Санкт-Петербургский Политехнический Университет Петра Великого

Кафедра компьютерных систем и программных технологий

Отчёт по лабораторной работе №2

Курс: «Операционные системы»

Тема: «Файловые системы»

Выполнил студент:

Бояркин Никита Сергеевич
Группа: 43501/3

Проверил:

Душутина Елена Владимировна

Санкт-Петербург
2016 г.

Содержание

Лабораторная работа №2

1.1 Цель работы

- Изучение принципов написания скриптов.
- Ознакомление с файловой системой.
- Изучение информации о правах владения и доступа к файлу.
- Изучение информации о точках монтирования.

1.2 Программа работы

1. Ознакомиться с типами файлов исследуемой ФС.
2. Получить все жесткие ссылки на заданный файл.
3. Проанализировать все возможные способы формирования символьных ссылок.
4. Получить все символьные ссылки на заданный в качестве входного параметра файл, не используя file.
5. Изучить утилиту find, используя ее ключи получить расширенную информацию о всех типах файлов.
6. Проанализировать содержимое заголовка файла, а также файла-каталога.
7. Определить максимальное количество записей в каталоге.
8. Ознакомиться с содержимым /etc/passwd, /etc/shadow, с утилитой /usr/bin/passwd.
9. Исследовать права владения и доступа, а также их сочетаемость.
   (a) Привести примеры применения утилит chmod, chown к специально созданному для этих целей отдельному каталогу с файлами.
   (b) Расширить права исполнения экспериментального файла с помощью флага SUID.
   (c) Экспериментально установить, как формируются итоговые права на использование файла, если права пользователя и группы, в которую он входит, различны.
   (d) 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.
11. Применяя утилиту df, а также информационные файлы типа fstab, получить информацию о файловых системах, возможных для монтирования.
    (a) Привести информацию об исследованных утилитах и информационных файлах с анализом их содержимого и форматов.
    (b) 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.
    (c) Привести «максимально возможное» дерево ФС, проанализировать, где это указывается
12. Проанализировать и пояснить принцип работы утилиты file.
    (a) Привести алгоритм её функционирования на основе информационной базы, размещение и полное имя которой указывается в описании утилиты в технической документации ОС.

(b) Утилиту file выполнить с разными ключами.

(c) Привести экспериментальную попытку с добавлением в базу собственного типа файла и его дальнейшей идентификацией.

1.3 Характеристики системы

Некоторая информация об операционной системе и текущем пользователе:

```
nikita@nikita-pc:~$ who
nikita   tty7         2016-11-01 10:08 (:0)
nikita@nikita-pc:~$ cat /proc/version
Linux version 4.4.0-31-generic (buildd@lgw01-16) (gcc version 5.3.1 20160413 (Ubuntu
    5.3.1-14ubuntu2.1) ) #50-Ubuntu SMP Wed Jul 13 00:07:12 UTC 2016
```

Информация об операционной системе и текущем пользователе компьютера в лаборатории:

```
g4081_12@SPOComp7:~$ who
g4081_12   tty7         2016-11-18 12:10 (:0)
g4081_12@SPOComp7:~$ cat /proc/version
Linux version 2.6.35-30-generic-pae (buildd@vernadsky) (gcc version 4.4.5 (Ubuntu/Linaro
    4.4.4-14ubuntu5) ) #56-Ubuntu SMP Mon Jul 11 21:51:12 UTC 2011
```

На домашнем и лабораторном компьютерах установлены реальные системы.

1.4 Ход работы

1.4.1 Фильтрация по одному примеру каждого типа файла

Решение в командной строке

Разработаем команду, которая выведет по одному примеру каждого типа файла из корневого каталога:

```
nikita@nikita-pc:~$ sudo ls / -l -R 2>/dev/null | awk '{if ($0~/^\//) path=substr($0, 0,
    length($0)); else { if($0~/^l/) $(NF-2)=path"/"$(NF-2); else {$NF=path"/"$NF} print $0
    } }' | grep -v ^/ | sort -k1.1,1.1 | uniq -w1
-r--r----- 1 root root 745 мар 30 2016 /etc/sudoers
brw-rw---- 1 root disk 1, 0 окт 22 09:43 /dev/ram0
c--------- 1 root root 5, 2 окт 22 09:43 /dev/pts/ptmx
drwx------ 2 root lp 4096 апр 21 2016 /etc/cups/ssl
lrwxrwxrwx 1 nikita nikita 10 июл 30 05:44 ControlPanel -> /home/nikita/files/software/
    altera/13.1/quartus/linux/jre/bin/jcontrol
prw------- 1 root root 0 окт 22 09:43 /run/systemd/inhibit/1.ref
srw-rw-rw- 1 root root 0 окт 22 09:43 /run/acpid.socket
```

Рассмотрим команду подробно:

- *ls / -l -R* - устанавливаем рекурсивный поиск по корневому каталогу с выводом полной информации.
- *2>/dev/null* - перенаправление потока ошибок в никуда.
- *awk* - скрипт, который добавляет полный путь в название файла.
- *if ($0~/^\//) path=substr($0, 0, length($0));* - если строка начинается с / (каталог), то сохраняем текущий путь в переменную.
- *else { if($0~/^1/) $(NF-2)=path"/"$(NF-2);* - иначе, если это символьная ссылка (начинается с l), то изменяем путь в столбце (NF-2).
- *else {$NF=path"/"$NF} print $0} }* - иначе заменяем путь в последнем столбце (NF).
- *grep -v ^/* - избавляемся от вывода каталогов.
- *sort -k1.1,1.1* - сортировка по первому символу.
- *uniq -w1* - уникальность по первому символу.

В результате работы команды были получены типы файлов с префиксами *-, b, c, d, l, p, s*. Рассмотрим каждый префикс подробнее:

- - файл, обеспечивает хранение символьных и двоичных данных.
- *b* - блочное устройство, обеспечивает обращение к аппаратному обеспечению компьютера. Пример блочного устройства - жесткий диск.
- *c* - символьное устройство, обеспечивает обращение к аппаратному обеспечению компьютера. Пример символьного устройства - терминал.
- *d* - каталог, обеспечивает организацию доступа к файлам.
- *l* - символьная ссылка, обеспечивает предоставление доступа к файлам, расположенным на любых носителях.
- *p* - канал (FIFO), обеспечивает организацию взаимодействия процессов в операционной системе.
- *s* - сокет, обеспечивает организацию взаимодействия процессов в операционной системе.

