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 №2
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. Сортировка внешней памяти. Сортировка больших файлов, которые не помещаются в оперативной памяти. Алгоритм: k-way merge sort.

Ход выполнения работы

Задание 1.

Подключаем виртуальный жесткий диск.

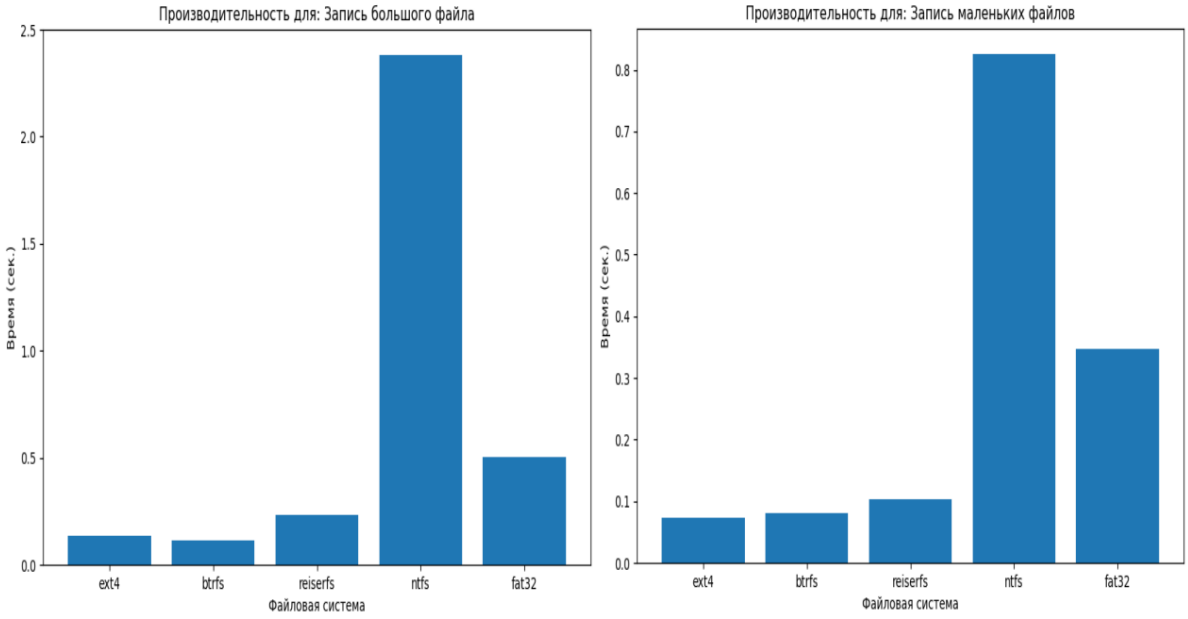
```
$  dd if=/dev/zero of=my_virtual_disk_os_lab3.img bs=1M count=10240
10240+0 records in
10240+0 records out
10737418240 bytes (11 GB, 10 GiB) copied, 18.8853 s, 569 MB/s
$ sudo losetup -fP  my_virtual_disk_os_lab3.img
$ losetup -a
/dev/loop1: []: (/home/andrev133/Desktop/lab3/my_virtual_disk_os_lab3.img)
…
$ sudo parted /dev/loop1
$ (parted) unit GB
$ (parted) mkpart primary ext4 0 2
$ (parted) mkpart primary btrfs 2 4
$ (parted) mkpart primary reiserfs 4 6
$ (parted) mkpart extended 6 10
$ (parted) mkpart logical ntfs 6 8
$ (parted) mkpart logical fat32 8 10
$ (parted) quit
$ lsblk /dev/loop1
NAME      MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop1       7:1    0   10G  0 loop
├─loop1p1 259:6    0  1.9G  0 part
├─loop1p2 259:7    0  1.9G  0 part
├─loop1p3 259:8    0  1.9G  0 part
├─loop1p4 259:9    0    1K  0 part
├─loop1p5 259:10   0  1.9G  0 part
└─loop1p6 259:11   0  1.9G  0 part
// Форматирование и создание ФС
```

