МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. Шухова»**

**(БГТУ им. В. Г. Шухова)**



Кафедра программного обеспечения вычислительной

техники и автоматизированных систем

**Лабораторная работа №3**

по дисциплине: «Операционные системы»

на тему: «Файловые системы в ОС Linux (Ubuntu): сравнение, области эффективности. Виртуальная файловая система. Пользовательская файловая система»

Выполнил: ст. группы ПВ-223

Дмитриев Андрей Александрович

Проверили:

доц. Островский Алексей Мичеславович,

асс. Четвертухин Виктор Романович

Белгород, 2024

**Цель работы:** Изучить популярные файловые системы в ОС Linux (ext4, Btrfs, ReiserFS, NTFS, FAT32), определить область эффективности каждой их них, разобраться как осуществляется работа с виртуальной файловой системой (VFS) ОС Linux и выполнить разработку пользовательской файловой системы в соответствии с индивидуальным заданием.

**Условие индивидуального задания:**

2. Реализовать пользовательскую файловую систему, которая генерирует случайные ошибки при вызовах read и write (может использоваться с целью отладки программ).

**Ход выполнения работы**

**Задание 2.**

**Текст программы** индивидуального заданияна языке С с комментариями.

#define \_POSIX\_C\_SOURCE 200809L // Стандарт POSIX версии 2008 года с некоторыми дополнениями

#define FUSE\_USE\_VERSION 30

#include <time.h>

#include <fuse3/fuse.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <errno.h>

#include <fcntl.h>

#include <unistd.h>

#include <sys/stat.h> // Для S\_IFDIR и S\_IFREG

static const char \*LOG\_FILE = "/tmp/os\_lab3\_tmp.txt";

typedef struct file\_node {

    char \*name;

    char \*contents;

    struct file\_node \*next;

} file\_node;

void flip\_random\_bit(char \*buffer, size\_t size) {

    if (size == 0) return;

    size\_t pos = rand() % size;

    int bit = rand() % 8;

    buffer[pos] ^= (1 << bit);

}

static file\_node \*file\_list = NULL;

void log\_operation(const char \*operation, const char \*path) {

    FILE \*fp = fopen(LOG\_FILE, "a");

    if (!fp) {

        perror("Ошибка открытия лог-файла");

        return;

    }

    time\_t now = time(NULL);

    struct tm \*t = localtime(&now);

    fprintf(fp, "[%04d-%02d-%02d %02d:%02d:%02d] PID: %d | Операция: %s | Путь: %s\n",

            t->tm\_year + 1900,

            t->tm\_mon + 1,

            t->tm\_mday,

            t->tm\_hour,

            t->tm\_min,

            t->tm\_sec,

            getpid(),

            operation,

            path);

    fclose(fp);

}

file\_node \*find\_file(const char \*name) {

    file\_node \*current = file\_list;

    while (current) {

        if (strcmp(current->name, name) == 0) {

            return current;

        }

        current = current->next;

    }

    return NULL;

}

file\_node \*create\_file(const char \*name) {

    file\_node \*new\_file = malloc(sizeof(file\_node));

    if (!new\_file) {

        return NULL;

    }

    new\_file->name = strdup(name);

    new\_file->contents = strdup("");

    new\_file->next = file\_list;

    file\_list = new\_file;

    return new\_file;

}

void create\_backup(const char \*name) {

    file\_node \*file = find\_file(name);

    if (!file) return;

    size\_t bak\_name\_len = strlen(name) + 5; // Длина имени + ".bak"

    char \*bak\_name = malloc(bak\_name\_len);

    if (!bak\_name) return;

    snprintf(bak\_name, bak\_name\_len, "%s.bak", name);

    // Проверяем, есть ли уже .bak файл, и удаляем его

    file\_node \*bak\_file = find\_file(bak\_name);

    if (bak\_file) {

        free(bak\_file->contents);

        bak\_file->contents = NULL;

    } else {

        bak\_file = create\_file(bak\_name);

    }

    if (bak\_file) {

        bak\_file->contents = strdup(file->contents);

    }

    free(bak\_name);

}

static int fs\_getattr(const char \*path, struct stat \*stbuf, struct fuse\_file\_info \*fi) {

    (void)fi; // Параметр не используется

    memset(stbuf, 0, sizeof(struct stat));

    if (strcmp(path, "/") == 0) {

        stbuf->st\_mode = S\_IFDIR | 0755;

        stbuf->st\_nlink = 2;

    } else {

        file\_node \*file = find\_file(path + 1);

        if (!file) {

            return -ENOENT;

        }

        stbuf->st\_mode = S\_IFREG | 0644;

        stbuf->st\_nlink = 1;

        stbuf->st\_size = strlen(file->contents);

    }

    return 0;

}

static int fs\_readdir(const char \*path, void \*buf, fuse\_fill\_dir\_t filler, off\_t offset,

                    struct fuse\_file\_info \*fi, enum fuse\_readdir\_flags flags) {

    (void) offset;

    (void) fi;

    (void) flags; // Не используются

    if (strcmp(path, "/") != 0) {

        return -ENOENT;

    }

    filler(buf, ".", NULL, 0, 0);

    filler(buf, "..", NULL, 0, 0);

    file\_node \*current = file\_list;

    while (current) {

        filler(buf, current->name, NULL, 0, 0);

        current = current->next;

    }

    return 0;

}

static int fs\_open(const char \*path, struct fuse\_file\_info \*fi) {

    (void) fi; // Не используются

    if (!find\_file(path + 1)) {

        return -ENOENT;

