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

Студент гр. 1304 _____ Заика Т.П.

Преподаватель _____ Душутина Е.В.

Санкт-Петербург

2023

Цель работы.

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

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. С помощью команды ls -R -l | grep ^[-, b, c, d, l, p, s] был произведен поиск файлов каждого типа.

Файлы системы могут быть следующих типов:

- Обычный файл

b Специальный файл блочного устройства

c Файл символьного устройства

d Директория

l Символьная ссылка

p FIFO

s Сокет

был написан скрипт, который выводит все типы файлов, которые есть в переданной в качестве параметра директории.

```
/bin/
ls -R -l | grep ^- > /home/ubstudy/Study/OS/lab2/commands.txt >>
-rwxr-xr-x 1 root root        18456 фев  7  2021 411toppm

/boot/
ls -R -l | grep ^b > /home/ubstudy/Study/OS/lab2/commands.txt >>
brw-rw----  1 root    disk      8,   1 фев 28 16:20 sda1

/dev/
ls -R -l | grep ^c > /home/ubstudy/Study/OS/lab2/commands.txt >>
crw-r--r--  1 root    root     10, 235 фев 28 16:20 autofs

/dev/
ls -R -l | grep ^d > /home/ubstudy/Study/OS/lab2/commands.txt >>
drwxr-xr-x  2 root    root         540 мар  1 14:09 block

/dev/
ls -R -l | grep ^l > /home/ubstudy/Study/OS/lab2/commands.txt >>
lrwxrwxrwx  1 root    root          11 фев 28 16:20 core -> /proc/kcore

/
sudo ls -R -l / | grep ^p > /home/ubstudy/Study/OS/lab2/commands.txt >>
prw------- 1 root root   0 фев 28 16:20 1.ref

/
sudo ls -R -l / | grep ^s > /home/ubstudy/Study/OS/lab2/commands.txt >>
srw-rw-rw-  1 root              root     0 фев 28 16:20 acpid.socket
```

```
#!/bin/bash
types='- c d l s b p' #типы файлов
for i in $types #ищем файл для каждого типа
do
    echo $i #пишем текущий ключ
    file_search=`ls -lR $1 | grep ^$i | head -1` #найдем файл
        if [[ -n $file_search ]] #если нашли, т.е. переменная не пуста
        then
        cmd=`ls -lR $1 | grep ^$i | head -1 | cut -b 53-1000` #запоминаем нужную информацию про файл
                echo "$file_search -- `pwd`/$cmd" #выводим информацию про файл с указанием полного пути
        else
                echo "notfound" #иначе говорим что не нашли файл
        fi
done
```

```
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts$ ./firstscript.sh /dev
-
notfound
c
crw-r--r--  1 root    root     10, 235 фев 28 16:20 autofs -- /home/ubstudy/Study/OS/lab2/scr
ipts/20 autofs
d
drwxr-xr-x  2 root    root         540 мар  2 21:41 block -- /home/ubstudy/Study/OS/lab2/scri
pts/41 block
l
lrwxrwxrwx  1 root    root           3 фев 28 16:21 cdrom -> sr0 -- /home/ubstudy/Study/OS/la
b2/scripts/21 cdrom -> sr0
s
notfound
b
brw-rw----  1 root    disk      7,   0 фев 28 16:20 loop0 -- /home/ubstudy/Study/OS/lab2/scri
pts/20 loop0
p
notfound
```

2. Написан скрипт, который находит все жесткие ссылки на заданный файл.

```
#!/bin/sh

if [ $# -lt 1 ] #если входных параметров не задано
then
        echo $0: error: File not specified #то говорим что файл не задан
else
        filename=$1 #запоминаем имя файла
        inode=`ls -i $filename | cut -d ' ' -f 1 | tr -d " "` #получаем жесткую ссылку на файл
        tmp=`ls -lRi /home/ubstudy | grep $inode` #рекурсивно просматриваем каталог, ищем жесткую ссылку
fi

echo $tmp #выводим жесткую ссылку на файл
```

```
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts$ sudo ./secondsript.sh /h
ome/ubstudy/Study/OS/lab1/test.txt
132988 -rw-rw-r-- 2 ubstudy ubstudy 0 фев 9 22:28 linktest.txt 132988 -rw-rw-r-- 2 ubstudy ub
study 0 фев 9 22:28 test.txt
```

3. Рассмотрим способы формирования символьных ссылок:

1) ln -s

```
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts$ sudo ln -s secondsript.s
h linkscript.sh
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts$ ls -l
total 8
-rwxrwxrwx 1 ubstudy ubstudy 341 мар  8 23:05 firstscript.sh
lrwxrwxrwx 1 root    root     14 мар  9 11:40 linkscript.sh -> secondsript.sh
-rwxrwxrwx 1 root    root    194 мар  9 11:25 secondsript.sh
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts$
```

2) cp -s

```
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts$ sudo cp -s secondsript.s
h cplinkscripts.sh
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts$ ls -l
total 8
lrwxrwxrwx 1 root    root     14 мар  9 11:42 cplinkscripts.sh -> secondsript.sh
-rwxrwxrwx 1 ubstudy ubstudy 341 мар  8 23:05 firstscript.sh
lrwxrwxrwx 1 root    root     14 мар  9 11:40 linkscript.sh -> secondsript.sh
-rwxrwxrwx 1 root    root    194 мар  9 11:25 secondsript.sh
```

