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. 1303 _____ Беззубов Д.В.

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. В UNIX-системах существуют следующие типы файлов:

Типы файлов в Linux

Типы файлов		Назначение
Обычные файлы	—	Хранение символьных и двоичных данных
Каталоги	d	Организация доступа к файлам
Символьные ссылки	l	Предоставление доступа к файлам, расположенных на любых носителях
Блочные устройства	b	Предоставление интерфейса для взаимодействия с аппаратным обеспечением компьютера
Символьные устройства	c	
Каналы	p	Организация взаимодействия процессов в операционной системе
Сокеты	s	

http://younglinux.info

Рисунок 1 – типы файлов в UNIX-системах

С помощью следующего скрипта отсортируем по одному файлу каждого типа:

```
#! /bin/bash

filetypes="-dcblps"
for (( i=0; i<${#filetypes}; i++ )); do
     exp="^${filetypes:$i:1}"
     `ls -lRa -Imnt -Iwsl / | grep $exp -m 1 -s >>
/root/lb2/file2.txt`
done
```

Результат работы скрипта:

```
-rw-r--r--   1 root root      56 Mar  2 00:38 filetypes.txt
drwxr-xr-x  24 root root    4096 Mar  8 17:45 .
crw-r--r--  1 root root  10, 235 Mar  8 17:45 autofs
brw-------  1 root root   7,   0 Mar  8 17:45 loop0
lrwxrwxrwx   1 root root       7 Aug 20  2021 bin -> usr/bin
srwxrwxrwx 1 root root   0 Mar  8 17:45 1_interop
```

Используя команду *readlink -f* можно найти полное имя некоторого файла. Примеры файлов с необходимым типом:

- Регулярный файл: /filetypes.txt
- Специальный файл блочного устройства: /dev/loop0
- Файл символьного устройства: /dev/autofs
- Директория: /.
- Символьная ссылка: /bin -> /usr/bin
- Сокет: /run/WSL/1_interop

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

```
#! /bin/sh
if [ $# -lt 1 ]
then echo $0: Error: No file
else
      filename=$1
      inode=`ls -i $filename | cut -d ' ' -f 1 | tr -d " "`
      tmp=`ls -lRi -Imnt / | grep $inode >> /root/lb2/file2.txt`
fi
echo $tmp
```

Проходя по директории рекурсивно, находим все файлы, inode которых совпадает с inode файла, переданного в качестве входного параметра.

Результат выполнения:

```
64165 -rw-r--r--  2 root root   28674 Feb 16 16:33 copy.txt
64165 -rw-r--r--  2 root root   28674 Feb 16 16:33 hard_link_copy
```

3. Сформируем символьные ссылки с помощью ln, link и cp для файла copy.txt

```
lrwxrwxrwx  1 root root       8 Mar  8 19:34  first_symb_link -> copy.txt
lrwxrwxrwx  1 root root       8 Mar  8 19:35  second_symb_link -> copy.txt
-rw-r--r--  3 root root   28674 Feb 16 16:33  third_symb_link
```

Как можно заметить, ln -s, cp -s создают символьные ссылки, а link – жесткую

Напишем скрипт, выполняющий подсчет символьных ссылок

```
#! /bin/sh

if [ $# -lt 1 ]

then echo $0: Error: no file
```

```
else
      path=$1
      tmp=`ls -lRa -Imnt / | grep "$path" | grep
^l >> ./file_links.txt`
fi
count=`wc -l ./file_links.txt | cut -d ' ' -f 1`
echo total $count >> ./file_links.txt
```

Проходя рекурсивно по директории, находим файлы, в атрибутах которых есть “l”, ссылающиеся на заданный файл.

Результат выполнения:

```
lrwxrwxrwx  1 root root       8 Mar  8 19:34 first_symb_link -> copy.txt
lrwxrwxrwx  1 root root      8 Mar  8 19:35 second_symb_link -> copy.txt
total 2
```

4. Найдем все символьные ссылки на заданный файл с помощью следующей команды:

```
ls -lRa -Imnt / | grep "copy.txt" | grep ^l
```

Результат выполнения команды:

```
root@DESKTOP-GHA60IS:~/lb2# ls -lRa -Imnt / | grep "copy.txt" | grep ^l
lrwxrwxrwx  1 root root       8 Mar  8 19:34 first_symb_link -> copy.txt
lrwxrwxrwx  1 root root      8 Mar  8 19:35 second_symb_link -> copy.txt
lrwxrwxrwx  1 root root     14 Mar  8 20:31 another_link -> /root/copy.txt
```

