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 задания на языке С с комментариями.

```c
#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;
```

```c
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.