Создадим скрипт, подсчитывающий и перечисляющий все полноименные символьные ссылки на файл, размещаемые в разных местах файлового дерева.

```
#!/bin/sh

filename=$1 #запоминаем имя файла, переданное через параметр
ls -lRa /home/ubstudy | grep $filename | grep ^l > symlinks.txt #рекурсивно в каталоге ищем символьные
                                                                #ссылки на файл, записываем в выходной файл

echo -n "total " >> symlinks.txt #дописываем в файл строчку total
wc -l symlinks.txt | cut -c 1 >> symlinks.txt #дописываем количество ссылок

com=`cat symlinks.txt` #запоминаем команду вывода содержимого выходного файла

echo $com #вызываем команду
```

```
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts$ sudo ./thirdscript.sh se
condsript.sh
lrwxrwxrwx 1 root root 14 мар 9 11:42 cplinkscripts.sh -> secondsript.sh lrwxrwxrwx 1 root ro
ot 14 мар 9 11:40 linkscript.sh -> secondsript.sh total 2
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts$ cat symlinks.txt
lrwxrwxrwx 1 root    root     14 мар  9 11:42 cplinkscripts.sh -> secondsript.sh
lrwxrwxrwx 1 root    root     14 мар  9 11:40 linkscript.sh -> secondsript.sh
total 2
```

4. Получим все символьные ссылки на заданный файл, переданный в качестве входного параметра.

```
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts$ ls -lRa | grep secondsri
pt.sh | grep ^l
lrwxrwxrwx 1 root    root     14 мар  9 11:42 cplinkscripts.sh -> secondsript.sh
lrwxrwxrwx 1 root    root     14 мар  9 11:40 linkscript.sh -> secondsript.sh
```

5. Пристум к изучениею утилиты find, используя ее ключи получим расширенную информацию о всех типах файлов.

```
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts$ find -ls
   915724      4 drwxrwxr-x   2 ubstudy  ubstudy      4096 мар  9 11:48 .
   131082      4 -rwxrwxrwx   1 ubstudy  ubstudy       341 мар  8 23:05 ./firstscript.sh
   915730      4 -rwxrwxrwx   1 root     root          194 мар  9 11:25 ./secondsript.sh
   915733      4 -rwxrwxrwx   1 root     root          203 мар  9 11:48 ./thirdscript.sh
   915731      0 lrwxrwxrwx   1 root     root           14 мар  9 11:42 ./cplinkscripts.sh ->
 secondsript.sh
   915732      4 -rw-r--r--   1 root     root          175 мар  9 11:48 ./symlinks.txt
   915729      0 lrwxrwxrwx   1 root     root           14 мар  9 11:40 ./linkscript.sh -> se
condsript.sh
```

Примеры вложенных комманд:

Создание папки с названием, которое возвращает команда whoami.

```
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts$ sudo mkdir $(whoami)
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts$ ls
cplinkscripts.sh  linkscript.sh  secondsript.sh  thirdscript.sh
firstscript.sh    osadmin        symlinks.txt
```

```
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts$ find `echo "*.sh"`
cplinkscripts.sh
firstscript.sh
linkscript.sh
secondsript.sh
thirdscript.sh
```

Поиск всех файлов с разширением sh.

Создание файла с именем Linux.

```
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts$ sudo touch `uname`
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts$ ls
cplinkscripts.sh  linkscript.sh  osadmin         symlinks.txt
firstscript.sh    Linux          secondsript.sh  thirdscript.sh
```

6. Проанализируем содержимое файла с помощью утилиты od. Для этого напишем скрипт, выполним его и посмотрим на содержимое выходного файла.

```
#!/bin/bash

> new.txt #создаем новый файл для изменений
echo "Создан новый регулярный файл:\n" > result.txt #создаем файл результатов
od -tc new.txt >> result.txt #записываем анализ содержимого файла для изменений в файл для результатов

# далее изменяем файл для изменения, результаты его анализа записываем в результирующий файл
echo "abc\nABCD\ntesttset" >> new.txt
echo "Файл был изменён (добавлен текст):\n" >> result.txt
od -tc new.txt >> result.txt

echo "abc\nABCD" > new.txt
echo "Файл был изменён (удалена часть текста):\n" >> result.txt
od -tc new.txt >> result.txt
```

```
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts$ sudo ./fourthscript.sh
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts$ cat result.txt
Создан новый регулярный файл:\n
0000000
Файл был изменён (добавлен текст):\n
0000000   a   b   c   \   n   A   B   C   D   \   n   t   e   s   t   t
0000020   s   e   t  \n
0000024
Файл был изменён (удалена часть текста):\n
0000000   a   b   c   \   n   A   B   C   D  \n
0000012
```

Использование флага -tc позволяет увидеть в начале каждой выводимой строки то количество байт на ячейку, которое в ней хранится. При манипуляциях с файлом можно увидеть изменение размера занимаемой памяти.

