

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное  
образовательное учреждение высшего образования  
«Комсомольский-на-Амуре государственный университет»

Факультет компьютерных технологий

Кафедра «Проектирование, управление и разработка информационных систем»

## КОНТРОЛЬНАЯ РАБОТА

по дисциплине «Альтернативные операционные системы»

Manjaro Linux

Студент группы ОИБ-1

Преподаватель

Д.В. Шутрин

Д.О. Журавлев

## Содержание

1 История дистрибутива Linux .....	3
1.1 История проекта Manjaro.....	3
1.2 Особенности Linux Manjaro .....	4
1.3 Достоинства дистрибутива.....	5
1.4 Недостатки Linux Manjaro .....	6
1.5 Выводы обзора дистрибутива Linux Manjaro .....	7
2 Установка Manjaro Linux на виртуальную машину.....	9
3 Командная строка Linux .....	19
4 Командные интерпретаторы .....	23
5 Файлы устройств и монтирование.....	27
6 Установка программ в Linux .....	42
6.1 Установка программ с помощью графического интерфейса.....	42
6.2 Установка программ с использованием командной строки .....	44
6.3 Установка программ из исходных кодов .....	45
7 Программирование в Linux .....	48
7.1 Создание консольного приложения .....	48
7.2 Ввод-вывод с помощью консоли .....	49
7.3 Передача параметров программе.....	52
Список использованных источников .....	55

# 1 История дистрибутива Linux

## 1.1 История проекта Manjaro

Manjaro — это дружелюбный пользователю дистрибутив Linux, основанный на независимо разработанной операционной системе Arch. В сообществе Linux, Arch известен как исключительно быстрый, мощный и легкий дистрибутив, предоставляющий доступ к самому последнему передовому и "революционному" программному обеспечению. Однако Arch также ориентирован на более опытных или технически подкованных пользователей. Поэтому считается, что он не под силу тем, кто не обладает достаточным техническим опытом (или упорством), необходимым для его использования.

Начало проекту положил австриец Роланд Зингер (Roland Singer), в середине 2011 года сообщивший на форуме Arch Linux о том, что собрал Live CD с уже известным нам названием. В качестве рабочего окружения по умолчанию он выбрал XFCE. Он и остался флагманским рабочим столом системы поныне. Вскоре к проекту примкнули единомышленники: француз Гийом Бенуа (Guillaume Benoit) и выходец из проекта Chakra Linux Филип Мюллер (Philip Müller). А спустя год после первого упоминания в Сети, 20 августа 2012-го, мир увидел первую стабильную версию дистрибутива — Manjaro 0.8. Разработчики неустанно повторяют, что Manjaro, несмотря на свое близкое родство с Arch Linux, по сути, полностью независимая система с собственными репозиториями программного обеспечения, инструментами настройки, командой разработчиков и своим видением направления развития системы. С самых первых дней проект ставил одной из своих главных целей сделать такую систему, которая обеспечит доступ широкому кругу пользователей к мощи и простоте Arch Linux, обойдя при этом все острые углы родительского дистрибутива, препятствующие его широкому распространению.

Разработанная в Австрии, Франции и Германии, Manjaro предоставляет все преимущества операционной системы Arch в сочетании с акцентом на удобство для пользователя и доступность. Manjaro следует Arch Linux и официально предлагает только 64-битную версию. Manjaro подходит как для новичков, так и для опытных пользователей Linux. Для новичков предоставляется удобная программа установки, а сама система разработана для полноценной работы "прямо из коробки" с такими функциями, как:

- Предустановленные среды рабочего стола
- предустановленные Графические менеджеры программного обеспечения для простой установки программного обеспечения и обновления системы
- Предустановленные кодеки для воспроизведения мультимедийных файлов

Для более опытных и авантюрных пользователей Manjaro также предлагает конфигурируемость и универсальность, которые можно формировать и лепить во всех отношениях в соответствии с личным вкусом и предпочтениями. Manjaro Architect - это и CLI-установщик, доступный бок о бок с графическим установщиком, и ISO на базе CLI, дающий возможность установить Manjaro на любой вкус, а также предлагает безвкусную установку DE, выбор файловой системы и загрузчика для тех, кто хочет полной свободы в формировании своей системы. Начиная с командной строки, вы можете свободно выбирать свои собственные приветствия, рабочие столы, драйверы оборудования, программные приложения и так далее!

## **1.2 Особенности Linux Manjaro**

Manjaro имеет много общих черт с Arch, включая:

- Скорость, мощность и эффективность

- Доступ к самому последнему передовому программному обеспечению.

- Модель разработки 'rolling release', которая обеспечивает самую современную систему без необходимости установки новых версий, и

- Доступ к Arch User Repository (AUR).

Однако Manjaro может похвастаться несколькими собственными дополнительными возможностями, включая:

- Упрощенный, удобный для пользователя процесс установки

- Автоматическое определение аппаратного обеспечения вашего компьютера (например, видеокарты)

- Автоматическая установка необходимого программного обеспечения (например, графических драйверов) для вашей системы

- Собственные специальные репозитории программного обеспечения для обеспечения поставки полностью протестированных и стабильных пакетов программного обеспечения, и

- Поддержка простой установки и использования нескольких ядер.

### **1.3 Достоинства дистрибутива**

1. Manjaro Linux - это роллинг-релиз дистрибутив, что означает, что вы всегда получаете самые свежие версии программ и компонентов системы, не нуждаясь в переустановке или обновлении<sup>12</sup>.

2. Manjaro Linux имеет отличную поддержку аппаратного обеспечения. Он автоматически определяет и устанавливает нужные драйверы для вашего компьютера, включая видеокарты, звуковые карты, сетевые адаптеры и т.д.<sup>1</sup>.

3. Manjaro Linux предоставляет вам доступ к AUR (Arch User Repository) - огромному репозиторию, содержащему тысячи программ и приложений, созданных и поддерживаемых сообществом Arch Linux<sup>12</sup>. Вы можете легко устанавливать и обновлять эти программы с помощью графического менеджера пакетов Pacman или командной строки Pacman<sup>2</sup>.

4. Manjaro Linux дает вам выбор рабочего окружения. Вы можете выбрать из нескольких официальных и общественных изданий Manjaro, которые предлагают различные рабочие столы, такие как XFCE, KDE, GNOME, Cinnamon, Budgie, MATE и другие<sup>13</sup>. Вы также можете настраивать внешний вид и поведение вашего рабочего стола по своему вкусу.

5. Manjaro Linux имеет дружелюбное и активное сообщество, которое готово помочь вам в решении любых проблем или вопросов, связанных с использованием дистрибутива<sup>23</sup>. Вы можете общаться с другими пользователями и разработчиками на форуме Manjaro, в чате Telegram, в группах социальных сетей или в канале IRC.

## **1.4 Недостатки Linux Manjaro**

1. Manjaro Linux может быть менее стабилен, чем другие дистрибутивы Linux, из-за того, что он постоянно обновляется и включает в себя последние версии программ и компонентов системы<sup>23</sup>. Это может привести к появлению ошибок, сбоев, несовместимостей или конфликтов, которые требуют внимания и исправления со стороны пользователя.

2. Manjaro Linux может быть несовместим с некоторыми программами или оборудованием, которые разработаны или оптимизированы для других дистрибутивов Linux, особенно для Ubuntu или Debian<sup>4</sup>. Например, некоторые игры или приложения могут не работать или работать некорректно на Manjaro, если они требуют определенных библиотек, зависимостей или драйверов, которые отсутствуют или отличаются в Manjaro.

3. Manjaro Linux может быть сложнее для новичков, чем другие дистрибутивы Linux, такие как Ubuntu или Linux Manjaro<sup>3</sup>. Хотя Manjaro упрощает установку и настройку Arch Linux, он все же требует от пользователя большего знания и понимания работы Linux, особенно при работе с AUR, Pacman или командной строкой. Также, пользователь должен

быть готов к тому, что ему придется часто обновлять систему и решать возникающие проблемы.

4. Manjaro Linux может быть небезопасен, если вы не доверяете источникам, из которых вы устанавливаете программы<sup>2</sup>. AUR содержит программы, которые не проверяются и не поддерживаются официальными разработчиками Manjaro или Arch Linux, а создаются и обновляются сообществом. Это означает, что вы не можете быть уверены в качестве, функциональности или безопасности этих программ. Вы должны всегда проверять, что вы устанавливаете, и избегать подозрительных или устаревших пакетов.

5. Manjaro Linux может быть неподходящим для критически важных задач, таких как работа с конфиденциальными данными, управление серверами, разработка программного обеспечения или другие профессиональные или академические цели<sup>3</sup>. Из-за того, что Manjaro постоянно обновляется и включает в себя последние версии программ и компонентов системы, он может быть нестабилен, несовместим или небезопасен для таких задач. Для этих целей лучше использовать более стабильные и проверенные дистрибутивы Linux, такие как Debian, Ubuntu, Fedora или CentOS.

## **1.5 Выводы обзора дистрибутива Linux Manjaro**

Manjaro Linux — это дистрибутив Linux, который имеет много преимуществ для тех, кто хочет иметь самые свежие версии программ и компонентов системы, отличную поддержку аппаратного обеспечения, доступ к огромному репозиторию AUR, выбор рабочего окружения и дружелюбное сообщество. Также дистрибутив Linux, который имеет некоторые недостатки для тех, кто ценит стабильность, совместимость, безопасность и подходит для критически важных задач. Он может быть менее стабилен, несовместим с некоторыми программами или

оборудованием, сложнее для новичков, не безопасен, если вы не доверяете источникам, из которых вы устанавливаете программы, и неподходящий для профессиональных или академических целей.

Manjaro Linux — это дистрибутив Linux, который подходит для тех, кто любит экспериментировать, изучать и настраивать свою систему, но не для тех, кто хочет иметь простую, надежную и безопасную систему. Вы должны быть готовы к тому, что вам придется часто обновлять систему и решать возникающие проблемы, а также проверять, что вы устанавливаете, и избегать подозрительных или устаревших пакетов.



## 2 Установка Manjaro Linux на виртуальную машину

Для установки Manjaro Linux на виртуальную машину в VM Ware Workstation необходимо сначала добавить новую виртуальную машину.

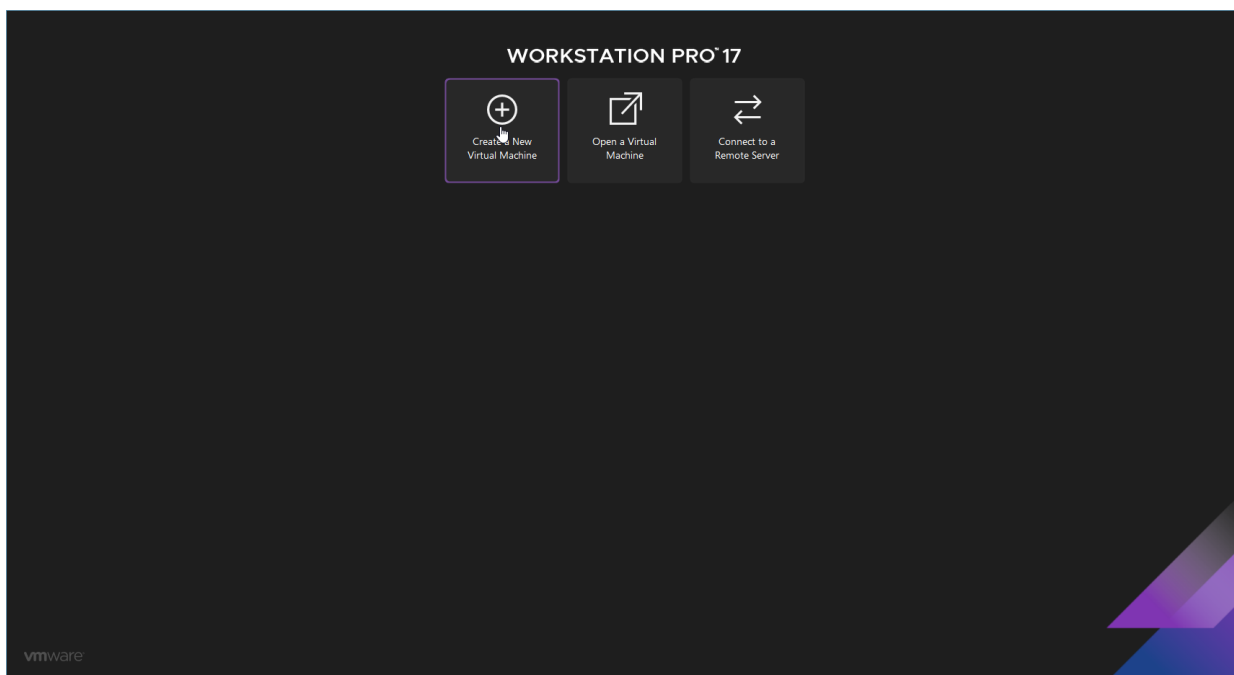


Рисунок 2.1 – Добавление новой виртуальной машины

После чего нужно выбрать ISO файл для установки и задать параметры для виртуальной машины, а именно: размер на диске, количество процессоров и количество оперативной памяти.

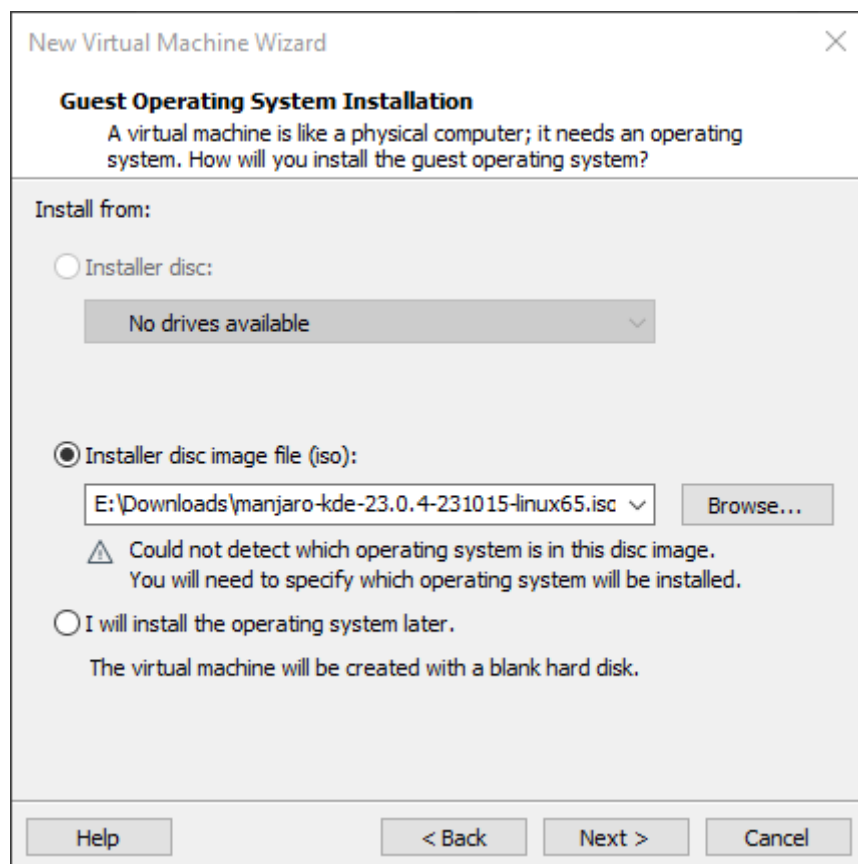


Рисунок 2.2 – Выбираем ISO файл Manjaro

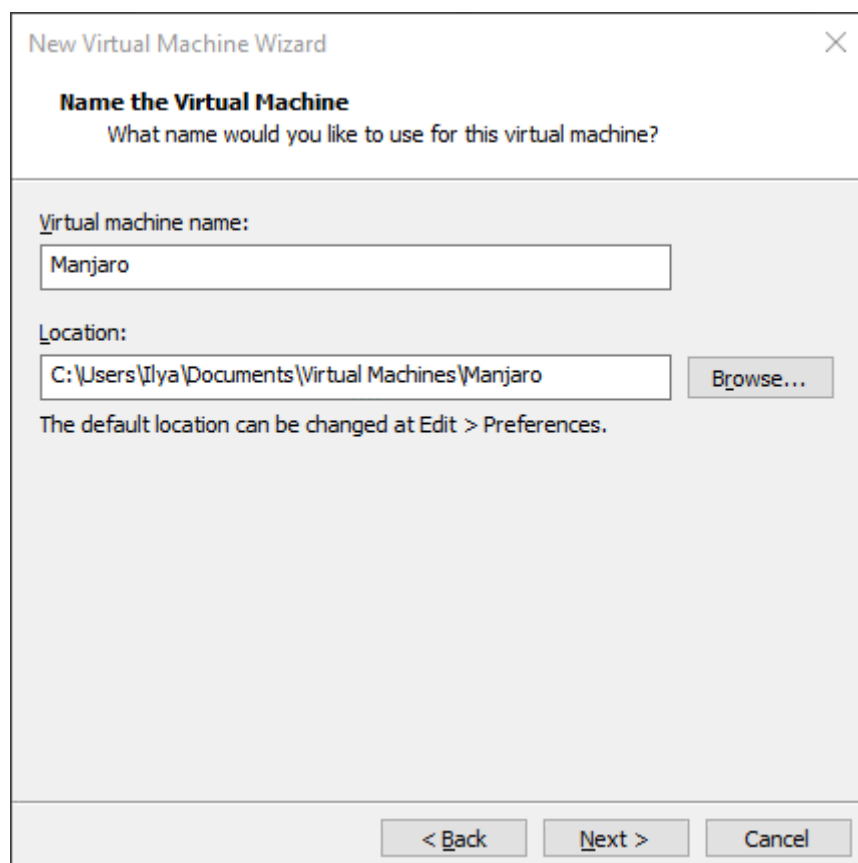


Рисунок 2.3 – Указываем путь до виртуальной машины

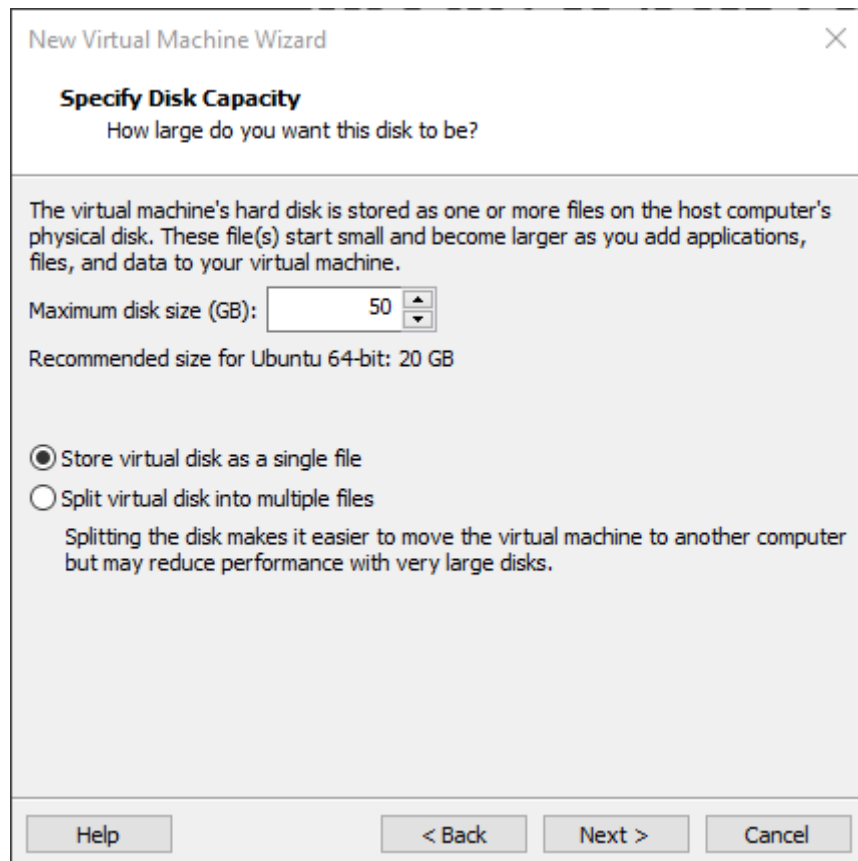


Рисунок 2.4 – Выделяем место на диске

Далее запускаем виртуальную машину и производим ее установку.

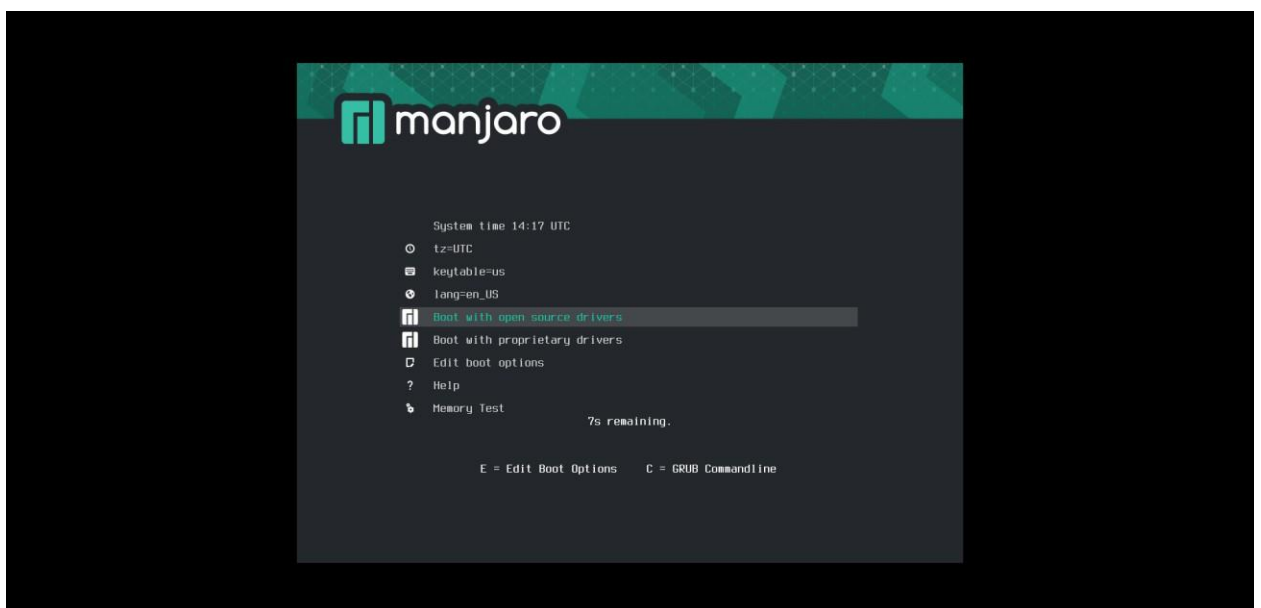


Рисунок 2.5 – Первый запуск системы

После завершения первичной установки запускается live версия Manjaro Linux.

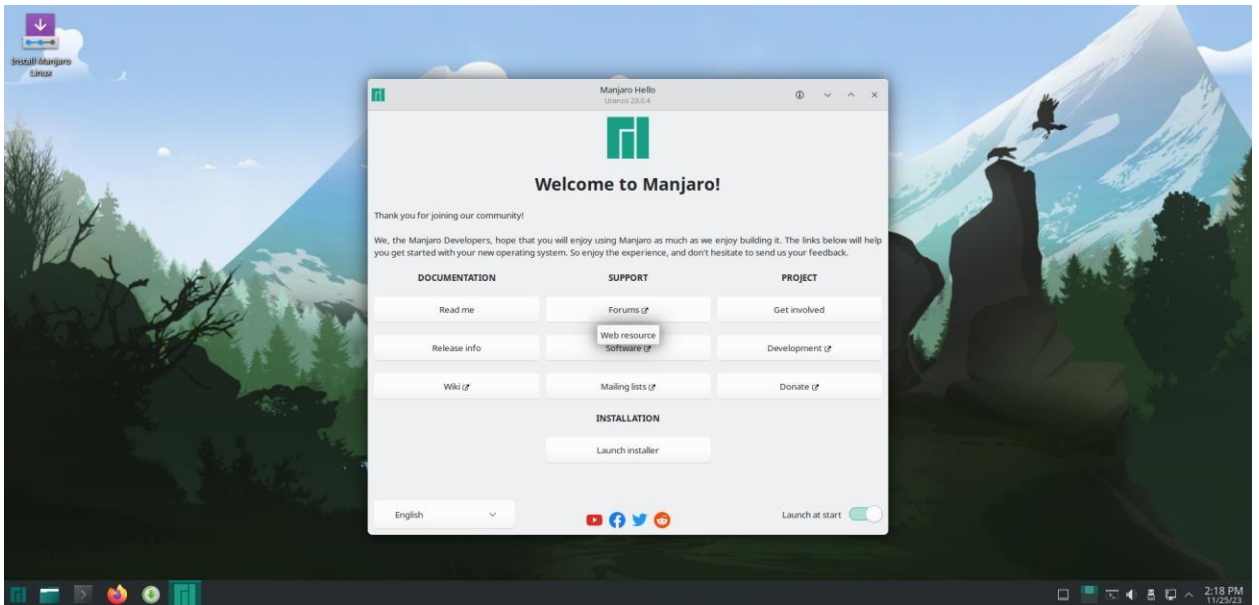


Рисунок 2.6 – Live версия Manjaro Linux

Далее мы нажимаем «Launch Installer» для установки полной версии системы. Первым делом выбираем язык системы.

