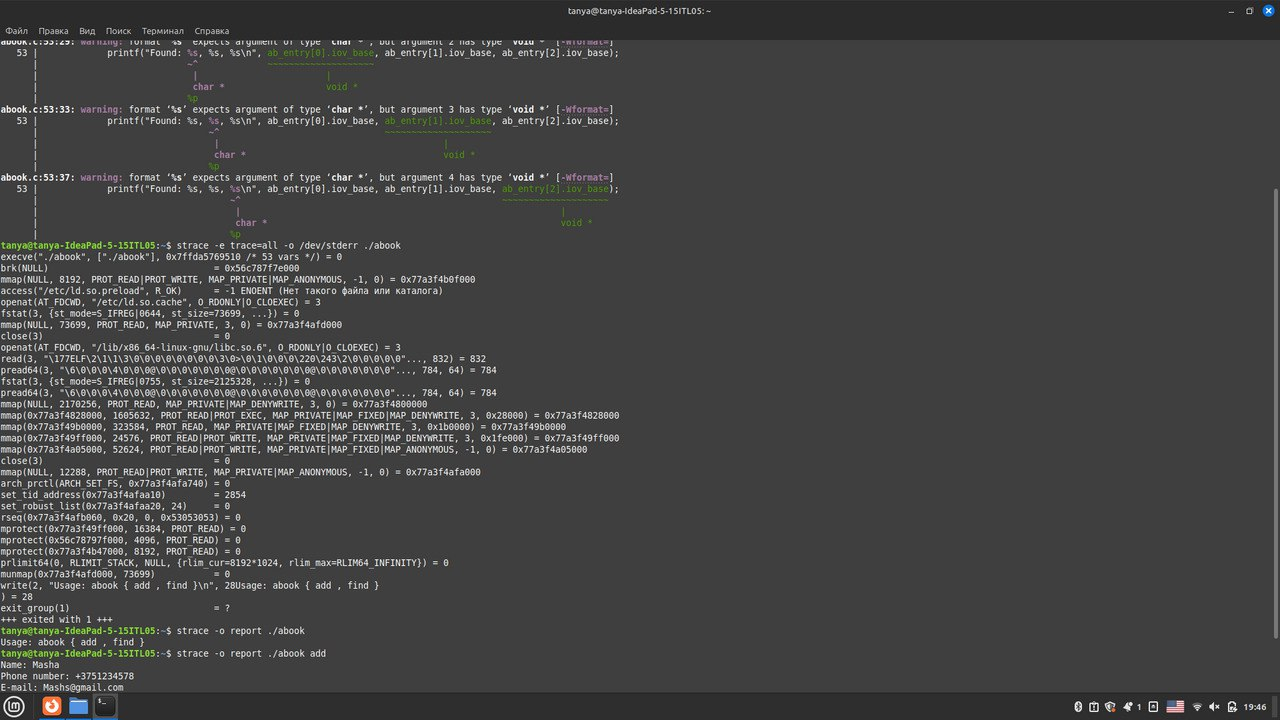
}

return 0;