7. Изменим размер каталога, варьируя количество записей (для этого создадим две программы, порождающие новые файлы и каталоги, а затем удаляющие их, предусмотрев промежуточный и конечный вывод информации о размере подопытного каталога, а также вывод информации о размере каталога после каждого добавления файла или каталога).

```
#!/bin/bash

#размер директории
size=`du -hs $1`

#выводим размер директории
echo "Текущий размер директории:" $size

#создаем в директории текстовые файлы произвольного диапозона
tmp_size=$(du -hs $1 | cut -c 1-3) #размер директории для сравнения
start_size=4 #начальный размер добавляемой единицы информации
for i in {1..1000}
do
        name=$1/$i.txt
        >$name

#записываем в файл информацию для объема
        echo "micro_test_1_2_3" >> $name

#сравниваем размер директории на каждом шаге для отслеживания скачков
        cur_size=$(du -hs $1 | cut -c 1-3)
        let "diff = $cur_size - $tmp_size"
        tmp_size=$(du -hs $1 | cut -c 1-3)
        if (("$diff" > "$start_size"))
        then
                echo "Скачок при добавлении " $i " файла, разница в " $diff " Кбайта"
                start_size=$diff
        fi
done

size=`du -hs $1`
echo "Размер директории после добавления текстовых файлов" $size

#удаляем часть файлов
for i in {1..800}
do
        name=$1/$i.txt
        rm -rf $name
done

size=`du -hs $1`

echo "Размер директории после удаления части текстовых файлов" $size
```

```
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts$ sudo ./fifthscript.sh test/
Текущий размер директории: 1,6M test/
Размер директории после добавления текстовых файлов 4,8M test/
Размер директории после удаления части текстовых файлов 1,7M test/
```

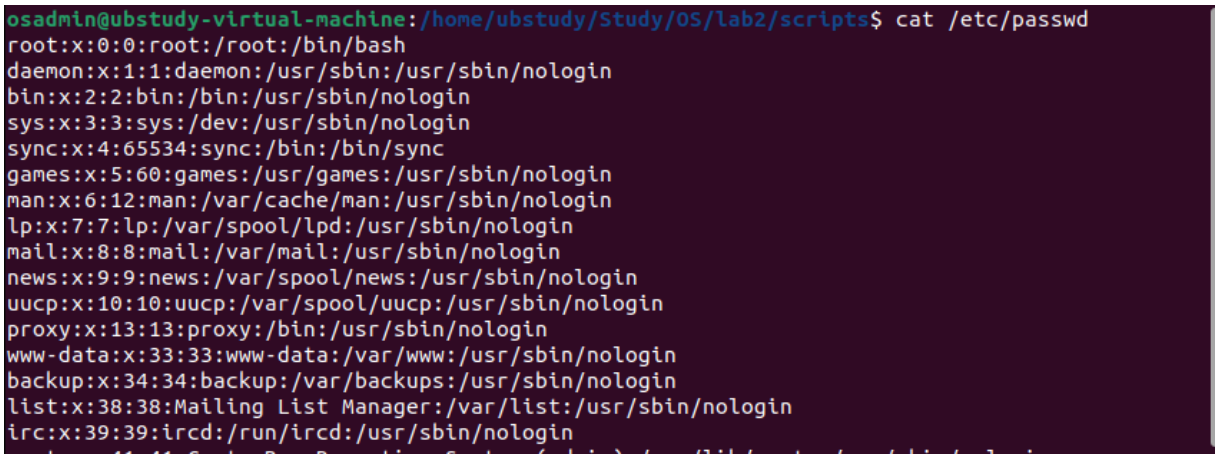
```
#!/bin/bash

#размер директории
size=`du -hs $1`

#выводим размер директории
echo "Текущий размер директории:" $size

#создаем в директории каталоги произвольного диапозона
tmp_size=$(du -hs $1 | cut -c 1-3) #размер директории для сравнения
start_size=4 #начальный размер добавляемой единицы информации
for i in {1..1000}
do
        mkdir $1/$i

#сравниваем размер директории на каждом шаге для отслеживания скачков
        cur_size=$(du -hs $1 | cut -c 1-3)
        let "diff = $cur_size - $tmp_size"
        tmp_size=$(du -hs $1 | cut -c 1-3)
        if (("$diff" > "$start_size"))
        then
                echo "Скачок при добавлении " $i " каталога, разница в " $diff " Кбайта"
                start_size=$diff
        fi
done

size=`du -hs $1`
echo "Размер директории после добавления каталогов" $size

#удаляем часть файлов
for i in {1..800}
do
        rm -rf $1/$i
done

size=`du -hs $1`

echo "Размер директории после удаления части каталогов" $size
```

```
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts$ sudo ./fifthscript_2.sh test/
Текущий размер директории: 820K test/
```

```
Размер директории после добавления каталогов 4,0M test/
Размер директории после удаления части каталогов 820K test/
```

Из вывода программ видно, что скачки при добавлении файлов или каталогов не происходят (начальный размер единицы информации в эксперемете в 4 Кбайта был получен эмперическим путем при наблюдении за ростом размера подопытного каталога при количестве файлов от 1 до 200) .

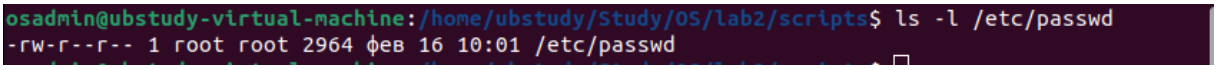
8. Ознакомимся с содержимым /etc/passwd



Данный файл содержим список пользовательских учетных записей в тексстовом формате. Является первым и основным источником информации о правах пользователя и операционной системы.

Файл содержит следующие записи, разделенные двоеточиями:

- Имя пользователя
- Зашированный пароль
- Цифровой идентификатор пользователя (UID)
- Цифровой идентификтор группы пользователя (GID)
- Полное имя пользователя (GECOS)
- Домашний каталог пользователя
- Оболочка входа в систему



Проанализируем права доступа

Файл принадлежит root, доступн для чтения всем пользователям, а для записи только root.