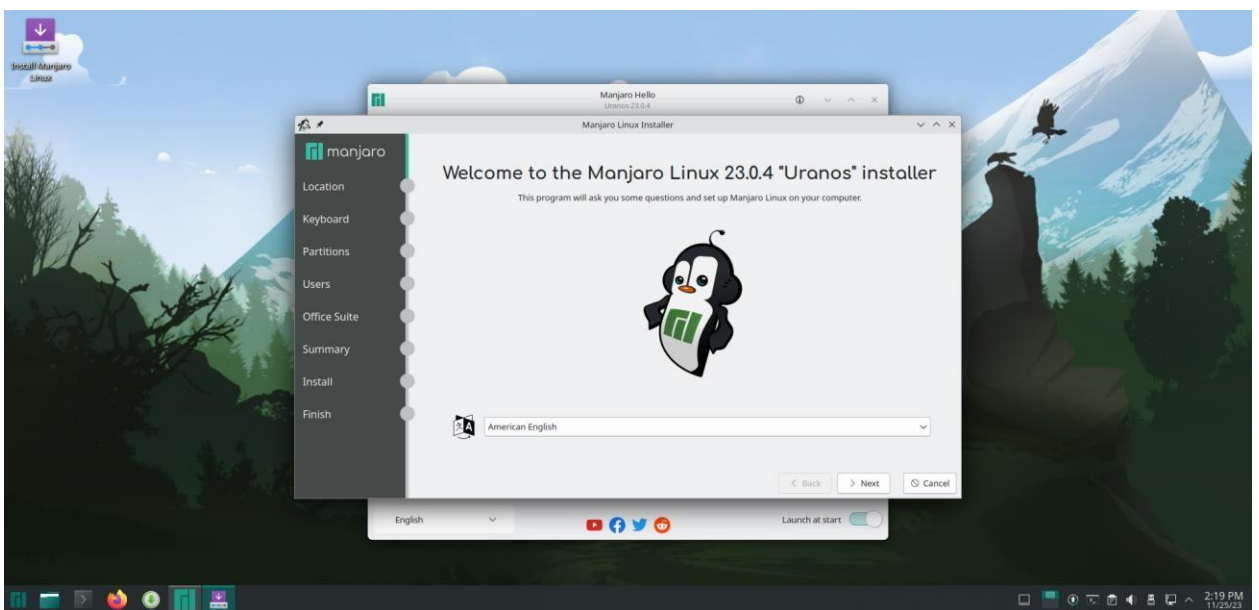


Рисунок 2.7 – Выбор языка

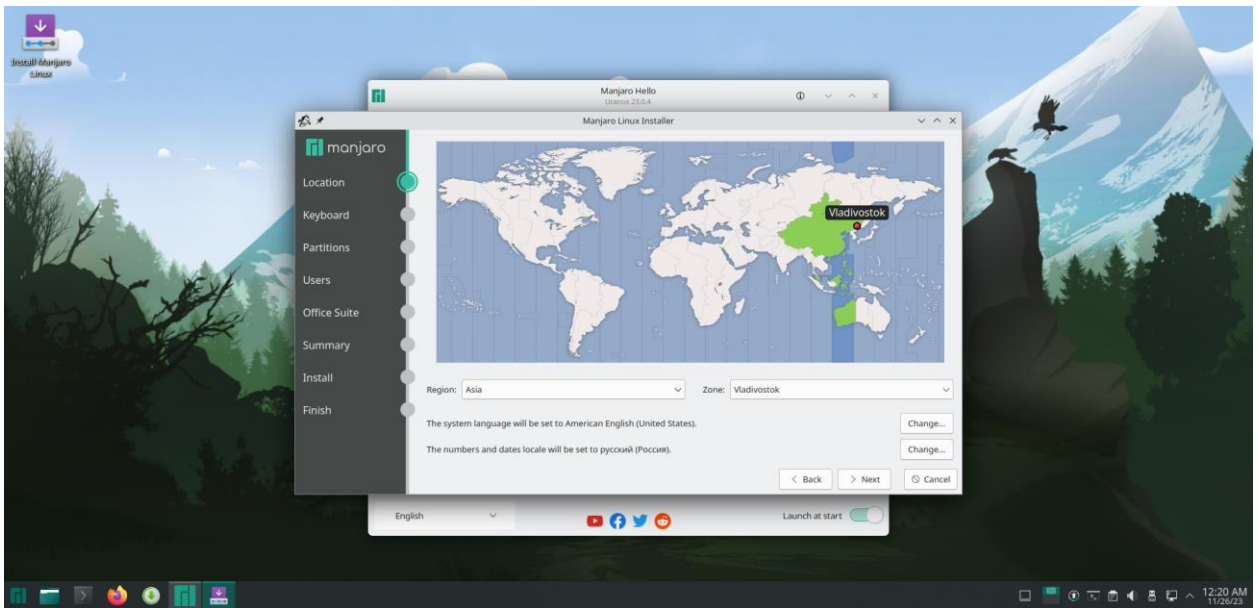


Рисунок 2.8 – Выбор часового пояса

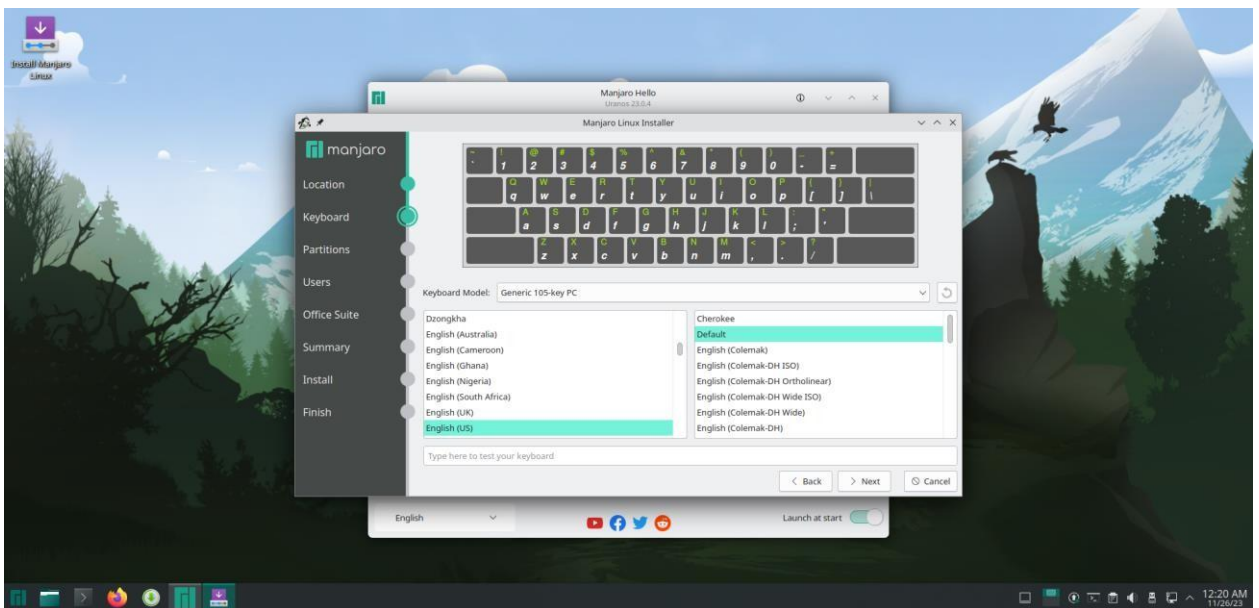


Рисунок 2.9 – Выбор раскладки

После чего настраиваем дисковое пространство. Выберем стереть все данные и установим линукс.

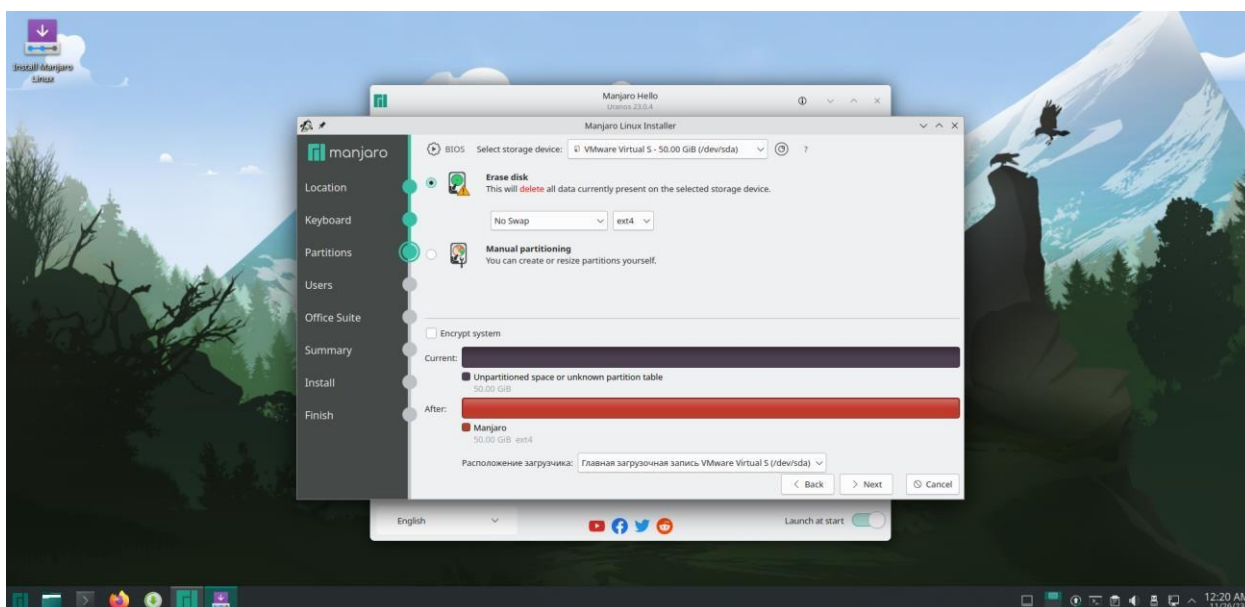


Рисунок 2.10 – Настройка дискового пространства

Далее настраиваем данные для входа. После этого мы можем начать установку.

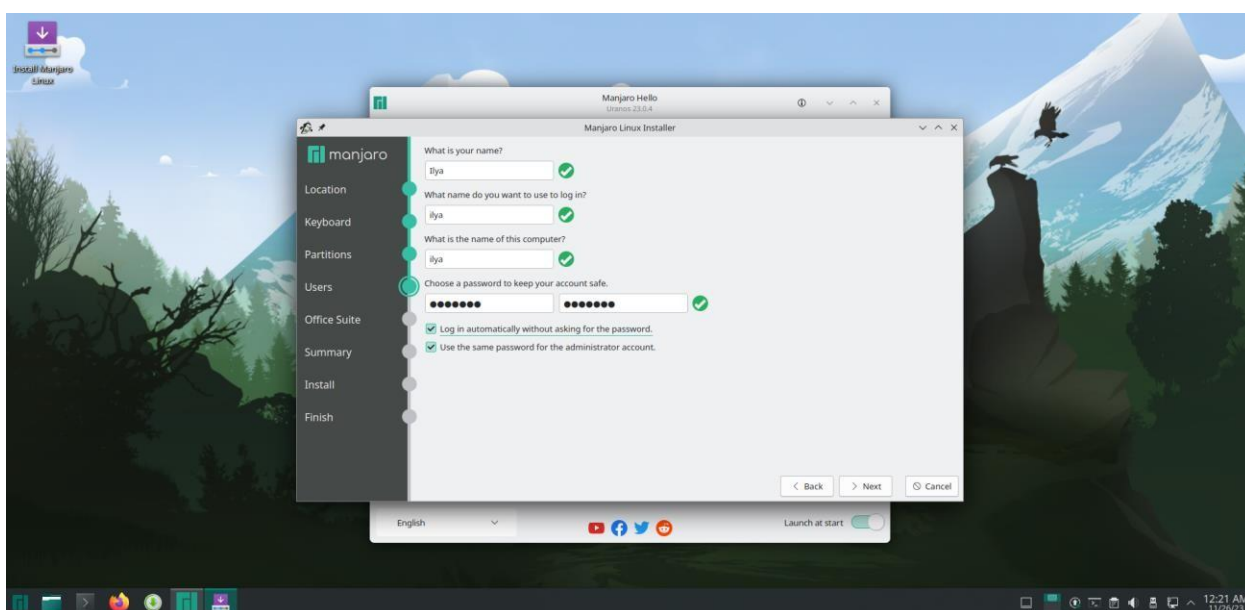


Рисунок 2.11 – Настройка учетной записи





Рисунок 2.12 – Выбор приложения для редактирования

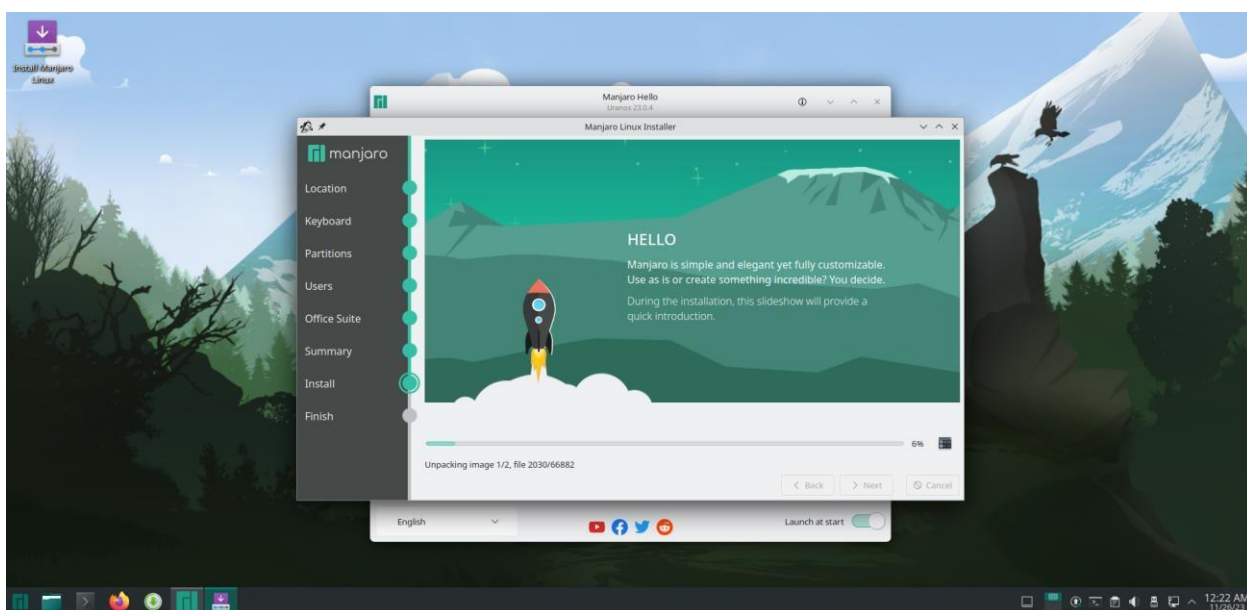


Рисунок 2.13 – Начало установки

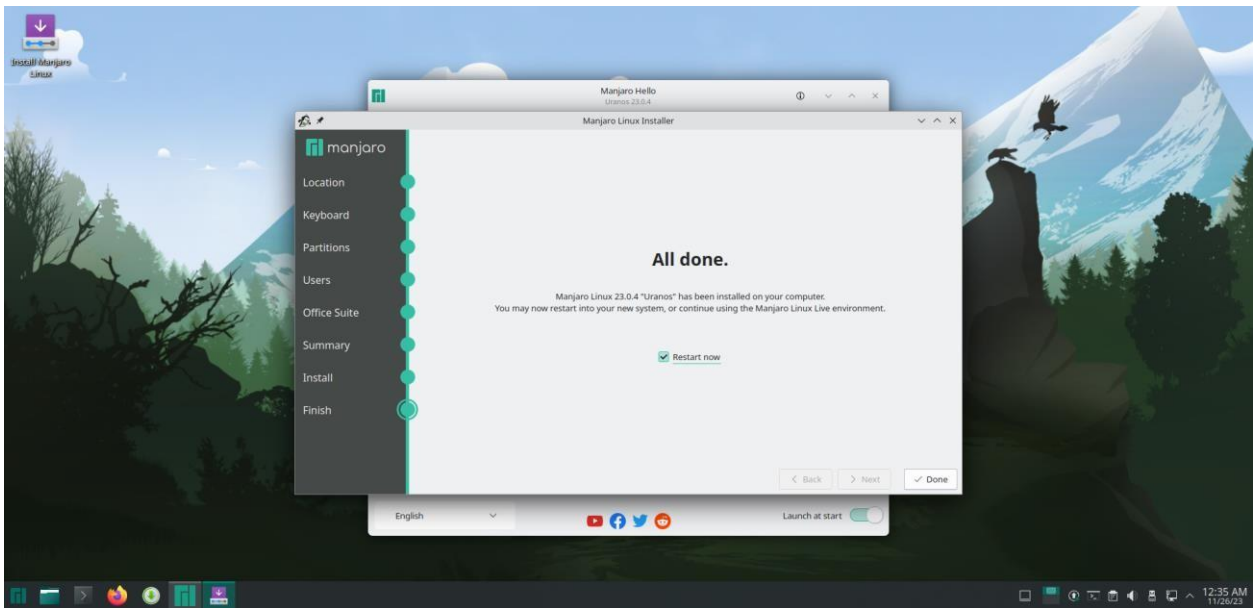


Рисунок 2.14 – Завершение установки

После перезагрузки системы мы видим приветственное окно, сообщаемое нам об успешной установке системы.

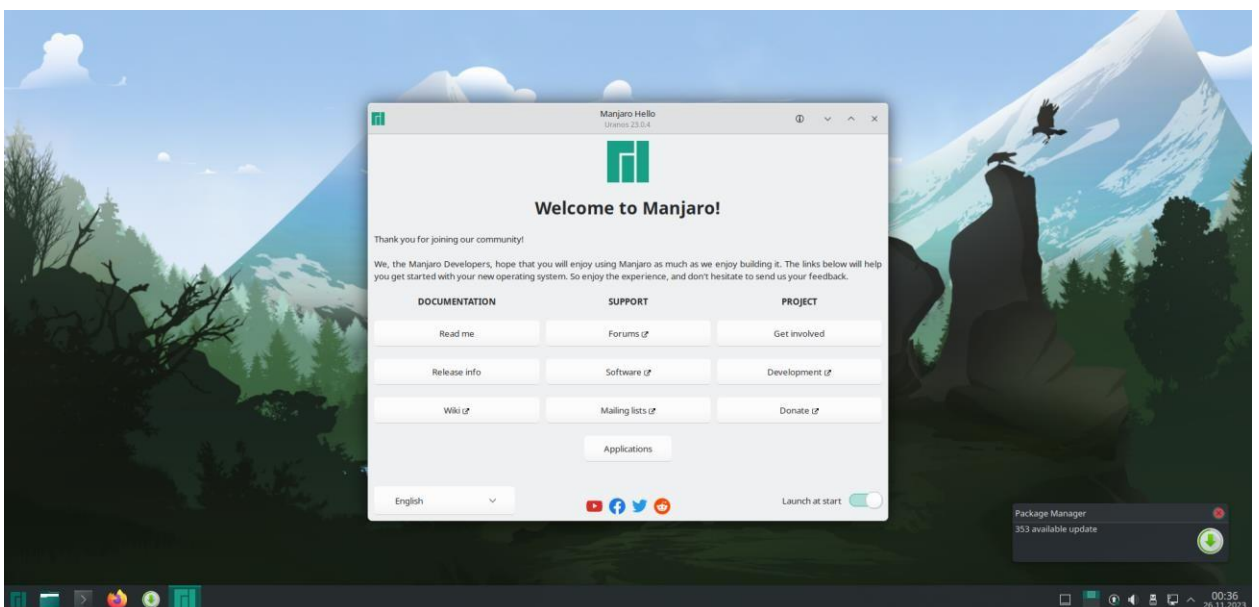


Рисунок 2.15 – Приветственное окно

Мы выбрали Manjaro Linux с окружением KDE Plasma, который является самой продвинутой и красивой менеджером окон. Дальше мы проведем обзор некоторых основных приложений.



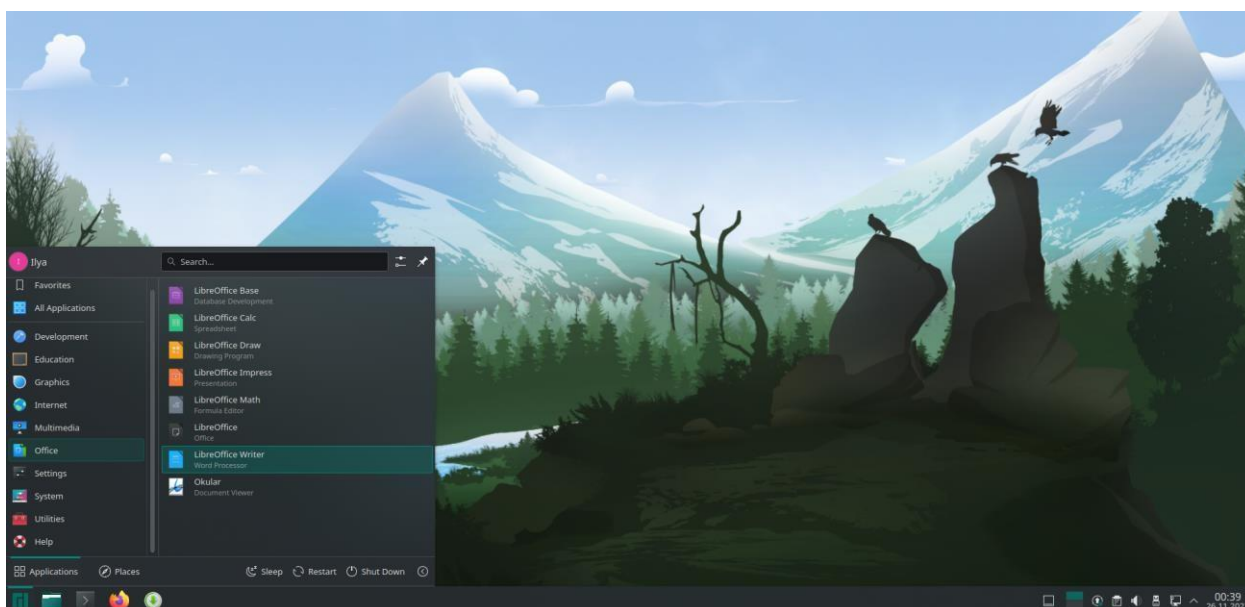


Рисунок 2.16 – Обзор системы

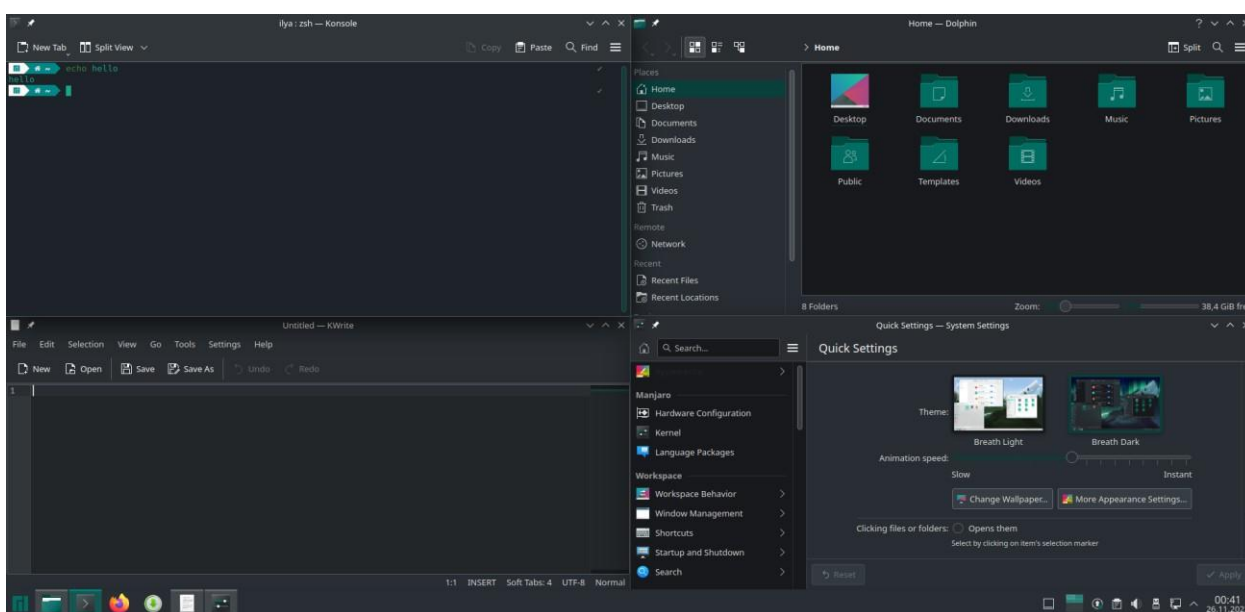


Рисунок 2.17 – Интерфейсы некоторых приложений

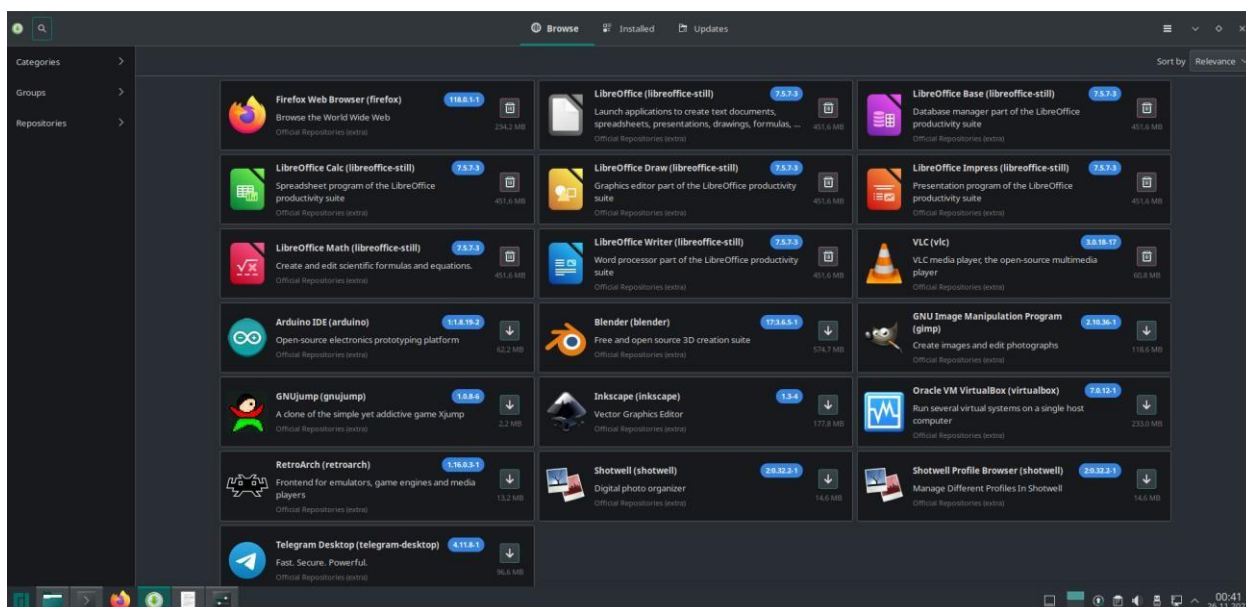


Рисунок 2.18 – Установщик приложений

### 3 Командная строка Linux

По заданию необходимо выполнить следующие действия:

1. Перейдите в свой домашний каталог
2. Создайте каталог A
3. Сделайте A текущим
4. Создайте пустые файлы a, b и c
5. Создайте каталог B
6. Сделайте B текущим
7. Создайте пустые файлы a, b и c
8. Перейдите в домашний каталог
9. Выведите список файлов в каталоге A
10. Выведите список файлов в каталоге A в подробном формате
11. Выведите подробную информацию о самом каталоге A
12. Выведите подробную информацию о каталоге A и всех вложенных каталогах и файлах
13. Создайте в своем домашнем каталоге символическую ссылку на каталог A (из предыдущего задания), используя абсолютный путь
14. Создайте в своем домашнем каталоге символическую ссылку на каталог A используя относительный путь
15. Переместите обе ссылки внутрь каталога A
16. Создайте в своем домашнем каталоге каталог tmp и повторите в нем структуру каталога A используя рекурсивное копирование
17. Удалите каталог tmp

Согласно варианту: каталог A – uuu, каталог B – vvv, файлы a, b, c – ww.txt, xx.txt, yy.txt соответственно.

Так как по заданию необходимо протоколировать вывод команд в файл с помощью перенаправления потока, то после первой команды мы введем оператор “>”, а после всех остальных команд мы вводим оператор “>>”. Разница из в том, что,> — стандартный вывод, а команды со знаком>> не

перезаписывают существующее содержимое файла, а присоединяют данные к нему. Исходя из этого получается следующий синтаксис.

### Листинг 3.1 – Команды

```
cd ~ > /home/ilya/Desktop/log3.txt
mkdir uuu >> /home/ilya/Desktop/log3.txt
cd uuu >> /home/ilya/Desktop/log3.txt
touch ww.txt xx.txt yy.txt >> /home/ilya/Desktop/log3.txt
mkdir vvv >> /home/ilya/Desktop/log3.txt
cd vvv >> /home/ilya/Desktop/log3.txt
touch ww.txt xx.txt yy.txt >> /home/ilya/Desktop/log3.txt
cd ~ >> /home/ilya/Desktop/log3.txt
ls uuu >> /home/ilya/Desktop/log3.txt
ls -l uuu >> /home/ilya/Desktop/log3.txt
ls -ld uuu >> /home/ilya/Desktop/log3.txt
ls -lR uuu >> /home/ilya/Desktop/log3.txt
ln -s /home/ilya/uuu uuu_abs_link >> /home/ilya/Desktop/log3.txt
ln -s ./uuu uuu_rel_link >> /home/ilya/Desktop/log3.txt
mv uuu_abs_link uuu_rel_link uuu/ >> /home/ilya/Desktop/log3.txt
mkdir tmp && cp -r uuu tmp/ >> /home/ilya/Desktop/log3.txt
rm -rf tmp/ >> /home/ilya/Desktop/log3.txt
```

Как только мы ввели все команды, на рабочем столе появился файл с названием lab3log.txt.

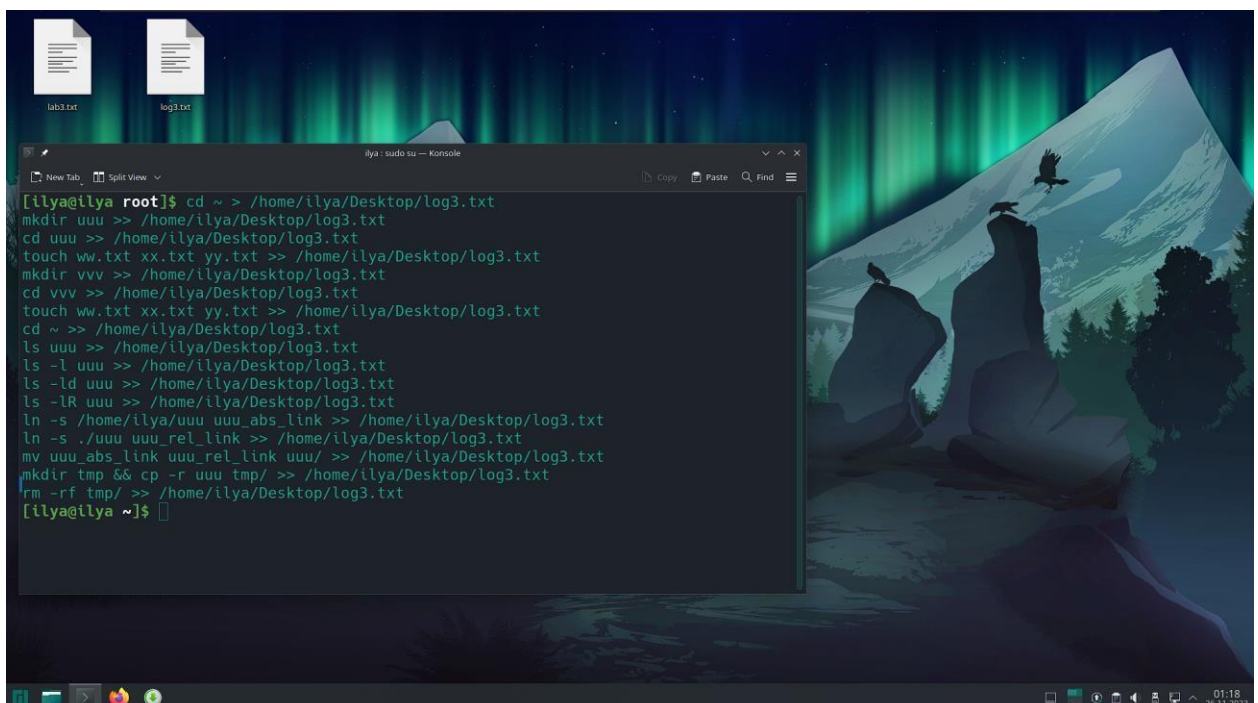


Рисунок 3.1 – Ввод команд в терминал и появление log файла

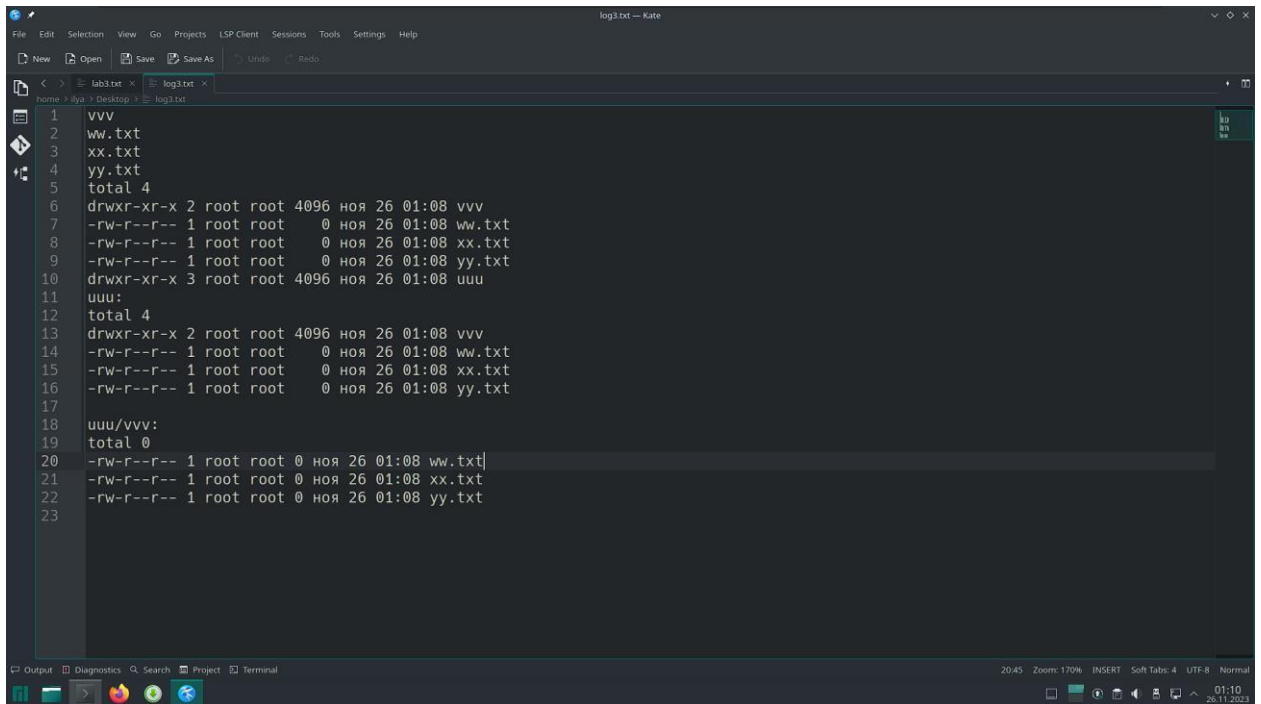


Рисунок 3.2 – Содержимое файла lab3log.txt

### Листинг 3.2 – Файл lab3log.txt

```
vuv
ww.txt
xx.txt
yy.txt
total 4
drwxr-xr-x 2 root root 4096 ноя 26 01:08 vuv
-rw-r--r-- 1 root root 0 ноя 26 01:08 ww.txt
-rw-r--r-- 1 root root 0 ноя 26 01:08 xx.txt
-rw-r--r-- 1 root root 0 ноя 26 01:08 yy.txt
drwxr-xr-x 3 root root 4096 ноя 26 01:08 uuu
uuu:
total 4
drwxr-xr-x 2 root root 4096 ноя 26 01:08 vuv
-rw-r--r-- 1 root root 0 ноя 26 01:08 ww.txt
-rw-r--r-- 1 root root 0 ноя 26 01:08 xx.txt
-rw-r--r-- 1 root root 0 ноя 26 01:08 yy.txt

uuu/vuv:
total 0
-rw-r--r-- 1 root root 0 ноя 26 01:08 ww.txt
-rw-r--r-- 1 root root 0 ноя 26 01:08 xx.txt
-rw-r--r-- 1 root root 0 ноя 26 01:08 yy.txt
```

Также если мы перейдем в домашний каталог, мы увидим директорию с названием `uuu`, и в ней также созданные файлы. А именно: каталог `B` – `vuv`, файлы `a`, `b`, `c` – `ww.txt`, `xx.txt` и `yy.txt` соответственно. По мимо этого также появились символические ссылки.

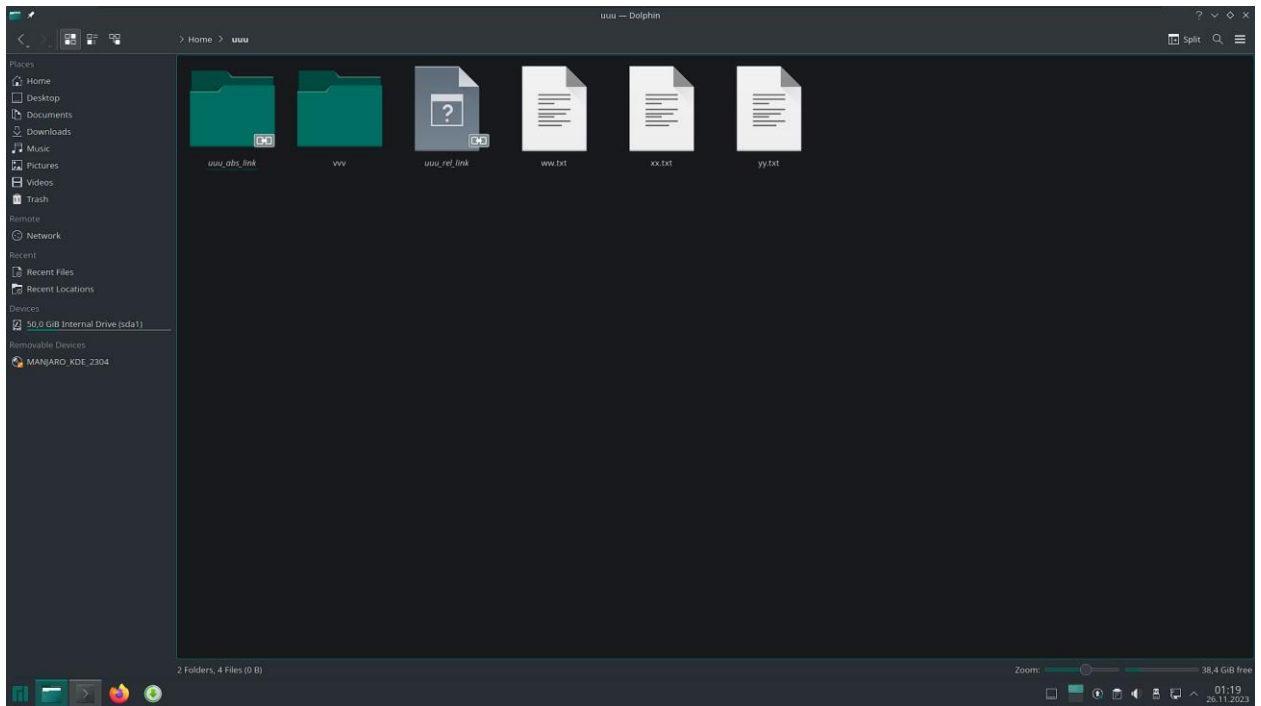


Рисунок 3.3 – Каталог uuu

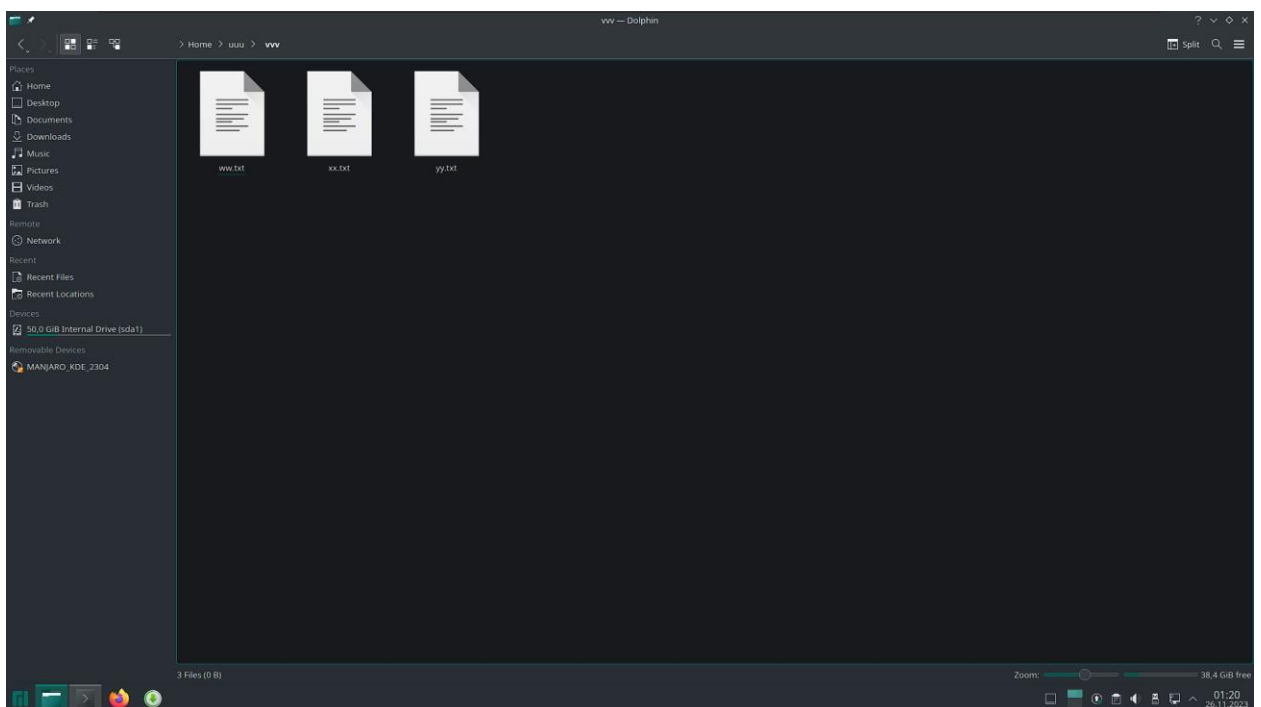


Рисунок 3.4 – Каталог vvv

## 4 Командные интерпретаторы

По заданию необходимо выполнить следующие действия:

1. Написать командный сценарий автоматизирующий ввод команд, использованных в лабораторной работе №3.
2. Набрать текст командного сценария в любом текстовом редакторе, снабдив его необходимыми комментариями.
3. Добиться работоспособности сценария и перенаправить его вывод в файл.
4. Сравнить вывод сценария с файлом, полученном в результате выполнения лабораторной работы №3, используя конвейеры и команду `cmp`.

Создадим `bash` скрипт для выполнения команд из третьей лабораторной работы, также снабдим файл необходимыми комментариями.

### Листинг 4.1 – Bash-file

```
#!/bin/bash
cd ~ # Переход в свой домашний каталог
mkdir uuu # Создание каталога uuu
cd uuu # Сделать uuu текущим
touch ww.txt xx.txt yy.txt # Создание пустых файлов ww.txt, xx.txt и yy.txt
mkdir vvv # Создание каталога vvv
cd vvv # Сделать vvv текущим
touch ww.txt xx.txt yy.txt # Создание пустых файлов ww.txt, xx.txt и yy.txt
cd ~ # Переход в домашний каталог
ls uuu # Вывод списка файлов в каталоге uuu
ls -l uuu # Вывод списка файлов в каталоге uuu в подробном формате
ls -ld uuu # Вывод подробной информации о самом каталоге uuu
ls -lR uuu # Вывод подробной информации о каталоге uuu и всех вложенных каталогах и файлах
ln -s /home/ilya/uuu uuu_abs_link # Создание в своем домашнем каталоге символическую ссылку на каталог uuu (из предыдущего задания), используя абсолютный путь
ln -s ./uuu uuu_rel_link # Создание в своем домашнем каталоге символическую ссылку на каталог uuu используя относительный путь
mv uuu_abs_link uuu_rel_link uuu/ # Перенос обоих ссылок внутрь каталога uuu
mkdir tmp && cp -r uuu tmp/ # Создание в своем домашнем каталоге каталога tmp и повторение в нем структуры каталога uuu используя рекурсивное копирование
rm -rf tmp/ # Удаление каталога tmp
```



Создадим файл lab4.bash на рабочем столе.

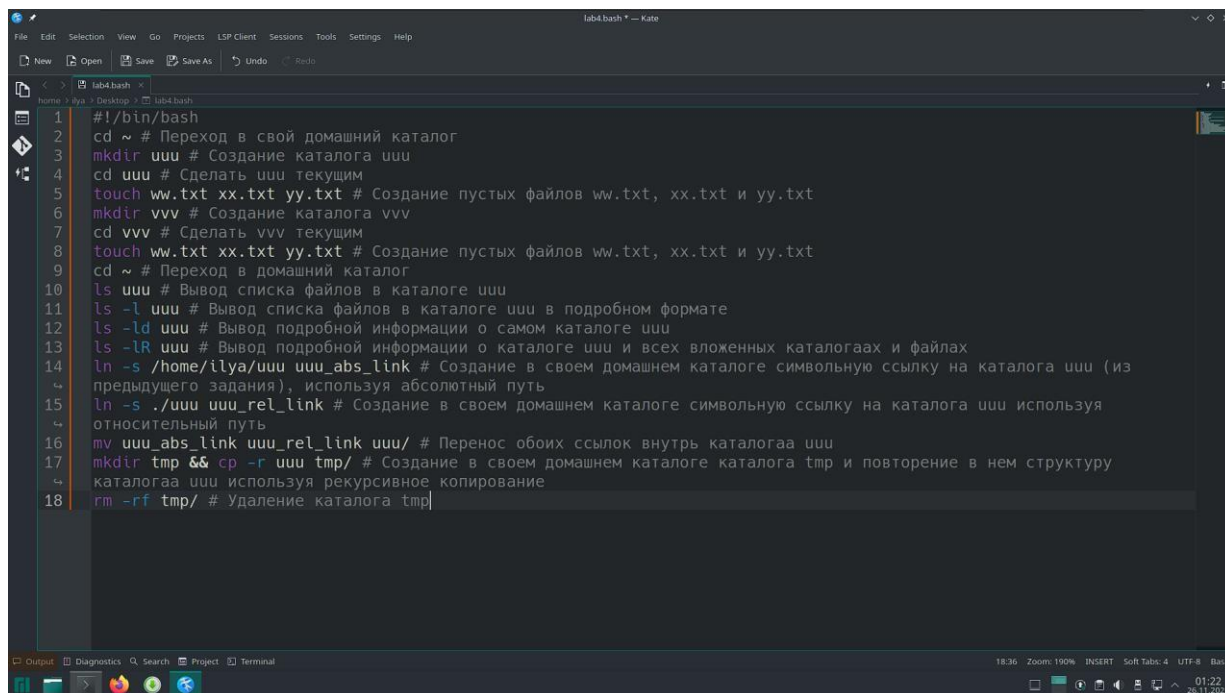


Рисунок 4.1 – Создание bash-file

Запустим текущий скрипт и перенаправим вывод в файл log4.txt.