5. Изучим утилиту *find*, используя ее ключи получим расширенную информацию о всех типах файлов.

```
root@DESKTOP-GHA60IS:~/lb2# find ./file2.txt -ls
66661    4 -rw-r--r--    1 root   root    923 Mar  8 20:06 ./file2.txt


root@DESKTOP-GHA60IS:~/lb2# find /root/lb2 -depth
/root/lb2/lb1_c++/lb1_c++
/root/lb2/lb1_c++/source.hpp
…
Вывод поддиректорий глубиной больше 5
```

```
root@DESKTOP-GHA60IS:~/lb2# find /root/lb2 -mindepth 5
/root/lb2/lb1_c++/build/.cmake/api/v1
/root/lb2/lb1_c++/build/.cmake/api/v1/reply
…
```

Примеры вложенных команд:

```
root@DESKTOP-GHA60IS:~/lb2# find $(ls | grep "txt")
D.txt
a.txt
file2.txt
file_links.txt
result.txt
test1.txt


root@DESKTOP-GHA60IS:~/lb2# find `ls | grep "txt"`
D.txt
a.txt
file2.txt
file_links.txt
result.txt
test1.txt
```

6. Проанализируем содержимое файла с помощью утилит od.

```
root@DESKTOP-GHA60IS:~/lb2# echo "file" > od_file.txt
root@DESKTOP-GHA60IS:~/lb2# od -tc od_file.txt
0000000   f   i   l   e  \n
0000005

root@DESKTOP-GHA60IS:~/lb2# echo "new_string" >> od_file.txt
root@DESKTOP-GHA60IS:~/lb2# od -tc od_file.txt
0000000   f   i   l   e  \n   n   e   w   _   s   t   r   i   n   g
\n
0000020
```

Команда od с опцией –t с выводит дамп памяти, ассоциированный с указанным файлом, побайтно в восьмеричном коде, заменяя код на символы там, где это возможно.

7. С помощью команды df -i найдем максимальное количество записей в каталоге – оно связано с количеством свободных inode.

```
8. root@DESKTOP-GHA60IS:~/lb2# mkdir nice_os
   root@DESKTOP-GHA60IS:~/lb2# df -i nice_os/
   Filesystem       Inodes IUsed    IFree IUse% Mounted on
   /dev/sdc       16777216 68116 16709100    1% /
```

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
echo "start size" `echo $(du -hs ./) | cut -d ' ' -f 1`
for (( i=0; i<250; i++ )); do
      touch $i.txt | echo "test" >> $i.txt
done
echo "current size" `echo $(du -hs ./) | cut -d ' ' -f 1`
for (( i=0; i<100; i++ )); do
      rm $i.txt
done
echo "final size" `echo $(du -hs ./) | cut -d ' ' -f 1`
```

Результат работы данного скрипта:

```
root@DESKTOP-GHA60IS:~/lb2/nice_os# bash script4
start size 8.0K
current size 1008K
final size 608K
```

8. Содержимое файла etc/passwd:

```
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
```

```
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System
(admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network
Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd
Resolver,,,:/run/systemd:/usr/sbin/nologin
systemd-timesync:x:102:104:systemd Time
Synchronization,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:106::/nonexistent:/usr/sbin/nologin
syslog:x:104:110::/home/syslog:/usr/sbin/nologin
_apt:x:105:65534::/nonexistent:/usr/sbin/nologin
tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false
uuidd:x:107:112::/run/uuidd:/usr/sbin/nologin
tcpdump:x:108:113::/nonexistent:/usr/sbin/nologin
sshd:x:109:65534::/run/sshd:/usr/sbin/nologin
landscape:x:110:115::/var/lib/landscape:/usr/sbin/nologin
pollinate:x:111:1::/var/cache/pollinate:/bin/false
```

Здесь записана информация о пользователях в следующем формате:

```
<Имя пользователя>:<пароль>:<UID>:<GID>:<комментарии>:<домашний каталог>:<интерпретатор shell>
```

Он доступен для чтения всем пользователям, на запись только администратору. В целях безопасности пароли в нём не хранятся, для них есть отдельный зашифрованный файл /etc/shadow, который доступен для чтения и записи только администратору. Для того, чтобы обычный пользователь мог изменить свой пароль, существует утилита /usr/bin/passwd, которая доступна администратору и группе root на чтение, запись и исполнение, а всем остальным – на чтение и исполнение. Эта программа может выполнять действия от имени администратора, независимо от того, кто её запустил.

Ознакомимся с содержимым /etc/shadow, включив права суперпользователя (т.к. он доступен для чтения только root)

```
root:*:18858:0:99999:7:::
daemon:*:18858:0:99999:7:::
bin:*:18858:0:99999:7:::
sys:*:18858:0:99999:7:::
sync:*:18858:0:99999:7:::
games:*:18858:0:99999:7:::
man:*:18858:0:99999:7:::
lp:*:18858:0:99999:7:::
mail:*:18858:0:99999:7:::
news:*:18858:0:99999:7:::
```

9. С помощью утилиты chmod установим права на чтение, запись и выполнение только для владельца директории, сделаем это рекурсивно с помощью флага -R

```
root@DESKTOP-GHA60IS:~/lb2# chmod 700 -R ./os_lab/
root@DESKTOP-GHA60IS:~/lb2# ll ./os_lab/
total 12
drwx------  3 root root 4096 Feb 16 16:40 ./
drwxr-xr-x 18 root root 4096 Mar  8 22:51 ../
drwx------  2 root root 4096 Feb 16 16:40 test/
```

Изменим права доступа, добавив такие же права группе, которой принадлежит директория.

```
root@DESKTOP-GHA60IS:~/lb2# chmod 770 -R ./os_lab/
root@DESKTOP-GHA60IS:~/lb2# ll ./os_lab/
total 12
drwxrwx---  3 root root 4096 Feb 16 16:40 ./
drwxr-xr-x 18 root root 4096 Mar  8 22:51 ../
drwxrwx---  2 root root 4096 Feb 16 16:40 test/
```

Теперь изменим владельца директории и поддиректорий

```
root@DESKTOP-GHA60IS:~/lb2# chown -R DaniLBez ./os_lab/
root@DESKTOP-GHA60IS:~/lb2# ll ./os_lab/
total 12
drwxrwx---  3 DaniLBez root 4096 Feb 16 16:40 ./
drwxr-xr-x 18 root     root 4096 Mar  8 22:51 ../
drwxrwx---  2 DaniLBez root 4096 Feb 16 16:40 test/
```

Создадим в директории файл и расширим права исполнения для суперпользователя с помощью флага SUID

```
root@DESKTOP-GHA60IS:~/lb2/os_lab# touch test_file.txt
root@DESKTOP-GHA60IS:~/lb2/os_lab# chmod u+s ./test_file.txt
root@DESKTOP-GHA60IS:~/lb2/os_lab# ll
total 12
drwxrwx---  3 DaniLBez root 4096 Mar  9 01:48 ./
drwxr-xr-x 18 root     root 4096 Mar  8 22:51 ../
drwxrwx---  2 DaniLBez root 4096 Feb 16 16:40 test/
-rwSr--r--  1 root     root    0 Mar  9 01:48 test_file.txt
```

Права доступа работают следующим образом:

1. Оболочка проверяет, являетесь ли вы владельцем файла, к которому вы хотите получить доступ. Если вы являетесь этим владельцем, вы получаете разрешения и оболочка прекращает проверку.
2. Если вы не являетесь владельцем файла, оболочка проверит, являетесь ли вы участником группы, у которой есть разрешения на этот файл. Если вы являетесь участником этой группы, вы получаете доступ к файлу с

разрешениями, которые для группы установлены, и оболочка прекратит проверку.

3. Если вы не являетесь ни пользователем, ни владельцем группы, вы получаете права других пользователей (Other).

10. Программа-«шлюз»

```
#include <iostream>
#include <fstream>

int main(){

std::ifstream file("./secret_file.txt");
if(file.is_open()){
        std::string pswd;
        std::getline(file, pswd);
        std::cout << pswd << std::endl;
}else{
std::cout << "access denied" << std::endl;
}
file.close();
return 0;
}
```

Для файла ./secret_file.txt заберем права доступа для всех. В таком случае при попытке открыть данный файл выводится сообщение об отказе в доступе.