Ознакомимся с содержимым /etc/shadow при помощи sudo.

```
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts$ sudo -s
[sudo] password for osadmin:
root@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts# cat /etc/shadow
root:!:19210:0:99999:7:::
daemon:*:19101:0:99999:7:::
bin:*:19101:0:99999:7:::
sys:*:19101:0:99999:7:::
sync:*:19101:0:99999:7:::
games:*:19101:0:99999:7:::
man:*:19101:0:99999:7:::
lp:*:19101:0:99999:7:::
mail:*:19101:0:99999:7:::
```

Данный файл является зашифрованным файлом паролей, в котором хранится зашифрованная информация о паролях для учетных записей пользователей. В дополнение хранит информацию о сроке действия (истечении) пароля.

Каждая строка представляет информацию о пользователе, состоящая из полей, разделенных двоеточием:

- Логин
- Зашифрованный пароль
- Количество дней, прошедших с 01.01.1970, когда последний раз меняли этот пароль.
- Дней до того, как пароль может быть изменен
- Кол-во дней, по истечении которых пароль необходимо изменить
- За сколько дней до истеченеия срока действия пароля пользователь получает предупрежденеие
- Через сколько дней после истечения срока действия пароля эта учетная запись будет отключена
- Дней с 01.0.1970, когда учетная запись была отключена
- Зарезервированное поле

Воспользуемся утилитой /usr/bin/passwd, которая позволяется менять пароль пользователя.

```
root@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts# cat /etc/shadow | grep
osadmin
osadmin:$y$j9T$7ONglCCpCExTtQMyOzprw1$ZcOVJ8iH.D6QGiock6xBnxrrPZKe/NlZB7wggWtedO5:19418:0
:99999:7:::
root@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts# /usr/bin/passwd osadmin
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: password updated successfully
root@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts# cat /etc/shadow | grep
osadmin
osadmin:$y$j9T$bDNSkQ7Yxpr3sJIuQQKiK0$.vza8domYWfsfYz1R3e.1kpfQUSVCxgOd3Aph/QtmN9:19429:0
:99999:7:::
```

```
root@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts# ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 59976 ноя 24 15:05 /usr/bin/passwd
```

Буква s в утилите означает разрешение SUID, которое позволяет устанавливать идентификтор пользователя, и применяется по умолчанию. При смене пароля пользователь временно получает права root для того, чтобы записывать информацию в файл /etc/shadow.

9.1 Приведем примеры применения утилит chmod и chown к каталогу land с файлами.

```
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts$ sudo chmod 700 land
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts$ ls -l
total 52
lrwxrwxrwx  1 root    root       14 мар  9 11:42 cplinkscripts.sh -> secondsript.sh
-rwxrwxrwx  1 root    root      800 мар  9 12:18 fifthscript.sh
-rwxrwxrwx  1 ubstudy ubstudy   341 мар  8 23:05 firstscript.sh
-rwxrwxrwx  1 root    root      435 мар  9 12:06 fourthscript.sh
drwx------  2 root    root     4096 мар  9 14:18 land
```

Изменение прав так, чтобы чтение, запись и исполнение были доступны только владельцу.

```
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts$ sudo chown osadmin land
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts$ ls -l
total 52
lrwxrwxrwx  1 root    root       14 мар  9 11:42 cplinkscripts.sh -> secondsript.sh
-rwxrwxrwx  1 root    root      800 мар  9 12:18 fifthscript.sh
-rwxrwxrwx  1 ubstudy ubstudy   341 мар  8 23:05 firstscript.sh
-rwxrwxrwx  1 root    root      435 мар  9 12:06 fourthscript.sh
drwx------  2 osadmin root     4096 мар  9 14:18 land
```

После выполенения данной команды владельцем директории является пользователь osadmin.

9.2. Расширим права исполнения эксперементального файла с помощью флага SUID командой chmod.

```
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts/land$ sudo chmod u+s 1.txt
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts/land$ ls -l 1.txt
-rwSr--r-- 1 root root 0 мар  9 14:18 1.txt
```

Буква S сигнализирует о флаге SUID в правах пользователя.

Другие виды расширенных прав:

Права	Числ. значение	Относит. режим	Применение к файлам	Применение к каталогам
SUID	4	u+s	Пользователь выполняет файл с разрешениями владельца файла.	Нет смысла применять
SGID	2	g+s	Пользователь выполняет файл с разрешениями владельца группы.	Файлы, созданные в каталоге, получают одного и того же владельца группы.
sticky bit	1	+t	Нет смысла применять	Запрещает пользователям удалять файлы от других пользователей.

Рассмотрим их в командной строке. Применим флаг SGID к файлам.

```
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts/land$ sudo chmod g+s
2.txt
[sudo] password for osadmin:
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts/land$ ls -l 2.txt
-rw-r-Sr-- 1 root root 0 мар  9 14:18 2.txt
```

Буква S сигнализирует о флаге SGID в правах группы. Применим флаг SGID к каталогу.

```
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts/land$ sudo chmod g+s
tree/
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts/land$ ls -l | grep tr
ee
drwxrwsr-x 2 osadmin osadmin 4096 мар 13 10:52 tree
```

Буква s сигнализирует о флаге SGID в правах группы. Применим флаг sticky bit к каталогу.

```
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts/land$ sudo chmod +t b
each/
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts/land$ ls -l | grep be
ach
drwxrwxr-t 2 osadmin osadmin 4096 мар 13 10:56 beach
```

Буква t сигнализирует о флаге SGID в правах остальных пользователей.