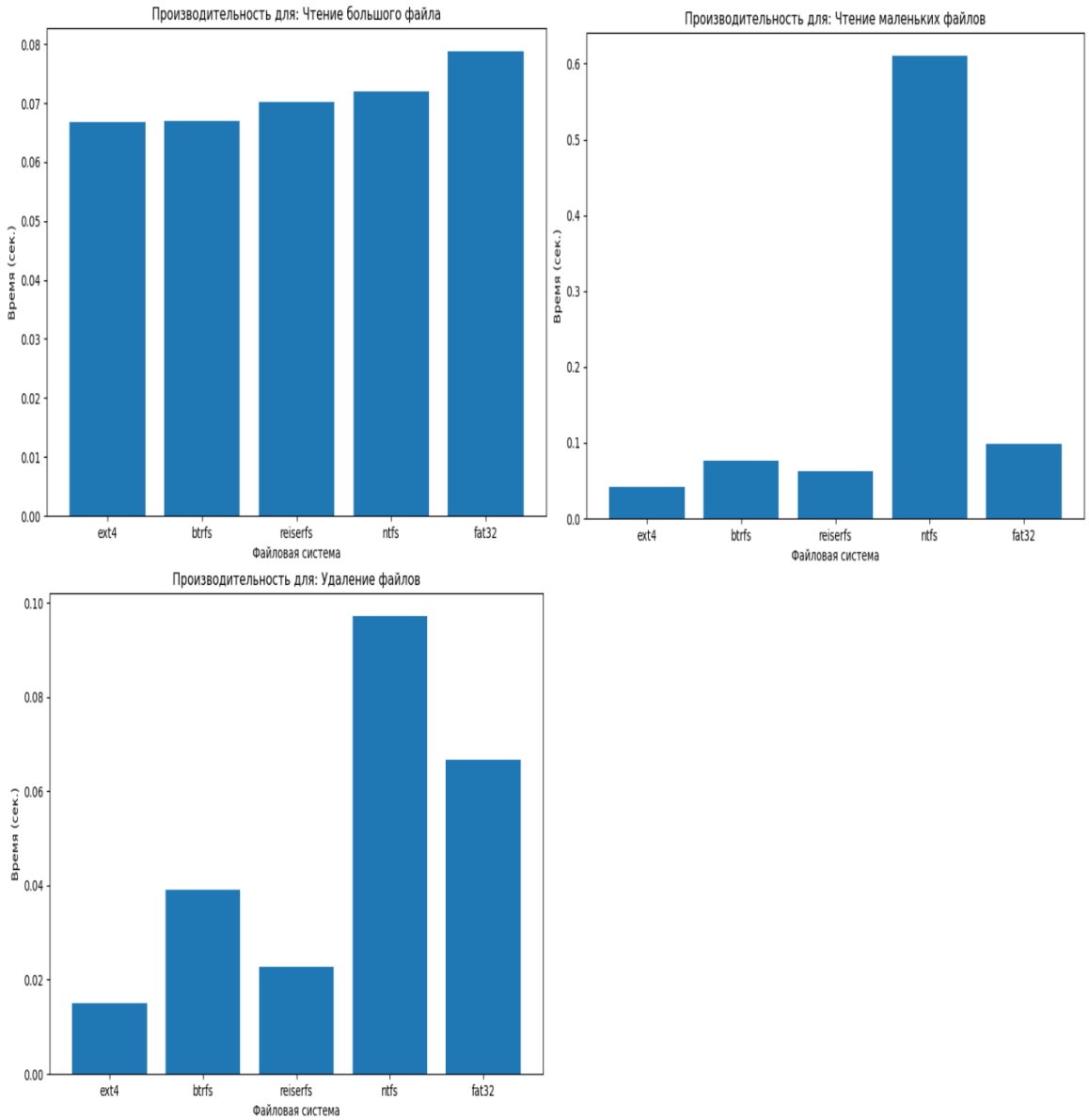
// Монтирование
$ lsblk -f /dev/loop1
NAME FSTYPE FSVER LABEL UUID FSAVAIL FSUSE%
MOUNTPOINTS
loop1

├─loop1p1
│ ext4 1.0 f3f422b2-d474-48a7-b7b5-09408377c986 1.7G 0%
/mnt/ext4
├─loop1p2
│ btrfs 3da0b8ee-d73e-42f4-9b70-e48eb791e8bc 1.7G 0%
/mnt/btrfs
├─loop1p3
│ reiser 3.6 215612f1-47fa-4402-b37a-48e802d62814 1.8G 2%
/mnt/reiserfs
├─loop1p4
│
├─loop1p5
│ ntfs 3CF77D7E695205FD 1.9G 1% /mnt/ntfs
└─loop1p6
vfat FAT32 F0A1-4491 1.9G 0% /mnt/fat32

Наблюдения.
Ntfs продолжительнее всех форматировался.

