МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Системное программирование»

Лабораторная работа №10

**«Диски и файловые системы»**

по дисциплине

Операционные системы

Выполнил: студент гр. БЭИ2202

Кулешов А. С.

Вариант 16

Проверил: Королькова Т.В.

Москва, 2025 г

Задание 1. Разметка диска, создание файловой системы

Подключите к виртуальной машине 2 новых диска, первый диск

произвольного размера, второй диск размером 3 ГБ.

1. На первом диске создайте таблицу разделов MBR. Создайте 4 раздела:

первый раздел занимает 50% диска, остальные - произвольного размера.

Один из разделов должен быть логическим.

2. На втором диске создайте таблицу разделов GPT. Создайте один раздел

размером 2 ГБ и разметьте его как xfs. Смонтируйте раздел по пути /mnt.

Создайте файл на смонтированной файловой системе. Сделайте снимок

экрана вывода команды df -hT.

Увеличьте раздел до 3 ГБ. Расширьте файловую систему на новое свободное

пространство. Убедитесь, что созданный файл остался внутри раздела и

файловой системы. Сделайте снимок экрана вывода команды df -hT.

В отчете приведите снимки экрана, демонстрирующие разметку диска

(например, командами lsblk -a; fdisk -l) и снимки экрана после выполнения

команд df.

Задание 2. Создание RAID-массива

Создайте программный RAID 1 при помощи утилиты mdadm. Объем

RAID выберите самостоятельно. В отчете приведите снимок экрана вывода

команды mdad m -D /dev/md0, где md0 - название созданного RAID

массива.

Задание 3. Получение информации о файловой системе

Определите тип основной файловой системы, ее размер, общее

количество и количество занятых inode.

В отчете приведите использованные команды и результаты их

выполнения в виде снимков экрана.

