

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра МО ЭВМ**

**ОТЧЕТ**  
**по лабораторной работе №2**  
**по дисциплине «Операционные системы»**  
**Тема: Файловые системы Unix-подобных ОС**

Студентка гр. 1381

\_\_\_\_\_

Рымарь М.И.

Преподаватель

\_\_\_\_\_

Душутина Е.В.

Санкт-Петербург

2023

## **Цель работы.**

Проанализировать функциональное назначение структурных элементов дерева файловой системы. Определить размещение корневого каталога (корневой файловой системы).

## **Задание.**

1. Ознакомиться с типами файлов исследуемой ФС. Применяя утилиту `ls`, отфильтровать по одному примеру каждого типа файла используемой вами ФС. Комбинируя различные ключи утилиты рекурсивно просканировать все дерево, анализируя крайнюю левую позицию выходной информации полученной посредством `ls -l`. Результат записать в выходной файл с указанием полного пути каждого примера. Выполнить задание сначала в консоли построчно, выбирая необходимые сочетания ключей (в командной строке), а затем оформить как скрипт с задаваемым в командной строке именем файла как параметр.

2. Получить все жесткие ссылки на заданный файл, находящиеся в разных каталогах пользовательского пространства (разными способами, не применяя утилиты `file` и `find`). Использовать конвейеризацию и фильтрацию. Оформить в виде скрипта.

3. Проанализировать все возможные способы формирования символьных ссылок (`ln`, `link`, `cp` и т.д.), продемонстрировать их экспериментально. Предложить скрипт, подсчитывающий и перечисляющий все полноименные символьные ссылки на файл, размещаемые в разных местах файлового дерева.

4. Получить все символьные ссылки на заданный в качестве входного параметра файл, не используя `file` (разными способами, не применяя утилиту `file`).

5. Изучить утилиту `find`, используя ее ключи получить расширенную информацию о всех типах файлов. Создать примеры вложенных команд.

6. Проанализировать содержимое заголовка файла, а также файла-каталога с помощью утилит `od` и `*dump`. Если доступ к файлу-каталогу возможен (для отдельных модификаций POSIX-совместимых ОС), проанализировать

изменение его содержимого при различных операциях над элементами, входящими в его состав (файлами и подкаталогами).

7. Определить максимальное количество записей в каталоге. Изменить размер каталога, варьируя количество записей (для этого создать программу, порождающую новые файлы и каталоги, а затем удаляющую их, предусмотрев промежуточный и конечный вывод информации о размере подопытного каталога).

8. Ознакомиться с содержимым `/etc/passwd`, `/etc/shadow`, с утилитой `/usr/bin/passwd`, проанализировать права доступа к этим файлам.

9. Исследовать права владения и доступа, а также их сочетаемость

9.1. Привести примеры применения утилит `chmod`, `chown` к специально созданному для этих целей отдельному каталогу с файлами.

9.2. Расширить права исполнения экспериментального файла с помощью флага `SUID`.

9.3. Экспериментально установить, как формируются итоговые права на использование файла, если права пользователя и группы, в которую он входит, различны.

9.4. Сопоставить возможности исполнения наиболее часто используемых операций, варьируя правами доступа к файлу и каталогу.

10. Разработать «программу-шлюз» для доступа к файлу другого пользователя при отсутствии прав на чтение информации из этого файла. Провести эксперименты для случаев, когда пользователи принадлежат одной и разным группам. Сравнить результаты.

Для выполнения задания применить подход, аналогичный для обеспечения функционирования утилиты `/usr/bin/passwd` (манипуляции с правами доступа, флагом `SUID`, а также размещением файлов).

11. Применяя утилиту `df` и аналогичные ей по функциональности утилиты, а также информационные файлы типа `fstab`, получить информацию о файловых системах, возможных для монтирования, а также установленных на компьютере реально.

11.1. Привести информацию об исследованных утилитах и информационных файлах с анализом их содержимого и форматов.

11.2. Привести образ диска с точки зрения состава и размещения всех ФС на испытуемом компьютере, а также образ полного дерева ФС, включая присоединенные ФС съемных и несъемных носителей. Проанализировать и указать формат таблицы монтирования.

11.3. Привести «максимально возможное» дерево ФС, проанализировать, где это указывается.

12. Проанализировать и пояснить принцип работы утилиты file.

12.1. Привести алгоритм её функционирования на основе информационной базы, размещение и полное имя которой указывается в описании утилиты в технической документации ОС (как правило, /usr/share/file/magic.\*), а также содержимого заголовка файла, к которому применяется утилита. Определить, где находятся магические числа и иные характеристики, идентифицирующие тип файла, применительно к исполняемым файлам, а также файлам других типов.

12.2. Утилиту file выполнить с разными ключами.

12.3. Привести экспериментальную попытку с добавлением в базу собственного типа файла и его дальнейшей идентификацией. Описать эксперимент и привести последовательность действий для расширения функциональности утилиты file и возможности встраивания дополнительного типа файла в ФС (согласовать содержимое информационной базы и заголовка файла нового типа).

### **Выполнение работы.**

1. Поиск файлов каждого типа был произведён с помощью команды:

```
ls -l | grep -E '^-[r, w, x, -]{9}'
```