    }

    return 0;

}

static int fs\_read(const char \*path, char \*buf, size\_t size, off\_t offset, struct fuse\_file\_info \*fi) {

    (void) fi;

    file\_node \*file = find\_file(path + 1);

    if (!file) {

        return -ENOENT;

    }

    log\_operation("read", path);

    size\_t len = strlen(file->contents);

    if ((size\_t)offset >= len) {

        return 0;

    }

    if ((size\_t)offset + size > len) {

        size = len - offset;

    }

    memcpy(buf, file->contents + offset, size);

    flip\_random\_bit(buf, size);

    return size;

}

static int fs\_write(const char \*path, const char \*buf, size\_t size, off\_t offset, struct fuse\_file\_info \*fi) {

    (void) fi;

    file\_node \*file = find\_file(path + 1);

    if (!file) {

        return -ENOENT;

    }

    create\_backup(path + 1);

    log\_operation("write", path);

    size\_t len = strlen(file->contents);

    if ((size\_t)offset + size > len) {

        char \*new\_contents = realloc(file->contents, offset + size + 1);

        if (!new\_contents) {

            return -ENOMEM;

        }

        file->contents = new\_contents;

        memset(file->contents + len, 0, offset + size - len);

    }

    memcpy(file->contents + offset, buf, size);

    file->contents[offset + size] = '\0';

    flip\_random\_bit(buf, size);

    return size;

}

static int fs\_create(const char \*path, mode\_t mode, struct fuse\_file\_info \*fi) {

    (void) mode;

    (void) fi;

    if (find\_file(path + 1)) {

        return -EEXIST;

    }

    log\_operation("create", path);

    if (!create\_file(path + 1)) {

        return -ENOMEM;

    }

    return 0;

}

static struct fuse\_operations fs\_oper = {

    .getattr    = fs\_getattr,

    .readdir    = fs\_readdir,

    .open       = fs\_open,

    .read       = fs\_read,

    .write      = fs\_write,

    .create     = fs\_create,

};

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

    return fuse\_main(argc, argv, &fs\_oper, NULL);

}

**Протоколы, логи, скриншоты, графики.**

Для решения задачи реализована функция flip\_random\_bit(char \*buffer, size\_t size), которая принимает buffer и случайно изменяет один бит. Заметно, как при каждой записи и чтении файлов из монтированной директории изменяется строка записанных данных.

Работа файловой системы:

andrev133@andrev133-VirtualBox:~/Desktop/lib3\_add$ echo "Это простой текстовый файл" > /tmp/lab3\_temp/test.txt

andrev133@andrev133-VirtualBox:~/Desktop/lib3\_add$ cat /tmp/lab3\_temp/test.txt

Это простой тевстовый файл

andrev133@andrev133-VirtualBox:~/Desktop/lib3\_add$ cat /tmp/lab3\_temp/test.txt.bak

andrev133@andrev133-VirtualBox:~/Desktop/lib3\_add$ echo "Это новый текстовый файл" > /tmp/lab3\_temp/test.txt

andrev133@andrev133-VirtualBox:~/Desktop/lib3\_add$ cat /tmp/lab3\_temp/test.txt.bak

Это простой текстов��й файл

andrev133@andrev133-VirtualBox:~/Desktop/lib3\_add$ cat /tmp/lab3\_temp/test.txt.bak

Это простой текстовый файл\*andrev133@andrev133-VirtualBox:~/De

andrev133@andrev133-VirtualBox:~/Desktop/lib3\_add$ cat /tmp/os\_lab3\_tmp.txt

[2024-12-31 15:05:22] PID: 5848 | Операция: create | Путь: /test.txt

[2024-12-31 15:05:22] PID: 5848 | Операция: write | Путь: /test.txt

[2024-12-31 15:05:34] PID: 5848 | Операция: read | Путь: /test.txt

[2024-12-31 15:06:07] PID: 5848 | Операция: write | Путь: /test.txt

[2024-12-31 15:06:10] PID: 5848 | Операция: read | Путь: /test.txt

[2024-12-31 15:56:38] PID: 6888 | Операция: create | Путь: /test.txt

[2024-12-31 15:56:38] PID: 6888 | Операция: write | Путь: /test.txt

[2024-12-31 15:56:50] PID: 6888 | Операция: read | Путь: /test.txt

[2024-12-31 15:57:15] PID: 6888 | Операция: write | Путь: /test.txt

[2024-12-31 15:57:28] PID: 6888 | Операция: read | Путь: /test.txt.bak

[2024-12-31 16:48:35] PID: 7508 | Операция: create | Путь: /test.txt

[2024-12-31 16:48:35] PID: 7508 | Операция: write | Путь: /test.txt

[2024-12-31 16:48:51] PID: 7508 | Операция: read | Путь: /test.txt

[2024-12-31 16:51:28] PID: 7508 | Операция: write | Путь: /test.txt

[2024-12-31 16:51:32] PID: 7508 | Операция: read | Путь: /test.txt.bak

[2024-12-31 16:51:51] PID: 7508 | Операция: read | Путь: /test.txt.bak

**Выводы**

В ходе лабораторной работы было выполнено индивидуальное задание. Изучены популярные файловые системы в ОС Linux (ext4, Btrfs, ReiserFS, NTFS, FAT32).

Также реализована собственная файловая система с помощью инструмента fuse, которая генерирует случайные ошибки.