9.3. Эксперементально установим, как формируются итоговые права на использование файла, если права пользователя и группы, в которую он входит, различны. Создадим файл leaf и рассмотрим пользователя osadmin и группу osadmin.

```
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts/land$ ls -l leaf
-rw-rw-r-- 1 osadmin osadmin 13 мар  9 14:39 leaf
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts/land$ cat leaf
leof of tree
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts/land$ whoami >> leaf
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts/land$ cat leaf
leof of tree
osadmin
```

Все операции успешно выполняются. Теперь дадим все права группе и уберем их у владельца.

```
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts/land$ sudo chmod u-rwx leaf
[sudo] password for osadmin:
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts/land$ sudo chmod g+rwx leaf
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts/land$ ls -l leaf
----rwxr-- 1 osadmin osadmin 21 мар  9 14:43 leaf
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts/land$ cat leaf
cat: leaf: Permission denied
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts/land$ whoami >> leaf
bash: leaf: Permission denied
```

После выполенения операций владелец не может что либо сделать, не смотря на принадлежность в группе, так как ему чтение, запись и исполнение запрещены.

10. Разработаем «программу-шлюз» для доступа к файлу дргуго пользователя при отсутствии прав на чтение информации из этого файла.

```c
#include <stdio.h>

int main(int argc, char* argv[]) {
        if (argc <= 1) { //Если не передано аргументов
                printf("%s: не передан входной параметр\n", argv[0]);
        } else {
                FILE* f; //Создаем указатель для работы с файлом
                f = fopen(argv[1], "r"); //Открываем файл для чтения
                if (f) { //Если открыт
                        printf("%s: %s файл открыт\n", argv[0], argv[1]);
                        char str[64]; //Создаем строку для содержимого файла
                        while (fgets(str, sizeof(str), f)) { //Пока можем записывать
                                                             //данные из файла в строку
                                printf("%s", str); //Выводим строку с содержиммым файла
                        }
                        fclose(f); //Закрываем файл
                } else {
                        printf("%s: %s файл не открыт\n", argv[0], argv[1]); //Иначе файл не открыт
                }
        }
        return 0;
}
```

Командой chmod +s ./a.out дадим исполняемому файлу программы разрешение SUID.

```
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts/program$ sudo chmod +s ./a.o
ut
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts/program$ ls -l
total 20
-rwsrwsr-x 1 osadmin osadmin 16136 мар  9 15:08 a.out
-rwxrwxrwx 1 root    root      473 мар  9 15:08 prog.c
```

Создадим файл clear.txt, убрав права чтения у пользователей и сменим текущего пользователя с osadmin на ubstudy.

```
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts/program$ touch clear.txt
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts/program$ ls -l clear.txt
-rw-rw-r-- 1 osadmin osadmin 0 мар  9 15:31 clear.txt
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts/program$ chmod o-r clear.txt
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts/program$ ls -l clear.txt
-rw-rw---- 1 osadmin osadmin 0 мар  9 15:31 clear.txt
osadmin@ubstudy-virtual-machine:/home/ubstudy/Study/OS/lab2/scripts/program$ sudo login ubstudy
Password:
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.19.0-32-generic x86_64)
```

Попробуем прочесть файл без программы и с помощью нее.

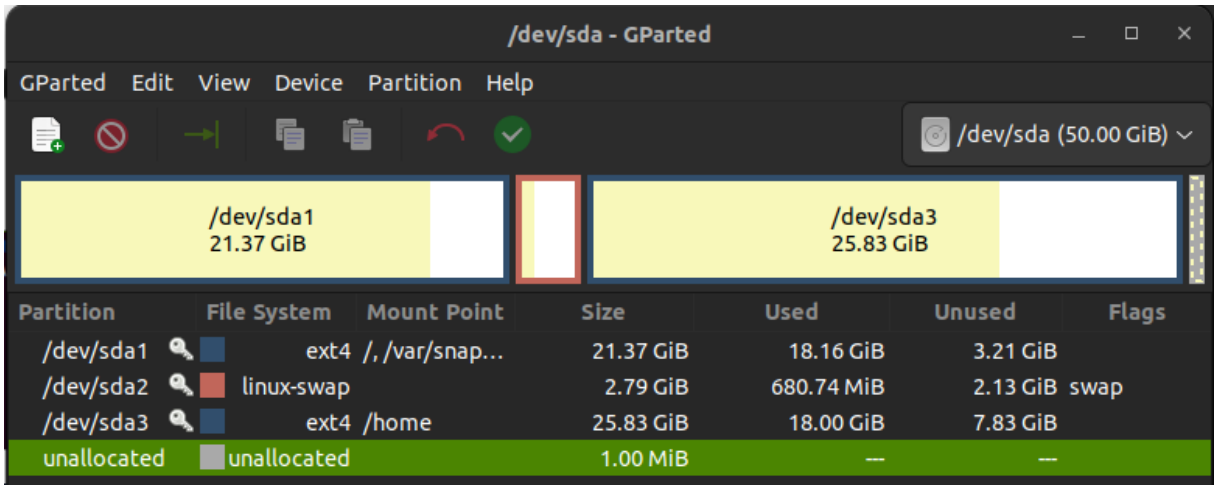
```
ubstudy@ubstudy-virtual-machine:~/Study/OS/lab2/scripts/program$ ./a.out clear.txt
./a.out: clear.txt файл открыт
ubstudy@ubstudy-virtual-machine:~/Study/OS/lab2/scripts/program$ cat clear.txt
cat: clear.txt: Permission denied
```

Нам удалось прочитать содержимое файла clear.txt из-за SUID у исполняемого файла.