#### Листинг 4.2 – Запуск bash скрипта

```
source lab4.bash > log4.txt
```

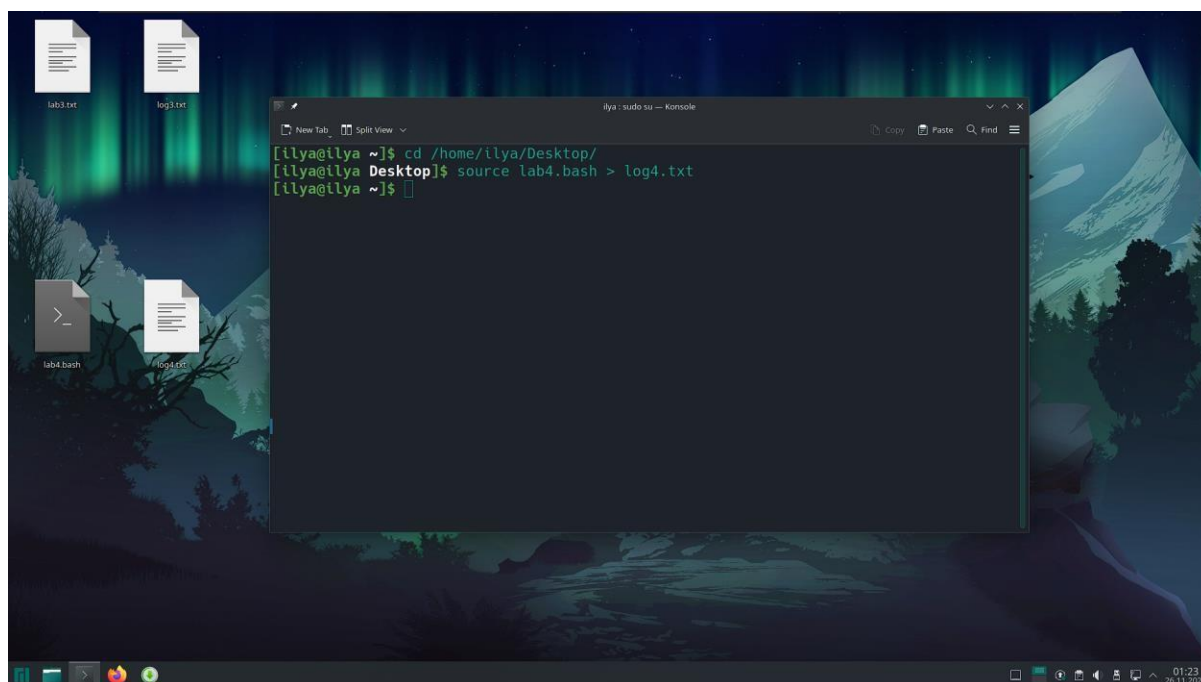


Рисунок 4.2 – Запуск командой source



Просмотрим содержимое файла log4.txt.

#### Листинг 4.3 – Содержимое файла log4.txt

```
vvv
ww.txt
xx.txt
yy.txt
total 4
drwxr-xr-x 2 ilya ilya 4096 ноя 26 01:23 vvv
-rw-r--r-- 1 ilya ilya 0 ноя 26 01:23 ww.txt
-rw-r--r-- 1 ilya ilya 0 ноя 26 01:23 xx.txt
-rw-r--r-- 1 ilya ilya 0 ноя 26 01:23 yy.txt
drwxr-xr-x 3 ilya ilya 4096 ноя 26 01:23 uuu
uuu:
total 4
drwxr-xr-x 2 ilya ilya 4096 ноя 26 01:23 vvv
-rw-r--r-- 1 ilya ilya 0 ноя 26 01:23 ww.txt
-rw-r--r-- 1 ilya ilya 0 ноя 26 01:23 xx.txt
-rw-r--r-- 1 ilya ilya 0 ноя 26 01:23 yy.txt

uuu/vvv:
total 0
-rw-r--r-- 1 ilya ilya 0 ноя 26 01:23 ww.txt
-rw-r--r-- 1 ilya ilya 0 ноя 26 01:23 xx.txt
-rw-r--r-- 1 ilya ilya 0 ноя 26 01:23 yy.txt
```

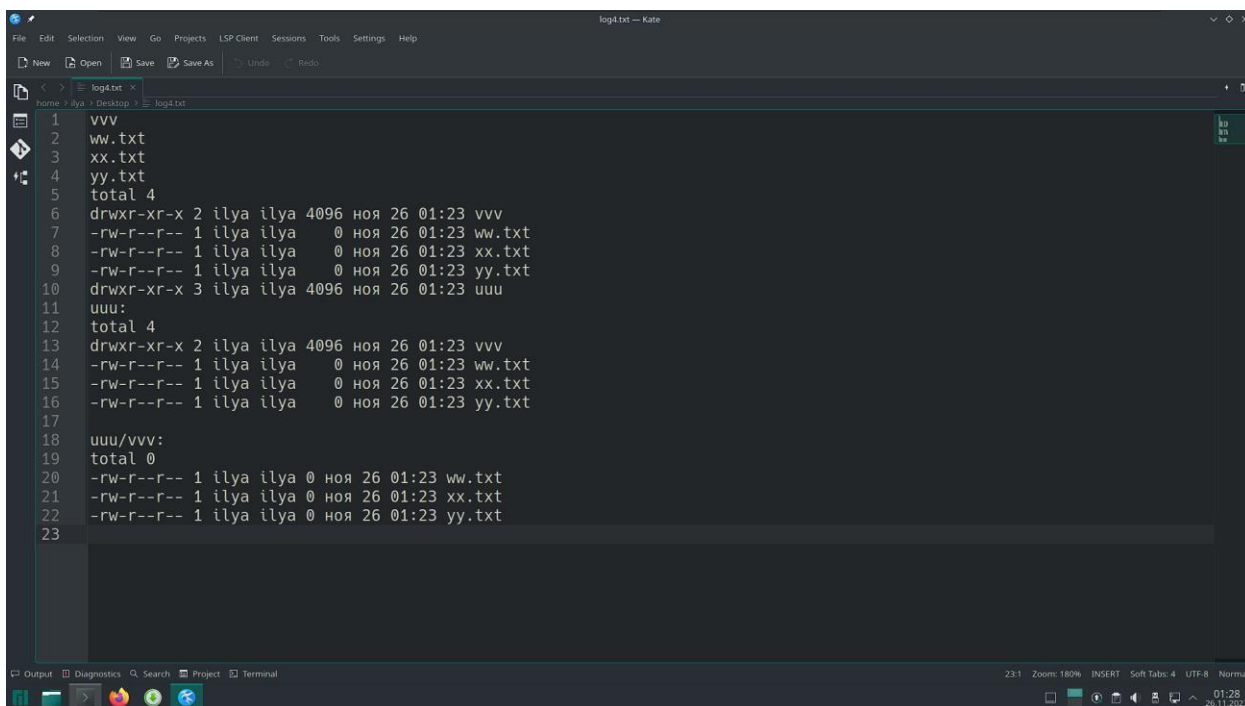


Рисунок 4.3 – Файл log4.txt

На первый взгляд файлы ничем не отличаются, но, чтобы в этом удостовериться проверим это командой `cmp` и конвейерами.

Введем команду `cmp -b log3.txt log4.txt`, чтобы сравнить их побайтово.

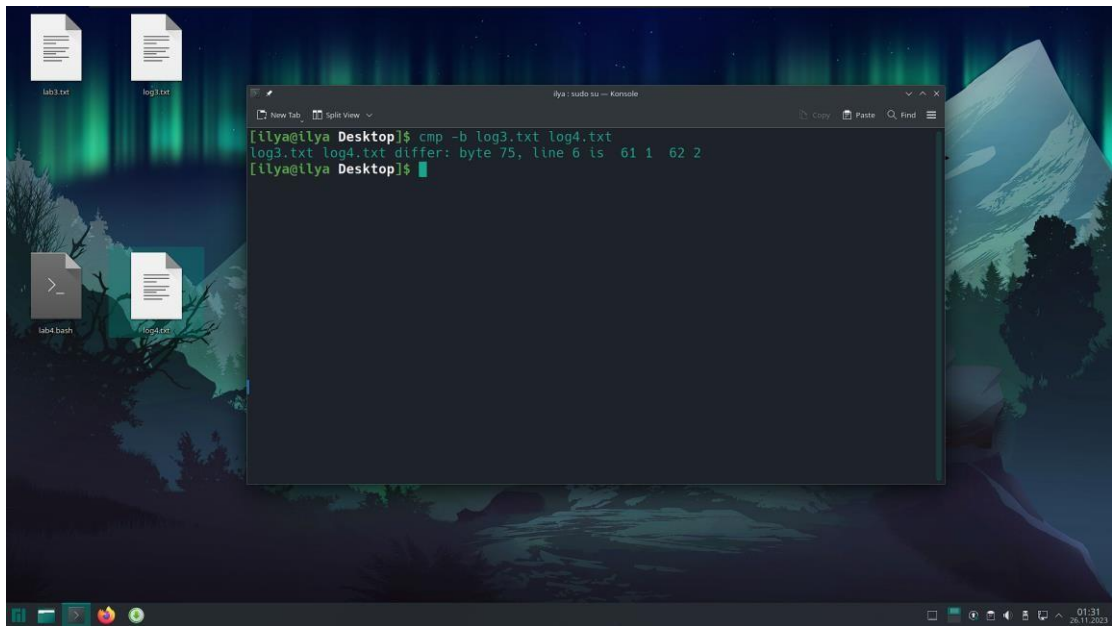


Рисунок 4.4 – Команда cmp

Видим, что файлы отличаются по 79 байту, предположительно это из-за разного времени создания файлов. Далее, используя конвейер diff, выведем какие строки в текстовых файлах нужно изменить чтобы они стали идентичными.

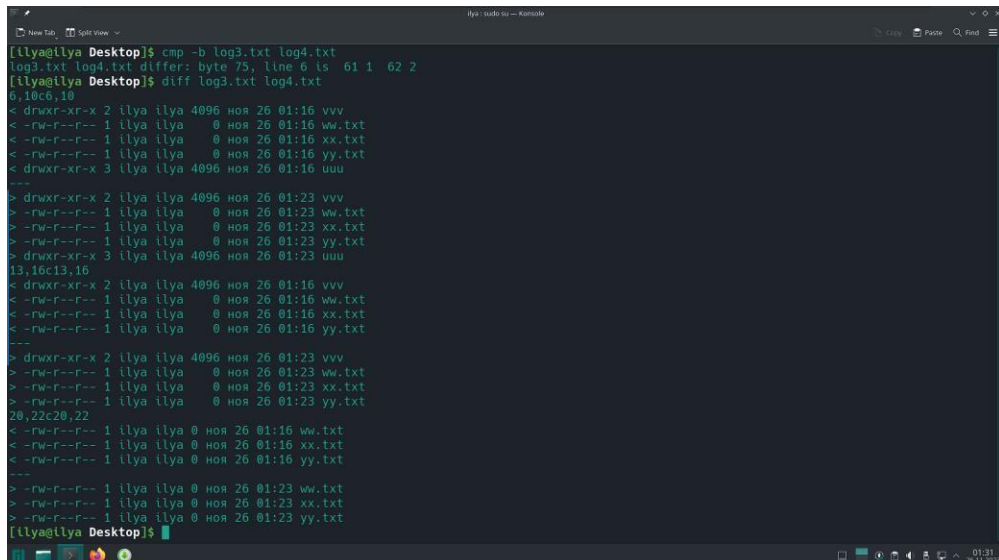
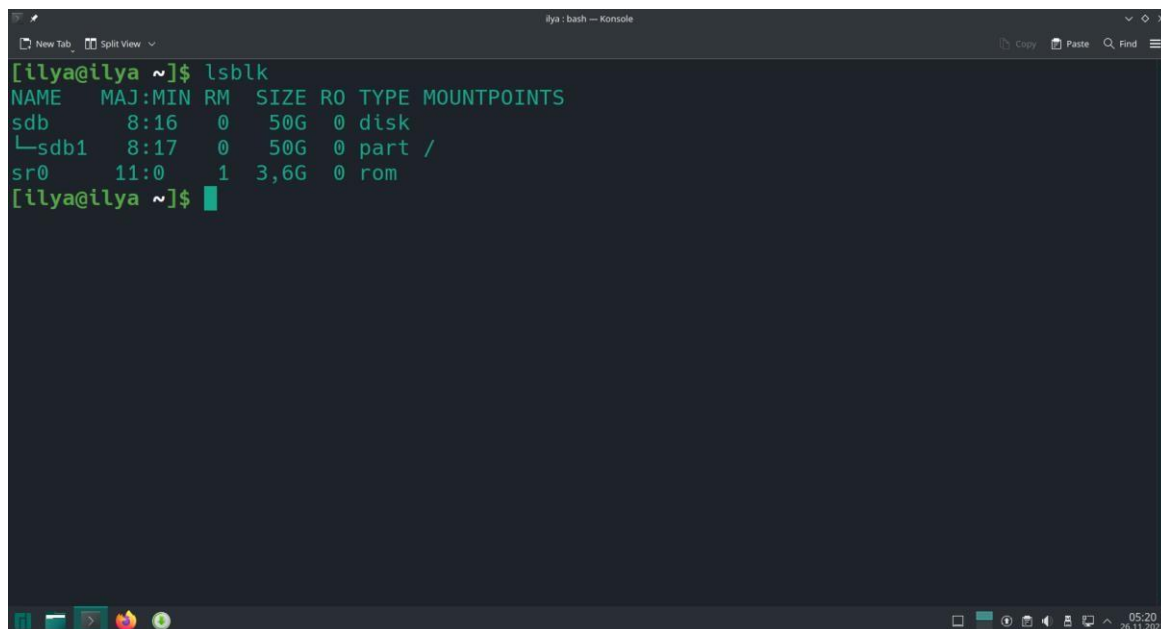


Рисунок 4.5 – Команда diff

Как мы видим, команда вывела только те строки, где есть список файлов, так как у них отличаются время создания.

## 5 Файлы устройств и монтирование

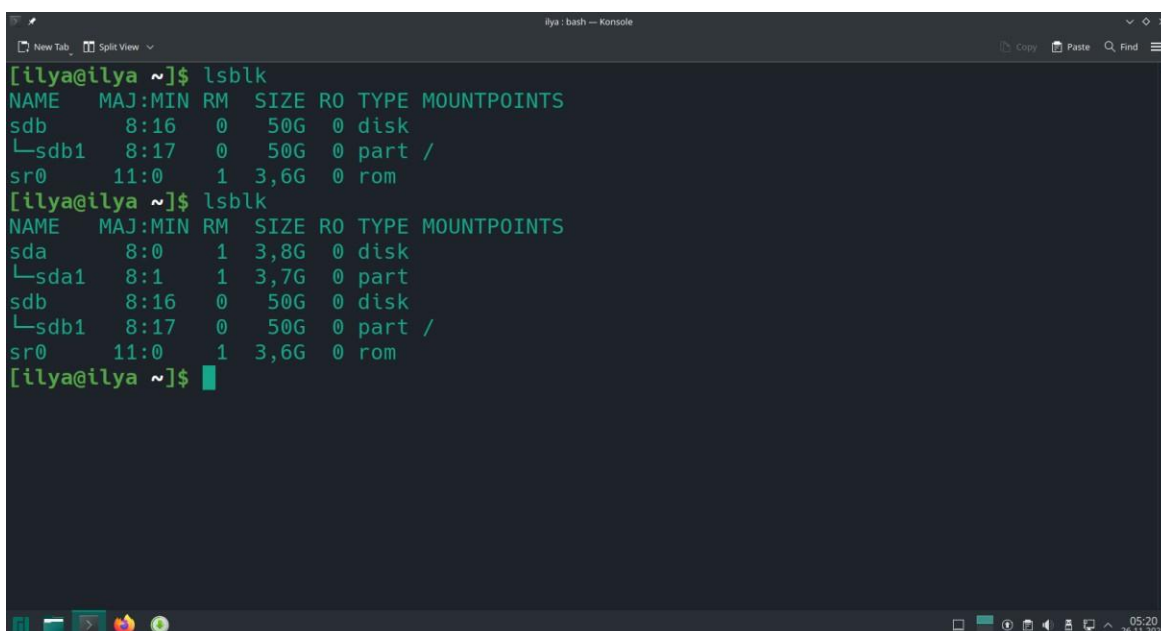
Получить список всех доступных устройств (Рисунок 5.1).



```
[ilya@ilya ~]$ lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS
sdb   8:16   0  50G  0 disk
└─sdb1 8:17   0  50G  0 part /
sr0   11:0   1  3,6G  0 rom
```

Рисунок 5.1 – Список доступных устройств

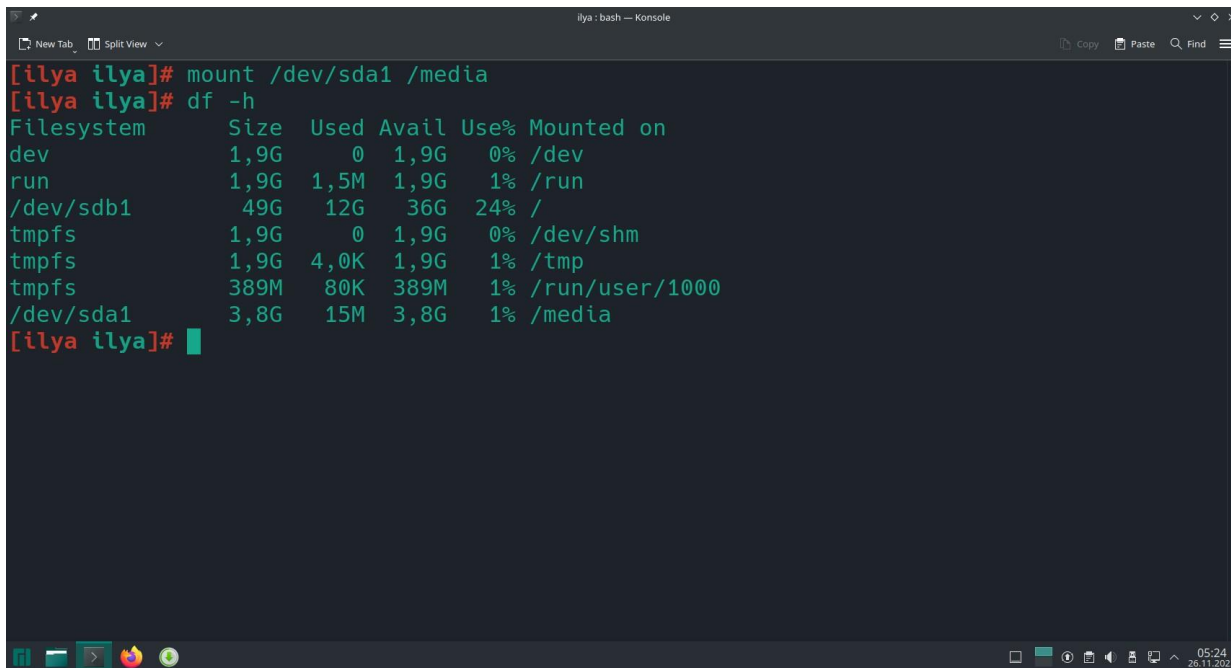
Подключить к компьютеру носители данных: Images-диск и DVD.  
Получить их идентификаторы всеми известными способами (Рисунок 5.2).



```
[ilya@ilya ~]$ lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS
sdb   8:16   0  50G  0 disk
└─sdb1 8:17   0  50G  0 part /
sr0   11:0   1  3,6G  0 rom
[ilya@ilya ~]$ lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS
sda   8:0    1  3,8G  0 disk
└─sda1 8:1    1  3,7G  0 part
sdb   8:16   0  50G  0 disk
└─sdb1 8:17   0  50G  0 part /
sr0   11:0   1  3,6G  0 rom
```

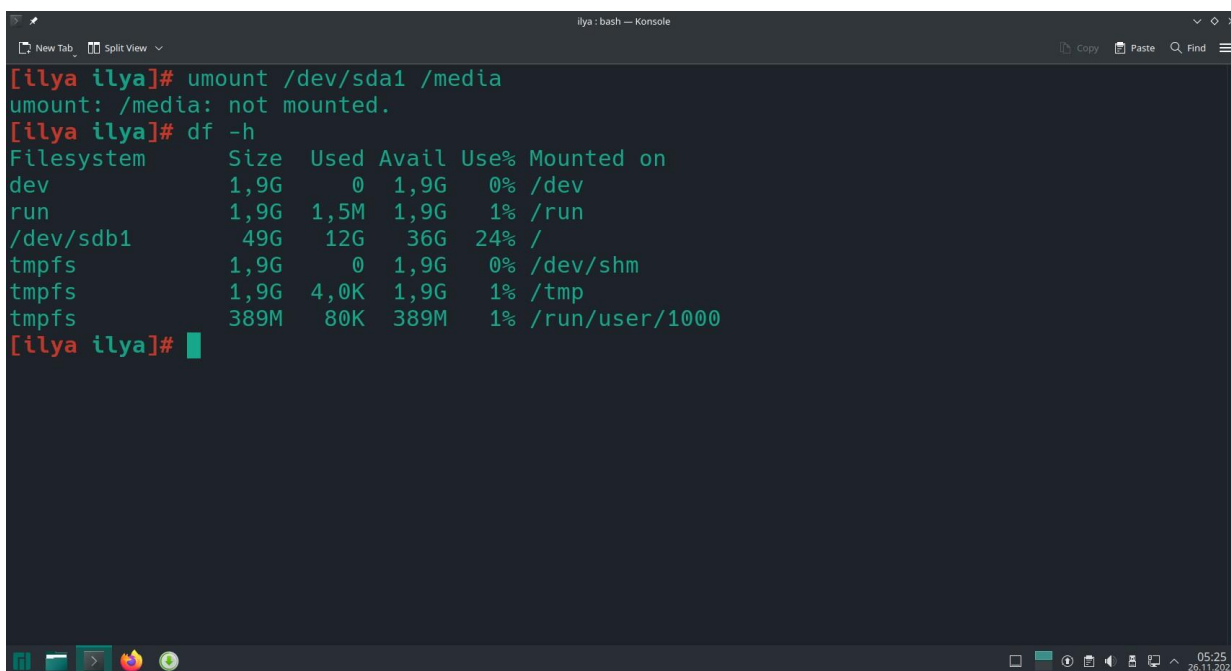
Рисунок 5.2 – Список доступных устройств после подключения usb

С помощью команд `mount` и `umount` выполнить монтирование и размонтирование файловой системы носителя к трём произвольным каталогам корневой файловой системы (Рисунок 5.3-8).