**ВЫПОЛНЕНИЕ**



Рисунок 1 – Задание 1. Добавление двух дисков

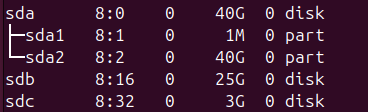


Рисунок 2 – Задание 1. Диски в системе. lsblk

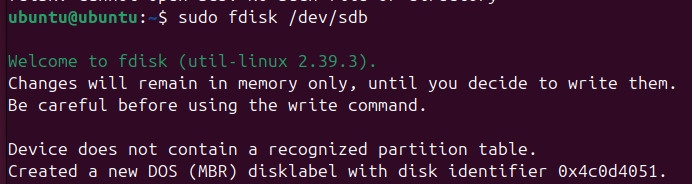


Рисунок 3 – Задание 1. Создание MBR

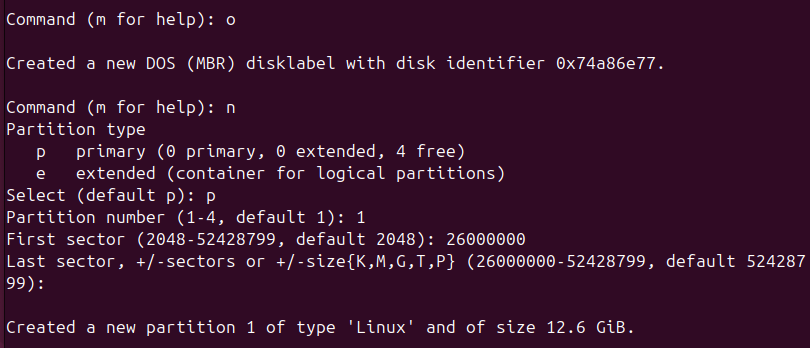


Рисунок 4 – Задание 1. Создание первого раздела на половину памяти

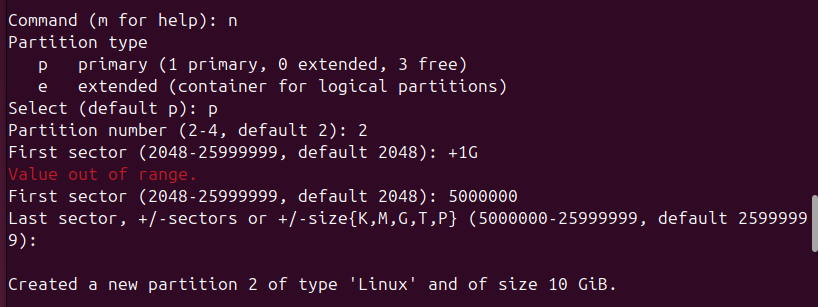


Рисунок 4 – Задание 1. Создание второго раздела

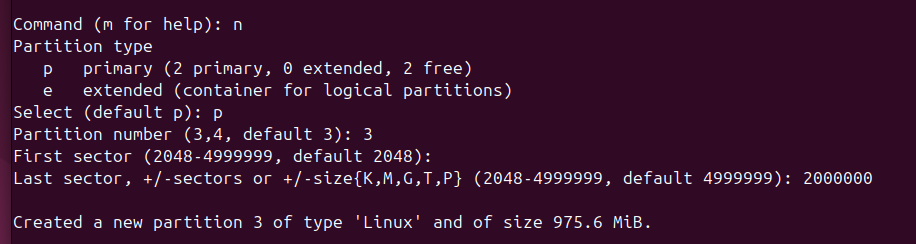


Рисунок 5 – Задание 1. Создание третьего раздела

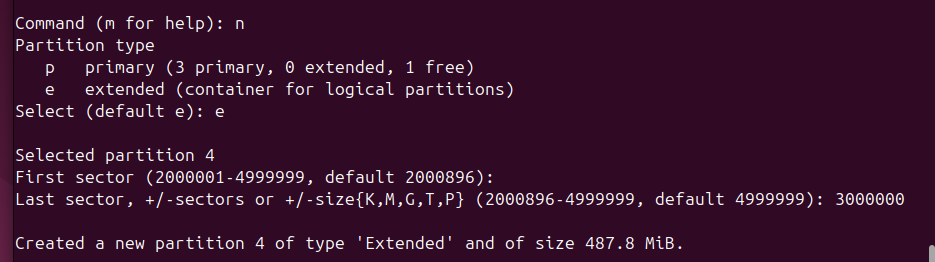


Рисунок 6 – Задание 1. Создание расширенного раздела (для логического)

