Федеральное государственное автономное

образовательное учреждение

Высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Институт космических и информационных технологий |
| институт |
| Информатика |
| кафедра |

**ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ №7**

|  |
| --- |
| Использование средств управления устройствами и файловой системой /proc |
| Тема |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Преподаватель | |  |  |  | А. Н. Пупков |
|  | |  | подпись, дата |  | инициалы, фамилия |
| Студент | КИ21-16/1б, 032155832 |  |  |  | Н. А. Терентьев |
|  | номер группы, зачётной книжки |  | подпись, дата |  | инициалы, фамилия |

Красноярск 2023

**СОДЕРЖАНИЕ**

1 Цели 3

2 Задачи 3

3 Описание варианта задания 4

4 Ход выполнения 4

4.1 Вывод справки 4

4.2 Работа с файловой системой /proc 5

4.3 Создание виртуальной файловой системы 8

4.4 Работа с программами 9

4.5 Doxygen документация 9

5 Выводы 10

Список использованных источников 11

**1 Цель**

Изучение особенностей средств управления специфичными объектами файловых систем в GNU/Linux.

**2 Задачи**

Для выполнения практической работы необходимо выполнить  
следующие задачи:

* ознакомиться с краткими теоретическими сведениями по управлению устройствами и файловой системой */*proc в ОС GNU/Linux;
* модифицировать результаты выполнения ЛР 6 добавлением использования программных средств для работы со специфичными объектами файловой системы. Обеспечить сборку обеих частей программы, как отдельно, так и полностью, с использованием инструментального набора GNU Autotools. Серверная и клиентская часть при запуске настраиваются аргументами командной строки. Для обработки аргументов командной строки рекомендовано использование функций семейства getopt. **Запретить** передачу аргументов командной строки через **перенаправление стандартных потоков ввод-вывода.** Один из аргументов командной строки - **вывод справки** по аргументам командной строки для серверной и клиентской частей программы, соответственно. Справка также должна выводиться в том случае, если не задан ни один из аргументов командной строки. Еще один аргумент - включает и отключает проверку заполнения "дисков", на которых хранятся log-файлы. Обе части должны разворачиваться **в виртуальной файловой системе** (ВФС). Изначально ВФС отсутствует.;
* используя изученные механизмы, разработать и отладить:
* серверную часть. Должен создаваться только единственный экземпляр процесса-сервера, при попытке запуска других экземпляров процесса выводится сообщение об ошибке.
* клиентскую часть;
* составить общее описание результатов, инструкции по сборке и использованию программы, а также инструкцию по получению документации, сформировать архив формата tar.gz и представить на проверку с исходными текстами программы. **Внимание**: исходные тексты программ должны соответствовать принятому стандарту кодирования, а также содержать комментарии в стиле системы Doxygen.

**3 Описание варианта задания**

Работа выполнена в соответствии с вариантом 20.

Клиент принимает от пользователя коэффициенты квадратного  
уравнения (ax2+bx+c=0) или кубического уравнения (ax3+bx2+cx+d=0), каждый  
из которых не равен нулю или единице, и отсылает их серверу.  
 Сервер принимает коэффициенты и выводит на экран разложение  
кубического или квадратного уравнения на множители.

**4 Ход выполнения**

**4.1 Вывод справки**

Для того, чтобы выводить справку по аргументам командной строки для серверной и клиентской части, был добавлен соответствующий код в файл interface.c. Добавленный код представлен в листинге 1.

Листинг 1 – Содержимое файла interface.c

void PrintClientHelp(void)

{

printf("Использование: ./client [опции] коэффициенты\n");

printf("Опции:\n");

printf("-h \t\tВывести эту справку и выйти\n");

printf("-l logFile\tЗадать имя файла для логирования\n");

printf("-t timeout\tЗадать таймаут для ожидания ответа\n");

printf("-f filesystem\tПроверка заполнения \"дисков\", на которых"

" хранятся log-файлы\n");

printf("Коэффициенты:\n");

printf("a b c\t\tКоэффициенты квадратного уравнения\n");

printf("a b c d\t\tКоэффициенты кубического уравнения\n");

}

void PrintServerHelp(void)

{

printf("Использование: ./server [опции]\n");

printf("Опции:\n");

printf("-h \t\tВывести эту справку и выйти\n");

printf("-l logFile\tЗадать имя файла для логирования\n");

printf("-t timeout\tЗадать таймаут для ожидания ответа\n");

printf("-f filesystem\tПроверка заполнения \"дисков\", на которых"

" хранятся log-файлы\n");

}

**4.2 Работа с файловой системой /proc**

Для того, чтобы добавить проверку заполнения «дисков», на которых хранятся log-файлы, создадим файл filesystem.c. Содержимое файла представлено на листинге 2.