После ^ указываются -,b,c,d,l,p,s в зависимости от того, какой тип файла нужен. На рисунках 1-5 показаны примеры работы такого конвейера.

```

rymary@rymary-VirtualBox:/bin$ ls -l | grep -E '^-[r, w, x, -]{9}'
-rwxr-xr-x 1 root root 1113504 Apr 18 2022 bash
-rwxr-xr-x 1 root root 748968 Aug 29 2018 brlty
-rwxr-xr-x 1 root root 34888 Jul 4 2019 bunzip2
-rwxr-xr-x 1 root root 2062296 Sep 18 2020 busybox

rymary@rymary-VirtualBox:/dev$ ls -l | grep -E '^b[r, w, x, -]{9}'
brw-rw---- 1 root disk 7, 0 Apr 17 06:10 loop0
brw-rw---- 1 root disk 7, 1 Apr 17 06:10 loop1
brw-rw---- 1 root disk 7, 10 Apr 17 06:10 loop10
brw-rw---- 1 root disk 7, 11 Apr 17 06:10 loop11

rymary@rymary-VirtualBox:/dev$ ls -l | grep -E '^c[r, w, x, -]{9}'
crw-r--r-- 1 root root 10, 235 Apr 17 06:10 autofs
crw----- 1 root root 10, 234 Apr 17 06:09 btrfs-control
crw----- 1 root root 5, 1 Apr 17 06:10 console
crw----- 1 root root 10, 59 Apr 17 06:10 cpu_dma_latency

rymary@rymary-VirtualBox:/dev$ ls -l | grep -E '^d[r, w, x, -]{9}'
drwxr-xr-x 2 root root 560 Apr 17 06:13 block
drwxr-xr-x 2 root root 80 Apr 17 06:09 bsg
drwxr-xr-x 3 root root 60 Apr 17 06:09 bus
drwxr-xr-x 2 root root 3660 Apr 17 06:09 char

rymary@rymary-VirtualBox:/run$ ls -l | grep -E '^s[r, w, x, -]{9}'
srw-rw-rw- 1 root root 0 Apr 17 06:09 acpid.socket
srw-rw-rw- 1 root root 0 Apr 17 06:09 snapd-snap.socket
srw-rw-rw- 1 root root 0 Apr 17 06:09 snapd.socket

```

#### Рисунки 1-5 – Примеры поиска файлов различных типов

С помощью команды `readlink -f` можно найти полное имя некоторого файла. Приведу примеры файлов с необходимым типом:

- 1). Специальный файл блочного устройства: `/dev/loop10`
- 2). Сокет: `/run/acpid.socket`
- 3). Регулярный файл: `/dev/vhost-net`
- 4). Директория: `/home/rymar`
- 5). FIFO: `/run/initctl`
- 6). Файл символического устройства: `/dev/vfio`

Был написан скрипт, который выводит все типы файлов, которые есть в введённой директории. На рисунке 6 представлен скрипт `1.sh`, на рисунке 7 – работа скрипта в директории `/bin`.

```

rymarmary@rymarmary-VirtualBox:~$ cat 1.sh
#!/bin/bash

types='- c d l s b p' # все типы файлов
for i in $types # цикл по всем типам
do
    echo $i # вывод некоторого типа
    file_search=`ls -lR $1 | grep ^$i | head -1` # поиск файла некоторого типа
и вывод первого попавшегося
    if [[ -n $file_search ]] # если строка не пустая
    then
        cmd=`ls -lR $1 | grep ^$i | head -1 | cut -b 53-1000`
        echo "$file_search -- `pwd`/$cmd" # вывод полного пути файла
    else
        echo "notfound" # если строка с файлом пустая
    fi
done

```

Рисунок 6 – Скрипт 1.sh

```

rymarmary@rymarmary-VirtualBox:~$ ./1.sh /bin
-
-rwxr-xr-x 1 root root 1113504 Apr 18 2022 bash -- /home/rymarmary/
c
notfound
d
notfound
l
lrwxrwxrwx 1 root root      6 Apr 10 17:09 bzip2 -- /home/rymarmary/
bzip2
s
notfound
b
notfound
p
notfound

```

Рисунок 7 – Работа скрипта в директории /bin

2. Был написан скрипт, который находит все жёсткие ссылки на заданный файл. Скрипт 2.sh показан на рисунке 8.

```

rymarmary@rymarmary-VirtualBox:~$ cat 2.sh
#!/bin/sh

if [ $# -lt 1 ] # сравнение количества введенных аргументов с 1
then
echo $0: error: File not specified # если они не были введены,
else                                     # то файл не специализирован
filename=$1 # в переменную filename записываем первый введенный аргумент
# вывод inode поданного файла
inode=`ls -li $filename | cut -d ' ' -f 1 | tr -d " "`
# поиск файлов по inode
tmp=`ls -lRi /home/rymar | grep $inode`
fi

# вывод найденных файлов
echo $tmp

```

Рисунок 8 – Скрипт 8.sh

3. Были проанализированы способы создания символьных ссылок. Две команды для создания символьных ссылок:

```
ln -s file1.txt file2.txt
```

```
link file1.txt file2.txt
```

Команда для копирования символьной ссылки:

```
cp -a file1.txt file2.txt
```

Скрипт 3.sh, который находит все символьные ссылки на файл, представлен на рисунке 9.

```
rymarmary@rymarmary-VirtualBox:~$ cat 3.sh
#!/bin/sh

filename=$1 # считывается имя файла
ls -lRa / | grep $filename | grep ^l > symlinks.txt # поиск символических ссылок
echo -n "total " >> symlinks.txt
wc -l symlinks.txt | cut -c 1 >> symlinks.txt # вывод количества ссылок

com='cat symlinks.txt'

echo $com # вывод результата в консоль
```

Рисунок 9 – Скрипт 3.sh

4. Скрипт 4.sh, который находит все символьные ссылки на файл, представлен на рисунке 10.

```
rymarmary@rymarmary-VirtualBox:~$ cat 4.sh
#!/bin/bash

find -L /home/rymar -samefile $1
```

Рисунок 10 – Скрипт 4.sh

5. Скрипт 5.sh, использующий утилиту find, которая в комбинации с ключами -type и -ls выводит расширенную информацию о всех типах файлов, представлен на рисунке 11.

```
rymarmary@rymarmary-VirtualBox:~$ cat 5.sh
#!/bin/sh

# все типы файлов
types='f c d l s b p'
# цикл по всем типам
for i in $types
do
    tmp='find / -type $i -ls | head -1' # команда, выводящая
    # расширенную информацию о всех типах файлов
    echo $i: # вывод типа
    if [[ -n $tmp ]] # если строка с данными не пустая
    then
        echo "$tmp" # вывод, если строка не пустая
    else
        echo "    notfound" # вывод, если не был найден файл
    fi
done
```