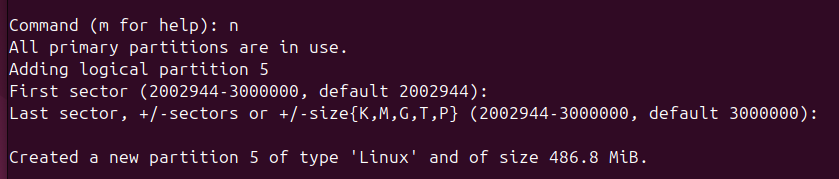


Рисунок 7 – Задание 1. Создание логического раздела

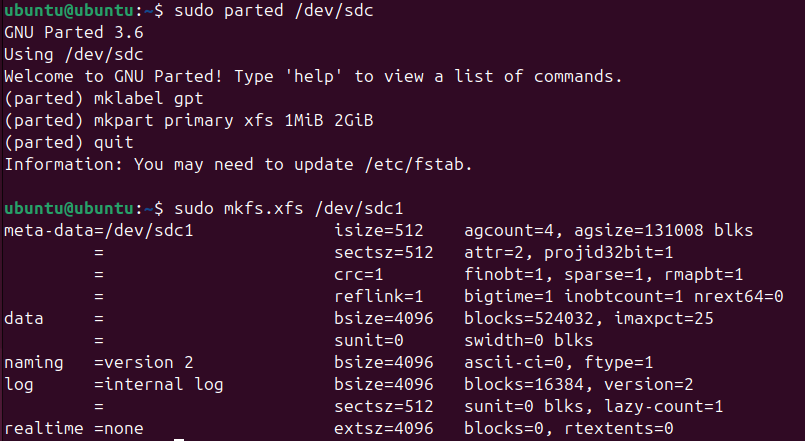


Рисунок 8 – Задание 1. Создание gpt раздела

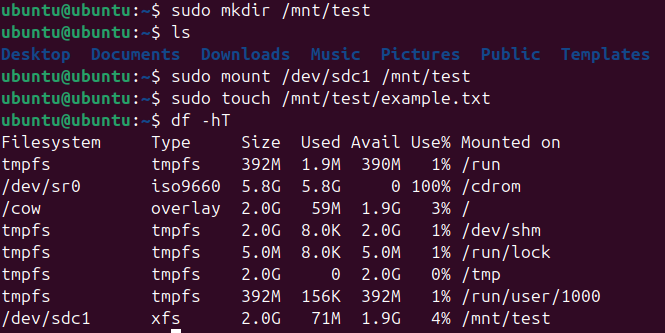


Рисунок 9 – Задание 1. Монтирование

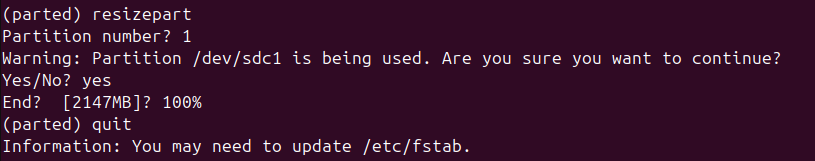


Рисунок 10 – Задание 1. Расширение диска

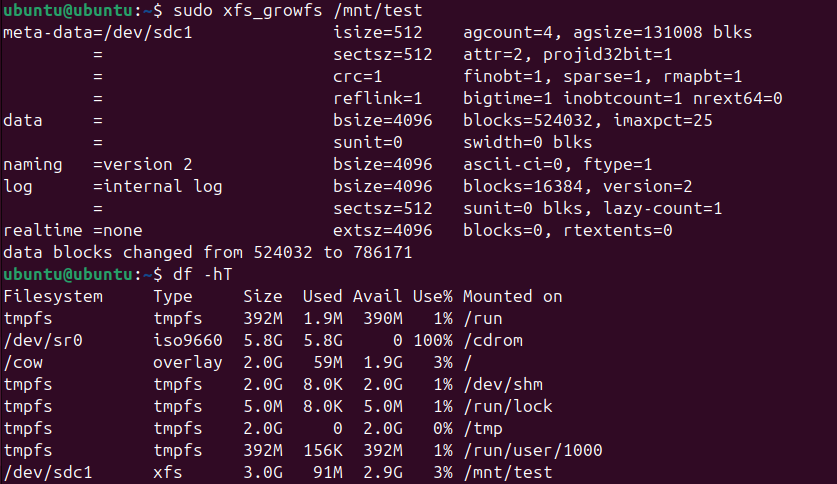


Рисунок 10 – Задание 1. Диск расширен

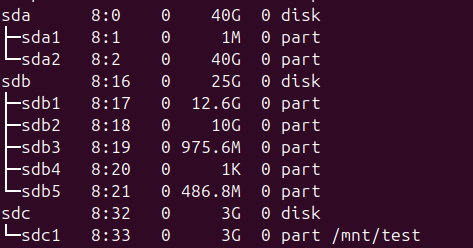


Рисунок 11 – Задание 1. Разметка диска

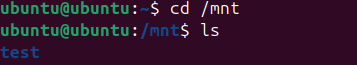


Рисунок 12 – Задание 1. Файл всё ещё тут

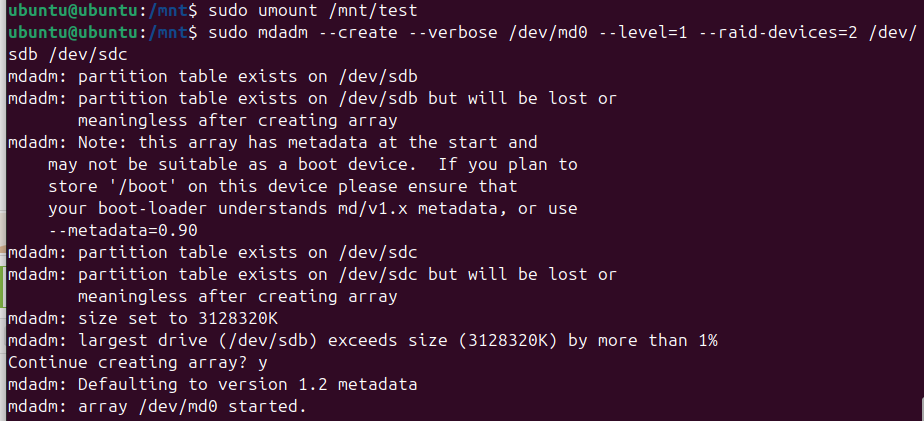