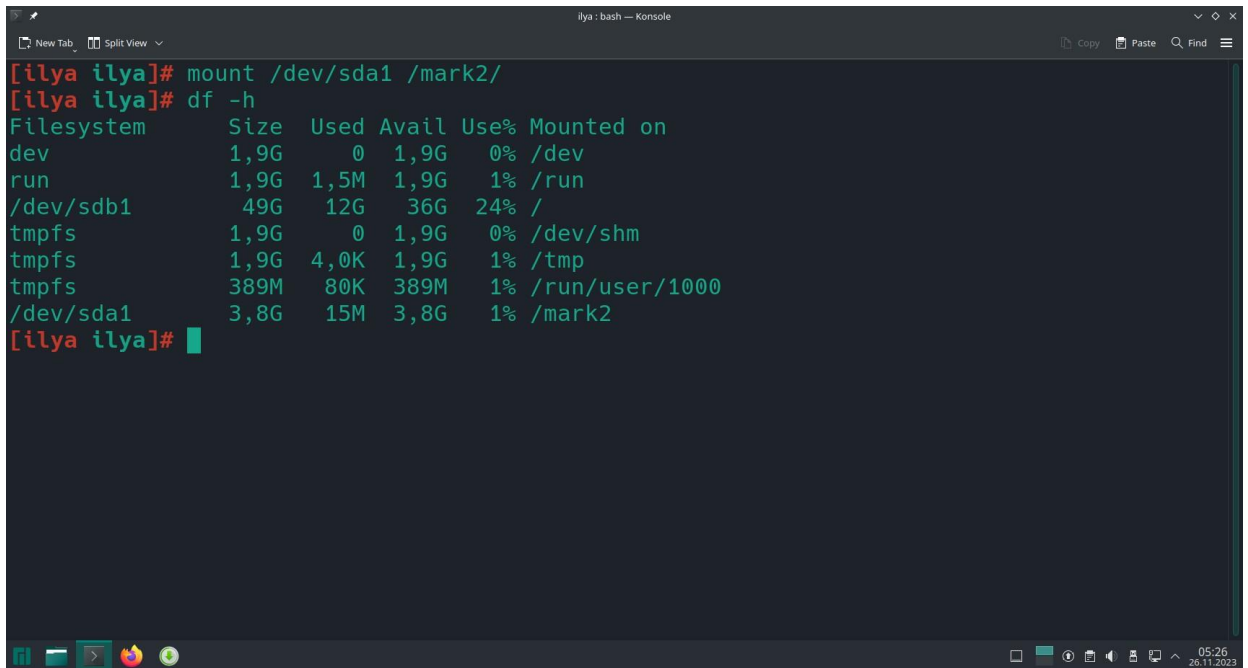
```
[ilya ilya]# mount /dev/sda1 /media
[ilya ilya]# df -h
Filesystem      Size  Used Avail Use% Mounted on
dev              1,9G   0    1,9G   0% /dev
run              1,9G  1,5M   1,9G   1% /run
/dev/sdb1        49G   12G   36G   24% /
tmpfs            1,9G   0    1,9G   0% /dev/shm
tmpfs            1,9G  4,0K   1,9G   1% /tmp
tmpfs            389M   80K   389M   1% /run/user/1000
/dev/sda1        3,8G   15M   3,8G   1% /media
[ilya ilya]#
```

Рисунок 5.3 – Монтирование usb в папку «media»



```
[ilya ilya]# umount /dev/sda1 /media
umount: /media: not mounted.
[ilya ilya]# df -h
Filesystem      Size  Used Avail Use% Mounted on
dev              1,9G   0    1,9G   0% /dev
run              1,9G  1,5M   1,9G   1% /run
/dev/sdb1        49G   12G   36G   24% /
tmpfs            1,9G   0    1,9G   0% /dev/shm
tmpfs            1,9G  4,0K   1,9G   1% /tmp
tmpfs            389M   80K   389M   1% /run/user/1000
[ilya ilya]#
```

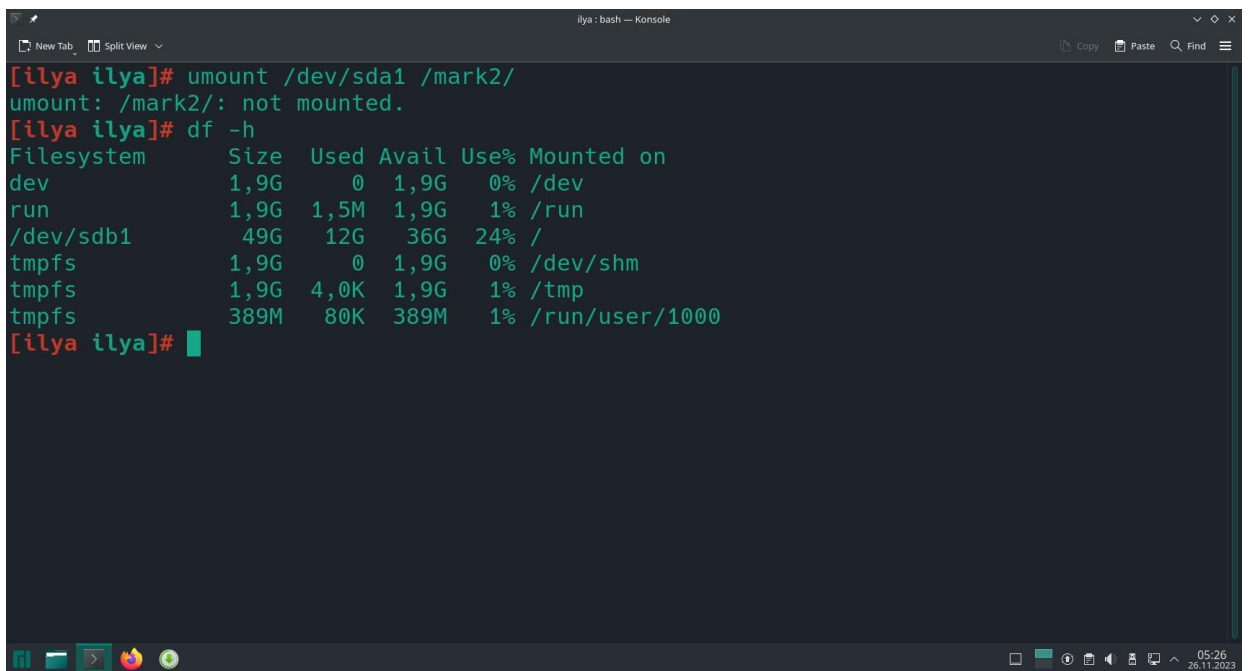
Рисунок 5.4 – Размонтирование usb из папки «media»



A terminal window titled 'ilya : bash — Konsole' showing the execution of two commands. The first command is 'mount /dev/sda1 /mark2/' and the second is 'df -h'. The output of 'df -h' is a table showing disk usage for various filesystems.

```
[ilya ilya]# mount /dev/sda1 /mark2/
[ilya ilya]# df -h
Filesystem      Size  Used Avail Use% Mounted on
dev              1,9G   0    1,9G   0% /dev
run              1,9G  1,5M   1,9G   1% /run
/dev/sdb1        49G   12G   36G   24% /
tmpfs            1,9G   0    1,9G   0% /dev/shm
tmpfs            1,9G  4,0K   1,9G   1% /tmp
tmpfs            389M   80K   389M   1% /run/user/1000
/dev/sda1        3,8G   15M   3,8G   1% /mark2
[ilya ilya]#
```

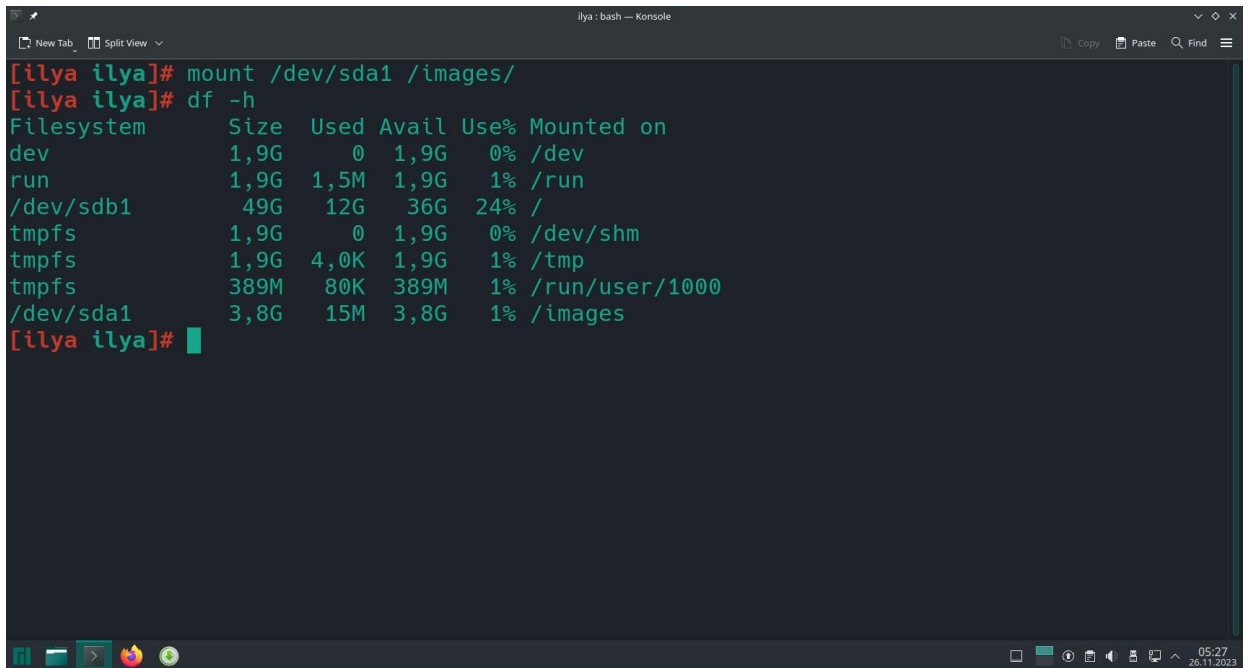
Рисунок 5.5 – Монтирование usb в папку «mark2»



A terminal window titled 'ilya : bash — Konsole' showing the execution of two commands. The first command is 'umount /dev/sda1 /mark2/' which returns an error message. The second command is 'df -h', which shows the same disk usage table as in the previous screenshot.

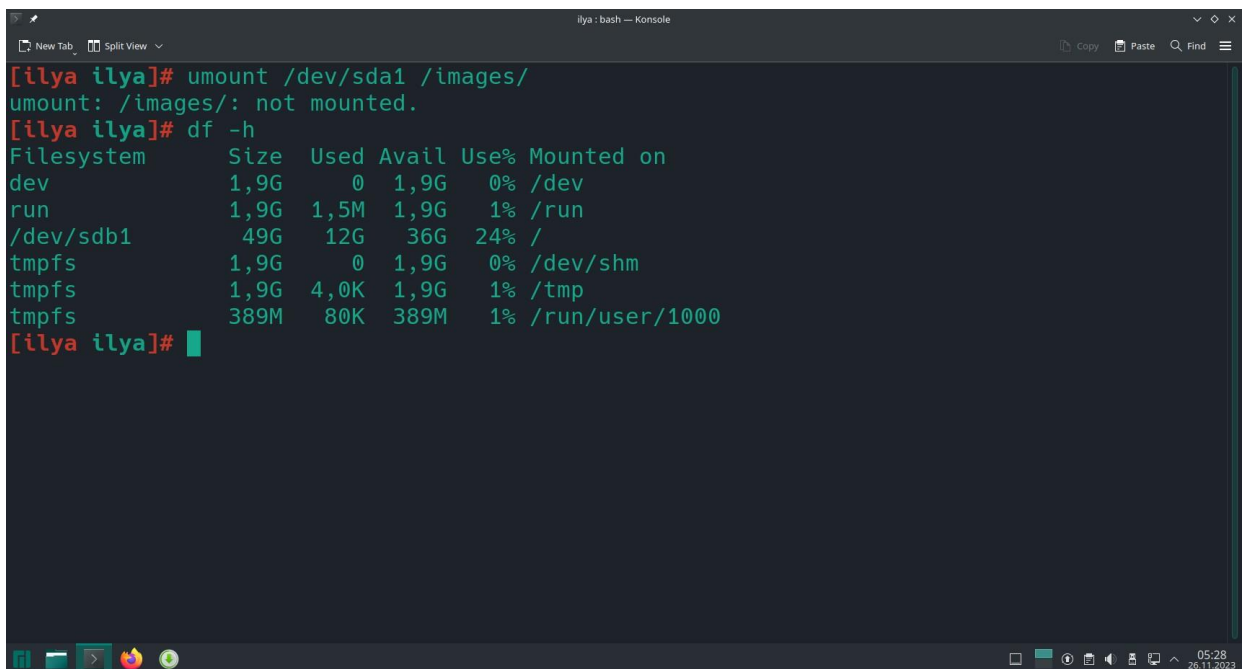
```
[ilya ilya]# umount /dev/sda1 /mark2/
umount: /mark2/: not mounted.
[ilya ilya]# df -h
Filesystem      Size  Used Avail Use% Mounted on
dev              1,9G   0    1,9G   0% /dev
run              1,9G  1,5M   1,9G   1% /run
/dev/sdb1        49G   12G   36G   24% /
tmpfs            1,9G   0    1,9G   0% /dev/shm
tmpfs            1,9G  4,0K   1,9G   1% /tmp
tmpfs            389M   80K   389M   1% /run/user/1000
[ilya ilya]#
```

Рисунок 5.6 – Размонтирование usb из папки «mark2»



```
[ilya ilya]# mount /dev/sda1 /images/
[ilya ilya]# df -h
Filesystem      Size  Used Avail Use% Mounted on
dev              1,9G   0    1,9G   0% /dev
run              1,9G  1,5M   1,9G   1% /run
/dev/sdb1        49G   12G   36G   24% /
tmpfs            1,9G   0    1,9G   0% /dev/shm
tmpfs            1,9G  4,0K   1,9G   1% /tmp
tmpfs            389M   80K   389M   1% /run/user/1000
/dev/sda1        3,8G   15M   3,8G   1% /images
[ilya ilya]#
```

Рисунок 5.7 – Монтирование usb в папку «images»



```
[ilya ilya]# umount /dev/sda1 /images/
umount: /images/: not mounted.
[ilya ilya]# df -h
Filesystem      Size  Used Avail Use% Mounted on
dev              1,9G   0    1,9G   0% /dev
run              1,9G  1,5M   1,9G   1% /run
/dev/sdb1        49G   12G   36G   24% /
tmpfs            1,9G   0    1,9G   0% /dev/shm
tmpfs            1,9G  4,0K   1,9G   1% /tmp
tmpfs            389M   80K   389M   1% /run/user/1000
[ilya ilya]#
```

Рисунок 5.8 – Размонтирование usb из папки «images»

Напишем bash-script файл, который отобразить состояния до и после команды mount, а также umount (Листинг 5.1), и выполним его, перенаправив поток в файл lab5.txt (Листинг 5.2).

## Листинг 5.1 – bash-script

```
#!/bin/bash
echo before mount
echo /media
echo lsblk:
echo
lsblk
echo
echo folder /media contains:
ls -la /media
echo
echo after mount
mount /dev/sda1 /media
echo
echo lsblk:
echo
lsblk
echo
echo folder /media contains:
echo
ls -la /media
echo
echo after umount
umount /dev/sda1 /media
echo
echo lsblk:
echo
lsblk
echo
echo folder /media contains:
ls -la /media
echo

echo before mount
echo /mark2
echo lsblk:
echo
lsblk
echo
echo folder /mark2 contains:
ls -la /mark2
echo
echo after mount
mount /dev/sda1 /mark2
echo
echo lsblk:
echo
lsblk
echo
echo folder /mark2 contains:
echo
ls -la /mark2
echo
echo after umount
umount /dev/sda1 /mark2
echo
echo lsblk:
echo
```

```

lsblk
echo
echo folder /mark2 contains:
ls -la /mark2
echo

echo before mount
echo /images
echo lsblk:
echo
lsblk
echo
echo folder /images contains:
ls -la /images
echo
echo after mount
mount /dev/sda1 /images
echo
echo lsblk:
echo
lsblk
echo
echo folder /images contains:
echo
ls -la /images
echo
echo after umount
umount /dev/sda1 /images
echo
echo lsblk:
echo
lsblk
echo
echo folder /images contains:
ls -la /images

```

## Листинг 5.2 – Файл lab5.txt

```

before mount
/media
lsblk:

NAME      MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
sda         8:0    1   3,8G  0 disk
└─sda1      8:1    1   3,7G  0 part
sdb         8:16    0    50G  0 disk
└─sdb1      8:17    0    50G  0 part /
sr0        11:0    1   3,6G  0 rom

folder /media contains:
total 8
drwxr-xr-x  2 root root 4096 ноя 26 05:23 .
drwxr-xr-x 20 root root 4096 ноя 26 05:23 ..

after mount

lsblk:

```



NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINTS
sda	8:0	1	3,8G	0	disk	
└sda1	8:1	1	3,7G	0	part	/media
sdb	8:16	0	50G	0	disk	
└sdb1	8:17	0	50G	0	part	/
sr0	11:0	1	3,6G	0	rom	

folder /media contains:

total 9

```
drwxrwxrwx  1 root root 4096 ноя 25 19:15 .
drwxr-xr-x 20 root root 4096 ноя 26 05:23 ..
-rwxrwxrwx  1 root root   52 ноя 25 19:15 file on flash.txt
drwxrwxrwx  1 root root    0 ноя 25 19:15 System Volume Information
```

after umount

lsblk:

NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINTS
sda	8:0	1	3,8G	0	disk	
└sda1	8:1	1	3,7G	0	part	
sdb	8:16	0	50G	0	disk	
└sdb1	8:17	0	50G	0	part	/
sr0	11:0	1	3,6G	0	rom	

folder /media contains:

total 8

```
drwxr-xr-x  2 root root 4096 ноя 26 05:23 .
drwxr-xr-x 20 root root 4096 ноя 26 05:23 ..
```

before mount

/mark2

lsblk:

NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINTS
sda	8:0	1	3,8G	0	disk	
└sda1	8:1	1	3,7G	0	part	
sdb	8:16	0	50G	0	disk	
└sdb1	8:17	0	50G	0	part	/
sr0	11:0	1	3,6G	0	rom	

folder /mark2 contains:

total 8

```
drwxr-xr-x  2 root root 4096 ноя 26 05:23 .
drwxr-xr-x 20 root root 4096 ноя 26 05:23 ..
```

after mount

lsblk:

NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINTS
sda	8:0	1	3,8G	0	disk	
└sda1	8:1	1	3,7G	0	part	/mark2
sdb	8:16	0	50G	0	disk	
└sdb1	8:17	0	50G	0	part	/
sr0	11:0	1	3,6G	0	rom	

folder /mark2 contains:

```
total 9
drwxrwxrwx  1 root root 4096 ноя 25 19:15 .
drwxr-xr-x 20 root root 4096 ноя 26 05:23 ..
-rwxrwxrwx  1 root root   52 ноя 25 19:15 file on flash.txt
drwxrwxrwx  1 root root    0 ноя 25 19:15 System Volume Information
```

after umount

lsblk:

NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINTS
sda	8:0	1	3,8G	0	disk	
└sda1	8:1	1	3,7G	0	part	
sdb	8:16	0	50G	0	disk	
└sdb1	8:17	0	50G	0	part	/
sr0	11:0	1	3,6G	0	rom	

folder /mark2 contains:

```
total 8
drwxr-xr-x  2 root root 4096 ноя 26 05:23 .
drwxr-xr-x 20 root root 4096 ноя 26 05:23 ..
```

before mount

/images

lsblk:

NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINTS
sda	8:0	1	3,8G	0	disk	
└sda1	8:1	1	3,7G	0	part	
sdb	8:16	0	50G	0	disk	
└sdb1	8:17	0	50G	0	part	/
sr0	11:0	1	3,6G	0	rom	

folder /images contains:

```
total 8
drwxr-xr-x  2 root root 4096 ноя 26 05:23 .
drwxr-xr-x 20 root root 4096 ноя 26 05:23 ..
```

after mount

lsblk:

NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINTS
sda	8:0	1	3,8G	0	disk	
└sda1	8:1	1	3,7G	0	part	/images
sdb	8:16	0	50G	0	disk	
└sdb1	8:17	0	50G	0	part	/
sr0	11:0	1	3,6G	0	rom	

folder /images contains:

```
total 9
drwxrwxrwx  1 root root 4096 ноя 25 19:15 .
drwxr-xr-x 20 root root 4096 ноя 26 05:23 ..
-rwxrwxrwx  1 root root   52 ноя 25 19:15 file on flash.txt
drwxrwxrwx  1 root root    0 ноя 25 19:15 System Volume Information
```

```

after umount

lsblk:

NAME      MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
sda         8:0    1   3,8G  0 disk
└─sda1      8:1    1   3,7G  0 part
sdb         8:16    0    50G  0 disk
└─sdb1      8:17    0    50G  0 part /
sr0        11:0    1   3,6G  0 rom

folder /images contains:
total 8
drwxr-xr-x  2 root root 4096 ноя 26 05:23 .
drwxr-xr-x 20 root root 4096 ноя 26 05:23 ..

```

Изучить файл `/etc/fstab` и объяснить приведенные в нем строки (Рисунок 5.9).

Файл `/etc/fstab` в Linux представляет собой таблицу, используемую операционной системой для определения, какие файловые системы и устройства должны быть автоматически монтированы при запуске системы. Каждая строка в этом файле описывает одну файловую систему и содержит различные параметры для ее монтирования.

**Устройство (Device):** Это устройство или раздел, которое будет монтировано. Может быть указано по имени (например, `/dev/sda1`) или по UUID (уникальному идентификатору) устройства.

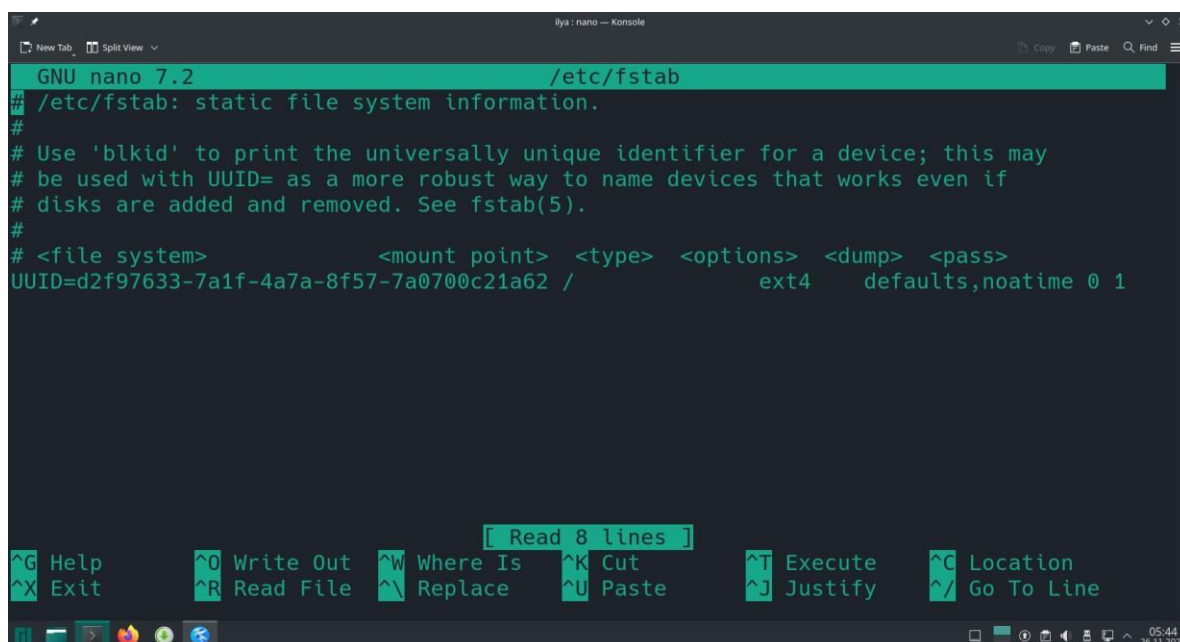
**Монтирование (Mount Point):** Это место, куда файловая система будет монтирована в файловой иерархии. Обычно это каталог в системе.

**Тип файловой системы (Mark2ystem Type):** Это тип файловой системы, находящейся на устройстве (например, `ext4`, `ntfs`, `vfat` и т. д.).

**Опции (Options):** Здесь можно указать различные опции для монтирования, такие как `ro` (только чтение) или `rw` (чтение и запись), а также другие опции, такие как `auto` (автоматическое монтирование при загрузке), `noauto` (не автоматически монтировать), `user` (разрешить пользователям монтировать), и так далее.

**Резерв (Dump):** Это резервная копия параметра, используемого для резервного копирования. Обычно ставится 0.

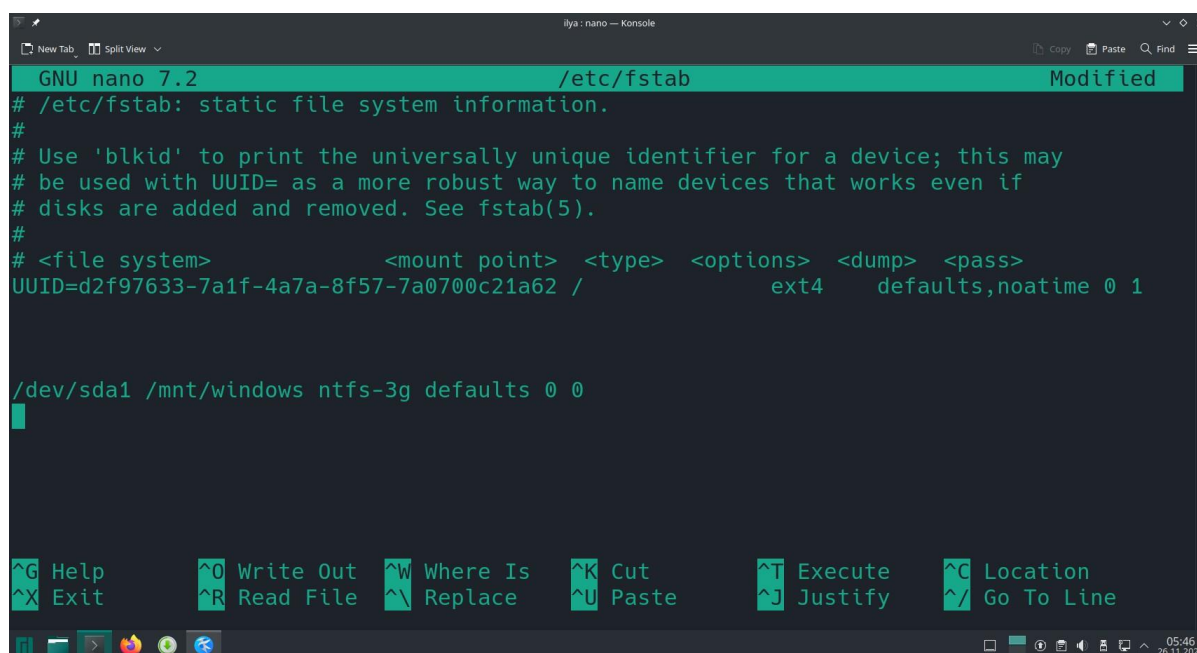
Период (Pass): Этот параметр используется для проверки целостности файловой системы при загрузке. Обычно ставится 2 (проверка по очереди).



```
GNU nano 7.2 /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a device; this may
# be used with UUID= as a more robust way to name devices that works even if
# disks are added and removed. See fstab(5).
#
# <file system>          <mount point> <type> <options> <dump> <pass>
UUID=d2f97633-7a1f-4a7a-8f57-7a0700c21a62 /          ext4    defaults,noatime 0 1
```

Рисунок 5.9 – Содержимое файла /etc/fstab

Добавить в файл строки, позволяющие монтировать разделы Windows/Images-диск (Рисунок 5.10).