Рисунок 11 – Скрипт 5.sh

Изучение ключей утилиты find:

1). -name – поиск по шаблону имени файла, показан на рисунке 12

```
rymarmary@rymarmary-VirtualBox:~$ find . -name "*.txt"
./os/w.txt
./mozilla/firefox/97dc36pg.default-release/SiteSecurityServiceState.txt
./mozilla/firefox/97dc36pg.default-release/pkcs11.txt
./mozilla/firefox/97dc36pg.default-release/AlternateServices.txt
./rymar/r.txt
./rymar/e.txt
```

Рисунок 12 – Утилита find с ключом name

2). -iname – поиск по шаблону имени файла без учёта регистра, показан на рисунке 13

```
rymarmary@rymarmary-VirtualBox:~$ find . -not -iname "*.txt"
.
./Music
./os
./examples.desktop
./5.sh
./3.sh
./Desktop
./config
```

Рисунок 13 – Утилита find с ключом iname

3). -chmod mode – изменяет права доступа к файлу

4). -type <...> - поиск файлов типа <...>, где <...> может быть равно b, c, d, l, n, s, p (FIFO), f (regular file).

5). -print – вывод полного имени файла, показан на рисунке 14

```
rymarmary@rymarmary-VirtualBox:~$ find . -name "*.txt" -print
./os/w.txt
./mozilla/firefox/97dc36pg.default-release/SiteSecurityServiceState.txt
./mozilla/firefox/97dc36pg.default-release/pkcs11.txt
./mozilla/firefox/97dc36pg.default-release/AlternateServices.txt
./rymar/r.txt
./rymar/e.txt
```

Рисунок 14 – Утилита find с ключами name и print

6). -ls – вывод полного имени файла в таком формате, как ls -l

7). -lname file – поиск символьных ссылок на файл с названием file

8). -inode file|n – поиск файлов с тем же серийным номером, как у файла с названием file, или с серийным номером n



9). -level n – поиск файлов, расположенных в дереве каталогов на n уровней ниже заданного каталога

Примеры вложенных команд:

1). На рисунке 15 показан поиск всех файлов с расширением txt

```
rymary@rymary-VirtualBox:~/rymar$ find `cat test1.txt`  
e.txt  
r.txt  
test1.txt  
rymary@rymary-VirtualBox:~/rymar$ cat test1.txt  
*.txt
```

Рисунок 15 – Поиск txt-файлов

2). На рисунке 16 показан поиск всех файлов с расширением sh

```
rymary@rymary-VirtualBox:~$ find `echo *.sh`  
1.sh  
2.sh  
3.sh  
4.sh  
5.sh  
script1.sh  
script.sh
```

Рисунок 16 – Поиск sh-файлов

6. Написан скрипт 6.sh, отслеживающий и анализирующий изменения файла с помощью утилиты od. Утилиту нельзя применять к директориям. Скрипт и его работа представлена на рисунках 17-18.

```
rymary@rymary-VirtualBox:~$ cat 6.sh  
#!/bin/bash  
  
> new.txt # создание нового регулярного файла  
echo "Создан новый регулярный файл:\n" > result.txt # запись вывода в результирующий файл  
od -tc new.txt >> result.txt # отслеживание состояния new.txt  
  
echo "rrr\nabcd\nRRR\n" >> new.txt  
echo "Файл был изменён(добавлен текст):\n" >> result.txt # запись вывода в результирующий файл  
od -tc new.txt >> result.txt # отслеживание состояния new.txt  
  
echo "rrr\nRRR\n" > new.txt  
echo "Файл был изменён(удалена часть текста):\n" >> result.txt # запись вывода в результирующий файл  
od -tc new.txt >> result.txt # отслеживание состояния new.txt
```

Рисунок 17 – Скрипт 6.sh

```

rymary@rymary-VirtualBox:~$ sudo chmod +x 6.sh
[sudo] password for rymary:
rymary@rymary-VirtualBox:~$ ./6.sh
rymary@rymary-VirtualBox:~$ cat result.txt
Создан новый регулярный файл:\n
0000000
Файл был изменён(добавлен текст):\n
0000000  г  г  г  \  н  а  б  с  d  \  н  R  R  R  \  н
0000020  \n
0000021
Файл был изменён(удалена часть текста):\n
0000000  г  г  г  \  н  R  R  R  \  н  \n
0000013

```

Рисунок 18 – Работа скрипта 6.sh

7. Максимальное количество записей в каталоге:  $2^{16}-1$ . Максимальное количество файлов в файловой системе и максимальное количество открытых файлов можно посмотреть в */proc/sys/fs/file-nr*. В моём случае им соответствуют числа 94988 и 8352.

На рисунке 19 представлен скрипт 7.sh, порождающий новые файлы и каталоги, а затем удаляющий их. В скрипте предусматривается отслеживание промежуточного и конечного вывода информации о размере тестового каталога. На рисунке 20 показана работа этого скрипта.

```

rymary@rymary-VirtualBox:~$ cat 7.sh
#!/bin/bash

size=`du -hs $1` # команда для нахождения текущего размера директории
echo Текущий размер директории: $size # вывод изначального размера

for i in {1..250} # цикл, создающий 250 файлов в директории
do
    name=$1/$i.txt
    > $name
    echo "czcxzxczaaaaaaaaaa" >> $name
    mkdir $1/$i
done

size=`du -hs $1` # команда для нахождения текущего размера директории
echo "Текущий размер директории(после добавления файлов):" $size # вывод измене
нного размера

for i in {101..251} # цикл, удаляющий 250 файлов в директории
do
    name=$1/$i.txt
    rm -rf $name $1/$i
done

size=`du -hs $1` # команда для нахождения текущего размера директории
echo "Текущий размер директории(после удаления файлов):" $size # вывод конечн
о размера

```

Рисунок 19 – Скрипт 7.sh

```

rymary@rymary-VirtualBox:~$ ./7.sh test
Текущий размер директории: 4.0K test
Текущий размер директории(после добавления файлов): 2.0M test
Текущий размер директории(после удаления файлов): 812K test