Рисунок 12 – Задание 2. Размаунт и создание массива

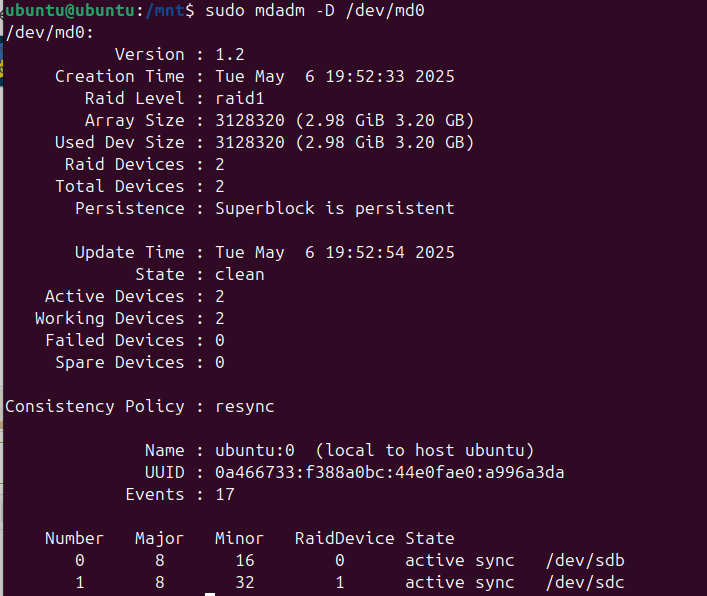


Рисунок 13 – Задание 2. Массив создан

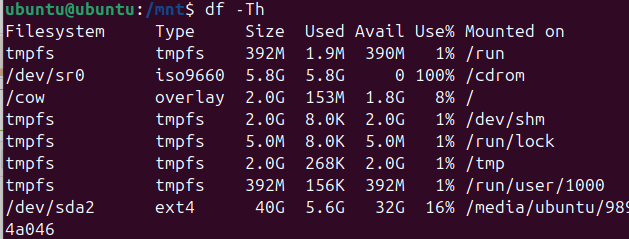


Рисунок 14 – Задание 3. Данные об основной файловой системы

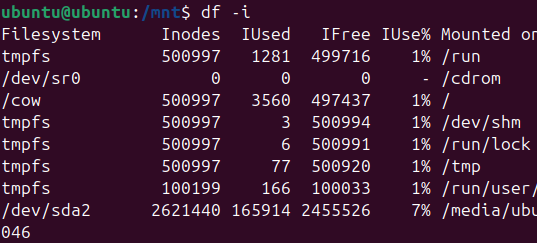


Рисунок 15 – Задание 3. Данные об inode



Рисунок 4 – Задание 2. Запуск без аргумента



Рисунок 5 – Задание 2. Получение pid



Рисунок 6 – Задание 2. Запуск. Успешный

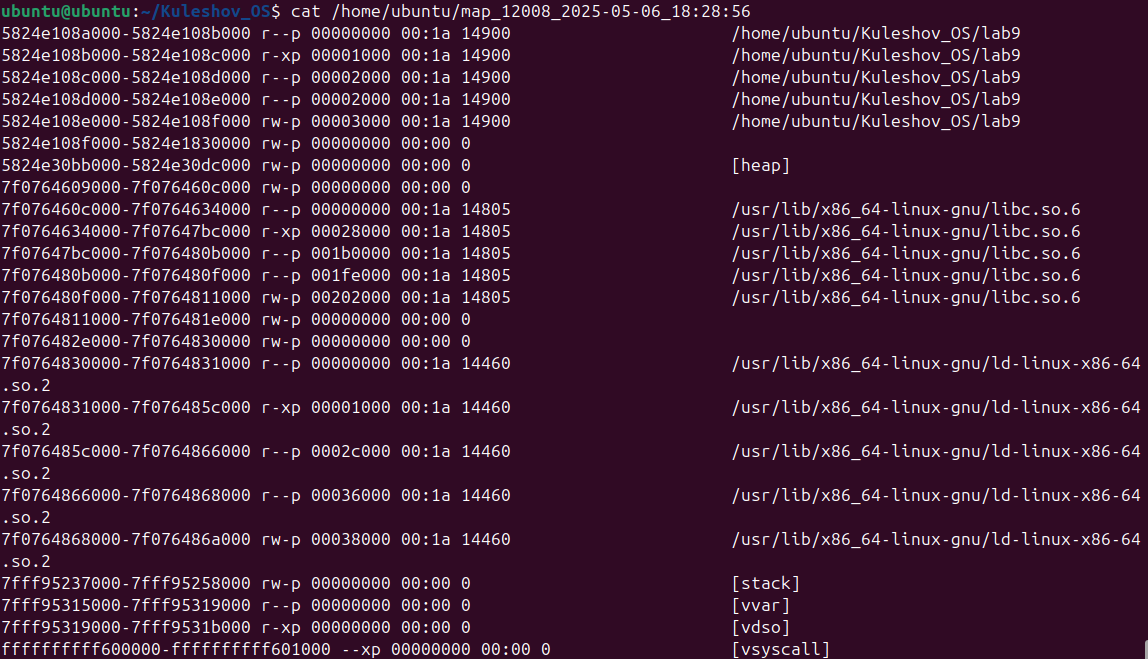


Рисунок 7 – Задание 2. Полученная карта

Вопрос 1. Какую информацию содержит карта памяти?

- Диапазоны адресов виртуальной памяти. Права доступа. Индекс файла в файловой системе.