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

```
ubstudy@ubstudy-virtual-machine:~/Study/OS/lab2$ df
Filesystem     1K-blocks     Used Available Use% Mounted on
tmpfs             810476     2140    808336   1% /run
/dev/sda1       21883384 18513528   2232912  90% /
tmpfs            4052380    99696   3952684   3% /dev/shm
tmpfs               5120        4      5116   1% /run/lock
/dev/sda3       26550336 18322316   6964060  73% /home
tmpfs             810476      196    810280   1% /run/user/1000
/dev/sr0         3569294  3569294         0 100% /media/ubstudy/Ubuntu 22.04 LTS amd64
```

Команда df выводит доступное пространство на всех разделах. В данном случае 7 ФС. Выводится информация о размере, используемом пространстве, доступном пространстве и точке монтирования. На моей виртаульной машине есть несколько разделов, графически их можно анализировать при помощи утилиты gparted.



Приведем информацию о данной утилите:

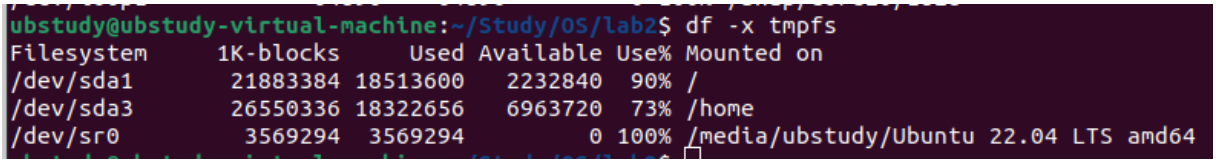
```
GPARTED(8)                          GParted Manual                          GPARTED(8)

NAME
       gparted - GNOME Partition Editor for manipulating disk partitions.

SYNOPSIS
       gparted [device]...

DESCRIPTION
       The  gparted  application is the GNOME partition editor for creating, reorganiz-
       ing, and deleting disk partitions.

       A disk device can be subdivided into one or more partitions.  The gparted appli-
       cation  enables  you to change the partition organization on a disk device while
       preserving the contents of the partition.

       With gparted you can accomplish the following tasks:
       - Create a partition table on a disk device.
       - Enable and disable partition flags such as boot and hidden.
       - Perform actions with partitions such as create, delete, resize,  move,  check,
       label, copy, and paste.

       More documentation can be found in the application help manual, and online at:
       https://gparted.org
```

tmpfs — временное файловое хранилище, предназначенное для монтирования ФС, размещается в ОЗУ вместо физического диска.

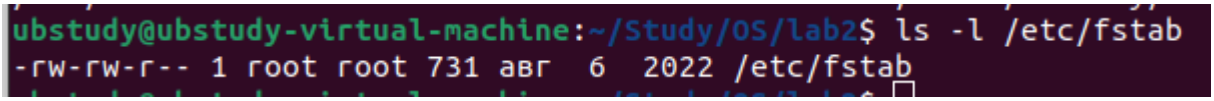
/dev/sda — разделы накопителя, установленнго в ноутбук.

```
ubstudy@ubstudy-virtual-machine:~/Study/OS/lab2$ df -a
df: /run/user/1000/doc: Operation not permitted
Filesystem        1K-blocks     Used Available Use% Mounted on
sysfs                     0        0         0    - /sys
proc                      0        0         0    - /proc
udev                4012652        0   4012652   0% /dev
devpts                    0        0         0    - /dev/pts
tmpfs                810476     2140    808336   1% /run
/dev/sda1          21883384 18513592   2232848  90% /
securityfs                0        0         0    - /sys/kernel/security
tmpfs               4052380    93132   3959248   3% /dev/shm
tmpfs                  5120        4      5116   1% /run/lock
cgroup2                   0        0         0    - /sys/fs/cgroup
pstore                    0        0         0    - /sys/fs/pstore
bpf                       0        0         0    - /sys/fs/bpf
systemd-1                 -        -         -    - /proc/sys/fs/binfmt_misc
```

С помощью флага -a можно получить информацию обо всех файловых системах, изветсных ядру.

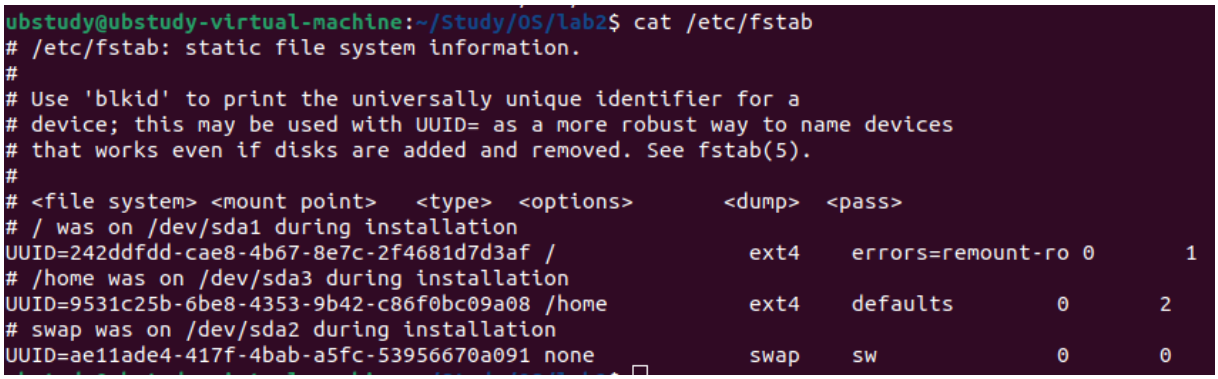


Команда df -x tmpfs выдает информацию про реальные файловые системы на жестком диске.