```

Рисунок 20 – Работа скрипта 7.sh

8. На рисунке 21 представлено частичное содержимое файла */etc/passwd*. В нём представлена информация о пользователях в следующем формате:

*<username>:<passwd>:<UID>:<GID>:<comments>:<homedir>:<shell>*

```

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync

```

Рисунок 21 – Частичное содержимое файла */etc/passwd*

Этот файл доступен всем пользователям для чтения, на запись и изменение – только администратору. В этом файле не хранятся пароли (в целях безопасности), пароли хранятся в зашифрованном файле */etc/shadow*, он доступен для чтения и записи только администратору. Чтобы обычный пользователь мог изменить свой пароль, существует утилита */usr/bin/passwd*, она доступна на чтение и исполнение, а администратору и root ещё и на запись. Права доступа к некоторому файлу можно узнать с помощью утилиты *ls -l <filename>*. Получить содержимое файла */etc/shadow* можно с помощью команды *sudo vim shadow*. Часть этого файла показана на рисунке 22.

```

root:!:19457:0:99999:7:::
daemon:!:18885:0:99999:7:::
bin:!:18885:0:99999:7:::
sys:!:18885:0:99999:7:::
sync:!:18885:0:99999:7:::
games:!:18885:0:99999:7:::

```

Рисунок 22 – Файл */etc/shadow*

9. В Linux есть 3 вида разрешений, они определяют права пользователя на три действия: чтение, запись и выполнение. Для них введены следующие обозначения:

r – read (чтение) – право просматривать содержимое файла;

w – write (запись) – право изменять файл;

x – execute (выполнение) – право запускать файл, обычно используется для программ или скриптов.

У каждой группы есть три типа пользователей, для которых можно устанавливать права доступа: owner, group, others. Owner – владелец, обычно это тот, кто создал файл, но можно сделать и кого-то другого владельцем. Group – группа пользователей с общими заданными правами. Others – все остальные пользователи, которые не являются владельцем или группой.

Для различных файлов есть различные наборы прав:

--- - нет прав, совсем;

--x – разрешено только выполнение файла, как программы, но не изменение и не чтение;

-w- - разрешено только изменение файла;

r-- - разрешён только просмотр файла;

-wx – разрешено только изменение и выполнение, но в случае с каталогом, просмотреть его содержимое нельзя;

r-x – разрешено только чтение и выполнение, запись ограничена;

rw- - разрешено чтение и запись, выполнение запрещено;

rw- - все права;

--s – установлен SUID или SGID бит, первый отображается в поле для владельца, второй для группы;

--t – установлен sticky-bit, а значит пользователи не могут удалить этот файл.

9.1. Команда `chmod` используется для изменения прав доступа к файлу или каталогу. Пример использования этой команды показан на рисунке 23. Команда `chown` меняет владельца файла, каталога или ссылки. Пример использования этой команды показан на рисунке 24. Несмотря на то, что утилита `chown` может быть выполнена любым пользователем, она позволяет изменить владельца только администратору.

```

rymary@rymary-VirtualBox:~/task9_1$ ls -l
total 8
-rw-rw-r-- 1 rymarmar yymarmar 61 Apr 17 17:45 text1.txt
-rw-rw-r-- 1 rymarmar yymarmar 23 Apr 17 17:46 text2.txt
rymary@rymary-VirtualBox:~/task9_1$ chmod +x text1.txt
rymary@rymary-VirtualBox:~/task9_1$ chmod -rw text2.txt
rymary@rymary-VirtualBox:~/task9_1$ ls -l
total 8
-rwxrwxr-x 1 rymarmar yymarmar 61 Apr 17 17:45 text1.txt
----- 1 rymarmar yymarmar 23 Apr 17 17:46 text2.txt

```

Рисунок 23 – Пример использования chmod

```

rymary@rymary-VirtualBox:~/task9_1$ sudo chown root text2.txt
[sudo] password for rymarmar:
rymary@rymary-VirtualBox:~/task9_1$ ls -l
total 8
-rwxrwxr-x 1 rymarmar yymarmar 61 Apr 17 17:45 text1.txt
----- 1 root      rymarmar 23 Apr 17 17:46 text2.txt
rymary@rymary-VirtualBox:~/task9_1$ sudo chown :root text2.txt
rymary@rymary-VirtualBox:~/task9_1$ ls -l
total 8
-rwxrwxr-x 1 rymarmar yymarmar 61 Apr 17 17:45 text1.txt
----- 1 root      root      23 Apr 17 17:46 text2.txt

```

Рисунок 24 – Пример использования chown

9.2. Расширим права исполнения файла с помощью флага SUID. Пример показан на рисунке 25.

```

rymary@rymary-VirtualBox:~/task9_1$ ls -l
total 8
-rwxrwxr-x 1 rymarmar yymarmar 61 Apr 17 17:45 text1.txt
----- 1 root      root      23 Apr 17 17:46 text2.txt
rymary@rymary-VirtualBox:~/task9_1$ chmod u+s text1.txt
rymary@rymary-VirtualBox:~/task9_1$ ls -l
total 8
-rwsrwxr-x 1 rymarmar yymarmar 61 Apr 17 17:45 text1.txt
----- 1 root      root      23 Apr 17 17:46 text2.txt

```

Рисунок 25 – Пример расширения прав с помощью флага SUID

9.3. Экспериментальным путём установим, как формируются итоговые права на использование файла, если права пользователя и группы, в которую он входит, различны. Результаты показаны на рисунках 26-27.

```

rymary@rymary-VirtualBox:~/task9_1$ ls -l
total 4
-rw-rw-r-- 1 rymarmar yymarmar 10 Apr 17 18:00 text1.txt
rymary@rymary-VirtualBox:~/task9_1$ chmod u-r text1.txt
rymary@rymary-VirtualBox:~/task9_1$ ls -l
total 4
--w-rw-r-- 1 rymarmar yymarmar 10 Apr 17 18:00 text1.txt
rymary@rymary-VirtualBox:~/task9_1$ chmod g+x text1.txt
rymary@rymary-VirtualBox:~/task9_1$ ls -l
total 4
--w-rwxr-- 1 rymarmar yymarmar 10 Apr 17 18:00 text1.txt
rymary@rymary-VirtualBox:~/task9_1$ vim text1.txt

```

```
~  
"text1.txt" [Permission Denied]
```

## Рисунки 26-27 – Формирование итоговых прав

9.4. Сопоставим возможности исполнения наиболее часто используемых операций, меняя права доступа к файлу и каталогу.

