МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

## Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных систем и технологий

**Отчёт по лабораторной работе № 21**

по дисциплине: ”Системное программирование”

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

Вариант -

Выполнил**:** студент группы 10701222 Зухта К.М.

Медведский Е.В.

Принял**:** пр. Давыденко Н.В.

# Лабораторная работа № 15. *Концепция «Файл» и операции ввода-вывода*.

## Цель: 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 1.

Перестроить программу из упражнений 1-3, проделав то же самое, но с ис-

пользованием механизма буферизации. Проведите компиляцию однофайло-

вого проекта. Исходные файлы программ дополните комментариями. Исход-

ные коды измененных программ, команды и результат компиляции программ,

а также результаты работы программ представьте в отчете.

### Листинг кода

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define ABOOK\_FNAME "abook"

#define NAME\_LENGTH 50

#define PHONE\_LENGTH 30

#define EMAIL\_LENGTH 30

struct ab\_entry {

char name[NAME\_LENGTH];

char phone[PHONE\_LENGTH];

char email[EMAIL\_LENGTH];

};

void abook\_failed(int retcode) {

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

exit(retcode);

}

void abook\_add(void) {

struct ab\_entry entry;

printf("Name: ");

scanf("%49s", entry.name); // Ограничиваем ввод до длины буфера

printf("Phone number: ");

scanf("%29s", entry.phone); // Ограничиваем ввод до длины буфера

printf("E-mail: ");

scanf("%29s", entry.email); // Ограничиваем ввод до длины буфера

FILE \*file = fopen(ABOOK\_FNAME, "a"); // Открываем файл для добавления

if (file == NULL) abook\_failed(1);

fprintf(file, "%s %s %s\n", entry.name, entry.phone, entry.email); // Записываем запись в файл

fclose(file); // Закрываем файл

}

void abook\_find(void) {

char find\_buffer[NAME\_LENGTH];

struct ab\_entry entry;

int found = 0;

printf("Name: ");

scanf("%49s", find\_buffer); // Ограничиваем ввод

FILE \*file = fopen(ABOOK\_FNAME, "r"); // Открываем файл для чтения

if (file == NULL) abook\_failed(1);

// Читаем построчно данные из файла

while (fscanf(file, "%49s %29s %29s", entry.name, entry.phone, entry.email) == 3) {

if (strcmp(find\_buffer, entry.name) == 0) {

printf("Phone: %s\n", entry.phone);

printf("E-mail: %s\n", entry.email);

found = 1;

break;

}

}

if (!found) {

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

}

fclose(file); // Закрываем файл

}

void abook\_delete(void) {

char find\_buffer[NAME\_LENGTH];

struct ab\_entry entry;

int found = 0;

printf("Name: ");

scanf("%49s", find\_buffer); // Ограничиваем ввод

FILE \*file = fopen(ABOOK\_FNAME, "r"); // Открываем файл для чтения

if (file == NULL) abook\_failed(1);

// Временный файл для записи оставшихся записей

FILE \*temp\_file = fopen("/tmp/abook\_temp", "w");

if (temp\_file == NULL) abook\_failed(1);

// Читаем записи из исходного файла

while (fscanf(file, "%49s %29s %29s", entry.name, entry.phone, entry.email) == 3) {

if (strcmp(find\_buffer, entry.name) == 0) {

printf("Deleted: %s\n", entry.name);

found = 1;

} else {

// Записываем все оставшиеся записи во временный файл

fprintf(temp\_file, "%s %s %s\n", entry.name, entry.phone, entry.email);

}

}

fclose(file); // Закрываем исходный файл

fclose(temp\_file); // Закрываем временный файл

// Заменяем исходный файл временным

if (remove(ABOOK\_FNAME) != 0) {

fprintf(stderr, "Error deleting original file\n");

exit(1);

}

if (rename("/tmp/abook\_temp", ABOOK\_FNAME) != 0) {

fprintf(stderr, "Error renaming temporary file\n");

exit(1);

}

if (!found) {

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

}

}

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

if (argc < 2) {

fprintf(stderr, "Too few arguments\n");

return 1;

}

if (strcmp(argv[1], "add") == 0) {

abook\_add();

} else if (strcmp(argv[1], "find") == 0) {

abook\_find();