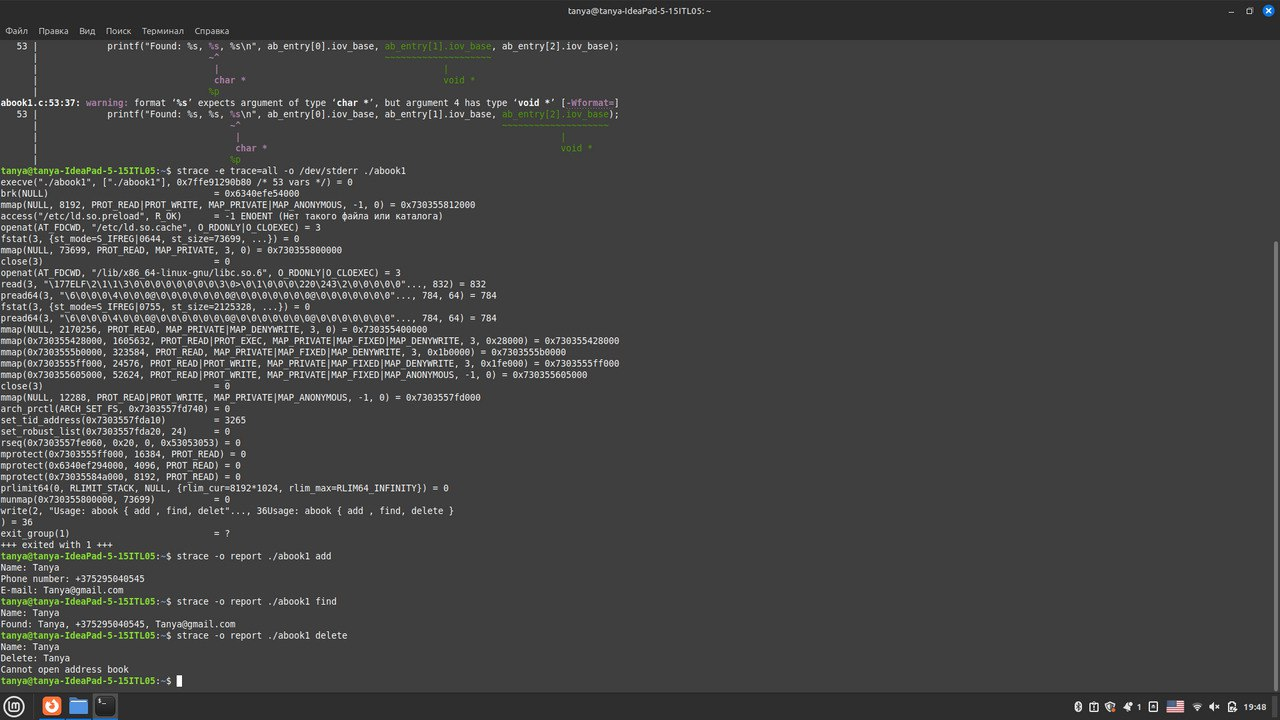
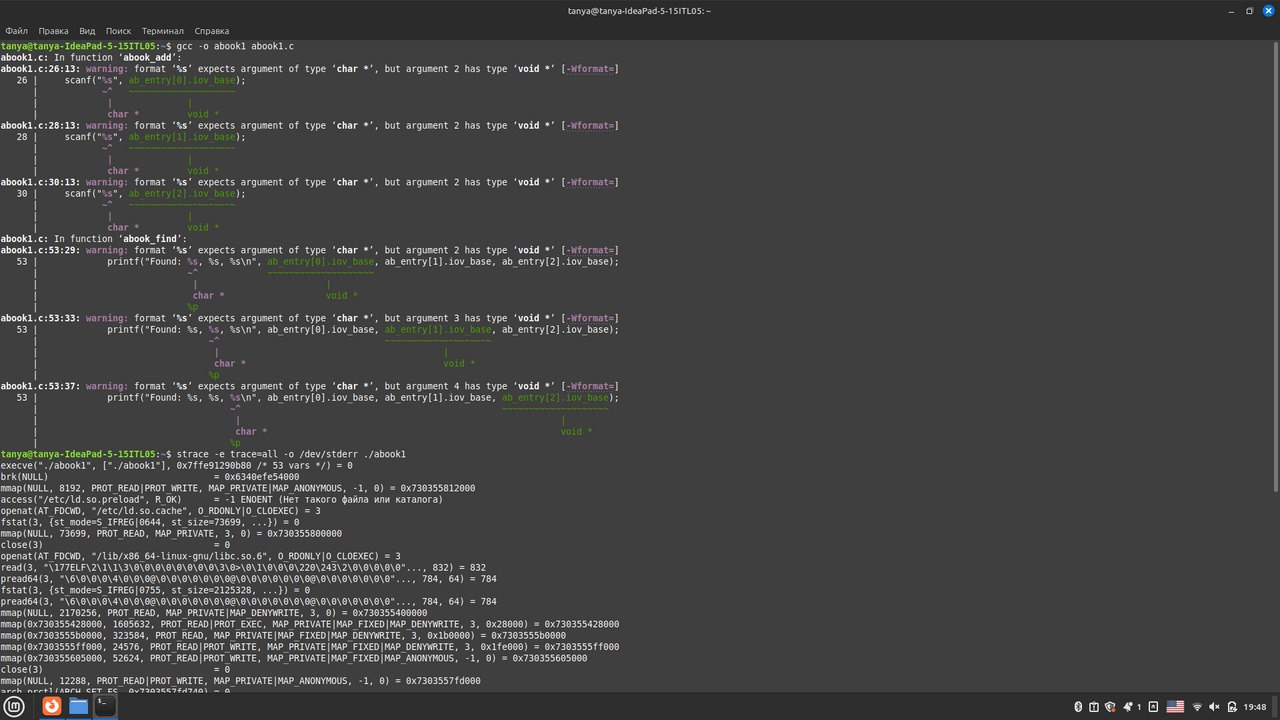
}