Скрипт был запущен на компьютере в лаборатории, ошибок в работе скрипта не возникало.

Решение в виде bash скрипта

Решение аналогично предыдущему пункту, однако оформлено в виде *bash* скрипта. Отличие заключается в получении имени файла из аргументов командной строки и запись решения в этот файл:

```
#!/bin/bash

# Если пользователь указал имя результирующего файла, то используем его

filename=$1
if [ -z $filenmame ]; then
  # Если имя файла не указано, то используем имя по умолчанию
  filename="1.out"
fi

# Исполняем команду из предыдущего пункта, перенаправляя вывод в файл

ls / -l -R 2>/dev/null | awk '{if ($0~/^\//) path=substr($0, 0, length($0)); else { if($0
    ~/^l/) $(NF-2)=path"/"$(NF-2); else {$NF=path"/"$NF} print $0} }' | grep -v ^/ | sort
    -k1.1,1.1 | uniq -w1 > $filename
```

Запуск скрипта на исполнение происходит следующим образом:

```
nikita@nikita-pc:~/temp1$ sudo sh 1.sh filename
```

В папке со скриптом создался файл *filename*, в котором находится результат работы скрипта, аналогичный предыдущему пункту.

Скрипт был запущен на компьютере в лаборатории, ошибок в работе скрипта не возникало.

1.4.2 Получение всех жестких ссылок на файл

С помощью использования индексного дескриптора найдем все ссылки на указанный файл:

```
#!/bin/bash

# Если пользователь не указал имя файла, то выходим с ошибкой

filename=$1;
if [ -z $filename ]; then
  exit 1
fi

# Получим индексный дескриптор указанного файла, если не получилось, то выходим с ошибкой

inode="$(ls -i $filename | awk '{print $1}')"

```

```
14 if [ -z $inode ]; then
15   exit 1
16 fi
17
18 # Рекурсивно ищем все жесткие ссылки на индексный дескриптор в домашнем каталоге пользователя
19
20 ls $HOME -l -R -i | grep ^$inode
```

Результаты работы скрипта:

```
1  nikita@nikita-pc:~/temp1$ >tempfile
2  nikita@nikita-pc:~/temp1$ mkdir tempdir
3  nikita@nikita-pc:~/temp1$ ln tempfile tempdir/link_to_tempfile
4  nikita@nikita-pc:~/temp1$ ln tempfile link_to_tempfile
5  nikita@nikita-pc:~/temp1$ ls -l
6  total 20
7  (...)
8  -rw-rw-r-- 3 nikita nikita    0 окт 18 11:01 link_to_tempfile
9  drwxrwxr-x 2 nikita nikita 4096 окт 18 11:02 tempdir
10 -rw-rw-r-- 3 nikita nikita    0 окт 18 11:01 tempfile
11 nikita@nikita-pc:~/temp1$ sudo sh 2.sh tempfile
12 11278137 -rw-rw-r-- 3 nikita nikita    0 окт 18 11:01 link_to_tempfile
13 11278137 -rw-rw-r-- 3 nikita nikita    0 окт 18 11:01 tempfile
14 11278137 -rw-rw-r-- 3 nikita nikita    0 окт 18 11:01 link_to_tempfile
```

Проверим работу скрипта на лабораторном компьютере:

```
1  g4081_12@SPOComp7:~/Boyarkin/2$ >tempfile
2  g4081_12@SPOComp7:~/Boyarkin/2$ mkdir folder
3  g4081_12@SPOComp7:~/Boyarkin/2$ ln tempfile folder/tempfile_link
4  g4081_12@SPOComp7:~/Boyarkin/2$ ln tempfile tempfile_link
5  g4081_12@SPOComp7:~/Boyarkin/2$ ls -l
6  total 164
7  (...)
8  -rw-r--r-- 3 g4081_12 g4081_12    0 2016-12-02 14:22 tempfile_link
9  drwxr-xr-x 2 g4081_12 g4081_12 4096 2016-12-02 14:23 folder
10 -rw-r--r-- 3 g4081_12 g4081_12    0 2016-12-02 14:22 tempfile
11 g4081_12@SPOComp7:~/Boyarkin/2$ sudo sh 2.sh tempfile
12 8391544 -rw-r--r-- 3 g4081_12 g4081_12    0 2016-12-02 14:22 tempfile_link
13 8391544 -rw-r--r-- 3 g4081_12 g4081_12    0 2016-12-02 14:22 tempfile
14 8391544 -rw-r--r-- 3 g4081_12 g4081_12    0 2016-12-02 14:22 tempfile_link
```

Скрипт успешно нашел все ссылки жесткие ссылки на файл.