Тип файла - обычный файл, разрешения для владельца - чтение и запись, разрешение для группы и для всех остальных - только чтение.

fstab - file systems table - один из конфигурационных файлов в UNIX - подобных системах, содержит информацию о ФС и устройствах хранения информации компьютера.



file_system — сообщает mount, что монтировать.

11.2. Проведем 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.

```
ubstudy@ubstudy-virtual-machine:~/Study/OS/lab2$ 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=4012652k,nr_inodes=1003163,mode=755,inode64)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,nodev,noexec,relatime,size=810476k,mode=755,inode64)
/dev/sda1 on / type ext4 (rw,relatime,errors=remount-ro)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev,inode64)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k,inode64)
cgroup2 on /sys/fs/cgroup type cgroup2 (rw,nosuid,nodev,noexec,relatime,nsdelegate,memory_recurs
iveprot)
pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relatime)
bpf on /sys/fs/bpf type bpf (rw,nosuid,nodev,noexec,relatime,mode=700)
```

mount — утилита командной строки в UNIX-подобных опреационных системах. Применяется для монтирования файловых систем.

```
ubstudy@ubstudy-virtual-machine:~/Study/OS/lab2$ cat /etc/mtab
sysfs /sys sysfs rw,nosuid,nodev,noexec,relatime 0 0
proc /proc proc rw,nosuid,nodev,noexec,relatime 0 0
udev /dev devtmpfs rw,nosuid,relatime,size=4012652k,nr_inodes=1003163,mode=755,inode64 0 0
devpts /dev/pts devpts rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000 0 0
tmpfs /run tmpfs rw,nosuid,nodev,noexec,relatime,size=810476k,mode=755,inode64 0 0
/dev/sda1 / ext4 rw,relatime,errors=remount-ro 0 0
securityfs /sys/kernel/security securityfs rw,nosuid,nodev,noexec,relatime 0 0
tmpfs /dev/shm tmpfs rw,nosuid,nodev,inode64 0 0
tmpfs /run/lock tmpfs rw,nosuid,nodev,noexec,relatime,size=5120k,inode64 0 0
cgroup2 /sys/fs/cgroup cgroup2 rw,nosuid,nodev,noexec,relatime,nsdelegate,memory_recursiveprot 0
 0
pstore /sys/fs/pstore pstore rw,nosuid,nodev,noexec,relatime 0 0
bpf /sys/fs/bpf bpf rw,nosuid,nodev,noexec,relatime,mode=700 0 0
```

```
ubstudy@ubstudy-virtual-machine:~/Study/OS/lab2$ cat /etc/mtab | grep /dev/sdb
/dev/sdb1 /media/ubstudy/MANJARO_KDEM_2203 iso9660 ro,nosuid,nodev,relatime,nojoliet,check=s,map
=n,blocksize=2048,uid=1000,gid=1000,dmode=500,fmode=400,iocharset=utf8 0 0
```

```
ubstudy@ubstudy-virtual-machine:~/Study/OS/lab2$ cat /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>
# / was on /dev/sda1 during installation
UUID=242ddfdd-cae8-4b67-8e7c-2f4681d7d3af /               ext4    errors=remount-ro 0       1
# /home was on /dev/sda3 during installation
UUID=9531c25b-6be8-4353-9b42-c86f0bc09a08 /home           ext4    defaults        0       2
# swap was on /dev/sda2 during installation
UUID=ae11ade4-417f-4bab-a5fc-53956670a091 none            swap    sw              0       0
```

mtab — mounted sufle system table — системный файл, в котором прописаны устройства, смонтированные в систему в настоящий момент. При подключении флещ накопителя (а также его монтирования) и применении данной команды, отображается соответствующая строка (/dev/sdb1 /media/ubstudy/MANJARO_KDEM_2203 ...). Стоит отметить, что в fstab изменений нет. Формат таблицы — имя устройства, режим включения, точка монтирования, тип ФС.

```
ubstudy@ubstudy-virtual-machine:~/Study/OS/lab2$ sudo fdisk -l
[sudo] password for ubstudy:
Disk /dev/loop0: 4 KiB, 4096 bytes, 8 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/loop1: 63,29 MiB, 66359296 bytes, 129608 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/loop2: 243,79 MiB, 255635456 bytes, 499288 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/loop3: 116,7 MiB, 122363904 bytes, 238992 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

```
ubstudy@ubstudy-virtual-machine:~/Study/OS/lab2$ sudo fdisk -l | grep /dev/sdb
Disk /dev/sdb: 14,51 GiB, 15581839360 bytes, 30433280 sectors
/dev/sdb1  *         64 6940703 6940640  3,3G  0 Empty
/dev/sdb2       6940704 6948895    8192    4M ef EFI (FAT-12/16/32)
```

Утилита fdisk позволяет посмотреть информацию о дисках, их разделах, промежутки занимаемых разделом секторов, размер блока каждого диска. Информация о флеш накопителе также отображается.

Сделаем так, чтобы нельзя было провести автомонтирование флеш-накопителя. Для этого применим следующую команду.

```
osadmin@ubstudy-virtual-machine:/$    sudo sh -c 'echo ATTRS{removable}=="1", SUBSYSTEMS=
="block", NAME="" > /etc/udev/rules.d/99-block-storage.rules'
[sudo] password for osadmin:
```

Выглядит наше правило:

ATTRS{removable}=="1", SUBSYSTEMS=="block", NAME=""

ATTRS{removable}=="1" - Нужны только отключаемые устройства (что бы жесткий диск случайно не отключить).

SUBSYSTEMS=="block" - Нас интересуют устройства хранения данных.

NAME="" - Собственно действие. Не создавать файл устройства.