Вопрос 2. Что такое стек и куча

- Стек хранит в себе как правило локальные данные при помощи механизма добавления и удаления по принципу LiFo (Last in, First out). Куча же предполагает ручное выделение и удаление памяти. Растёт как правило в положительную сторону адресов

Вопрос 3. Как происходит статическое и динамическое выделение памяти?

- Статическое выделение происходит на стадии компиляции. Данные существуют в весь период жизни программы. Динамические выделются во время работы программы

Вопрос 4. Какие плюсы и минусы статического и динамического выделения памяти?

- Статическая память быстрее, а ещё с ней меньше проблем с утечками. Однако выделить нефиксированное число памяти не выйдет. Также размер меньше чем у динамической

Код для задания 1.

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <signal.h>

#include <string.h>

#include <time.h>

#define MAX\_PTRS 1000000

**volatile** **sig\_atomic\_t** stop = **0**;

**void**\* allocated\_ptrs[MAX\_PTRS];

**size\_t** allocated\_count = **0**;

**void** **handle\_sigint**(**int** sig) {

stop = **1**;

printf("**\n**[INFO] Прерывание получено. Освобождение всей памяти...**\n**");

**for** (**size\_t** i = **0**; i < allocated\_count; ++i) {

free(allocated\_ptrs[i]);

}

printf("[INFO] Освобождено %zu блоков памяти. Выход.**\n**", allocated\_count);

exit(**0**);

}

**void** **nanosleep\_random**() {

**struct** timespec ts;

ts.tv\_sec = **0**;

ts.tv\_nsec = **100** + rand() % **901**; // от 100 до 1000 нс

nanosleep(&ts, NULL);

}

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

**if** (argc != **2**) {

fprintf(stderr, "Использование: %s режим(1 или 2)**\n**", argv[**0**]);

**return** **1**;

}

**int** mode = atoi(argv[**1**]);

**if** (mode != **1** && mode != **2**) {

fprintf(stderr, "Неверный режим. Используйте 1 или 2.**\n**");

**return** **1**;

}

signal(SIGINT, handle\_sigint);

srand(time(NULL));

**size\_t** alloc\_rate = (mode == **1**) ? **1** : **2**;

**size\_t** free\_rate = **1**;

**while** (!stop) {

// Выделяем память

**for** (**size\_t** i = **0**; i < alloc\_rate; ++i) {

**if** (allocated\_count < MAX\_PTRS) {

**void**\* ptr = malloc(**64**); // Выделим 64 байта

**if** (ptr != NULL) {

allocated\_ptrs[allocated\_count++] = ptr;

}

}

nanosleep\_random();

}

// Освобождаем память

**for** (**size\_t** i = **0**; i < free\_rate; ++i) {

**if** (allocated\_count > **0**) {

free(allocated\_ptrs[--allocated\_count]);

}

nanosleep\_random();

}

}

**return** **0**;

}

Код для задания 2.

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <time.h>

#include <string.h>

#include <sys/stat.h>

#include <dirent.h>

**void** **create\_memory\_map**(**pid\_t** pid, **const** **char** \*output\_dir) {

**time\_t** now = time(NULL);

**struct** tm \*tm\_info = localtime(&now);

**char** timestamp[**20**];

strftime(timestamp, **sizeof**(timestamp), "%Y-%m-%d\_%H:%M:%S", tm\_info);

**char** filename[**256**];

snprintf(filename, **sizeof**(filename), "%s/map\_%d\_%s", output\_dir, pid, timestamp);

**char** proc\_path[**64**];

snprintf(proc\_path, **sizeof**(proc\_path), "/proc/%d/maps", pid);

**FILE** \*src = fopen(proc\_path, "r");

**if** (!src) {

perror("Не смогли открыть /proc/[pid]/maps");

exit(**1**);

}

**FILE** \*dst = fopen(filename, "w");

**if** (!dst) {

perror("Не смогли создать выходной файл");

fclose(src);

exit(**1**);

}

**char** buffer[**4096**];

**size\_t** bytes;

**while** ((bytes = fread(buffer, **1**, **sizeof**(buffer), src)) > **0**) {

fwrite(buffer, **1**, bytes, dst);

}

fclose(src);

fclose(dst);

printf("Карта памяти создана: %s**\n**", filename);

}

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

**if** (argc != **3**) {

printf("Использование: %s <pid> <output\_directory>**\n**", argv[**0**]);

**return** **1**;

}

**pid\_t** pid = atoi(argv[**1**]);

**const** **char** \*output\_dir = argv[**2**];

**DIR** \*dir = opendir(output\_dir);

**if** (!dir) {

perror("Выходная директория не существует");

**return** **1**;

}

closedir(dir);

create\_memory\_map(pid, output\_dir);

**return** **0**;

}