} else if (strcmp(argv[1], "delete") == 0) {

abook\_delete();

} else {

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

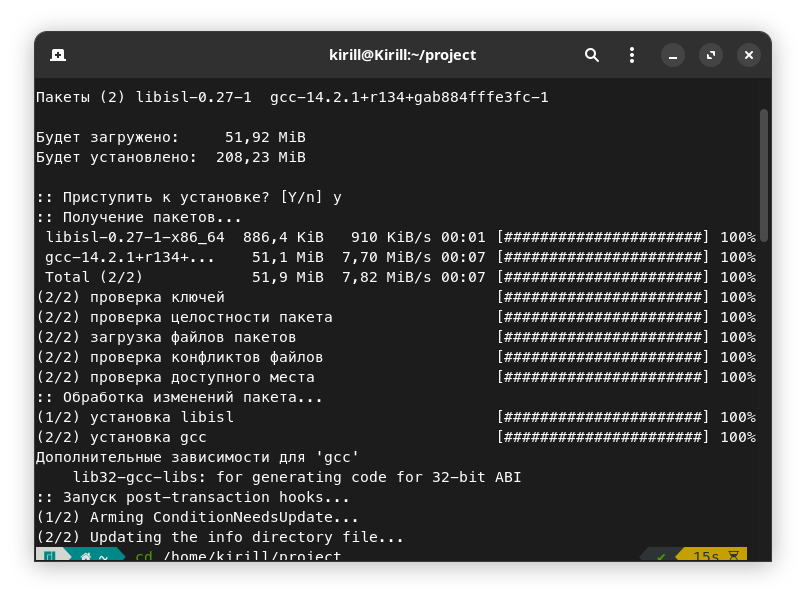
return 1;

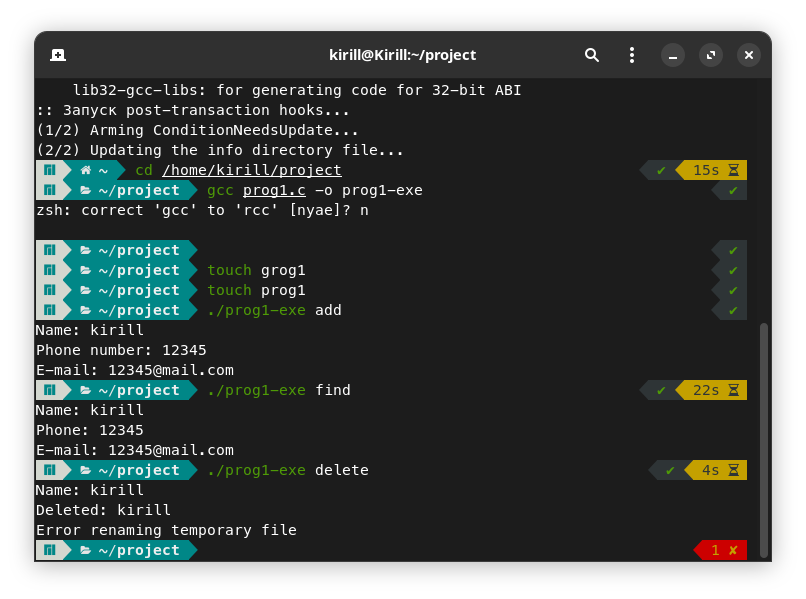
}

return 0;

}

# Скриншоты результатов

Установка gcc

  
Компиляция и запуск файла prog1

## Задание 2.

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

Листинг prod 2

#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 "abook"

#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\_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, ab\_entry[0].iov\_base)) {

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

continue;

}

writev(tfd, ab\_entry, 3);

}

close(fd);

lseek(tfd, 0, SEEK\_SET);

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

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

while (read(tfd, &ch, 1) > 0) write(fd, &ch, 1);

close(fd);

close(tfd);

}

void abook\_add(void) {

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;

printf("Name: ");

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

printf("Phone number: ");

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

printf("E-mail: ");

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

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("Phone: %s\n", (char\*)ab\_entry[1].iov\_base);

printf("E-mail: %s\n", (char\*)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, "Too few arguments\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]);