```
daniilbez@daniilbez-VirtualBox:~$ cat secret
cat: secret: Отказано в доступе
daniilbez@daniilbez-VirtualBox:~$
```

Попробуем запустить программу без установки файла SUID:

```
daniilbez@daniilbez-VirtualBox:~$ ./heck
Access denied
daniilbez@daniilbez-VirtualBox:~$
```

Затем с помощью флага SUID добавим программе права суперпользователя и запустим ее вновь:

```
daniilbez@daniilbez-VirtualBox:~$ sudo chmod u+s ./heck
daniilbez@daniilbez-VirtualBox:~$ ./heck
123456
```

Таким образом, благодаря правам суперпользователя мы получили доступ к файлу, доступ к которому не имел никто.

11. 1) С помощью утилиты df проанализируем занятое дисковое пространство. По умолчанию дисковое пространство измеряется в количестве 512-байтных блоков. Опция –k используется для измерения пространства количеством 1024-байтных блоков. Опция –h используется для измерения пространства в единицах, удобных для чтения человеком (байты, килобайты и т.д.). Опция –P используется для отображения заголовков столбцов таблицы. С помощью ключей -a и -T выведем информацию о файловых системах на компьютере

```
daniilbez@daniilbez-VirtualBox:~$ df
Файл.система    1K-блоков Использовано Доступно Использовано% Смонтировано в
udev               473896           0   473896            0% /dev
tmpfs              100884        5144    95740            6% /run
/dev/sda1         9204224     4453524  4260104           52% /
tmpfs              504420         212   504208            1% /dev/shm
tmpfs                5120           4     5116            1% /run/lock
tmpfs              504420           0   504420            0% /sys/fs/cgroup
tmpfs              100884         120   100764            1% /run/user/1000
tmpfs              100884          56   100828            1% /run/user/108
```

В данном случае 8 файловых систем. Выводится информация о размере, используемом пространстве, доступном пространстве и точке монтирования.

```
daniilbez@daniilbez-VirtualBox:~$ sudo fdisk -l
[sudo] пароль для daniilbez:
Диск /dev/sda: 10 GiB, 10737418240 байтов, 20971520 секторов
Единицы измерения: секторов из 1 * 512 = 512 байтов
Размер сектора (логический/физический): 512 байт / 512 байт
I/O size (minimum/optimal): 512 bytes / 512 bytes
Тип метки диска: dos
Идентификатор диска: 0xb41d1be3

Устр-во    Загрузочный    Start Конец Секторы  Size Id Тип
/dev/sda1  *               2048 18970623 18968576    9G 83 Linux
/dev/sda2             18972670 20969471  1996802  975M  5 Расширенный
/dev/sda5             18972672 20969471  1996800  975M 82 Linux своп / Solaris
```

Проанализируем ФС на компьютере с помощью утилиты fdisk. Данная утилита предназначена для управления разделами жёсткого диска. С помощью ключа -l получим информацию об установленных дисках и разделах. Как мы видим – установлен 1 физический диск, разделенный на 3 логических раздела, загрузочным из них является /dev/sda1.

Затем проанализируем файл /etc/fstab

```
root@DESKTOP-GHA60IS:/# cat ./etc/fstab
LABEL=cloudimg-rootfs   /        ext4   defaults        0 1
```

Первое поле – метка показывающая, что монтировать.

Второе поле показывает, что монтировать нужно в корневой каталог.

Третье поле указывает тип монтируемой файловой системы.

Четвертое – опции монтирования, в данном случае никаких опций нет, default выполняет роль «заглушки».

Пятое - используется утилитой dump для того, чтобы определить, когда делать резервную копию.

Шестое – указывает утилите fsck в каком порядке проверять файловую систему, «1» - наибольший приоритет.

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.