На рисунке 28 представлен следующий случай: у директории забрали права на запись, но пытаемся изменить файлы в ней. Итог – доступ запрещён.

На рисунке 29 представлен другой случай: у директории забрали права на чтение и пытаемся вывести её содержимое. Доступ запрещён, но удалить, например, файл возможно, так как директория доступна на запись.

```
rymary@rymary-VirtualBox:~/task9_1$ ls -l  
total 8  
drwxrwxr-x 2 rymary rymary 4096 Apr 17 18:28 test  
--w-rwxr-- 1 rymary rymary 10 Apr 17 18:00 text1.txt  
rymary@rymary-VirtualBox:~/task9_1$ chmod ugo-w test/  
rymary@rymary-VirtualBox:~/task9_1$ ls -l  
total 8  
dr-xr-xr-x 2 rymary rymary 4096 Apr 17 18:28 test  
--w-rwxr-- 1 rymary rymary 10 Apr 17 18:00 text1.txt  
rymary@rymary-VirtualBox:~/task9_1$ cd test  
rymary@rymary-VirtualBox:~/task9_1/test$ ls  
a.txt b.txt  
rymary@rymary-VirtualBox:~/task9_1/test$ rm -rf a.txt  
rm: cannot remove 'a.txt': Permission denied  
rymary@rymary-VirtualBox:~/task9_1/test$ mv a.txt c.txt  
mv: cannot move 'a.txt' to 'c.txt': Permission denied
```

Рисунок 28 – Забрали права на запись у директории

```
rymary@rymary-VirtualBox:~/task9_1$ ls -l  
total 8  
dr-xr-xr-x 2 rymary rymary 4096 Apr 17 18:28 test  
--w-rwxr-- 1 rymary rymary 10 Apr 17 18:00 text1.txt  
rymary@rymary-VirtualBox:~/task9_1$ chmod +w test  
rymary@rymary-VirtualBox:~/task9_1$ ls -l  
total 8  
drwxrwxr-x 2 rymary rymary 4096 Apr 17 18:28 test  
--w-rwxr-- 1 rymary rymary 10 Apr 17 18:00 text1.txt  
rymary@rymary-VirtualBox:~/task9_1$ chmod ugo-r test  
rymary@rymary-VirtualBox:~/task9_1$ cd test  
rymary@rymary-VirtualBox:~/task9_1/test$ ls  
ls: cannot open directory '.': Permission denied  
rymary@rymary-VirtualBox:~/task9_1/test$ rm -rf a.txt  
rymary@rymary-VirtualBox:~/task9_1/test$ ls  
ls: cannot open directory '.': Permission denied  
rymary@rymary-VirtualBox:~/task9_1/test$ sudo ls  
[sudo] password for rymary:  
b.txt
```

Рисунок 29 – Забрали права на чтение у директории



10. Создадим программу-шлюз, которая пытается открыть файл и вывести его содержимое. Исходный код показан на рисунке 30.

```
rymary@rymary-VirtualBox:~/rymar/10task$ cat prog.c
#include <stdio.h>

int main(int argc, char* argv[]) {
    if (argc <= 1) {
        printf("%s: файл не был введен\n", argv[0]);
    } else {
        FILE* f;
        f = fopen(argv[1], "r");
        if (f) {
            printf("%s: %s файл был открыт\n", argv[0], argv[1]);
            char str[64];
            while (fgets(str, sizeof(str), f)) {
                printf("%s", str);
            }
            fclose(f);
        } else {
            printf("%s: %s файл не был открыт\n", argv[0], argv[1]);
        }
    }

    return 0;
}
```

Рисунок 30 - Исходный код программы-шлюза

Командой *chmod o-r a.txt* были отобраны права на чтение файла *a.txt* у других пользователей. Затем после компиляции *prog.c* исполняемому файлу *a.out* был выдан SUID командой *chmod +s ./a.out*.

Далее был создан новый пользователь *tester*. Авторизуемся через него с помощью команды *sudo login tester* и попытаемся посмотреть содержимое файла. Доступ к файлу ограничен, что показано на рисунке 31. Теперь попробуем с помощью программы-шлюза получить доступ к содержимому файла, будучи авторизованным под другим пользователем. Содержимое файла было выведено, что показано на рисунке 32, так как был выдан SUID исполняемому файлу, который временно выдал права администратора.

```
tester@rymary-VirtualBox:/home/rymary/rymar/10task$ cat a.txt
cat: a.txt: Permission denied
```

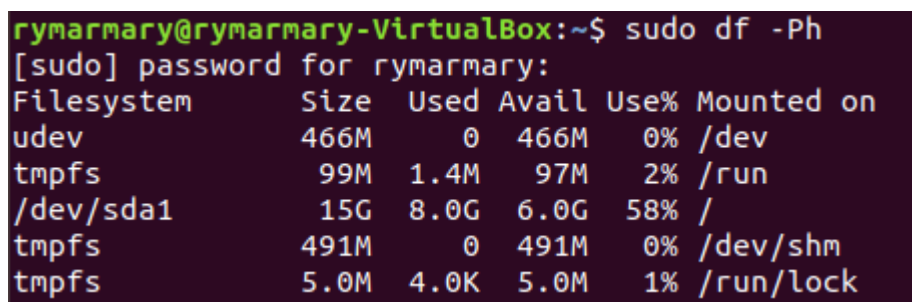
Рисунок 31 – Попытка другим пользователем открыть файл

```
tester@rymary-VirtualBox:/home/rymary/rymar/10task$ ./a.out a.txt
./a.out: a.txt файл был открыт
муау
```

Рисунок 32 – Вывод содержимого файла благодаря программе-шлюзу

11. В UNIX-подобных системах утилита `df` показывает список всех файловых систем по именам устройств, их размер, занятое и свободное пространство и точки монтирования. Применение этой утилиты показано на рисунке 33. Применяя эту утилиту и аналогичные ей по функциональности, можно получить информацию о файловых системах, доступных для монтирования, а также установленных на компьютере реально.