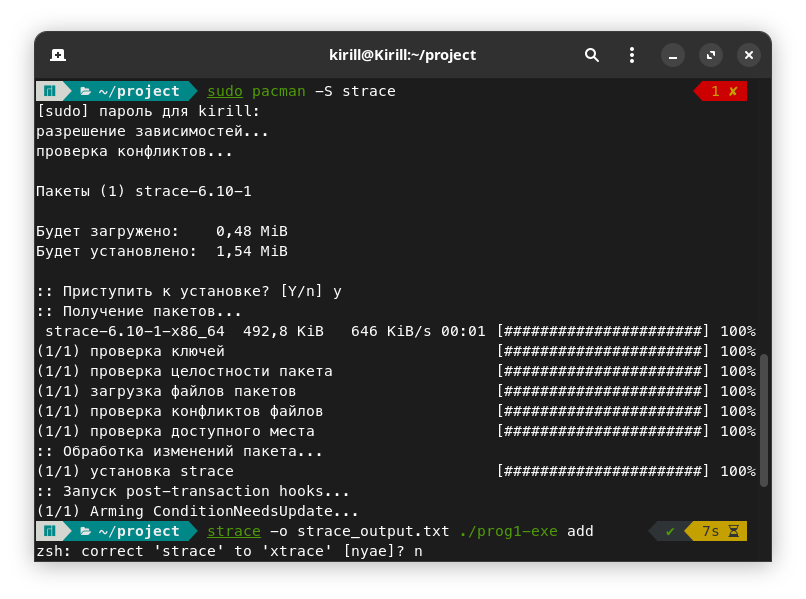
return 1;

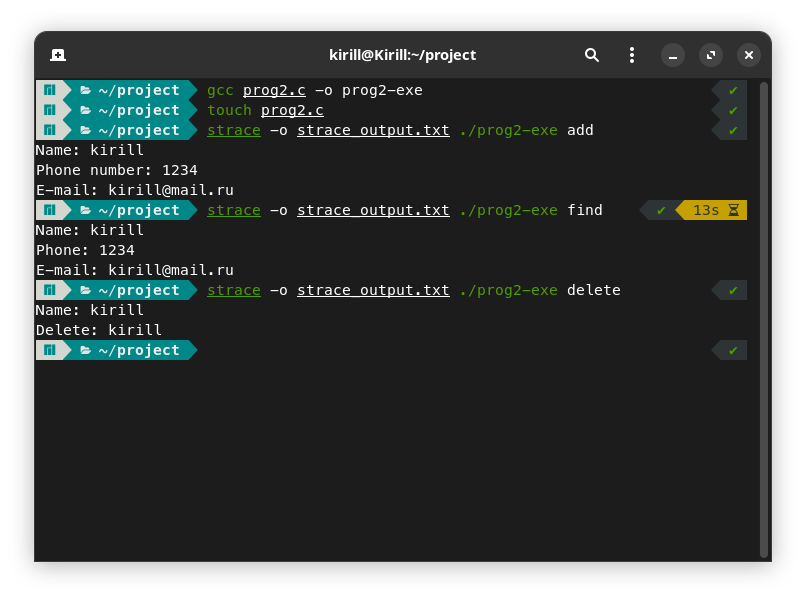
}

return 0;