1.4.3 Анализ всех способов формирования ссылок

Рассмотрим действия команд *link, ln, ln -s, cp*. С помощью команды *ls -l* выясним какого рода объекты они порождают:

```
1  nikita@nikita-pc:~/temp1$ >tempfile
2  nikita@nikita-pc:~/temp1$ link tempfile templink
3  nikita@nikita-pc:~/temp1$ ls -l
4  (...)
5  -rw-rw-r-- 2 nikita nikita   0 окт 18 11:49 tempfile
6  -rw-rw-r-- 2 nikita nikita   0 окт 18 11:49 templink
7  nikita@nikita-pc:~/temp1$ rm templink tempfile
8
9  nikita@nikita-pc:~/temp1$ >tempfile
10 nikita@nikita-pc:~/temp1$ ln tempfile templink
11 nikita@nikita-pc:~/temp1$ ls -l
12 (...)
13 -rw-rw-r-- 2 nikita nikita   0 окт 18 11:49 tempfile
14 -rw-rw-r-- 2 nikita nikita   0 окт 18 11:49 templink
15 nikita@nikita-pc:~/temp1$ rm templink tempfile
16
17 nikita@nikita-pc:~/temp1$ >tempfile
18 nikita@nikita-pc:~/temp1$ ln -s tempfile templink
```

```
nikita@nikita-pc:~/temp1$ ls -l
(...)
-rw-rw-r-- 1 nikita nikita    0 окт 18 11:59 tempfile
lrwxrwxrwx 1 nikita nikita    8 окт 18 11:59 templink -> tempfile
nikita@nikita-pc:~/temp1$ rm templink tempfile

nikita@nikita-pc:~/temp1$ >tempfile
nikita@nikita-pc:~/temp1$ cp tempfile templink
nikita@nikita-pc:~/temp1$ ls -l
(...)
-rw-rw-r-- 1 nikita nikita    0 окт 18 12:02 tempfile
-rw-rw-r-- 1 nikita nikita    0 окт 18 12:02 templink
nikita@nikita-pc:~/temp1$ rm templink tempfile
```

Сделаем вывод о назначении команд *link, ln, ln -s, cp*:

- *link* - позволяет создавать только жесткие ссылки.
- *ln* - без ключей создает жесткую ссылку на файл.
- *ln -s* - с ключем *-s* создает символьную ссылку на файл.
- *cp* - создает новый файл.

Вывод всех полноименных символьных ссылок на файл

Напишем скрипт подсчитывающий все полноименные символьные ссылки на указанный файл:

```
#!/bin/bash

# Если пользователь не указал имя файла, то выходим с ошибкой

filename=$1;
if [ -z $filename ]; then
  exit 1
fi

# Рекурсивно ищем все символьные ссылки на файл в домашнем каталоге пользователя, отсеивая поток
    ошибок и добавляя полный путь файла

ls $HOME -l -R 2>/dev/null | awk '{if ($0~/^\//) path=substr($0, 0, length($0)); else {
    if($0~/^l/) $(NF-2)=path"/"$(NF-2); else {$NF=path"/"$NF} print $0} }' | grep '\-> '$1
```

Алгоритм работы скрипта, который добавляет полный путь файла и алгоритм перенаправления потока ошибок идентичен скрипту фильтрации каждого типа файла из начала работы.

Результат работы скрипта:

```
nikita@nikita-pc:~/temp1$ >tempfile
nikita@nikita-pc:~/temp1$ ln -s tempfile tempfolder/dddd
nikita@nikita-pc:~/temp1$ ln -s tempfile gggg
nikita@nikita-pc:~/temp1$ ln -s tempfile ../yyyy
nikita@nikita-pc:~/temp1$ sh 3.sh tempfile
lrwxrwxrwx 1 nikita nikita     8 окт 18 12:30 /home/nikita/yyyy -> tempfile
lrwxrwxrwx 1 nikita nikita     8 окт 18 12:30 /home/nikita/temp1/gggg -> tempfile
lrwxrwxrwx 1 nikita nikita     8 окт 18 12:30 /home/nikita/temp1/tempfolder/dddd ->
    tempfile
```

Проверим работу скрипта на лабораторном компьютере:

```
g4081_12@SPOComp7:~/Boyarkin/2$ >tempfile
g4081_12@SPOComp7:~/Boyarkin/2$ ln -s tempfile folder/aaaa
g4081_12@SPOComp7:~/Boyarkin/2$ ln -s tempfile bbbb
g4081_12@SPOComp7:~/Boyarkin/2$ ln -s tempfile ../cccc
g4081_12@SPOComp7:~/Boyarkin/2$ sh 3.sh tempfile
lrwxrwxrwx 1 g4081_12 g4081_12 8 2016-12-02 14:29 /home/Boyarkin/cccc -> tempfile
lrwxrwxrwx 1 g4081_12 g4081_12 8 2016-12-02 14:29 /home/Boyarkin/2/bbbb -> tempfile
lrwxrwxrwx 1 g4081_12 g4081_12 8 2016-12-02 14:29 /home/Boyarkin/2/folder/aaaa ->
    tempfile
```

Скрипт успешно нашел все ссылки символьные ссылки на заданный файл.

1.4.4 Вывод всех символьных ссылок на файл

Напишем скрипт подсчитывающий все символьные ссылки на указанный файл:

```
#!/bin/bash

# Если пользователь не указал имя файла, то выходим с ошибкой

filename=$1;
if [ -z $filename ]; then
  exit 1
fi

# Рекурсивно ищем все символьные ссылки на файл в домашнем каталоге пользователя, отсеивая поток ошибок

ls $HOME -l -R 2>/dev/null | grep '\-> '$1
```

Результат работы скрипта:

```
nikita@nikita-pc:~/temp1$ >tempfile
nikita@nikita-pc:~/temp1$ ln -s tempfile tempfolder/dddd
nikita@nikita-pc:~/temp1$ ln -s tempfile gggg
nikita@nikita-pc:~/temp1$ ln -s tempfile ../yyyy
nikita@nikita-pc:~/temp1$ sh 4.sh tempfile
lrwxrwxrwx 1 nikita nikita    8 окт 18 12:30 yyyy -> tempfile
lrwxrwxrwx 1 nikita nikita    8 окт 18 12:30 gggg -> tempfile
lrwxrwxrwx 1 nikita nikita    8 окт 18 12:30 dddd -> tempfile
```

Скрипт был запущен на компьютере в лаборатории, ошибок в работе скрипта не возникало.

1.4.5 Утилита find

find - утилита для поиска файлов по имени и другим свойствам в UNIX-подобных ОС. Может проводить поиск в одном или нескольких каталогах, с использованием критериев, заданных пользователем. По умолчанию возвращает все файлы в рабочей директории. также *find* позволяет применять действия ко всем найденным файлам.