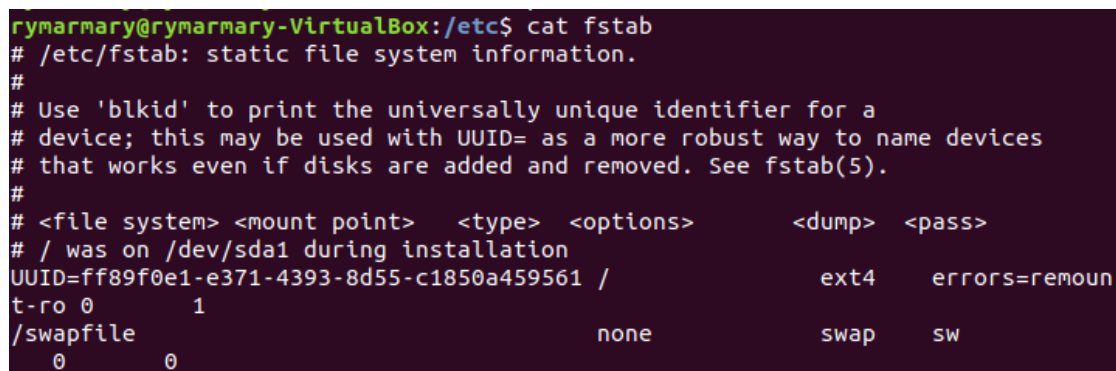
Команда `df` предоставляет опцию для отображения размеров в удобочитаемых форматах с ключом `-h` (выводит результаты в удобном формате, например, 1М, 3К). Ключ `-P` использует POSIX формат вывода. В первом столбце вывода написаны названия ФС, в следующей – их размеры, далее – сколько памяти используется, в четвёртой – сколько доступно памяти, затем сколько занято в процентном соотношении, в последней – где смонтирована система.



```
rymarmary@rymarmary-VirtualBox:~$ sudo df -Ph
[sudo] password for rymarmary:
Filesystem      Size  Used Avail Use% Mounted on
udev            466M     0  466M   0% /dev
tmpfs           99M    1.4M   97M   2% /run
/dev/sda1       15G    8.0G   6.0G  58% /
tmpfs           491M     0  491M   0% /dev/shm
tmpfs           5.0M    4.0K   5.0M   1% /run/lock
```

Рисунок 33 – Утилита `df`

На рисунке 34 выведена информация файла `/etc/fstab`. В нём обычно перечислены все доступные разделы диска и другие типы файловых систем и источников данных, которые могут не обязательно располагаться на дисках, также указано, как они должны быть инициализированы или интегрированы иным образом в более крупную структуру файловой системы.



```
rymarmary@rymarmary-VirtualBox:/etc$ cat 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>
# / was on /dev/sda1 during installation
UUID=ff89f0e1-e371-4393-8d55-c1850a459561 / ext4 errors=remoun
t-ro 0 1
/swapfile none swap sw
0 0
```

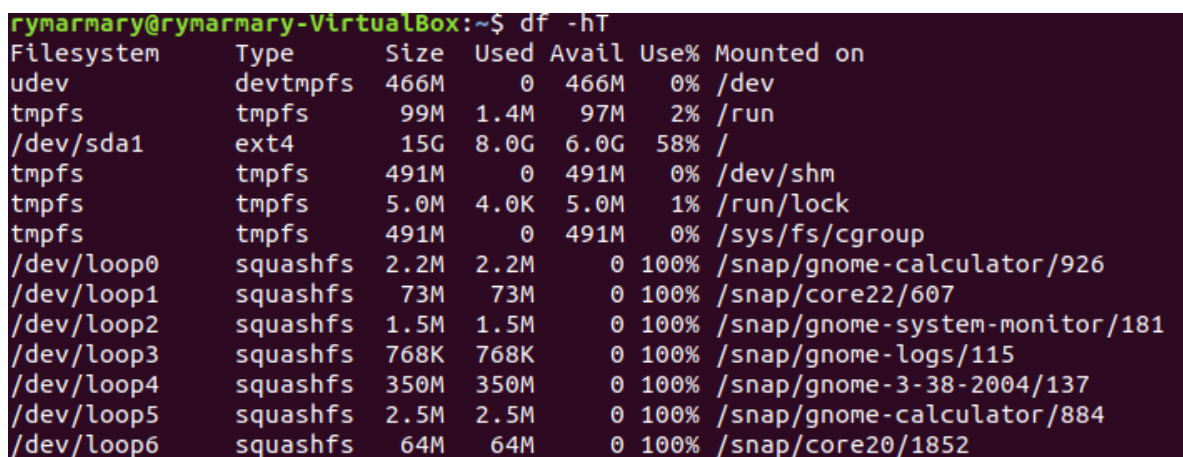
Рисунок 34 – Файл `fstab`



11.1. Во многих Unix-подобных системах tmpfs – временное файловое хранилище. Оно предназначено для монтирования ФС, но размещается в ОЗУ вместо физического диска. Подобная конструкция является RAM диском.

Для новых версий ядра Linux менеджером устройств является udev. Его основная задача – обслуживание файлов устройств в каталоге /dev и обработка всех действий, выполняемых в пространстве пользователя при добавлении или отключении внешних устройств. /dev/nvme0n1p8 – раздел жёсткого диска.