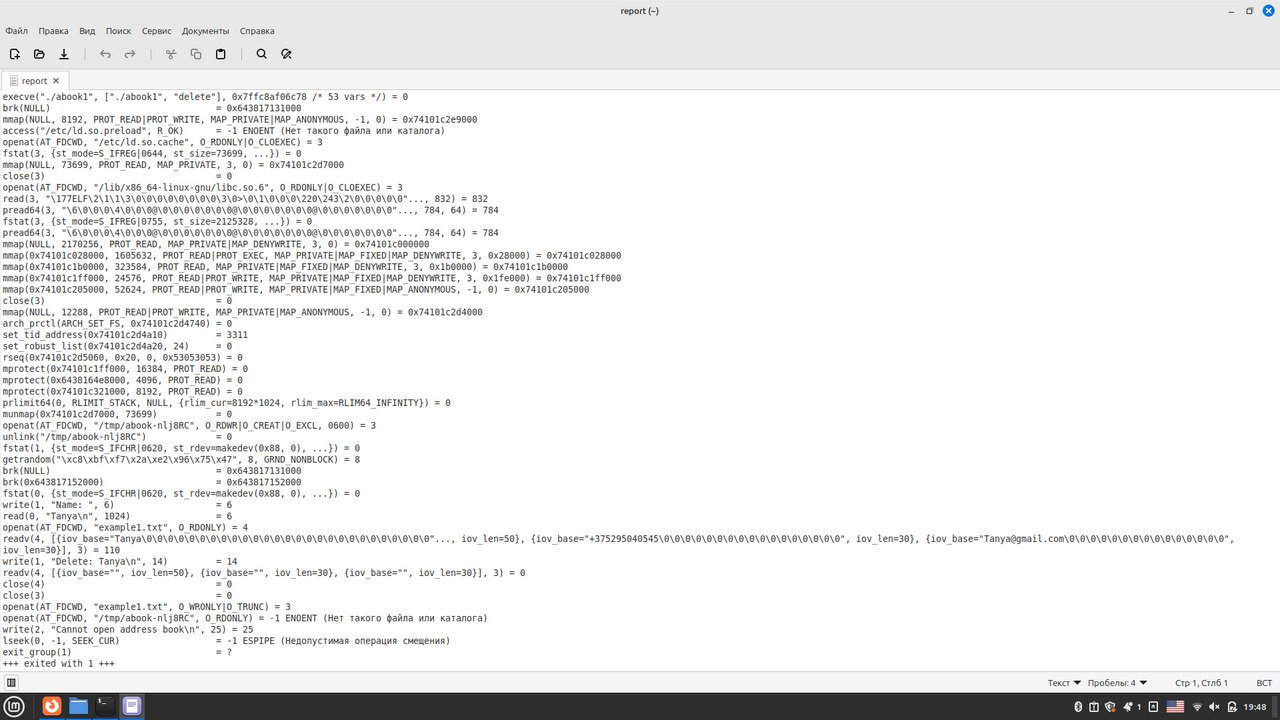
Уровень 2

Abook



Abook1





strace -e trace=all -o /dev/stderr <название\_программы>

### Параметры:

- -e trace=all — указывает strace отслеживать все системные вызовы. Вы можете изменить это на конкретные вызовы, такие как file, network и т. д., если вам нужно что-то конкретное.

- -o /dev/stderr — указывает strace выводить отчет об системных вызовах в стандартный поток ошибок (stderr).

**Контрольные вопросы**

В чем разница использования двух форм директивы: #include или #include “filename”?

Ответ:

Директива #include имеет две формы:

   #include <filename>, которая сообщает препроцессору искать файл в системных путях (в местах хранения системных библиотек языка С++).

   #include "filename", которая сообщает препроцессору искать файл в текущей директории проекта. Если его там не окажется, то препроцессор начнет проверять системные пути и любые другие, которые вы указали в настройках вашей IDE Эта форма используется для подключения пользовательских заголовочных файлов.