Замеры с использованием прилагаемого кода:





Текст программы индивидуального задания на языке С с комментариями.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_LINE_LENGTH 1024
#define MEMORY_LIMIT 2500  // Пример ограничения памяти

// Функция для чтения строк из файла и записи их в массив
int read_lines(FILE *file, char **lines, int max_lines) {
    int count = 0;
    char buffer[MAX_LINE_LENGTH];
    while (fgets(buffer, MAX_LINE_LENGTH, file) != NULL && count < max_lines) {
        lines[count] = strdup(buffer);
        count++;
    }
    return count;
}
```

```c
// Функция для сортировки массива строк
void sort_lines(char **lines, int count) {
    for (int i = 0; i < count - 1; i++) {
        for (int j = i + 1; j < count; j++) {
            if (strcmp(lines[i], lines[j]) > 0) {
                char *temp = lines[i];
                lines[i] = lines[j];
                lines[j] = temp;
            }
        }
    }
}

// Функция для записи отсортированных строк в файл
void write_lines(FILE *file, char **lines, int count) {
    for (int i = 0; i < count; i++) {
        fputs(lines[i], file);
        free(lines[i]);
    }
}

// Функция для разделения файла на части и сортировки каждой части
int split_and_sort(const char *input_filename, const char *output_prefix) {
    FILE *input_file = fopen(input_filename, "r");
    if (!input_file) {
        perror("Ошибка открытия входного файла");
        exit(1);
    }

    char **lines = malloc(MEMORY_LIMIT * sizeof(char *));
    int part_number = 0;
    while (1) {
        int count = read_lines(input_file, lines, MEMORY_LIMIT);
        if (count == 0) break;

        sort_lines(lines, count);

        char output_filename[1024];
        sprintf(output_filename, "%s%d.txt", output_prefix, part_number++);
        FILE *output_file = fopen(output_filename, "w");
        if (!output_file) {
            perror("Ошибка создания временного файла");
            exit(1);
        }

        write_lines(output_file, lines, count);
        fclose(output_file);
    }

    free(lines);
```

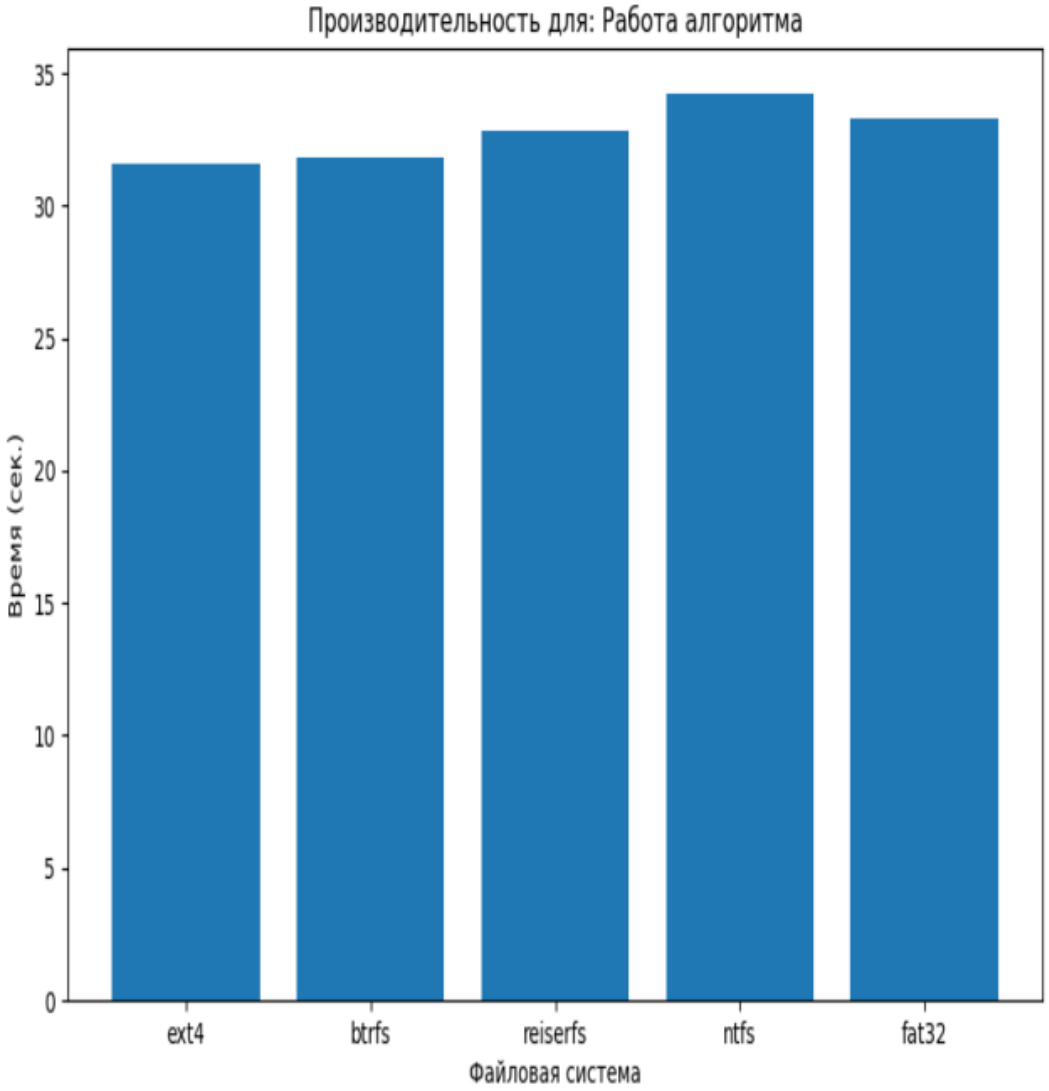
```
    fclose(input_file);