Рассмотрим возможности команды *find* с несколькими ключами:

```
nikita@nikita-pc:~/temp1$ ls
0.log  1.log  1.sh  2.log  2.sh  3.log  3.sh  5.log
nikita@nikita-pc:~/temp1$ find
.
./2.sh
./5.log
./2.log
./1.log
./3.sh
./3.log
./0.log
./tempfolder
./1.sh
nikita@nikita-pc:~/temp1$ find -type d
.
./tempfolder
nikita@nikita-pc:~/temp1$ find -name "*.log"
./5.log
./2.log
./1.log
./3.log
./0.log
nikita@nikita-pc:~/temp1$ find -name "3*"
./3.sh
./3.log
nikita@nikita-pc:~/temp1$ ls ../temp
cat.txt       file_name    gdbtest    my_open.c  prog1.c    v1
defigned.txt  file_name.c  gdbtest.c  pipe       res.txt    v2
```

```
eqv             foo.c         libfoo.so  pipe.c      result
example         foo.o         my_open    prog        script.sh
nikita@nikita-pc:~/temp1$ find ../temp -name "*.txt"
../temp/cat.txt
../temp/defigned.txt
../temp/res.txt
nikita@nikita-pc:~/temp1$ find -size +500c
.
./2.sh
./5.log
./2.log
./3.log
./1.sh
nikita@nikita-pc:~/temp1$ find -size -500c
./1.log
./3.sh
./0.log
nikita@nikita-pc:~/temp1$ find -size -500c -exec ls -l {} \;
-rw-rw-r-- 1 nikita nikita 206 окт 18 10:28 ./1.log
-rw-rw-r-- 1 nikita nikita 360 окт 18 12:34 ./3.sh
-rw-rw-r-- 1 nikita nikita 257 окт 18 10:49 ./0.log
```

Рассмотрим результат исследования команды с несколькими ключами подробнее:

- *find -type* - осуществляет поиск по типу файла.
- *find -name* - осуществляет поиск файлов по имени, в основном используется для поиска по маске.
- *find -size* - осуществляет поиск файлов по размеру. Можно устанавливать нижнюю границу размера файла, верхнюю или обе вместе.
- *find -exec* - позволяет создавать вложенные команды. Аргумент "{}"заменяется на имя рассматриваемого файла, каждый раз, когда он встречается среди аргументов команды. Все символы за флагом *-exec* считаются ее аргументами до символа ";".

1.4.6 Утилиты od и hexdump

Рассмотрим команду *od* с флагами *-c, -bc* на примере тестового файла:

```
nikita@nikita-pc:~/temp1$ echo "content">tempfile
nikita@nikita-pc:~/temp1$ ls -l tempfile
-rw-rw-r-- 1 nikita nikita 8 окт 18 12:55 tempfile
nikita@nikita-pc:~/temp1$ od tempfile
0000000 067543 072156 067145 005164
0000010
nikita@nikita-pc:~/temp1$ od -c tempfile
0000000   c   o   n   t   e   n   t  \n
0000010
nikita@nikita-pc:~/temp1$ od -bc tempfile
0000000 143 157 156 164 145 156 164 012
          c   o   n   t   e   n   t  \n
0000010s
```

Рассмотрим используемые команды:

- *od* - выводит содержимое файла в восьмеричном формате.
- *od -c* - флаг *-c* печатает те символы, которые может напечатать.
- *od -bc* - флаг *-b* разбивает результат на октавы.

Рассмотрим команду *hexdump*. Она похожа на команду *od*, однако имеет больше возможностей для отображения файла:

```
nikita@nikita-pc:~/temp1$ hexdump -C tempfile
0000000  63 6f 6e 74 65 6e 74 0a                           |content.|
0000008
nikita@nikita-pc:~/temp1$ hexdump -e '"%2_ad | " 2 "%_c" "|\n"' tempfile
```

```
 0 | co|
 2 | nt|
 4 | en|
 6 | t\n|
```

Рассмотрим используемые флаги и форматы:

- *-C* - выводит содержимое файла в шестнадцатиричном формате и выводит ASCII символы.
- *-e* - позволяет выводить в настраиваемом формате.
- *"2_ad | "* - вывод двух символов смещения в десятичном формате с разделителем ".
- *2 "%_c"* - далее двух символов из файла.
- *"| \n"* - далее разделитель и символ перевода строки.

1.4.7 Определение максимального количества записей в каталоге

В первой лабораторной работе я выяснил, что размер каталога при создании равен 4096 байт. Это обусловлено типом файловой системы (в данном случае *ext4*). Однако размер каталога можно увеличить, наполняя его файлами или другими каталогами. С помощью скрипта создадим промежуточную папку и будем увеличивать ее размер, наполняя другими каталогами. Если размер папки изменился, то выходим из цикла:

```
#!/bin/bash

# Создаем папку
mkdir tempfolder

# Фиксируем в переменной первоначальный размер пустой папки (4096 для текущей ОС)
defaultsize=$(ls -l -d tempfolder | cut -d ' ' -f5)
currentsize=$defaultsize
index=0

# Продолжаем цикл, пока размер папки не изменится
while [ "$defaultsize" -eq "$currentsize" ]
do
  # Создаем новые папки
  mkdir ./tempfolder/$index
  # Инкрементируем счетчик
  index=$((index+1))
  # Фиксируем в переменной размер заполняющейся папки
  currentsize=$(ls -l -d tempfolder | cut -d ' ' -f5)
done

# Удаляем папку
rm -rf tempfolder

echo "Count of new directories to change size $index"
```

Результат подсчета количества вложенных каталогов для изменения размера папки:

```
nikita@nikita-pc:~/temp1$ sudo sh 7.sh
Count of new directories to change size 340
```

Для изменения размера папки с 4096 байт на новое значение понадобилось 340 вложенных каталогов.

Для файловой системы *ext4* максимальное количество каталогов, которое может быть помещено в папку равно $2^{32} - 1$.

Скрипт был запущен на компьютере в лаборатории, ошибок в работе скрипта не возникало.

1.4.8 Содержимое /etc/passwd, /etc/shadow, утилита passwd

Рассмотрим содержимое файла */etc/passwd*:

```
nikita@nikita-pc:~$ sudo cat /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
(...)
```

Этот файл содержит строки следующего вида:

`login:passwd:UID:GID:GECOS:home:shell`

- *login* - имя пользователя.
- *passwd* - хэш пароля.
- *UID* - уникальный идентификатор пользователя.
- *GID* - уникальный идентификатор группы.
- *GECOS* - расширенное описание пользователя.
- *home* - домашний каталог
- *shell* - интерпретатор командной строки.