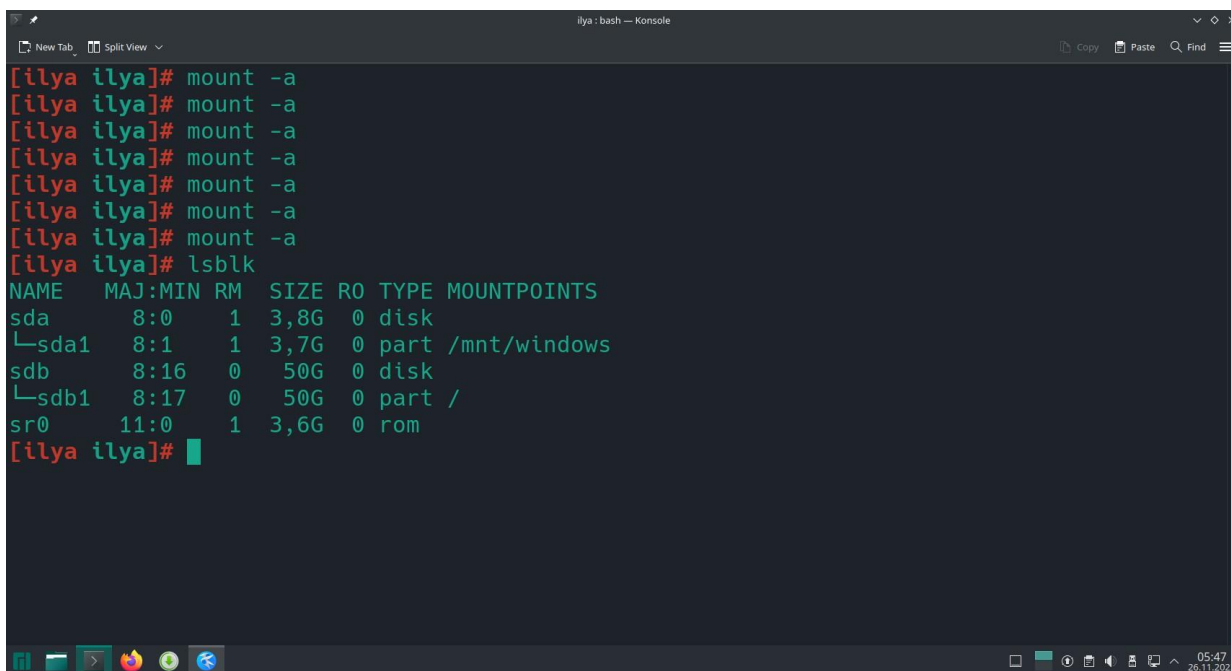


```
GNU nano 7.2 /etc/fstab Modified
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a device; this may
# be used with UUID= as a more robust way to name devices that works even if
# disks are added and removed. See fstab(5).
#
# <file system>          <mount point> <type> <options> <dump> <pass>
UUID=d2f97633-7a1f-4a7a-8f57-7a0700c21a62 /          ext4    defaults,noatime 0 1

/dev/sda1 /mnt/windows ntfs-3g defaults 0 0
```

Рисунок 5.10 – Добавление строки для монтирования раздела windows

Проверим корректность созданного файла командой `mount -a` (Рисунок 5.11).



```
[ilya ilya]# mount -a
[ilya ilya]# mount -a
[ilya ilya]# mount -a
[ilya ilya]# mount -a
[ilya ilya]# mount -a
[ilya ilya]# mount -a
[ilya ilya]# mount -a
[ilya ilya]# mount -a
[ilya ilya]# lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda           8:0    1   3,8G  0 disk
└─sda1        8:1    1   3,7G  0 part /mnt/windows
sdb           8:16   0    50G  0 disk
└─sdb1        8:17   0    50G  0 part /
sr0          11:0    1   3,6G  0 rom
[ilya ilya]#
```

Рисунок 5.11 – Применение изменений в файле

История создания файловой системы VFAT:

VFAT, что расшифровывается как «Virtual File Allocation Table», представляет собой файловую систему, используемую в операционных системах Microsoft Windows. История создания VFAT связана с появлением Windows 95 в середине 1990-х годов.

**FAT12 и FAT16:** Ранние версии MS-DOS и Windows (до Windows 95) использовали файловые системы FAT12 и FAT16 (File Allocation Table). Они имели множество ограничений, такие как максимальный размер раздела и ограничения на количество файлов и директорий в одной директории.

**Windows 95 и поддержка длинных имён файлов:** Появление Windows 95 потребовало новой файловой системы, которая поддерживала длинные имена файлов (до 255 символов), в отличие от ограничений на 8.3 символов в более ранних версиях. Эта система получила название VFAT и была представлена вместе с Windows 95.

**Обратная совместимость:** VFAT была спроектирована так, чтобы она была обратно совместима с более ранними версиями FAT. Это означает, что диски, отформатированные под FAT12 или FAT16, могли быть использованы на компьютерах с Windows 95, и в них можно было записывать файлы с длинными именами, хотя они не могли быть прочитаны на более старых системах.

**Долгожданное обновление:** VFAT предоставила долгожданное обновление файловой системы FAT, что улучшило совместимость и возможности для хранения файлов на дисках.

**Современные итерации:** на протяжении времени были представлены более современные итерации файловых систем, такие как NTFS, используемые в более новых версиях Windows. Однако VFAT до сих пор используется для обеспечения совместимости с разными операционными системами и устройствами, так как она общепризнана и широко поддерживается.

Назначение и особенности файловой системы VFAT:

**Совместимость:** Одним из основных назначений VFAT является обеспечение совместимости с различными операционными системами, включая Windows, Linux и другие. Это позволяет перемещать носители данных (например, флэш-накопители) между разными системами и устройствами, сохраняя доступность файлов.

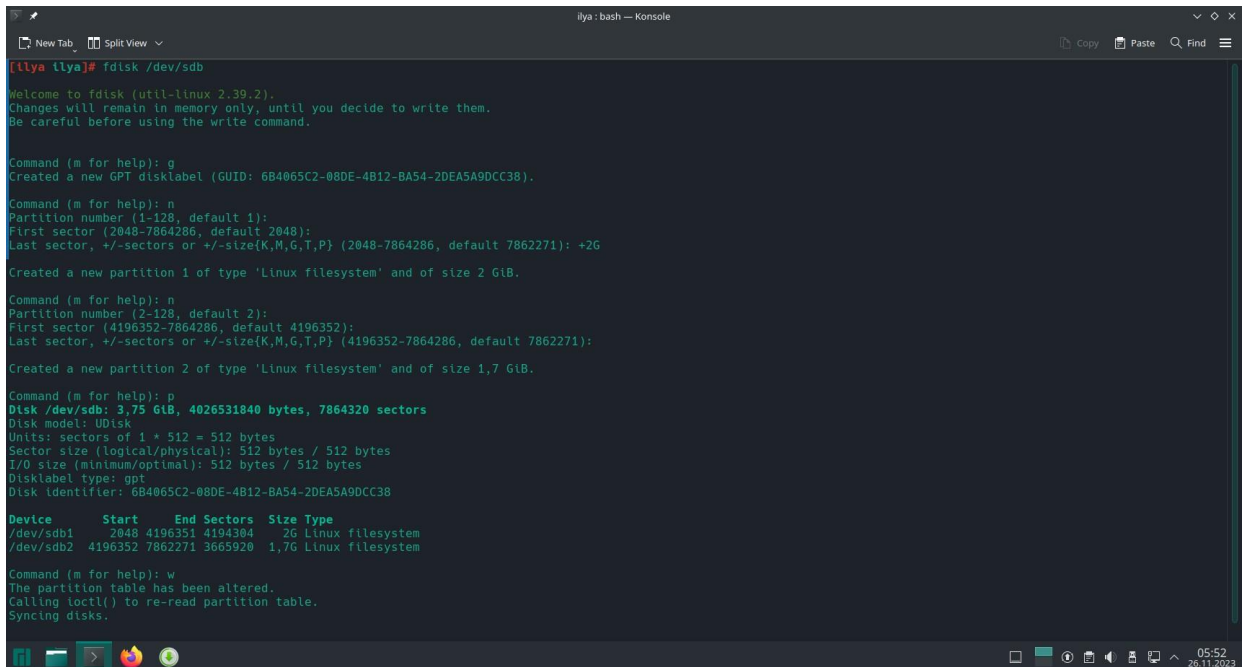
**Поддержка длинных имен файлов:** VFAT позволяет использовать длинные и понятные имена файлов, что упрощает организацию файлов и каталогов.

**Сохранение обратной совместимости:** VFAT остается обратно совместимой с более ранними версиями FAT, что означает, что устройства, отформатированные под VFAT, могут быть использованы на более старых компьютерах.

**Простота и эффективность:** VFAT относительно проста и легко поддерживается. Это делает ее привлекательной для внешних устройств, таких как USB-накопители и карты памяти.

Хотя VFAT несколько уступает более новым файловым системам, таким как NTFS, она продолжает играть важную роль в обеспечении совместимости и удобства использования носителей данных на разных платформах.

При помощи утилиты fdisk создать раздел и отформатировать его в заданной файловой системе (Рисунок 5.12-14).



```
[ilya ilya]# fdisk /dev/sdb

Welcome to fdisk (util-linux 2.39.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Command (m for help): g
Created a new GPT disklabel (GUID: 6B4065C2-08DE-4B12-BA54-2DEA5A9DCC38).

Command (m for help): n
Partition number (1-128, default 1):
First sector (2048-7864286, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-7864286, default 7862271): +2G

Created a new partition 1 of type 'Linux filesystem' and of size 2 GiB.

Command (m for help): n
Partition number (2-128, default 2):
First sector (4196352-7864286, default 4196352):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (4196352-7864286, default 7862271):

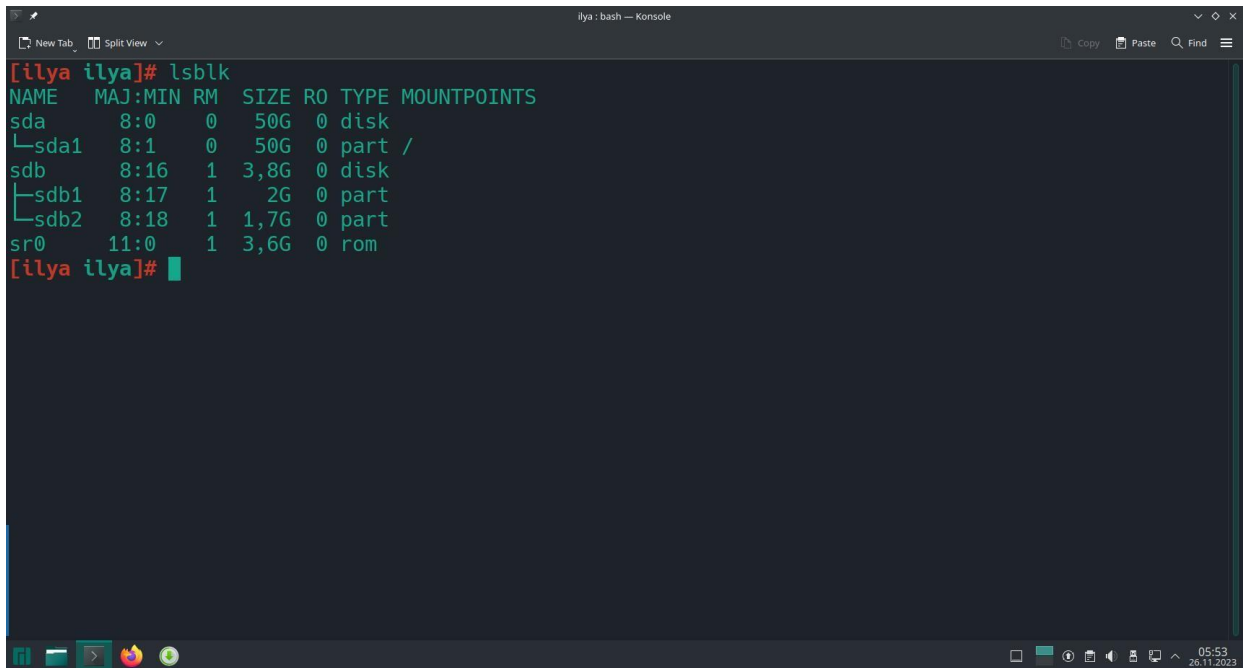
Created a new partition 2 of type 'Linux filesystem' and of size 1,7 GiB.

Command (m for help): p
Disk /dev/sdb: 3,75 GiB, 4026531840 bytes, 7864320 sectors
Disk model: UDisk
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 6B4065C2-08DE-4B12-BA54-2DEA5A9DCC38

Device      Start      End Sectors  Size Type
/dev/sdb1   2048 4196351 4194304    2G Linux filesystem
/dev/sdb2 4196352 7862271 3665920    1,7G Linux filesystem

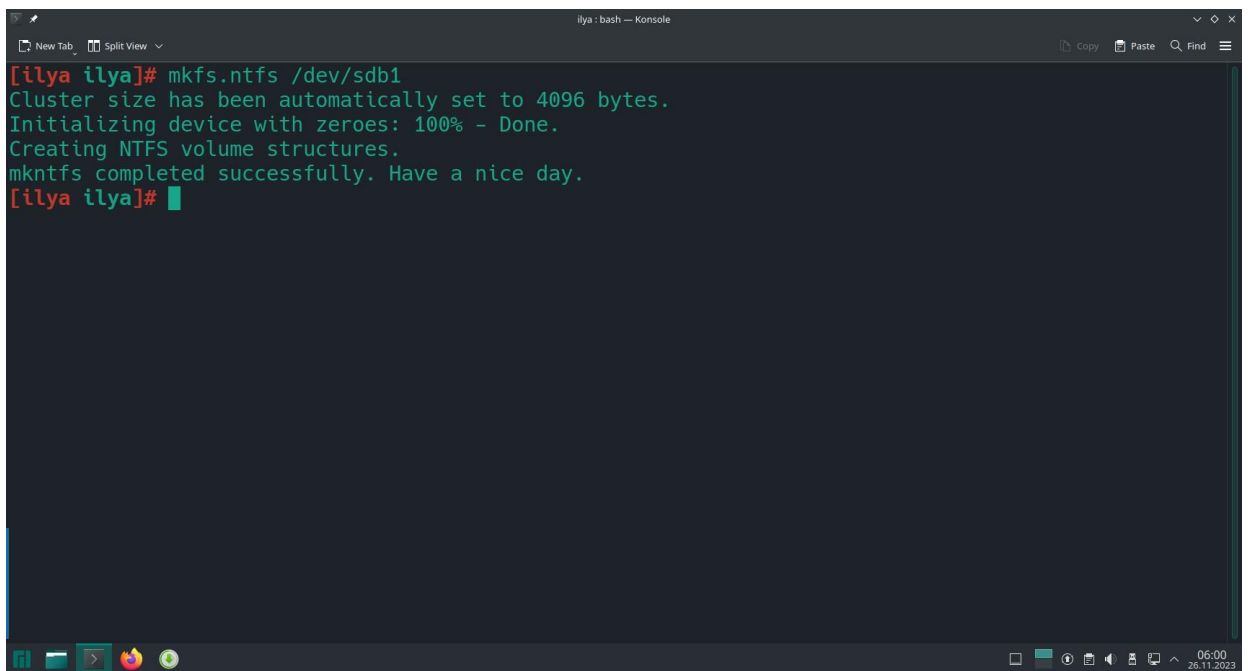
Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

Рисунок 5.12 – Создание раздела



```
[ilya ilya]# lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS
sda   8:0    0  50G  0 disk
└─sda1 8:1    0  50G  0 part /
sdb   8:16   1  3,8G  0 disk
├─sdb1 8:17   1    2G  0 part
└─sdb2 8:18   1  1,7G  0 part
sr0   11:0   1  3,6G  0 rom
```

Рисунок 5.13 – Завершение процесса создания

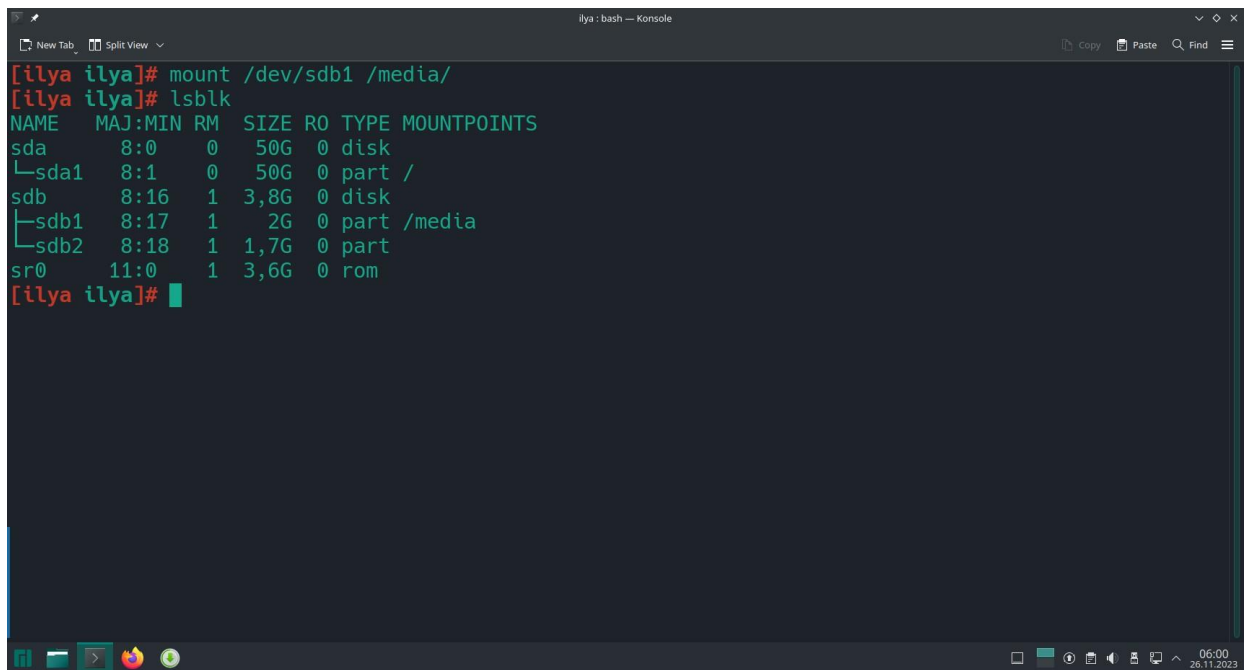