    return part_number;
}

// Структура для хранения строки и номера файла
typedef struct {
    char line[1024];
    int file_index;
} LineEntry;

// Функция для сравнения строк
int compare_lines(const void *a, const void *b) {
    return strcmp(((LineEntry *)a)->line, ((LineEntry *)b)->line);
}

// Функция для слияния отсортированных файлов
void merge_files(const char *output_prefix, int part_count, const char *out-
put_filename) {
    FILE *output_file = fopen(output_filename, "w+");
    if (!output_file) {
        perror("Ошибка создания выходного файла");
        exit(1);
    }

    FILE **input_files = malloc(part_count * sizeof(FILE *));
    LineEntry *heap = malloc(part_count * sizeof(LineEntry));
    for (int i = 0; i < part_count; i++) {
        char filename[1024];
        sprintf(filename, "%s%d.txt", output_prefix, i);
        input_files[i] = fopen(filename, "r");
        if (!input_files[i]) {
            perror("Ошибка открытия временного файла");
            exit(1);
        }

        if (fgets(heap[i].line, MAX_LINE_LENGTH, input_files[i]) != NULL) { //
crit
            heap[i].file_index = i;
        } else {
            heap[i].line[0] = '\0';
            heap[i].file_index = -1;
        }
    }

    qsort(heap, part_count, sizeof(LineEntry), compare_lines);

    while (heap[0].file_index != -1) {
        fputs(heap[0].line, output_file);

        int index = heap[0].file_index;
```

```
        if (fgets(heap[0].line, MAX_LINE_LENGTH, input_files[index]) == NULL) {
            heap[0].line[0] = '\0';
            heap[0].file_index = -1;
        }

        qsort(heap, part_count, sizeof(LineEntry), compare_lines);
    }

    for (int i = 0; i < part_count; i++) {
        fclose(input_files[i]);
    }

    free(input_files);
    free(heap);
    fclose(output_file);
}

int main(int argc, char *argv[]) {
    if (argc != 4) {
        fprintf(stderr, "Использование: %s <входной_файл> <выходной_файл> <папка
для временных файлов>\n", argv[0]);
        return 1;
    }

    const char *input_filename = argv[1];
    const char *output_filename = argv[2];
    char output_prefix[60];
    strcpy(output_prefix, argv[3]);
    strcat(output_prefix, "sorted_part_");

    int part_count = split_and_sort(input_filename, output_prefix);;

    merge_files(output_prefix, part_count, output_filename);

    for (int i = 0; i < part_count; i++)
    {
        char filename[1024];
        sprintf(filename, "%s%d.txt", output_prefix, i);
        remove(filename);
    }

    return 0;
}
```

Протоколы, логи, скриншоты, графики.

Основная идея работы алгоритма в том, что выделяется максимально возможное кол-во строк, сортируется и сохраняется в несколько файлов. Затем открываем поток чтения для каждого временного файла и сверху берём подходящую строку, повторяем так пока все строки не закончатся.

Для теста был взят файл на 500 тысяч строк по 10 символов, и ограничение памяти на 1000 строк. В процессе работы алгоритма появляется множество файлов среднего размера (500шт. по 12.3 kB). График производительности:



Так как большую часть времени занимает слияние файлов разница в скорости кажется незначительной, но точно заметен «проигрыш» в скорости файловой системы ntfs.

Выводы

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