```
osadmin@ubstudy-virtual-machine:/$ cat /etc/mtab | grep /dev/sdb
osadmin@ubstudy-virtual-machine:/$ sudo fdisk -l | grep /dev/sdb
```

После выполнения mount и fdisk устройство не было монтировано.

11.3. Проведем «максимально возможное» дерево ФС, проанализируем, где это указывается.

В файле /usr/include/linux/limits.h определена максимальная длина пути.

```
ubstudy@ubstudy-virtual-machine:~/Study/OS/lab2$ cat /usr/include/linux/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
```

Маскимальная длина имени файла 255 байт, маскимальная длина полного пути до файла включая имя 4096 байт. То есть сама вложенность не лимитируется, но длина пути не ограничена.

12.1. Приведем алгоритм функционирования утилиты file 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. Определим, где находятся магические числа и иные характеристики, идентифицирующие тип файла, применительно к исполняемым файлам, а также файлам других типов.

file — команда, которая позволяется узнать тип данных, которые на самом деле содержатся внутри файла. Принцип работы команды следующий:

1) Тестирование на файловую систему

2) Тестирование на магические цифры

3) Языковые тесты

Если темт прошел проверку, то она прерывается и возвращается результат. Информация о тестах на магические числа содержится в файле /usr/share/mime/magic.

```
ubstudy@ubstudy-virtual-machine:~/Study/OS/lab2$ cat /usr/share/mime/magic
MIME-Magic
[90:application/vnd.stardivision.writer]
>2089=
StarWriter
[90:application/x-docbook+xml]
>0=<?xml
1>0=-//OASIS//DTD DocBook XML+101
1>0=-//KDE//DTD DocBook XML+101
[90:image/x-eps]
>0=%!
1>15=EPS
>0=%!
1>16=EPS
>0=◆◆◆◆
[80:application/prs.plucker]
>60DataPlkr
[80:application/vnd.corel-draw]
>8CDRXvrsn&◆◆◆◆◆◆◆
[80:application/x-fictionbook+xml]
>0=
   <FictionBook+257
[80:application/x-mobipocket-ebook]
>60BOOKMOBI
```

Приведем пример работы утилиты file.

```
ubstudy@ubstudy-virtual-machine:~/Study/OS/lab2$ mv results.txt results.png
ubstudy@ubstudy-virtual-machine:~/Study/OS/lab2$ file results.png
results.png: Unicode text, UTF-8 text
```

12.2. Выполним утилиту file с разными ключами.

```
ubstudy@ubstudy-virtual-machine:~/Study/OS/lab2$ file -b results.png
Unicode text, UTF-8 text
```

Ключ -b позволяет не выводит имя файла.

```
ubstudy@ubstudy-virtual-machine:~/Study/OS/lab2$ file -i results.png
results.png: text/plain; charset=utf-8
```

Ключ -i выводит MIME — тип файла.

```
6: > 0 regex/l/100,!^[^Cc \t].*$,""]
0 != 0 = 0
bb=[0x564cab9a2690,957,0], 0 [b=0x564cab9a2690,957,0], [o=0, c=0]
mget(type=20, flag=0x40, offset=0, o=0, nbytes=957, il=0, nc=0)
mget/128 @0: 1. \n\n/bin/\nls -R -l | grep ^- > /home/ubstudy/Study/OS/lab2/commands.txt >>\n-rw
xr-xr-x 1 root root        18456 \321\204\320\265\320\262  7  2021\000

28: > 0 search/1,!p,""]
1 != 0 = 1
bb=[0x564cab9a2690,957,0], 0 [b=0x564cab9a2690,957,0], [o=0, c=1]
mget(type=17, flag=0, offset=0, o=0, nbytes=957, il=0, nc=0)
mget/128 @0: 1. \n\n/bin/\nls -R -l | grep ^- > /home/ubstudy/Study/OS/lab2/commands.txt >>\n-rw
xr-xr-x 1 root root        18456 \321\204\320\265\320\262  7  2021\000

29: >> 0 regex,=^package[ \t]+req,"Tcl script"]
1 == 0 = 0
[try ascmagic 1]
results.png: Unicode text, UTF-8 text
ubstudy@ubstudy-virtual-machine:~/Study/OS/lab2$ file -d results.png
```

Ключ -d выводит сообщения дебага.

12.3. Проведем эксперементальную попытку с добавлением в базу собственного типа файла и его дальнейшей идентификацией.

Откроем /etc/magic с помощью утилиты nano и впишем тип файла hash, который определен магическим числом MB в своем содержании.

```
ubstudy@ubstudy-virtual-machine:~/Study/OS/lab2$ sudo nano /etc/magic
ubstudy@ubstudy-virtual-machine:~/Study/OS/lab2$ cat /etc/magic
# Magic local data for file(1) command.
# Insert here your local magic data. Format is described in magic(5).
0 string MB hash
```

Создадим новый файл с магическим числом и определим его тип.

```
ubstudy@ubstudy-virtual-machine:~/Study/OS/lab2$ touch newfile.void
ubstudy@ubstudy-virtual-machine:~/Study/OS/lab2$ echo "MB" >> newfile.void
ubstudy@ubstudy-virtual-machine:~/Study/OS/lab2$ cat newfile.void
MB
ubstudy@ubstudy-virtual-machine:~/Study/OS/lab2$ file newfile.void
newfile.void: hash
```

Файл опредляется с типом hash, следовательно новый тип успешно добавлен.

Вывод.

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