Листинг 2 – Содержимое файла filesystem.c

#include <stdio.h>

#include <string.h>

#include <sys/statvfs.h>

#include <unistd.h>

// Файл с информацией о смонтированных файловых системах

#define MOUNT\_FILE "/proc/mounts"

#define THRESHOLD 90 // Порог заполнения диска в процентах

int CheckDiskFull()

{

FILE \*fp; // Указатель на файл

char line[256]; // Буфер для чтения строки из файла

char device[256]; // Буфер для хранения имени устройства

char mountPoint[256]; // Буфер для хранения точки монтирования

struct statvfs buf; // Структура для хранения данных о файловой системе

unsigned long freeSpace; // Свободное пространство на диске в байтах

unsigned long totalSpace; // Общее пространство на диске в байтах

int usage; // Процент заполнения диска

char buffer[256]; // Буфер для хранения пути к текущей директории

char\* directory; // Указатель для хранения результата функции getcwd()

directory = getcwd(buffer, sizeof(buffer));

if (directory == NULL)

{

perror("getcwd");

return 1;

}

printf("Текущая директория: %s\n", buffer);

// Открываем файл с информацией о смонтированных файловых системах

fp = fopen(MOUNT\_FILE, "r");

if (fp == NULL)

{

perror("Ошибка открытия файла");

return 1;

}

// Читаем файл построчно

while (fgets(line, sizeof(line), fp) != NULL)

{

// Разбиваем строку на токены по пробелам и получаем имя устройства и

// точку монтирования

sscanf(line, "%s %s", device, mountPoint);

// Проверяем, совпадает ли точка монтирования с директорией с log

// файлами или ее родителем

if (strcmp(mountPoint, directory) == 0 || strcmp(mountPoint, "/") == 0)

{

// Закрываем файл

fclose(fp);

// Получаем данные о файловой системе по точке монтирования

if (statvfs(mountPoint, &buf) == -1)

{

perror("Ошибка получения данных о файловой системе");

return 1;

}

// Вычисляем свободное и общее пространство на диске в байтах

freeSpace = buf.f\_bsize \* buf.f\_bavail;

totalSpace = buf.f\_bsize \* buf.f\_blocks;

// Вычисляем процент заполнения диска

usage = 100 \* (totalSpace - freeSpace) / totalSpace;

// Выводим информацию о диске и его заполнении

printf("Устройство: %s\n", device);

printf("Точка монтирования: %s\n", mountPoint);

printf("Свободное пространство: %lu байт\n", freeSpace);

printf("Общее пространство: %lu байт\n", totalSpace);

printf("Процент заполнения: %d%%\n", usage);

// Проверяем, превышает ли заполнение порог

if (usage > THRESHOLD)

{

printf("Предупреждение: диск заполнен более чем "

"на %d%%\n", THRESHOLD);

}

return 0;

}

}

// Если не нашли нужный диск, выводим сообщение об ошибке

printf("Не найден диск с log файлами\n");

return 1;

}

**4.3 Создание виртуальной файловой системы**

Была написана инструкция по создание ВФС, развертыванию программ в ней и удаления ВФС. На листинге 3 представлена инструкция.

Листинг 3 – Инструкция по созданию ВФС

Все команды от имени суперпользователя

Создаём директории для хранения виртуальных дисков

mkdir -p /mnt/vfs1

mkdir -p /mnt/vfs2

Создаём файлы для виртуальных дисков размером 100 МБ

dd if=/dev/zero of=vfs1.img bs=1M count=100

dd if=/dev/zero of=vfs2.img bs=1M count=100

Создаём файловую систему ext4 на виртуальном диске

mkfs.ext4 vfs1.img

mkfs.ext4 vfs2.img

Монтируем виртуальные диски в директории vfs1 и vfs2

mount -o loop vfs1.img /mnt/vfs1

mount -o loop vfs2.img /mnt/vfs2

Копируем программы в директории

cp ./server.c /mnt/vfs1

cp ./client.c /mnt/vfs2

Переходим в нужную директорию

cd /mnt/vfs1

cd /mnt/vfs2 rmgfgf

Компилируем программы

Запускаем программы

./client

./server

Возвращаемся в корневую директорию

cd /

Размонтируем виртуальные диски с помощью команды umount и удаляем директорию для хранения виртуальных дисков с помощью команды rm -rf.

umount /mnt/vfs1

umount /mnt/vfs2

Удаляем файлы виртуальных дисков и директорию vfs

rm -f vfs1.img

rm -f vfs2.img

rm -rf /mnt/vfs1

rm -rf /mnt/vfs2

**4.4 Работа с программами**

Работа с программами представлена на рисунке 1.