11.2. Фильтрация вывода по типу файловой системы происходит также с помощью утилиты df только с ключом -T. Применение данного ключа показано на рисунке 35. Приведён образ диска с точки зрения состава и размещения всех ФС на компьютере, а также образ полного дерева ФС, включая присоединённые ФС съёмных и несъёмных носителей. В первом столбце вывода указаны названия файловых систем, потом – их тип, далее – размер, в четвёртой – сколько используется, потом – сколько памяти доступно, затем – сколько занято в процентном отношении, в последней – где смонтирована ФС.



Filesystem	Type	Size	Used	Avail	Use%	Mounted on
udev	devtmpfs	466M	0	466M	0%	/dev
tmpfs	tmpfs	99M	1.4M	97M	2%	/run
/dev/sda1	ext4	15G	8.0G	6.0G	58%	/
tmpfs	tmpfs	491M	0	491M	0%	/dev/shm
tmpfs	tmpfs	5.0M	4.0K	5.0M	1%	/run/lock
tmpfs	tmpfs	491M	0	491M	0%	/sys/fs/cgroup
/dev/loop0	squashfs	2.2M	2.2M	0	100%	/snap/gnome-calculator/926
/dev/loop1	squashfs	73M	73M	0	100%	/snap/core22/607
/dev/loop2	squashfs	1.5M	1.5M	0	100%	/snap/gnome-system-monitor/181
/dev/loop3	squashfs	768K	768K	0	100%	/snap/gnome-logs/115
/dev/loop4	squashfs	350M	350M	0	100%	/snap/gnome-3-38-2004/137
/dev/loop5	squashfs	2.5M	2.5M	0	100%	/snap/gnome-calculator/884
/dev/loop6	squashfs	64M	64M	0	100%	/snap/core20/1852

Рисунок 35 – Утилита df с ключом -hT

Fstab – постоянная информация для монтирования файловой системы (выше прикреплён скриншот). Файл также содержит информацию о смонтированных файловых системах.

Динамически изменяющаяся информация о реально смонтированных файловых системах находится в mtab. Проведём опыт с флеш-накопителем. В файл 1.txt записаны данные /etc/mtab с использованием флеш-накопителя, а в

файл 2.txt без него. С помощью утилиты diff можно увидеть разницу в файлах. Это строка с указанием нового устройства – флеш-накопителя, типа файловой системы, точкой его монтирования и опциями.

Посмотреть список всех монтированных устройств можно утилитой mount. Пример её работы показан на рисунке 36. Формат строки таблицы: имя устройства, режим включения, точка монтирования и тип файловой системы. С помощью этой утилиты можно также отследить флеш-накопитель.

```
rymarmary@rymarmary-VirtualBox:~$ mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
udev on /dev type devtmpfs (rw,nosuid,relatime,size=476188k,nr_inodes=119047,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=100484k,mode=755)
/dev/sda1 on / type ext4 (rw,relatime,errors=remount-ro)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
```

Рисунок 36 – Список монтированных устройств

Файл, содержащий перечень всех поддерживаемых ядром типов файловых систем, - /proc/filesystems показан на рисунке 37. Строки, которые начинаются с nodev значат то, что файловая система не является физической.

```
rymarmary@rymarmary-VirtualBox:/proc$ cat filesystems
nodev    sysfs
nodev    tmpfs
nodev    bdev
nodev    proc
nodev    cgroup
nodev    cgroup2
nodev    cpuset
nodev    devtmpfs
nodev    configfs
nodev    debugfs
nodev    tracefs
nodev    securityfs
nodev    sockfs
nodev    bpf
nodev    pipefs
```

Рисунок 37 – Файл /proc/filesystems

11.3. В Linux наибольшая длина пути определена в файле /usr/include/linux/limits.h. Содержимое этого файла представлено на рисунке 38. Таким образом, наибольшее количество символов пути файла равняется 4096.

```

rymary@rymary-VirtualBox:/usr/include/linux$ cat limits.h
/* SPDX-License-Identifier: GPL-2.0 WITH Linux-syscall-note */
#ifndef _LINUX_LIMITS_H
#define _LINUX_LIMITS_H

#define NR_OPEN          1024

#define NGROUPS_MAX      65536 /* supplemental group IDs are available */
#define ARG_MAX          131072 /* # bytes of args + environ for exec() */
#define LINK_MAX         127 /* # links a file may have */
#define MAX_CANON         255 /* size of the canonical input queue */
#define MAX_INPUT         255 /* size of the type-ahead buffer */
#define NAME_MAX          255 /* # chars in a file name */
#define PATH_MAX          4096 /* # chars in a path name including nul */
#define PIPE_BUF          4096 /* # bytes in atomic write to a pipe */
#define XATTR_NAME_MAX    255 /* # chars in an extended attribute name */
#define XATTR_SIZE_MAX    65536 /* size of an extended attribute value (64k) */
#define XATTR_LIST_MAX    65536 /* size of extended attribute namelist (64k) */

#define RTSIG_MAX         32

#endif

```

Рисунок 38 – Файл limits.h

Зная, что имя каталога содержит хотя бы два символа (/ + имя), то максимальный уровень вложенности равен 2047 директорий.

Исходя из вывода прошлой команды можно увидеть, что максимальное дерево файловой системы - /dev/nvme0n1p8, так как его размер максимален относительно всех файловых систем. Также он смонтирован в корневой директории, поэтому имеет дерево максимального размера.

12. Тип введенного файла определяется утилитой `file`. Она позволяет получить информацию о типе файла: она сканирует начало файла и пытается определить его тип. Есть три типа тестов: тест файловой системы, тест магических чисел и тест языка. Они применяются именно в таком указанном порядке, результат выдаёт первый успешно закончившийся тест. Синтаксис:

*file [-bcLnvz] [-f namefile] [-m magicfile] file ...*

12.1. Для определения типа файла выполняются тесты, которые можно разделить на три группы:

Filesystem tests – тесты, основанные на анализе кода возврата системного вызова `stat()`. Программа тестов основана на проверке на пустоту файла и на принадлежность к одному из специальных типов файлов. Все известные типы файлов распознаются, если они определены в системном файле `/usr/include/sys/stat.h`.