```
[ilya ilya]# mkfs.ntfs /dev/sdb1
Cluster size has been automatically set to 4096 bytes.
Initializing device with zeroes: 100% - Done.
Creating NTFS volume structures.
mkntfs completed successfully. Have a nice day.
[ilya ilya]#
```

Рисунок 5.14 – Форматирование раздела

Привести пример монтирования данной файловой системы командой mount (Рисунок 5.15).





The image shows a terminal window titled "ilya : bash — Konsole". The user has executed the command `mount /dev/sdb1 /media/` and then `lsblk`. The output of `lsblk` is as follows:

NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINTS
sda	8:0	0	50G	0	disk	
└─sda1	8:1	0	50G	0	part	/
sdb	8:16	1	3,8G	0	disk	
└─sdb1	8:17	1	2G	0	part	/media
└─sdb2	8:18	1	1,7G	0	part	
sr0	11:0	1	3,6G	0	rom	

The terminal prompt is now `[ilya ilya]#`.

Рисунок 5.15 – Результат монтирования

## 6 Установка программ в Linux

### 6.1 Установка программ с помощью графического интерфейса

Запустим менеджер приложений и выведем список всех приложений, доступных для нашей операционной системы (Рисунок 6.1).

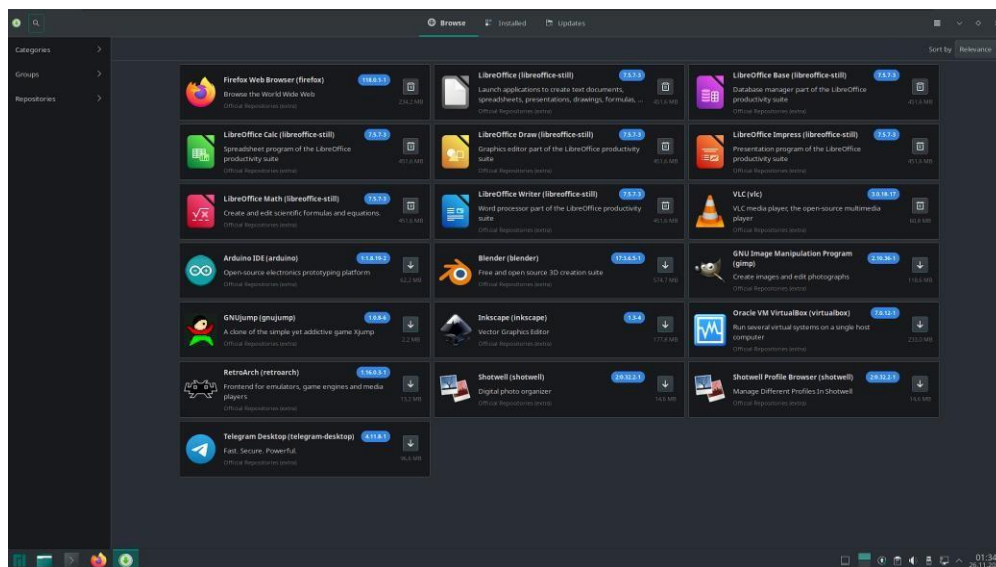


Рисунок 6.1 – Менеджер приложений

Далее заходим в поиск и вводим mc в поисковую строку (Рисунок 6.2).

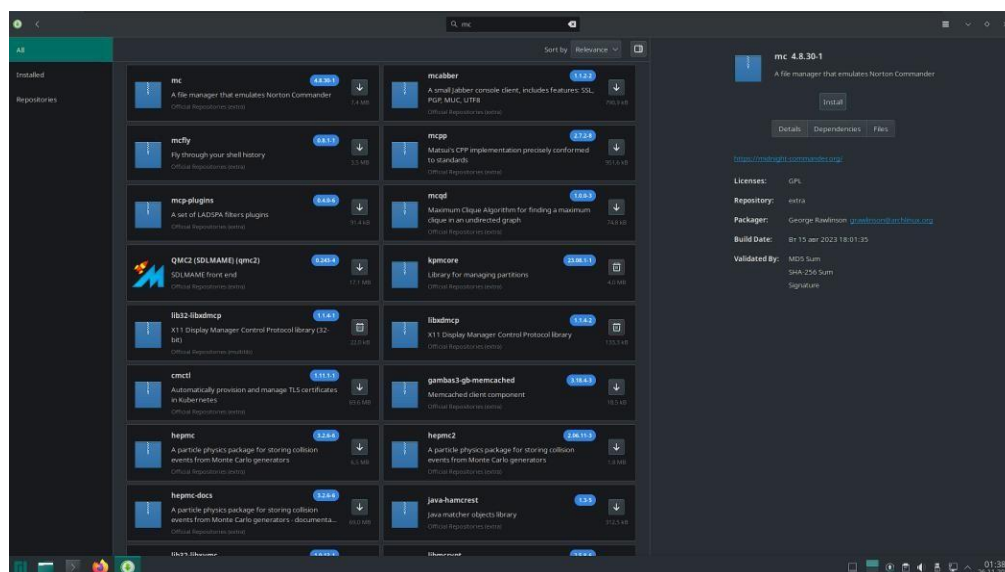


Рисунок 6.2 – Поиск MidnightCommander

Нажимаем установить. Менеджер приложений предлагает установить все зависимости, нажимаем принять и ждем окончание установки (Рисунок 6.3).

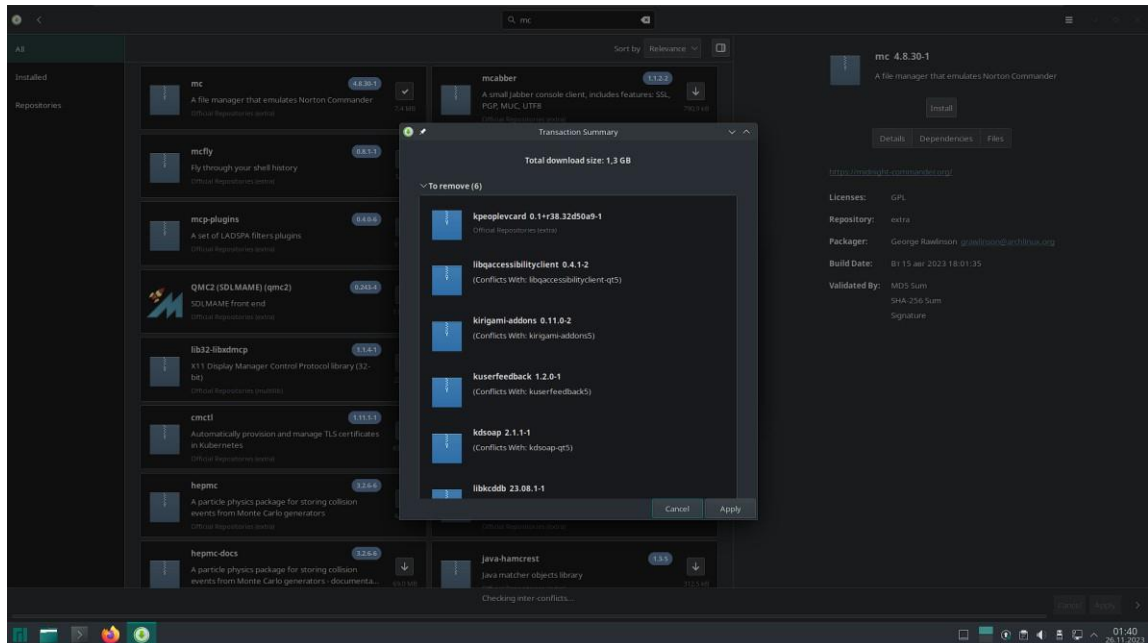


Рисунок 6.3 – Установка

Далее находим это приложение в меню пуск и запускаем. Как видно приложение спокойно запускается и работает исправно (Рисунок 6.4).

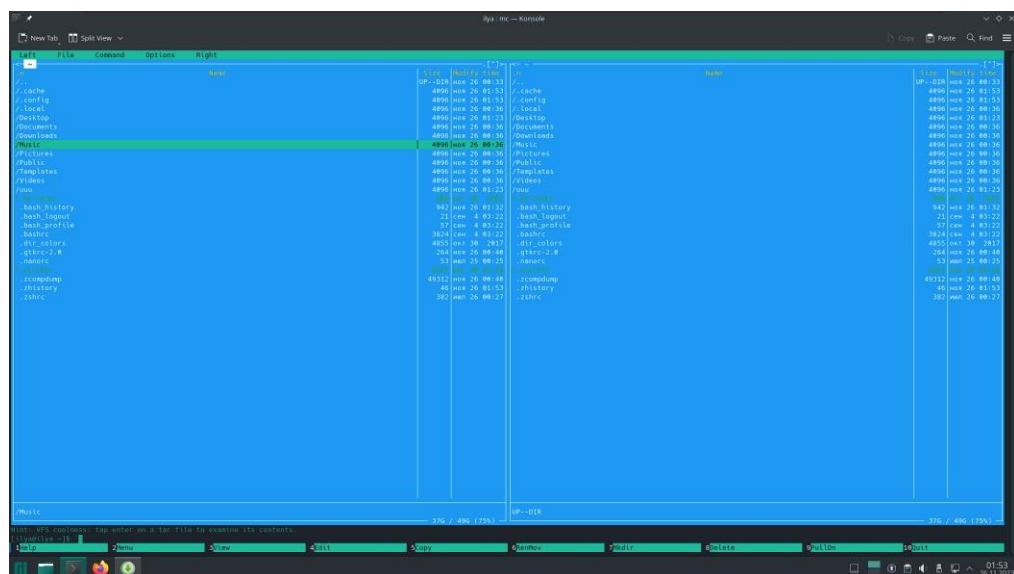


Рисунок 6.4 – Запуск «MidnightCommander»

## 6.2 Установка программ с использованием командной строки

Запускаем терминал и воспользуемся `pacman install mc` для установки нового приложения (Рисунок 6.5).

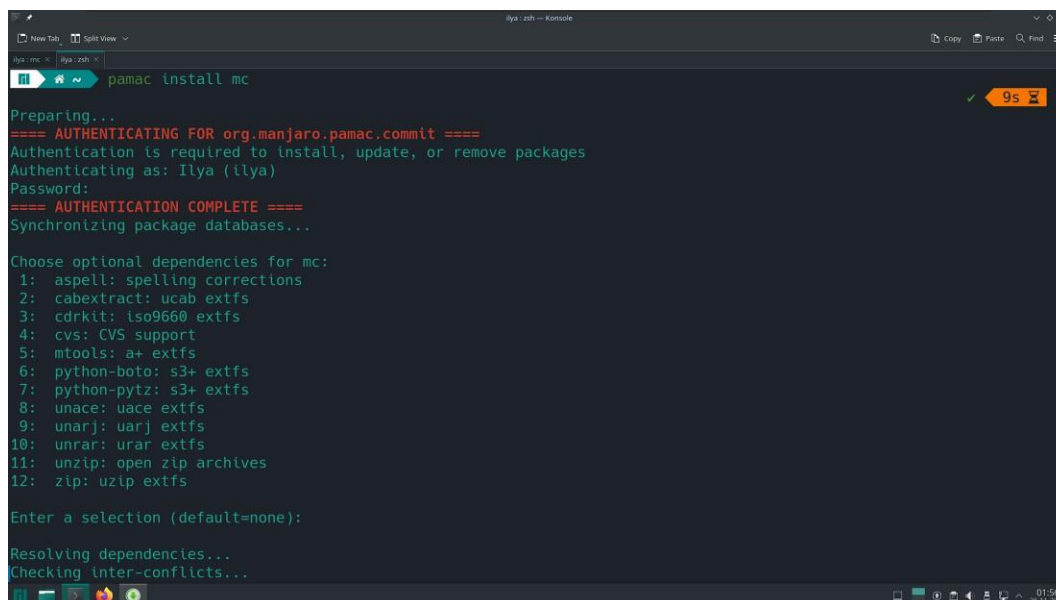


Рисунок 6.5 – Установка в терминале

Чтобы проверить установился ли MidnightCommander мы введем в терминал команду `mc` и увидим интерфейс (Рисунок 6.6).

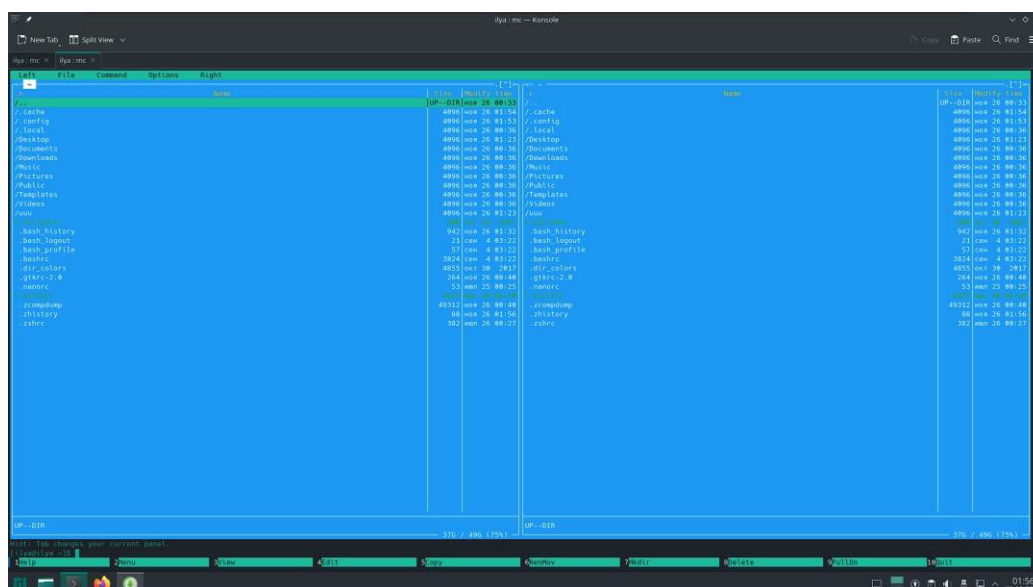


Рисунок 6.6 – MidnightCommander

## 6.3 Установка программ из исходных кодов

Из исходников мы будем восстанавливать приложение TestDisk, для этого мы переходим на официальный сайт и скачиваем файл в формате tar.bz2 (Рисунок 6.7).

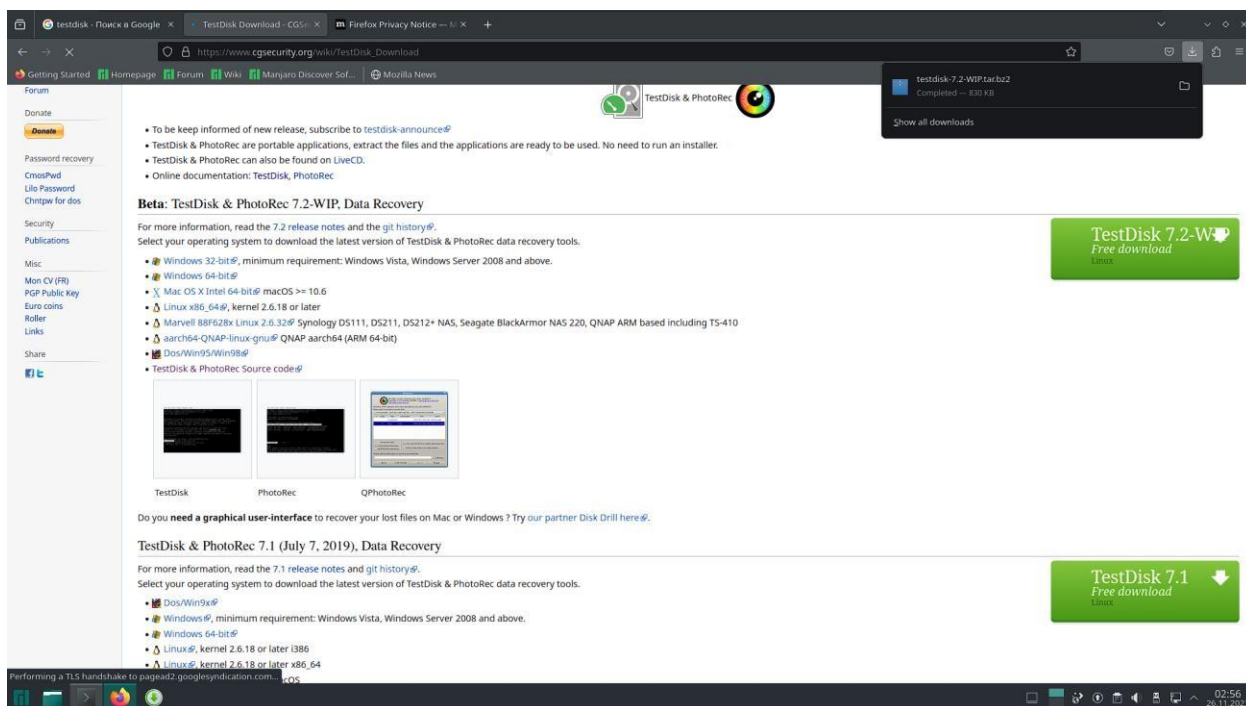


Рисунок 6.7 – Скачивание tar.bz2 архива

Далее мы распаковываем архив (Рисунок 6.8). И переходим в распакованную директорию.

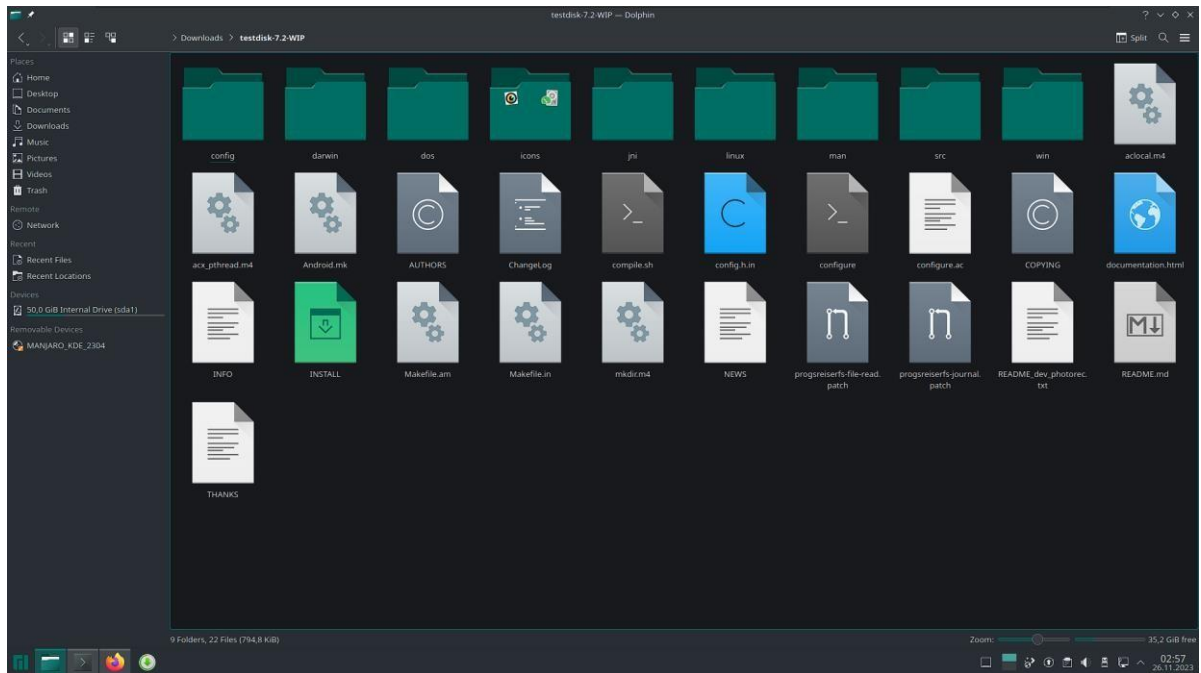


Рисунок 6.8 – Распаковка исходников

Далее мы сконфигурируем нашу программу для этого мы пропишем следующее (Листинг 6.1). Запускаем конфигурацию и ждем окончание установки (Рисунок 6.9).

Листинг 6.1 – Конфигурация

```
sudo ./configure
```

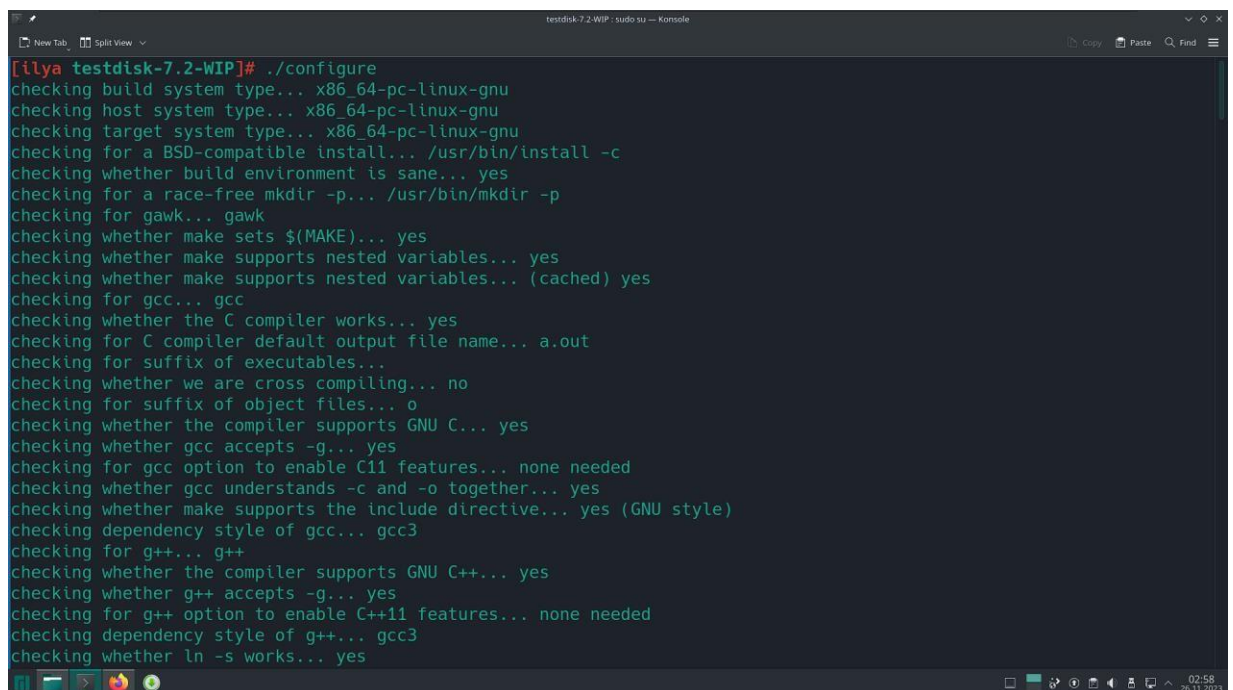
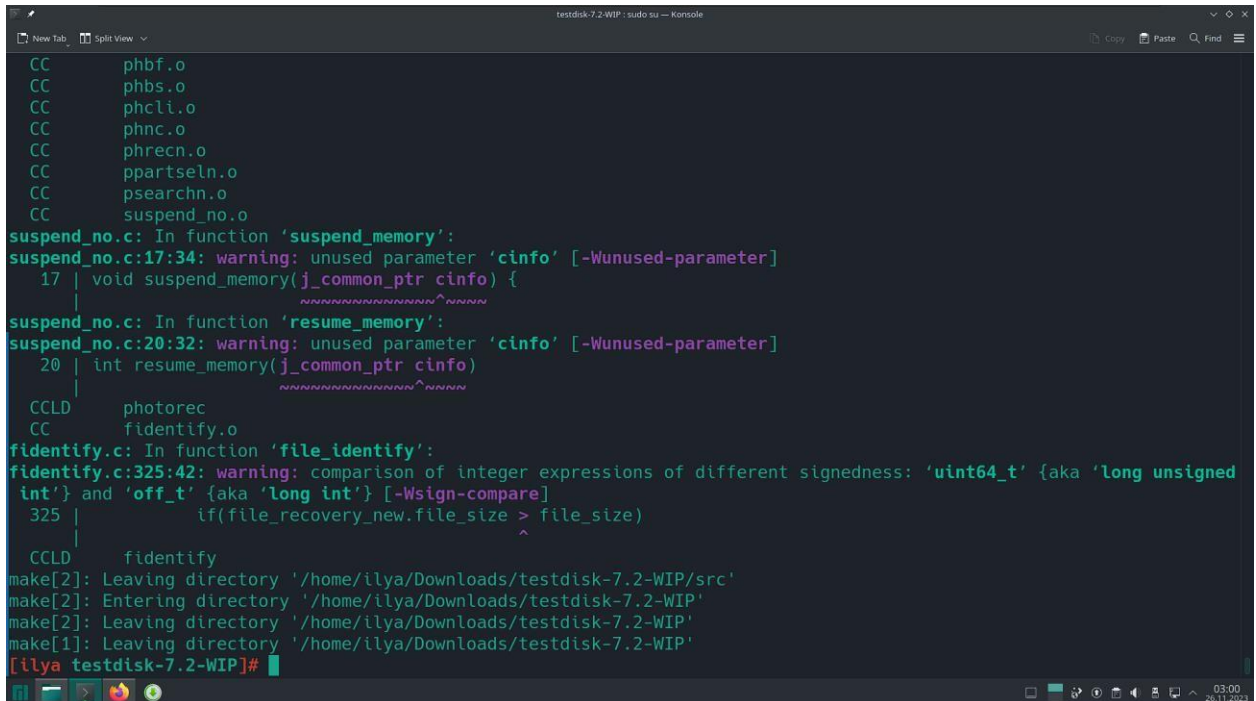


Рисунок 6.9 – Конфигурация

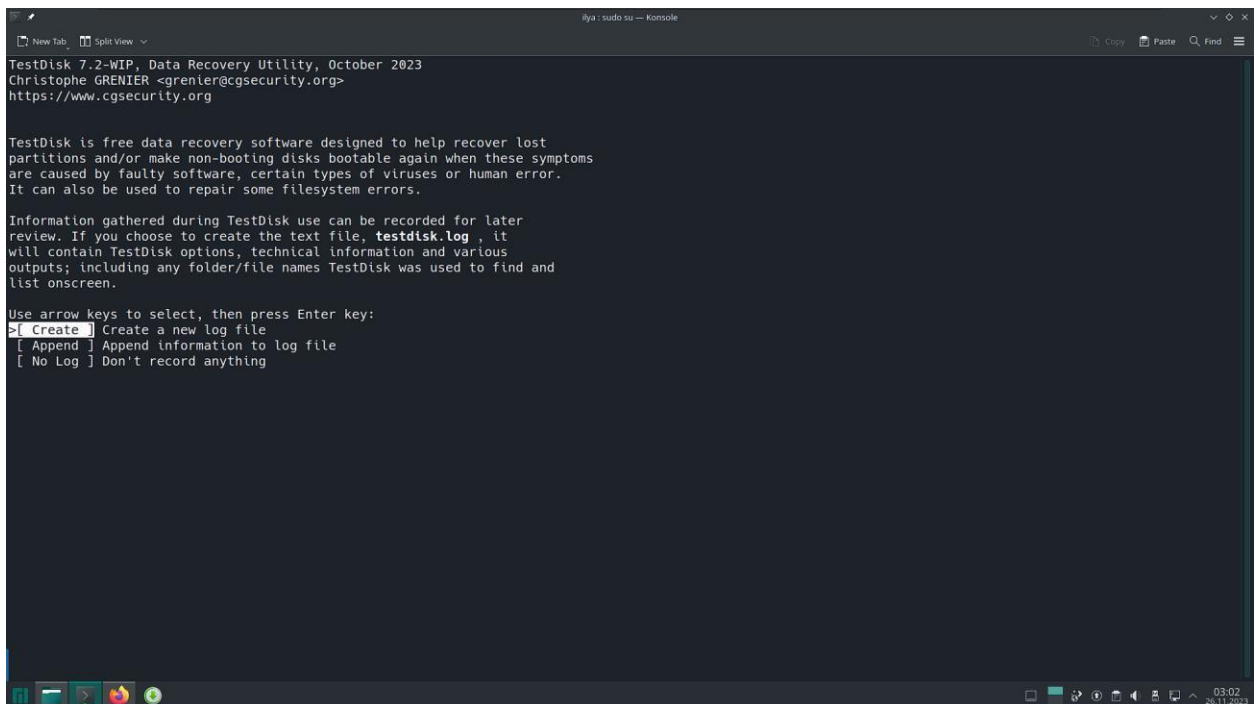


Manjaro Linux сам установит все зависимости. После запускаем команду `make` и `make install` (Рисунок 6.10). После чего мы сможем запустить TestDisk (Рисунок 6.11).



```
CC      phbf.o
CC      phbs.o
CC      phcli.o
CC      phnc.o
CC      phrecln.o
CC      ppartseln.o
CC      psearchn.o
CC      suspend_no.o
suspend_no.c: In function 'suspend_memory':
suspend_no.c:17:34: warning: unused parameter 'cinfo' [-Wunused-parameter]
   17 | void suspend_memory(j_common_ptr cinfo) {
      |                                ^~~~~~
suspend_no.c: In function 'resume_memory':
suspend_no.c:20:32: warning: unused parameter 'cinfo' [-Wunused-parameter]
   20 | int resume_memory(j_common_ptr cinfo)
      |                        ^~~~~~
CCLD    photorec
CC      fidentify.o
fidentify.c: In function 'file_identify':
fidentify.c:325:42: warning: comparison of integer expressions of different signedness: 'uint64_t' {aka 'long unsigned int'} and 'off_t' {aka 'long int'} [-Wsign-compare]
   325 |         if(file_recovery_new.file_size > file_size)
      |                                     ^
CCLD    fidentify
make[2]: Leaving directory '/home/ilya/Downloads/testdisk-7.2-WIP/src'
make[2]: Entering directory '/home/ilya/Downloads/testdisk-7.2-WIP'
make[2]: Leaving directory '/home/ilya/Downloads/testdisk-7.2-WIP'
make[1]: Leaving directory '/home/ilya/Downloads/testdisk-7.2-WIP'
[ilya testdisk-7.2-WIP]#
```

Рисунок 6.10 – Команды make



```
TestDisk 7.2-WIP, Data Recovery Utility, October 2023
Christophe GRENIER <grenier@cgsecurity.org>
https://www.cgsecurity.org

TestDisk is free data recovery software designed to help recover lost
partitions and/or make non-booting disks bootable again when these symptoms
are caused by faulty software, certain types of viruses or human error.
It can also be used to repair some filesystem errors.