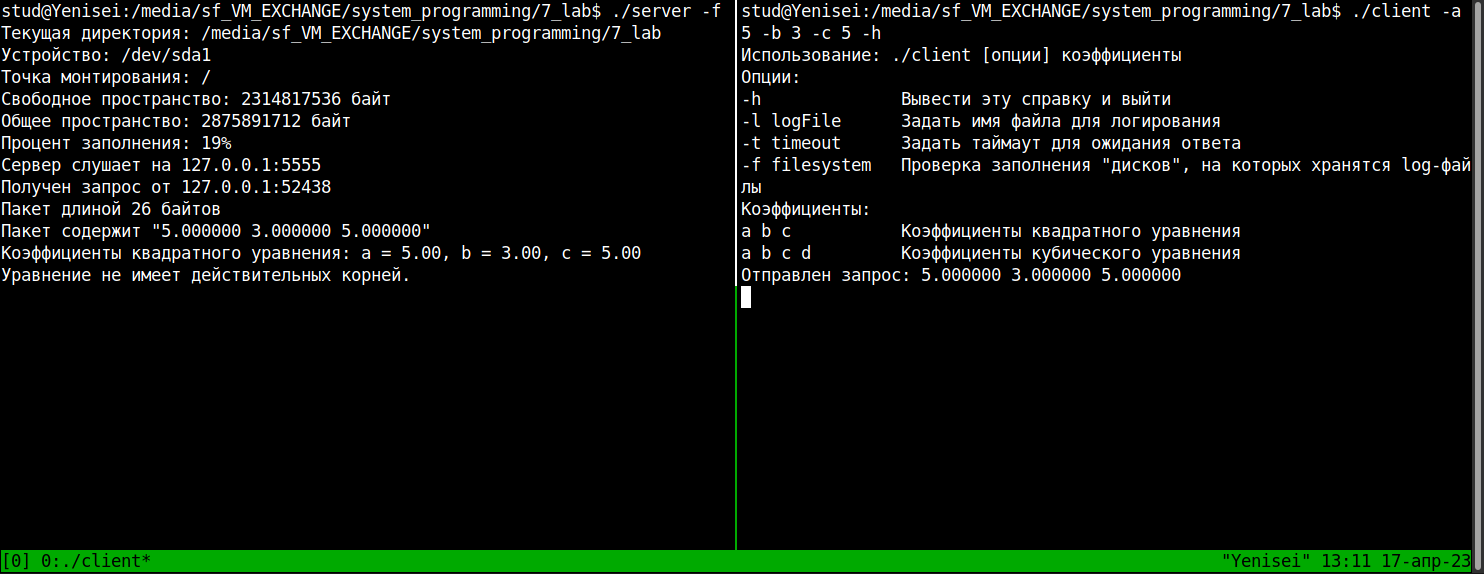


Рисунок 1 – Работа с программой

**4.5 Doxygen документация**

Все файлы содержат комментарии в стиле системы doxygen. Благодаря конфигурационному файлу была сгенерирована документация. Её внешний вид представлен на рисунке 2.

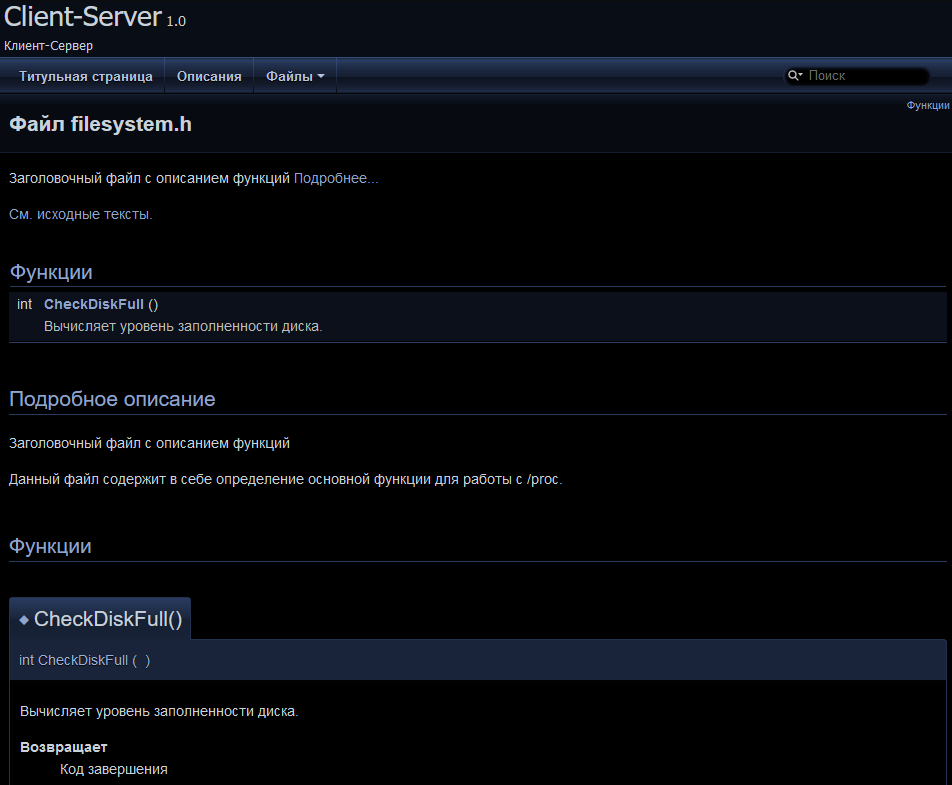


Рисунок 2 – Сгенерированная документация

**5** **Выводы**

В ходе работы были изучены средства управления специфичными объектами файловых систем в OC GNU/Linux.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1 СТО 4.2-07-2021 Система менеджмента качества. Общие требования  
к построению, изложении и оформлению документов учебной деятельности.  
Дата введения – 30.12.2-13.