Magic number tests – тесты, которые используются для проверки файлов, данные которых записаны в определённом формате. В определённом месте в начале таких файлов записано магическое число, которое позволяет ОС определить тип файла. Все известные ОС магические числа по умолчанию хранятся в файле */usr/share/file/magic.\**.

Language tests – тесты, которые используются для анализа языка, на котором написан файл, если этот файл в формате ASCII. Выполняется поиск стандартных строк, которые могут соответствовать определённому языку.

Первый, успешно завершившийся тест, и выводит тип файла. Типы файлов можно разделить на три основные группы:

Текстовые – файл содержит только ASCII символы, он может быть безопасно прочитан на терминале.

Исполняемые – файл содержит результаты компилирования программы в той форме, которая понятна ядру операционной системы.

Данные – всё, что не входит в первые две группы (обычно это бинарные и непечатаемые файлы). Исключение составляют well-known форматы, используемые для хранения бинарных данных.

## 12.2. Все ключи:

-b, --brief – запрет на демонстрацию имён и адресов файлов в выводе команды;

-i, --mime – определение MIME-типа документа по его заголовку;

--mime-type, --mime-encoding – определение конкретного элемента MIME;

-f, --files-from – анализ документов, адреса которых указаны в простом текстовом файле;

-l, --list – список паттернов и их длина;

-s, --special-files – предотвращение проблем, которые могут возникнуть при чтении утилитой специальных файлов;

-P – анализ определённой части файла, которая обозначается различными параметрами;

-r, --raw – отказ от вывода /ooo вместо напечатанных символов;

-z – анализ содержимого сжатых документов.

Пример стандартного применения утилиты file показан на рисунке 39. Вывод только типа файла показан на рисунке 40. Выполнение поиска, используя диапазон, представлено на рисунке 41. Итоговая информация расширенной формы magic file показана на рисунке 42.

```
rymary@rymary-VirtualBox:~$ file result.txt
result.txt: UTF-8 Unicode text
```

Рисунок 39 – Утилита file

```
rymary@rymary-VirtualBox:~$ file -b result.txt
UTF-8 Unicode text
```

Рисунок 40 – Вывод типа файла

```
rymary@rymary-VirtualBox:~$ file [p-t]*
Pictures:  directory
Public:    directory
result.txt: UTF-8 Unicode text
rymar:     directory
script1.sh: empty
script.sh: POSIX shell script, ASCII text executable
task9_1:   directory
test:      directory
```

Рисунок 41 – Поиск с использованием диапазона

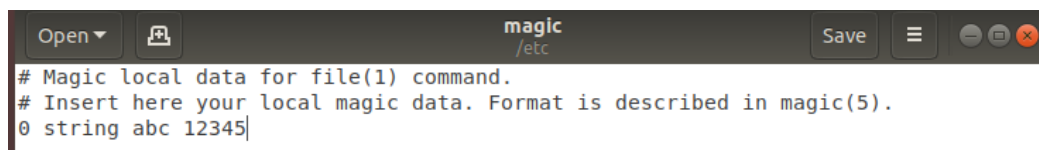
```
rymary@rymary-VirtualBox:~$ file -c
cont  offset type  opcode mask  value  desc
```

Рисунок 42 – Утилита file с ключом -c

12.3. Добавлен новый тип файла на рисунке 43 показана команда. На рисунке 44 представлено содержимое файла.

```
rymary@rymary-VirtualBox:~$ sudo gedit /etc/magic
[sudo] password for rymary:
```

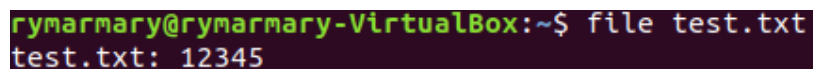
Рисунок 43 – Команда для добавления нового типа файла



```
# Magic local data for file(1) command.
# Insert here your local magic data. Format is described in magic(5).
0 string abc 12345
```

Рисунок 44 – Файл /etc/magic

Выбран тип 12345, после был создан файл, в начало которого записали строку “abc”. Используя утилиту file, определили новый тип файла. Это представлено на рисунке 45.



```
rymary@rymary-VirtualBox:~$ file test.txt
test.txt: 12345
```

Рисунок 45 – Новый тип файла

### **Выводы.**

В ходе выполнения лабораторной работы было проанализировано функциональное назначение структурных элементов дерева файловой системы и определено размещение корневого каталога (корневой файловой системы).

Кроме того, были рассмотрены различные команды и утилиты для работы с файловой системой Linux, включая команды для создания, копирования, перемещения и удаления файлов и директорий, а также для изменения прав доступа к файлам и директориям.

Были изучены различные типы файловых систем, такие как ext2, ext3, ext4, NTFS и FAT, и их особенности. Также были рассмотрены различные форматы файловых систем, такие как fdisk, mkfs и mke2fs, и способы их использования для создания и форматирования разделов жесткого диска. Также были рассмотрены команды для монтирования файловых систем.

Таким образом, лабораторная работа по работе с файловой системой Linux позволяет получить практические навыки работы с файловой системой, что является важным для администраторов систем и других пользователей Linux.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Собель М. Руководство администратора Linux. - СПб.: Питер, 2015. - 1056 с.
2. Барретт Дж. Linux. Карманный справочник. - СПб.: Питер, 2013. - 320 с.
3. Линукс в деталях / Раймонд Эрик. - М.: ООО "Диалектика", 2012. - 422 с.
4. Шотт Б. Командная строка Linux. - СПб.: Питер, 2013. - 416 с.
5. Официальная документация Linux:  
<https://www.kernel.org/doc/html/latest/>
6. Руководства и документация по Linux на сайте Linux.org:  
<https://www.linux.org/docs/>
7. Руководства и документация по Linux на сайте Ubuntu:  
<https://help.ubuntu.com/>
8. Руководства и документация по Linux на сайте Red Hat:  
<https://access.redhat.com/documentation/>