Рассмотрим содержимое файла */etc/shadow*:

```
nikita@nikita-pc:~$ sudo cat /etc/shadow
root:!:16950:0:99999:7:::
daemon:*:16911:0:99999:7:::
bin:*:16911:0:99999:7:::
sys:*:16911:0:99999:7:::
(...)
usbmux:*:16911:0:99999:7:::
nikita:$6$rQrZ9lk/$MnoiobKSEiHH3ot0gXa8Gf9cYQuBPoG8ouhFYTVHjAb3oCwL.MAm5Nq/wfTW0RWIWgt/
    mED0sSR65JI6bU9.u/:16950:0:99999:7:::
```

Этот файл содержит зашифрованную информацию о паролях для всех аккаунтов.
Рассмотрим поля каждой строки файла:

- Имя пользователя.
- Хэш пароля.
- Дата последнего изменения пароля.
- Дни до возможности смены пароля.
- Дни до устаревания пароля.
- За сколько дней до того, как пароль устаревает начинает напоминать о необходимости смены пароля.
- Через сколько дней после того, как пароль устареет, заблокировать учетную запись пользователя.
- Дата, при достижении которой учетная запись блокируется.
- Зарезервированное поле.

Утилита *passwd* позволяет изменить пароль текущего пользователя, информацию об учетной записи и срок действия пароля. Суперпользователь может работать с паролями всех пользователей, а остальные пользователи - только со своими паролями. Рассмотрим утилиту *passwd*:

```
nikita@nikita-pc:~/temp1$ sudo passwd
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully

nikita@nikita-pc:~/temp1$ sudo passwd nikita
Changing password for nikita.
passwd: Authentication token manipulation error
passwd: password unchanged

nikita@nikita-pc:~/temp1$ sudo passwd
Changing password for nikita.
Enter new UNIX password:
Retype new UNIX password:
Password unchanged
```

Некоторые ключи утилиты *passwd*:

- *-d* - удалить пароль, учетная запись станет беспарольной.
- *-e* - сделать пароль устаревшим.
- *-l* - заблокировать пароль пользователя.
- *-S* - показать состояние учетной записи.
- *-u* - разблокировать пароль пользователя.

1.4.9 Исследование прав владения и доступа

Утилиты chmod, chown

Исследуем утилиты *chmod* и *chown*:

```
nikita@nikita-pc:~/temp1$ mkdir tempfolder
nikita@nikita-pc:~/temp1$ ls -ld tempfolder/
drwxrwxr-x 2 nikita nikita 4096 окт 20 10:52 tempfolder/
nikita@nikita-pc:~/temp1$ >tempfolder/file1
nikita@nikita-pc:~/temp1$ >tempfolder/file2
nikita@nikita-pc:~/temp1$ ls -ld tempfolder/file1
-rw-rw-r-- 1 nikita nikita 0 окт 20 10:52 tempfolder/file1
nikita@nikita-pc:~/temp1$ ls -ld tempfolder/file2
-rw-rw-r-- 1 nikita nikita 0 окт 20 10:52 tempfolder/file2

nikita@nikita-pc:~/temp1$ sudo chmod -R u=rwx,g=,o=rwx tempfolder/
nikita@nikita-pc:~/temp1$ ls -ld tempfolder/
drwx---rwx 2 nikita nikita 4096 окт 20 10:52 tempfolder/
nikita@nikita-pc:~/temp1$ ls -ld tempfolder/file1
-rwx---rwx 1 nikita nikita 0 окт 20 10:52 tempfolder/file1
nikita@nikita-pc:~/temp1$ ls -ld tempfolder/file2
-rwx---rwx 1 nikita nikita 0 окт 20 10:52 tempfolder/file2

nikita@nikita-pc:~/temp1$ sudo chmod u=rx,g=rwx,o= tempfolder/file1
nikita@nikita-pc:~/temp1$ ls -ld tempfolder/file1
-r-xrwx--- 1 nikita nikita 0 окт 20 10:52 tempfolder/file1

nikita@nikita-pc:~/temp1$ sudo chown root:root tempfolder/file2
nikita@nikita-pc:~/temp1$ ls -ld tempfolder/file2
-rwx---rwx 1 root root 0 окт 20 10:52 tempfolder/file2
```

Утилита *chmod* предназначена для изменения прав доступа к файлам и директориям. При применении данной команды к каталогу, права вложенных файлов и папок не изменятся. Для изменения прав доступа всех вложенных файлов и папок используется флаг *-R*.

Утилита *chmod* позволяет задавать права доступа несколькими способами. Рассмотрим эти способы на примере задания прав *rwx—rwx*:

```
sudo chmod u=rwx,g=,o=rwx filename
sudo chmod u+rwx,g-rwx,o+rwx filename
sudo chmod 707 filename
```

Для наглядности в примере была использована символьная форма, однако быстрее и удобнее использовать числовую форму задания прав.

Утилита *chown* предназначена для изменения владельца и/или группы для указанного файла. Владелец и группа задаются через разделитель ": например *root:root*.

Расширение прав флагом SUID

Помимо флага разрешения исполнения x существует флаг расширения прав SUID (символ s или S). Этот флаг необходим для запуска пользователем файла, который ему не принадлежит. Флаг S в верхнем регистре используется, если нет прав на выполнение. Попробуем задать расширенные права:

```
nikita@nikita-pc:~/temp1$ sudo chmod -R u+s tempfolder/
nikita@nikita-pc:~/temp1$ ls -ld tempfolder/
drws------ 2 nikita nikita 4096 окт 20 11:06 tempfolder/
nikita@nikita-pc:~/temp1$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 54256 мар 29  2016 /usr/bin/passwd
```

Также были рассмотрены права утилиты *passwd*. Она также использует флаг расширения прав SUID, для использования несколькими пользователями.