```
danillbez@danillbez-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=473896k,nr_inodes=118474,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=100884k,mode=755)
/dev/sda1 on / type ext4 (rw,relatime,errors=remount-ro,data=ordered)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noexec,relatime,xattr,release_agent=/lib/systemd/systemd-cgroups-agent,name=systemd)
pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relatime)
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf_event)
cgroup on /sys/fs/cgroup/rdma type cgroup (rw,nosuid,nodev,noexec,relatime,rdma)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpuset)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,cpu,cpuacct)
cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,devices)
cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,memory)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,hugetlb)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (rw,nosuid,nodev,noexec,relatime,net_cls,net_prio)
cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer)
cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,blkio)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=31,pgrp=1,timeout=0,minproto=5,maxproto=5,direct,pipe_ino=12625)
mqueue on /dev/mqueue type mqueue (rw,relatime)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,pagesize=2M)
configfs on /sys/kernel/config type configfs (rw,relatime)
fusectl on /sys/fs/fuse/connections type fusectl (rw,relatime)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=100884k,mode=700,uid=1000,gid=1000)
gvfsd-fuse on /run/user/1000/gvfs type fuse.gvfsd-fuse (rw,nosuid,nodev,relatime,user_id=1000,group_id=1000)
tmpfs on /run/user/108 type tmpfs (rw,nosuid,nodev,relatime,size=100884k,mode=700,uid=108,gid=114)
binfmt_misc on /proc/sys/fs/binfmt_misc type binfmt_misc (rw,relatime)
```

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

mtab — mounted sufle system table — системный файл, в котором прописаны устройства, смонтированные в систему в настоящий момент.

При подключении флеш-накопителя (а также его монтирования) и применении данной команды, отображается соответствующая строка (/dev/sdb1 /media/daniilbez/Transcend ...). Стоит отметить, что в fstab изменений нет. Формат таблицы — имя устройства, режим включения, точка монтирования, тип ФС.

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

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

```
daniilbez@daniilbez-VirtualBox:~$ cat /usr/include/linux/limits.h
#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 ограничена.

12. *File* определяет тип файла. Для этого она выполняет разные тесты, которые можно разделить на 3 группы:

- Filesystem tests – основаны на анализе кода возврата системного вызова stat(). Программа проверяет не пустой ли файл, и не принадлежит ли он 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 – используются для проверки файлов, данные в 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/misc/magic.
- Language tests – используются для анализа языка, на котором написан файл, если это файл в формате ASCII. Выполняется поиск стандартных строк, которые могут соответствовать определённому языку.

Первый тест, который завершится успешно, выводит тип файла. Типы файлов можно разделить на 3 основные группы:

- Текстовые – файл содержит только ASCII символы и может быть безопасно прочитан на терминале.
- Исполняемые – файл содержит результаты компилирования программы в форме понятной ядру ОС.
- Данные – всё, что не подходит в первые 2 группы (обычно это бинарные или непечатаемые файлы). Исключение составляют well-known форматы, используемые для хранения бинарных данных.

  Синтаксис:

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

  Опции:

- -b – не выводить имя файла перед его типом;
- -m magicfile – определяет альтернативный файл с магическими числами;
- -c – обычно используется для дебага нового файла с магическими числами перед его использованием;
- -f namefile – определить типы файлов, имена которых записаны в файле namefile;
- -L – определять типы файлов, на которые ссылаются заданные символические ссылки, а не типы ссылок;
- -n – выводить имя файла перед его типом (по умолчанию);
- -v – вывести версию программы и выйти;
- -z – пытаться смотреть внутри сжатых файлов.

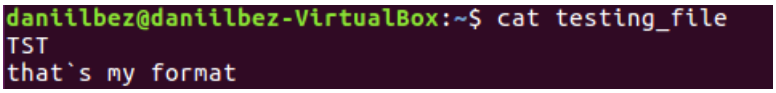
Запуск утилиты с разными ключами:

```
root@DESKTOP-GHA60IS:~# file -b test.cpp
C++ source, ASCII text
root@DESKTOP-GHA60IS:~# file -n test.cpp
test.cpp: C++ source, ASCII text
root@DESKTOP-GHA60IS:~# file -c test.cpp
cont    offset  type    opcode  mask    value   desc
```

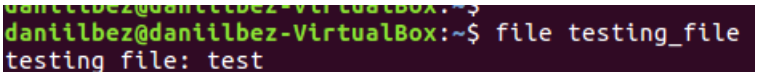
Проведем экспериментальную попытку с добавлением в базу собственного типа файла и его дальнейшей идентификацией. Откроем

/etc/magic с помощью утилиты nano и впишем тип файла test, который определен строкой TST в своем содержании.

Создадим файл со следующим содержанием:



С помощью утилиты *file* узнаем тип данного файла.



Так как тип файла определился как test – свой собственный тип данных был добавлен успешно

Вывод

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