Information gathered during TestDisk use can be recorded for later
review. If you choose to create the text file, testdisk.log, it
will contain TestDisk options, technical information and various
outputs; including any folder/file names TestDisk was used to find and
list onscreen.

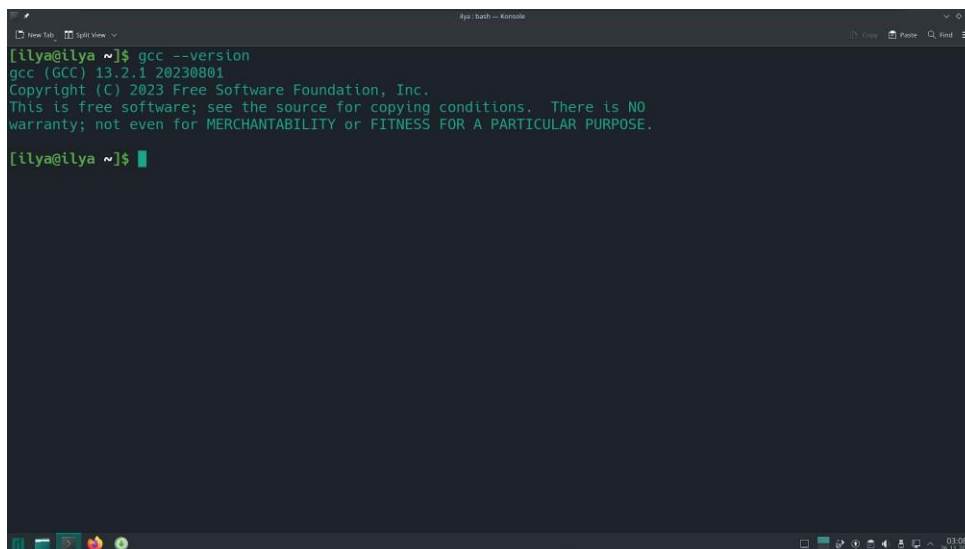
Use arrow keys to select, then press Enter key:
> [ Create ] Create a new log file
  [ Append ] Append information to log file
  [ No Log ] Don't record anything
```

Рисунок 6.11 – Собранное приложение

## 7 Программирование в Linux

### 7.1 Создание консольного приложения

В первую очередь проверим установлен ли у нас компилятор gcc (Рисунок 7.1). Как мы видим он установлен.

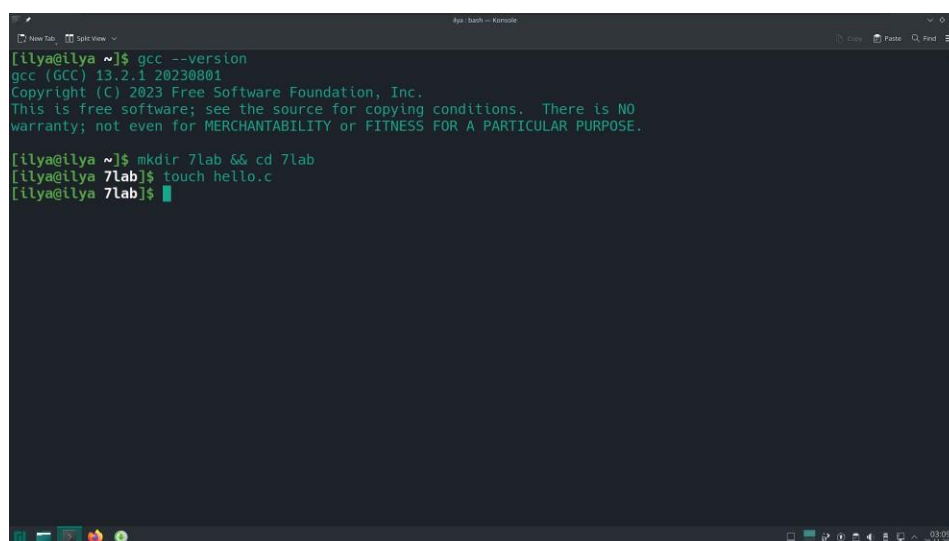


```
[ilya@ilya ~]$ gcc --version
gcc (GCC) 13.2.1 20230801
Copyright (C) 2023 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

[ilya@ilya ~]$
```

Рисунок 7.1 – Проверка наличия gcc

Создаем директорию с названием «7lab» и создаем в ней файл hello.c (Рисунок 7.2). И вписываем в него следующий код (Листинг 7.1).



```
[ilya@ilya ~]$ gcc --version
gcc (GCC) 13.2.1 20230801
Copyright (C) 2023 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

[ilya@ilya ~]$ mkdir 7lab && cd 7lab
[ilya@ilya 7lab]$ touch hello.c
[ilya@ilya 7lab]$
```

Рисунок 7.2 – Создание файла



## Листинг 7.1 – Hello world!

```
#include <stdio.h>

int
main (void)
{
    printf ("Hello, world!\n");
    return 0;
}
```

Компилируем данную программу и запускаем (Рисунок 7.3).

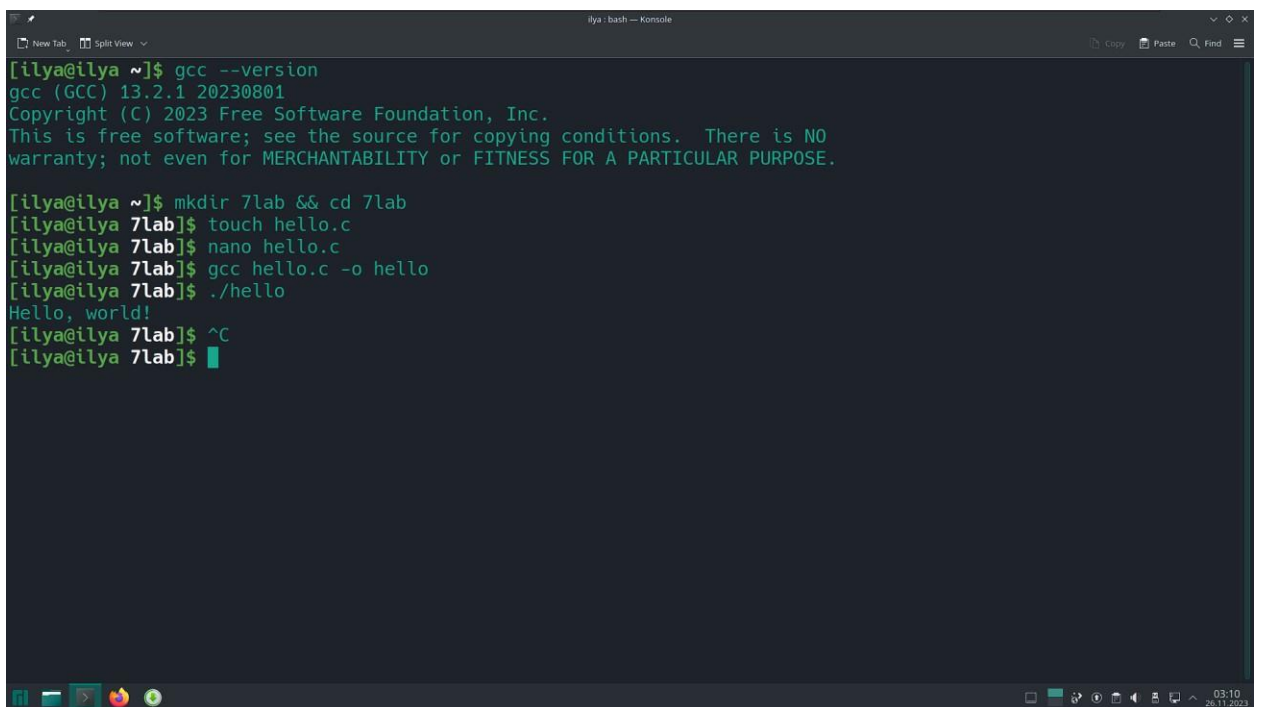
A screenshot of a terminal window titled 'ilya: bash — Konsole'. The terminal shows the following commands and output:   
[ilya@ilya ~]\$ gcc --version   
gcc (GCC) 13.2.1 20230801   
Copyright (C) 2023 Free Software Foundation, Inc.   
This is free software; see the source for copying conditions. There is NO   
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.   
[ilya@ilya ~]\$ mkdir 7lab && cd 7lab   
[ilya@ilya 7lab]\$ touch hello.c   
[ilya@ilya 7lab]\$ nano hello.c   
[ilya@ilya 7lab]\$ gcc hello.c -o hello   
[ilya@ilya 7lab]\$ ./hello   
Hello, world!   
[ilya@ilya 7lab]\$ ^C   
[ilya@ilya 7lab]\$   
The terminal window has a dark background with green text. At the bottom, there is a taskbar with various icons and a system clock showing 03:10 on 26.11.2023.

Рисунок 7.3 – Компиляция и запуск

## 7.2 Ввод-вывод с помощью консоли

Создаем файл `input_output.c`, где создадим приложение для хранения и вывода данных в консоль. Код предоставлен ниже (Листинг 7.2).

### Листинг 7.2 – Код файла `input_output.c`

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

// Функция для ввода строки с консоли с проверкой длины
char* input_string(int max_len) {
    char* str = (char*)malloc(max_len + 1); // Выделяем память для
```

```

строки
    if (str == NULL) { // Проверяем, что память выделена успешно
        printf("Ошибка выделения памяти\n");
        exit(1);
    }
    fgets(str, max_len + 1, stdin); // Считываем строку с консоли
    int len = strlen(str); // Определяем длину строки
    if (len == max_len && str[len - 1] != '\n') { // Проверяем, что
строка не превышает максимальную длину
        printf("Введенная строка слишком длинная. Повторите ввод.\n");
        while (getchar() != '\n'); // Очищаем буфер ввода
        return input_string(max_len); // Рекурсивно вызываем функцию
заново
    }
    else {
        str[len - 1] = '\0'; // Убираем символ переноса строки из
конца строки
        return str; // Возвращаем строку
    }
}

// Функция для ввода целого числа с консоли с проверкой корректности
int input_int() {
    char* str = input_string(10); // Вводим строку с консоли
    for (int i = 0; i < strlen(str); i++) { // Проверяем, что все
символы строки являются цифрами
        if (!isdigit(str[i])) {
            printf("Введенная строка не является целым числом.
Повторите ввод.\n");
            return input_int(); // Рекурсивно вызываем функцию заново
        }
    }
    int num = atoi(str); // Преобразуем строку в целое число
    free(str); // Освобождаем память из-под строки
    return num; // Возвращаем число
}

// Функция для вывода на консоль текстовых и числовых данных
void output_data(char* text, int num) {
    printf("Введенный текст: %s\n", text); // Выводим текст на консоль
    printf("Введенное число: %d\n", num); // Выводим число на консоль
}

// Функция для вывода главного меню на консоль и выбора пункта меню
пользователем
int main_menu() {
    printf("Главное меню:\n");
    printf("1. Ввести данные\n");
    printf("2. Вывести данные\n");
    printf("3. Выход\n");
    printf("Выберите пункт меню: ");
    int choice = input_int(); // Вводим выбор пользователя с консоли
    if (choice < 1 || choice > 3) { // Проверяем, что выбор
пользователя в допустимом диапазоне
        printf("Неверный выбор. Повторите ввод.\n");
        return main_menu(); // Рекурсивно вызываем функцию заново
    }
    else {

```

```

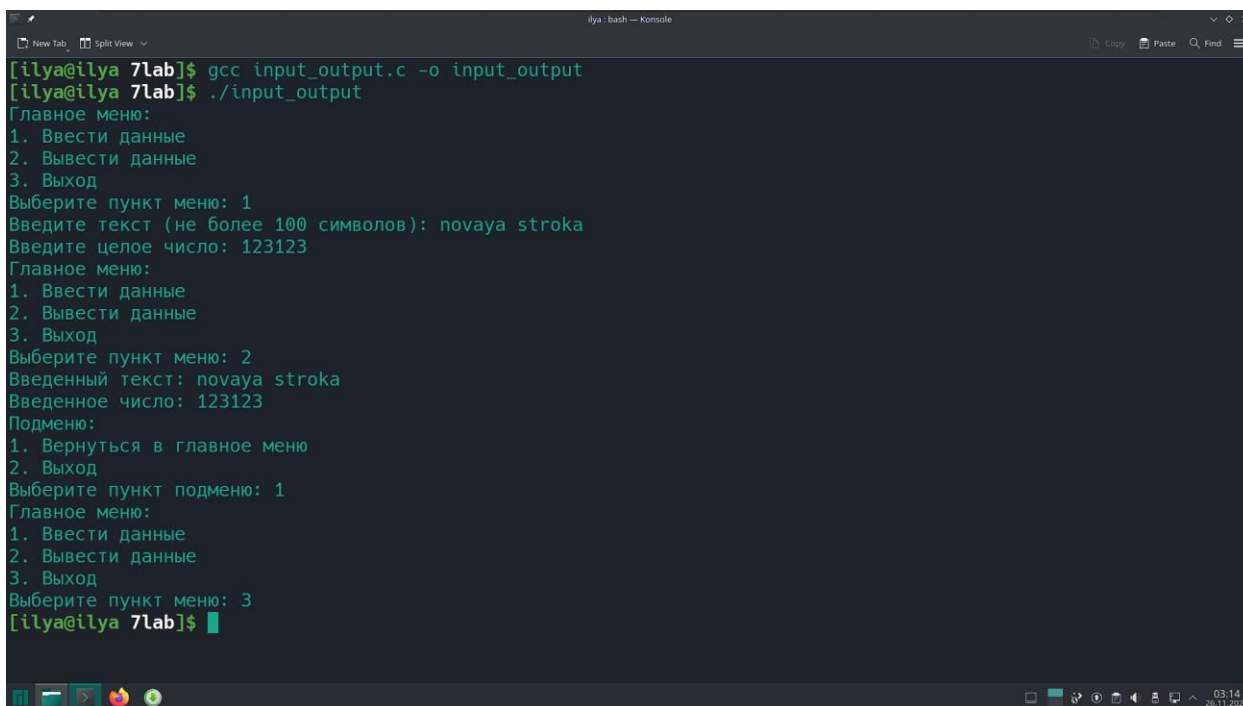
        return choice; // Возвращаем выбор пользователя
    }
}

// Функция для вывода подменю на консоль и выбора пункта подменю
пользователем
int sub_menu() {
    printf("Подменю:\n");
    printf("1. Вернуться в главное меню\n");
    printf("2. Выход\n");
    printf("Выберите пункт подменю: ");
    int choice = input_int(); // Вводим выбор пользователя с консоли
    if (choice < 1 || choice > 2) { // Проверяем, что выбор
пользователя в допустимом диапазоне
        printf("Неверный выбор. Повторите ввод.\n");
        return sub_menu(); // Рекурсивно вызываем функцию заново
    }
    else {
        return choice; // Возвращаем выбор пользователя
    }
}

// Главная функция программы
int main() {
    char* text = NULL; // Переменная для хранения текстовых данных
    int num = 0; // Переменная для хранения числовых данных
    int choice = 0; // Переменная для хранения выбора пользователя
    int exit = 0; // Переменная для хранения флага выхода из программы
    while (!exit) { // Пока не выходим из программы
        choice = main_menu(); // Выводим главное меню и получаем выбор
пользователя
        switch (choice) { // Обрабатываем выбор пользователя
            case 1: // Если пользователь выбрал ввести данные
                printf("Введите текст (не более 100 символов): ");
                text = input_string(100); // Вводим текст с консоли
                printf("Введите целое число: ");
                num = input_int(); // Вводим число с консоли
                break;
            case 2: // Если пользователь выбрал вывести данные
                output_data(text, num); // Выводим данные на консоль
                choice = sub_menu(); // Выводим подменю и получаем
выбор пользователя
                if (choice == 2) { // Если пользователь выбрал выйти
из программы
                    exit = 1; // Устанавливаем флаг выхода в 1
                }
                break;
            case 3: // Если пользователь выбрал выйти из программы
                exit = 1; // Устанавливаем флаг выхода в 1
                break;
        }
    }
    free(text); // Освобождаем память из-под текста
    return 0; // Завершаем программу с кодом 0
}

```

Далее мы проверим работоспособность нашей программы (Рисунок 7.4).



```
[ilya@ilya 7lab]$ gcc input_output.c -o input_output
[ilya@ilya 7lab]$ ./input_output
Главное меню:
1. Ввести данные
2. Вывести данные
3. Выход
Выберите пункт меню: 1
Введите текст (не более 100 символов): novaya stroka
Введите целое число: 123123
Главное меню:
1. Ввести данные
2. Вывести данные
3. Выход
Выберите пункт меню: 2
Введенный текст: novaya stroka
Введенное число: 123123
Подменю:
1. Вернуться в главное меню
2. Выход
Выберите пункт подменю: 1
Главное меню:
1. Ввести данные
2. Вывести данные
3. Выход
Выберите пункт меню: 3
[ilya@ilya 7lab]$
```

Рисунок 7.4 – Проверка работоспособности программы

### 7.3 Передача параметров программе

Необходимо создать программу, в которую можно передавать аргументы. Наша программа будет перемножать два числа. Код предоставлен ниже (Листинг 7.3).

Листинг 7.3 – Код файла args.c

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

// Функция для считывания параметров из файла конфигурации
void read_config_file(char* file_name, int* a, int* b) {
    FILE* file = fopen(file_name, "r"); // Открываем файл для чтения
    if (file == NULL) { // Проверяем, что файл открыт успешно
        printf("Ошибка открытия файла %s\n", file_name);
        exit(1);
    }
    fscanf(file, "%d %d", a, b); // Считываем два целых числа из файла
    fclose(file); // Закрываем файл
}
```

```

// Функция для записи отладочного вывода в файл
void write_debug_file(char* file_name, int a, int b, int c) {
    FILE* file = fopen(file_name, "a"); // Открываем файл для дозаписи
    if (file == NULL) { // Проверяем, что файл открыт успешно
        printf("Ошибка открытия файла %s\n", file_name);
        exit(1);
    }
    fprintf(file, "Возведение %d в степень %d дает %d\n", a, b, c); //
    Записываем результат в файл
    fclose(file); // Закрываем файл
}

// Главная функция программы
int main(int argc, char* argv[]) {
    int a = 0; // Переменная для хранения основания
    int b = 0; // Переменная для хранения степени
    int c = 0; // Переменная для хранения возведения в степень

    if (argc == 3) { // Если переданы два параметра через имя
        программы из командной строки
        a = atoi(argv[1]); // Преобразуем первый параметр в целое
        число и присваиваем его переменной a
        b = atoi(argv[2]); // Преобразуем второй параметр в целое
        число и присваиваем его переменной b
    }
    else if (argc == 2) { // Если передан один параметр через имя
        программы из командной строки
        read_config_file(argv[1], &a, &b); // Считываем два числа из
        файла конфигурации и присваиваем их переменным a и b
    }
    else { // Если не переданы параметры через имя программы из
        командной строки
        printf("Неверное количество параметров. Используйте следующий
        формат:\n");
        printf("%s <число> <число>\n", argv[0]); // Выводим формат для
        передачи двух чисел через имя программы из командной строки
        printf("%s <имя_файла>\n", argv[0]); // Выводим формат для
        передачи имени файла конфигурации через имя программы из командной
        строки
        exit(1);
    }

    c = pow(a, b); // Возведение в степень

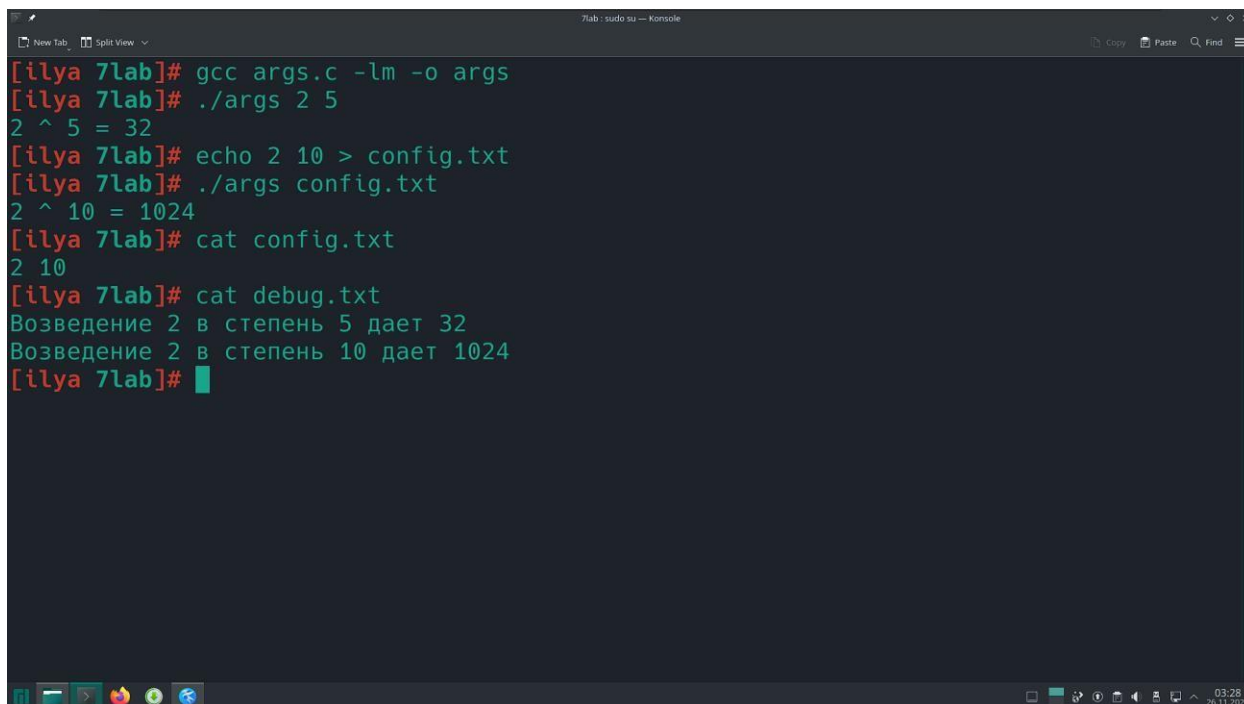
    printf("%d ^ %d = %d\n", a, b, c); // Выводим результат на консоль

    write_debug_file("debug.txt", a, b, c); // Записываем отладочный
    вывод в файл

    return 0; // Завершаем программу с кодом 0
}

```

Запускаем программу и передаем в нее нужные аргументы, и как мы видим все работает корректно (Рисунок 7.5).



```
[ilya 7lab]# gcc args.c -lm -o args
[ilya 7lab]# ./args 2 5
2 ^ 5 = 32
[ilya 7lab]# echo 2 10 > config.txt
[ilya 7lab]# ./args config.txt
2 ^ 10 = 1024
[ilya 7lab]# cat config.txt
2 10
[ilya 7lab]# cat debug.txt
Возведение 2 в степень 5 дает 32
Возведение 2 в степень 10 дает 1024
[ilya 7lab]#
```

Рисунок 7.5 – Проверка работоспособности программы

## Список использованных источников

- 1 И.А. Трещев, Г.Ф. Вильдяйкин, И.А. Кожин Безопасность операционных систем. Часть 1. Raid, восстановление файлов, metasploit // Издательские решения 2020 - 140с.
- 2 И.А. Трещев, С.В. Прокофьев. Безопасность операционных систем. Часть 2. Операционные системы, уязвимости. // Издательские решения 2021 - 262с.
- 3 И.А. Трещев Анализ защищенности распределенных информационных систем. // Издательские решения 2020 - 102с.
- 4 В.А. Тихомиров Операционные системы. Ч. 2. Операционные системы защищенного режима работы процессора: Учеб. пособие. – Комсомольск-на-Амуре: ГОУВПО «КнАГТУ», 2003. - 206 с.
- 5 А.А. Хусаинов, Н.Н. Михайлова Архитектура вычислительных систем: Учеб. пособие / А.А. Хусаинов, Н.Н. Михайлова. – Комсомольск-на-Амуре: Государственное образовательное учреждение высшего профессионального образования «Комсомольский-на-Амуре гос. техн. ун-т», 2007. – 123 с.