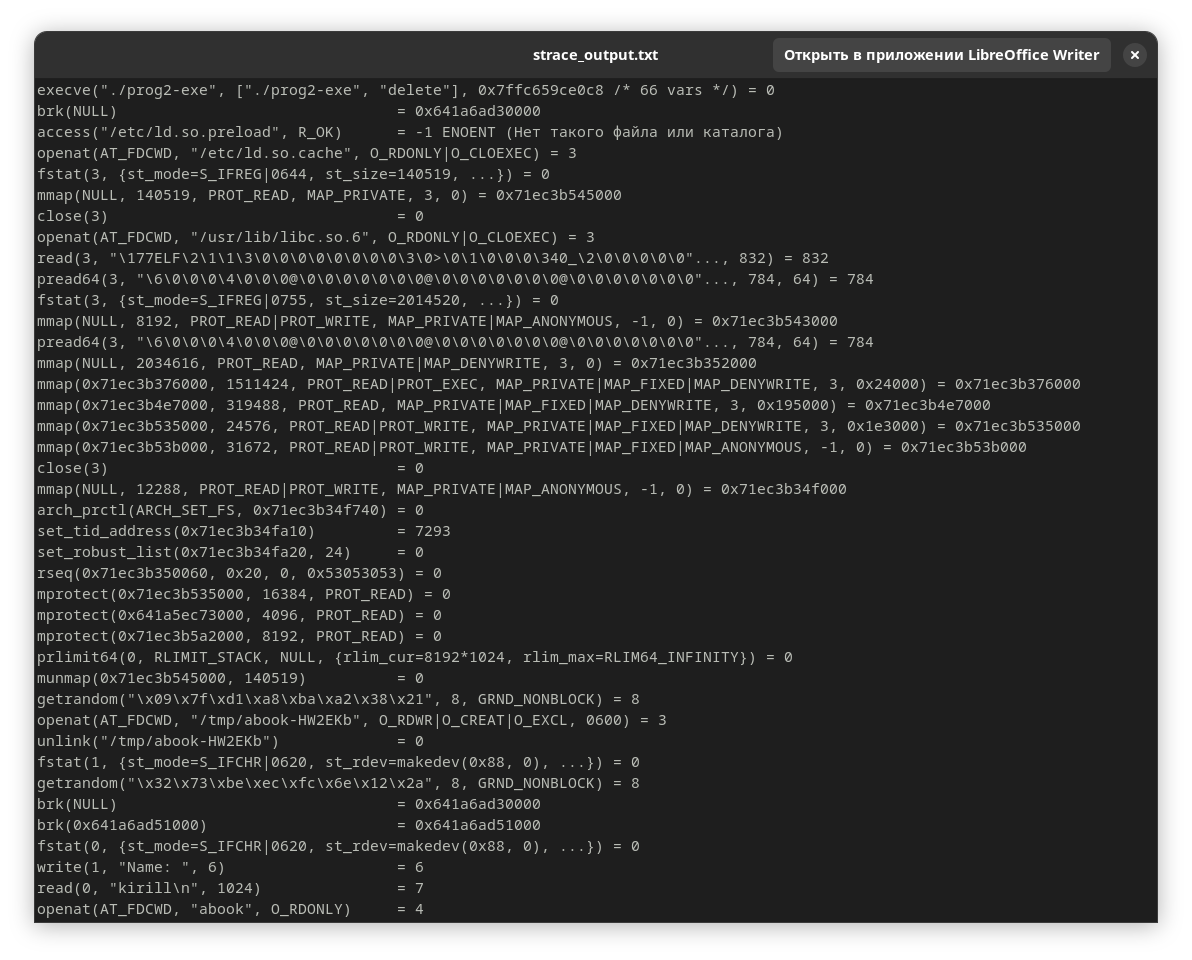
}

### Скриншоты результатов

Установка утилиты Strace



Компиляция Prog2 и использование strace

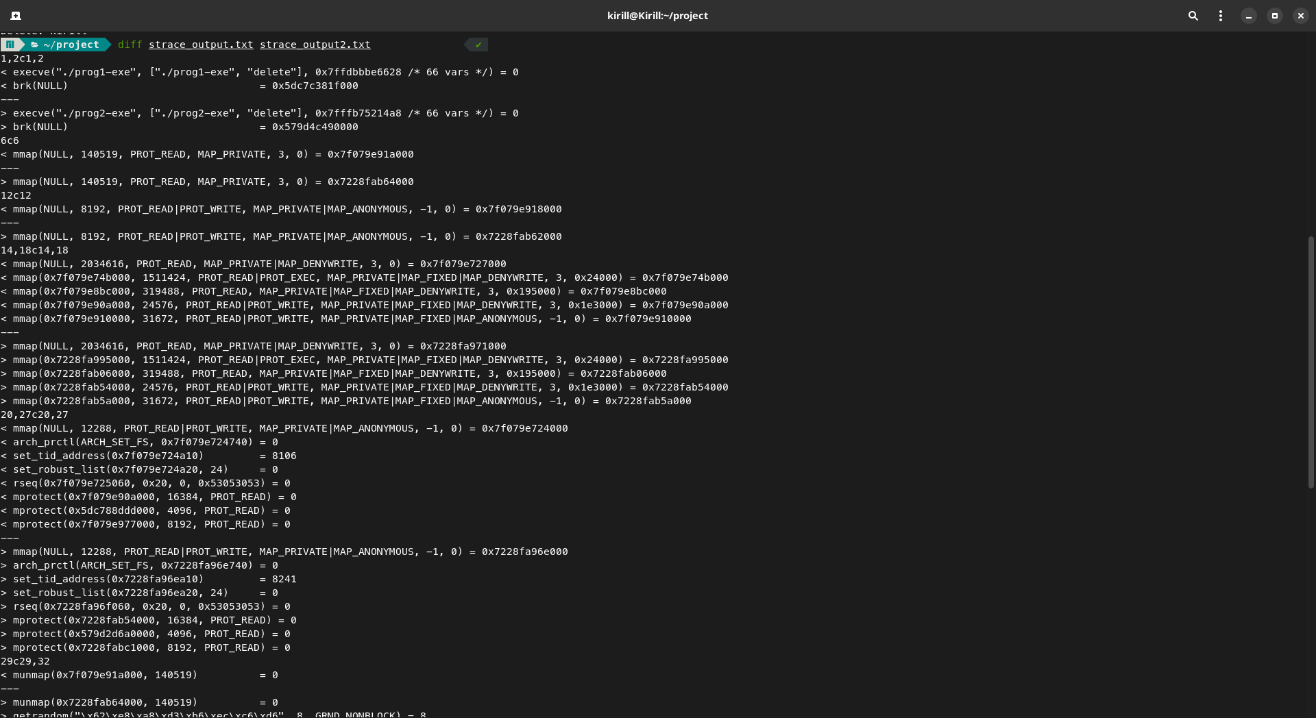


Полученный txt файл

## Задание 3.

Утилитой diff или другой подобной утилитой сравните полученные отчеты

Скриншоты



Команда diff strace\_output.txt strace\_output2.txt

Сравнение изменений в двух файлах

## Контрольные вопросы

Зухта

7. Перечислите составляющие модели файлового ввода-вывода

Ответ: включает в себя несколько ключевых составляющих:

Файлы:

- Основной объект, с которым происходит взаимодействие. Файлы могут быть текстовыми или бинарными.

Потоки:

- Представляют собой абстракции для передачи данных. Потоки могут быть входными (для чтения) и выходными (для записи).

Драйверы:

- Программные компоненты, которые управляют взаимодействием с устройствами хранения (например, жесткими дисками, SSD, сетевыми устройствами).

Буферизация:

- Механизм временного хранения данных в памяти для оптимизации операций чтения и записи. Это позволяет уменьшить количество обращений к устройствам.

Системные вызовы:

- Интерфейсы, через которые приложения взаимодействуют с операционной системой для выполнения операций ввода-вывода.

Управление памятью:

- Процессы, связанные с выделением и освобождением памяти для хранения данных, передаваемых через потоки.

Управление ошибками:

- Механизмы для обработки ошибок, возникающих во время операций ввода-вывода, таких как отсутствие доступа к файлам или переполнение буфера.

Эти компоненты работают вместе, чтобы обеспечить эффективное и надежное выполнение операций ввода-вывода в компьютерных системах.

Медведский

В Linux временные файлы используются для хранения данных, которые необходимы программам и системным процессам на короткий срок. Эти файлы обычно создаются в каталоге /tmp или /var/tmp.

Вот несколько ключевых моментов о временных файлах в Linux:

Каталог /tmp: Этот каталог предназначен для временных файлов, которые могут быть удалены при перезагрузке системы. Программы и процессы используют его для хранения временных данных, которые не нужно сохранять после завершения работы1.

Каталог /var/tmp: В отличие от /tmp, файлы в этом каталоге могут сохраняться между перезагрузками системы. Он используется для временных файлов, которые могут понадобиться дольше, чем один сеанс работы системы1.

tmpfs: Это временная файловая система, которая хранит файлы в оперативной памяти. Она используется для ускорения доступа к временным файлам и автоматического удаления их при перезагрузке системы2.

Временные файлы играют важную роль в работе системы, обеспечивая временное хранение данных для различных операций и процессов.