Формирование итоговых прав, если права пользователя и группы различны

Сбросим пользовательские права доступа и предоставим все права группе и остальным:

```
nikita@nikita-pc:~/temp1$ sudo chmod -R u=,g=rwx,o=rwx tempfolder/
nikita@nikita-pc:~/temp1$ ls -l tempfolder/
----rwxrwx 1 nikita nikita 0 окт 20 10:52 tempfolder/file1
----rwxrwx 1 nikita nikita 0 окт 20 10:52 tempfolder/file2
nikita@nikita-pc:~/temp1$ cat tempfolder/file1
cat: tempfolder/file1: Permission denied
```

Несмотря на то, что пользователь принадлежит группе, отсутствие у него прав не позволяет получить доступ к файлу. Это означает, что приоритет пользовательских прав выше прав группы.

Использование команд записи, чтения и удаления, в зависимости от прав

Исследуем действие команд записи и чтения файла при различных правах доступа к нему:

```
nikita@nikita-pc:~/temp1$ sudo chmod -R u=rx,g=rx,o=rx tempfolder/
nikita@nikita-pc:~/temp1$ ls -ld tempfolder/
dr-xr-xr-x 2 nikita nikita 4096 окт 20 11:06 tempfolder/
nikita@nikita-pc:~/temp1$ echo "test" > tempfolder/file1
bash: tempfolder/file1: Permission denied
nikita@nikita-pc:~/temp1$ cat tempfolder/file1
content
nikita@nikita-pc:~/temp1$ rm tempfolder/file1
rm: remove write-protected regular file 'tempfolder/file1'? n

nikita@nikita-pc:~/temp1$ sudo chmod -R u=wx,g=wx,o=wx tempfolder/
nikita@nikita-pc:~/temp1$ ls -ld tempfolder/
d-wx-wx-wx 2 nikita nikita 4096 окт 20 11:36 tempfolder/
nikita@nikita-pc:~/temp1$ echo "content">tempfolder/file1
nikita@nikita-pc:~/temp1$ cat tempfolder/file1
cat: tempfolder/file1: Permission denied

nikita@nikita-pc:~/temp1$ sudo chmod -R u=,g=,o= tempfolder/
nikita@nikita-pc:~/temp1$ ls -ld tempfolder/
d---------- 2 nikita nikita 4096 окт 20 11:36 tempfolder/
nikita@nikita-pc:~/temp1$ echo "content">tempfolder/file1
bash: tempfolder/file1: Permission denied
nikita@nikita-pc:~/temp1$ cat tempfolder/file1
cat: tempfolder/file1: Permission denied
```

В результате эксперимента было выявлено, что команду *ls -l file* можно использовать к файлу с любыми правами доступа, команду *echo "some"> file* можно использовать только при наличии прав на запись, команду *cat file* можно использовать только при наличии прав на чтение, команда *rm file* выдаст предупреждение.

1.4.10 Разработка программы-шлюза для доступа к файлу другого пользователя

Разработаем программу-шлюз, которая читает содержимое файла и выводит его в поток вывода:

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

int main(int argc, char** argv) {
  // Если нет аргументов командной строки
  if(argc < 2) {
    // Выводим сообщение об ошибке в поток ошибок и выходим с кодом 0x1
    std::cerr << "Error: No input file." << std::endl;
    return 0x1;
  }

  // Открываем поток на чтение из файла
  std::ifstream stream(argv[1]);

  // Если не удалось открыть файл
  if(!stream.is_open()) {
    // Выводим сообщение об ошибке в поток ошибок и выходим с кодом 0x2
    std::cerr << "Error: It's impossible to open file." << std::endl;
    return 0x2;
  }

  // Посимвольно читаем содержимое файла и выводим те же симводы в поток вывода
  char symbol;
  while(stream >> symbol)
    std::cout << symbol;

  std::cout << std::endl;

  // Закрываем поток чтения из файла
  stream.close();
  return 0x0;
}
```

Скомпилируем программу на C++ командой:

```
g++ -std=c++11 -static-libgcc -static-libstdc++ -o 10.cpp 10.exe
```

Результат работы программы, запущенной от владельца должен быть аналогичен команде *cat*, убедимся в этом:

```
nikita@nikita-pc:~/temp1$ sudo chmod -R u=rwx,g=,o= tempfolder/
nikita@nikita-pc:~/temp1$ ls -ld tempfolder/
drwx------ 2 nikita nikita 4096 окт 20 11:36 tempfolder/
nikita@nikita-pc:~/temp1$ sudo chmod u=rwx,g=,o= 10.exe
nikita@nikita-pc:~/temp1$ ls -ld 10.exe
-rwx------ 1 nikita nikita 1227384 окт 20 12:08 10.exe
nikita@nikita-pc:~/temp1$ cat tempfolder/file1
content
nikita@nikita-pc:~/temp1$ ./10.exe tempfolder/file1
content
```

Создадим нового пользователя и от его имени попробуем запустить программу с теми же правами доступа:

```
nikita@nikita-pc:~/temp1$ sudo chmod -R 700 tempfolder/
nikita@nikita-pc:~/temp1$ sudo chmod 700 10.exe

nikita@nikita-pc:~/temp1$ sudo useradd bratishka
nikita@nikita-pc:~/temp1$ sudo passwd bratishka
```

```
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
nikita@nikita-pc:~/temp1$ sudo su bratishka

bratishka@nikita-pc:/home/nikita/temp1$ cat tempfolder/file1
cat: tempfolder/file1: Permission denied
bratishka@nikita-pc:/home/nikita/temp1$ ./10.exe tempfolder/file1
bash: ./10.exe: Permission denied
```

Добавим флаг SUID для программы-шлюза и повторим эксперимент:

```
nikita@nikita-pc:~/temp1$ sudo chmod u=rwx,g=x,o=x 10.exe
nikita@nikita-pc:~/temp1$ sudo chmod u+s 10.exe
nikita@nikita-pc:~/temp1$ sudo chmod g+s 10.exe
nikita@nikita-pc:~/temp1$ ls -l 10.exe
-rws--s--x 1 nikita nikita 1227384 окт 20 12:08 10.exe
nikita@nikita-pc:~/temp1$ sudo chmod -R 700 tempfolder/
nikita@nikita-pc:~/temp1$ ls -l tempfolder/
total 8
-rwx------ 1 nikita nikita 8 окт 20 11:39 file1
-rwx------ 1 nikita nikita 8 окт 20 11:36 file2
nikita@nikita-pc:~/temp1$ sudo su bratishka
bratishka@nikita-pc:/home/nikita/temp1$ cat tempfolder/file1
cat: tempfolder/file1: Permission denied
bratishka@nikita-pc:/home/nikita/temp1$ ./10.exe tempfolder/file1
content
```

Эксперимент с несколькими пользователями был проведен на компьютере в лаборатории. Результаты эксперимента совпали с ожидаемыми.

1.4.11 Получение информации о файловых системах, точках монтирования

Утилиты и информационные файлы

Исследуем утилиту *df* с различными флагами. Утилита *df* предоставляет информацию о состоянии жесткого диска и точках монтирования:

```
nikita@nikita-pc:~/temp1$ df
Filesystem     1K-blocks     Used Available Use% Mounted on
udev              998788        0    998788   0% /dev
tmpfs             203612     6420    197192   4% /run
/dev/sda1      305481976 59911132 230030212  21% /
tmpfs            1018056      220   1017836   1% /dev/shm
tmpfs               5120        4      5116   1% /run/lock
tmpfs            1018056        0   1018056   0% /sys/fs/cgroup
tmpfs             203612       60    203552   1% /run/user/1000
/dev/sdb1        2002624     5664   1996960   1% /media/nikita/4B52-FD2E

nikita@nikita-pc:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            976M     0  976M   0% /dev
tmpfs           199M  6,4M  193M   4% /run
/dev/sda1       292G   58G  220G  21% /
tmpfs           995M  184K  995M   1% /dev/shm
tmpfs           5,0M  4,0K  5,0M   1% /run/lock
tmpfs           995M     0  995M   0% /sys/fs/cgroup
tmpfs           199M   44K  199M   1% /run/user/1000
/dev/sdb1       1,9G  5,5K  1,9G   1% /media/nikita/4B52-FD2E

nikita@nikita-pc:~$ df -i
Filesystem       Inodes   IUsed    IFree IUse% Mounted on
udev             249697     562   249135    1% /dev
tmpfs            254514     780   253734    1% /run
/dev/sda1      19406848 1095765 18311083    6% /
tmpfs            254514       7   254507    1% /dev/shm
tmpfs            254514       5   254509    1% /run/lock
```

```
tmpfs             254514       16   254498    1% /sys/fs/cgroup
tmpfs             254514       27   254487    1% /run/user/1000
/dev/sdb1              0        0        0     - /media/nikita/4B52-FD2E

nikita@nikita-pc:~$ df -ih
Filesystem     Inodes IUsed IFree IUse% Mounted on
udev             244K   562  244K    1% /dev
tmpfs            249K   780  248K    1% /run
/dev/sda1         19M  1,1M   18M    6% /
tmpfs            249K     7  249K    1% /dev/shm
tmpfs            249K     5  249K    1% /run/lock
tmpfs            249K    16  249K    1% /sys/fs/cgroup
tmpfs            249K    27  249K    1% /run/user/1000
/dev/sdb1           0     0     0     - /media/nikita/4B52-FD2E

nikita@nikita-pc:~$ df -hT
Filesystem     Type       Size  Used Avail Use% Mounted on
udev           devtmpfs  976M     0  976M   0% /dev
tmpfs          tmpfs     199M  6,4M  193M   4% /run
/dev/sda1      ext4      292G   58G  220G  21% /
tmpfs          tmpfs     995M  184K  995M   1% /dev/shm
tmpfs          tmpfs     5,0M  4,0K  5,0M   1% /run/lock
tmpfs          tmpfs     995M     0  995M   0% /sys/fs/cgroup
tmpfs          tmpfs     199M   48K  199M   1% /run/user/1000
/dev/sdb1       vfat     1,9G  5,5K  1,9G   1% /media/nikita/4B52-FD2E
```

Утилита *df* выдала 8 точек монтирования, рассмотрим их подробнее:

- *udev* - менеджер устройств ядра Linux. Имеет тип *devtmpfs*.
- *tmpfs* - одна из разновидностей ФС, отличающаяся быстрой скоростью работы и надежностью. Располагается в оперативной памяти. Имеет тип *tmpfs*.
- */dev/sda1* - основной раздел, отформатированный под Linux. Имеет тип *ext4*.
- */dev/sdb1* - подключенная флешка. Имеет тип *vfat*.

Рассмотрим флаги подробнее:

- *df -h* - вывод памяти в читабельном формате (килобайты, мегабайты и др.).
- *df -i* - вывод памяти в блоках.
- *df -T* - вывод информации о типе файловой системы.

Рассмотрим ФС с точки зрения физических устройств:

```
nikita@nikita-pc:~/temp1$ lsblk -o +FSTYPE
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT                FSTYPE
sda      8:0    0 298,1G  0 disk
|-sda1   8:1    0 296,1G  0 part /                        ext4
|-sda2   8:2    0     1K  0 part
|-sda5   8:5    0     2G  0 part [SWAP]                   swap
sdb      8:16   1   1,9G  0 disk
|-sdb1   8:17   1   1,9G  0 part /media/nikita/4B52-FD2E vfat
```

В выводе *lsblk* два диска: sda - основной жесткий диск, sdb - флеш накопитель. Основной жесткий диск разделен на три раздела: sda1 - корневой раздел системы в формате *ext4*, sda5 - раздел для виртуальной памяти, sda2 - расширенный раздел, включающий в себя sda5.

Флеш накопитель имеет тип файловой системы *vfat*, который характерен для портативных носителей небольшого объема, и один раздел.

Рассмотрим файл */etc/fstab*, который содержит информацию о различных файловых системах и устройствах хранения информации:

```
nikita@nikita-pc:~/temp1$ 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=8a48f05f-7dd3-45d8-946b-567d6661e2a7 /               ext4    errors=remount-ro 0
       1
# swap was on /dev/sda5 during installation
UUID=bc8eabbf-fae3-45b4-bd1d-93c3862f9937 none            swap    sw              0
      0
```

Каждая строка файла */etc/fstab* имеет структуру:

```
<filesystem> <mount point> <type> <options> <dumb> <pass>
```

- *filesystem* - физическое место размещения файловой системы, по которому определяется конкретный раздел или устройство хранения для монтирования.
- *mount point* - точка монтирования, куда монтируется корень файловой системы.
- *type* - тип файловой системы. Список всех доступных для монтирования файловых систем определен в файле */proc/filesystems*.
- *options* - параметры монтирования файловой системы.
- *dump* - используется утилитой *dump* для определения, нужно ли создать резервную копию данных в файловой системе. Если указана единица, то *dump* создаст резервную копию.
- *pass* - используется утилитой *fsck* для определения нужно ли проверять целостность ФС. Значение 1 указывается только для корневой файловой системы. Для остальных ФС следует указывать 2, которое имеет менее высокий приоритет. Если указано 0, то ФС не будет проверяться *fsck*.

В файле */etc/mtab* прописаны устройства, смонтированные в систему в настоящий момент. При монтировании новой ФС в файл будет добавлена соответствующая запись. Формат файла аналогичен формату файла */etc/fstab*, за исключением того, что в */etc/mtab* указывается не физический адрес, а путь.

Рассмотрим файл */etc/mtab*:

```
nikita@nikita-pc:~/temp1$ 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=998788k,nr_inodes=249697,mode=755 0 0
devpts /dev/pts devpts rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000 0 0
tmpfs /run tmpfs rw,nosuid,noexec,relatime,size=203612k,mode=755 0 0
/dev/sda1 / ext4 rw,relatime,errors=remount-ro,data=ordered 0 0
securityfs /sys/kernel/security securityfs rw,nosuid,nodev,noexec,relatime 0 0
tmpfs /dev/shm tmpfs rw,nosuid,nodev 0 0
tmpfs /run/lock tmpfs rw,nosuid,nodev,noexec,relatime,size=5120k 0 0
tmpfs /sys/fs/cgroup tmpfs ro,nosuid,nodev,noexec,mode=755 0 0
cgroup /sys/fs/cgroup/systemd cgroup rw,nosuid,nodev,noexec,relatime,xattr,release_agent
    =/lib/systemd/systemd-cgroups-agent,name=systemd 0 0
pstore /sys/fs/pstore pstore rw,nosuid,nodev,noexec,relatime 0 0
cgroup /sys/fs/cgroup/cpu,cpuacct cgroup rw,nosuid,nodev,noexec,relatime,cpu,cpuacct 0 0
cgroup /sys/fs/cgroup/devices cgroup rw,nosuid,nodev,noexec,relatime,devices 0 0
cgroup /sys/fs/cgroup/hugetlb cgroup rw,nosuid,nodev,noexec,relatime,hugetlb 0 0
cgroup /sys/fs/cgroup/blkio cgroup rw,nosuid,nodev,noexec,relatime,blkio 0 0
cgroup /sys/fs/cgroup/net_cls,net_prio cgroup rw,nosuid,nodev,noexec,relatime,net_cls,
    net_prio 0 0
cgroup /sys/fs/cgroup/memory cgroup rw,nosuid,nodev,noexec,relatime,memory 0 0
cgroup /sys/fs/cgroup/cpuset cgroup rw,nosuid,nodev,noexec,relatime,cpuset 0 0
cgroup /sys/fs/cgroup/perf_event cgroup rw,nosuid,nodev,noexec,relatime,perf_event 0 0
cgroup /sys/fs/cgroup/pids cgroup rw,nosuid,nodev,noexec,relatime,pids 0 0
cgroup /sys/fs/cgroup/freezer cgroup rw,nosuid,nodev,noexec,relatime,freezer 0 0
systemd-1 /proc/sys/fs/binfmt_misc autofs rw,relatime,fd=26,pgrp=1,timeout=0,minproto=5,
    maxproto=5,direct 0 0
hugetlbfs /dev/hugepages hugetlbfs rw,relatime 0 0
debugfs /sys/kernel/debug debugfs rw,relatime 0 0
mqueue /dev/mqueue mqueue rw,relatime 0 0
```

```
fusectl /sys/fs/fuse/connections fusectl rw,relatime 0 0
tmpfs /run/user/1000 tmpfs rw,nosuid,nodev,relatime,size=203612k,mode=700,uid=1000,gid=1000 0 0
gvfsd-fuse /run/user/1000/gvfs fuse.gvfsd-fuse rw,nosuid,nodev,relatime,user_id=1000,group_id=1000 0 0
/dev/sdb1 /media/nikita/4B52-FD2E vfat rw,nosuid,nodev,relatime,uid=1000,gid=1000,fmask=0022,dmask=0022,codepage=437,iocharset=iso8859-1,shortname=mixed,showexec,utf8,flush,errors=remount-ro 0 0
```

Список всех доступных для монтирования файловых систем определен в файле */proc/filesystems*. Рассмотрим его содержимое:

```
nikita@nikita-pc:~/temp1$ cat /proc/filesystems
nodev sysfs
nodev rootfs
nodev ramfs
nodev bdev
nodev proc
nodev cpuset
nodev cgroup
nodev tmpfs
nodev devtmpfs
nodev debugfs
nodev tracefs
nodev securityfs
nodev sockfs
nodev bpf
nodev pipefs
nodev devpts
  ext3
  ext2
  ext4
  squashfs
nodev hugetlbfs
  vfat
nodev ecryptfs
  fuseblk
nodev fuse
nodev fusectl
nodev pstore
nodev mqueue
nodev autofs
```

Пометка *nodev* говорит о том, что это виртуальная файловая система.

Максимально возможное дерево файловой системы

На диске может быть создано не более четырех разделов, это обусловлено тем, что под таблицу разделов в *MBR* выделено 64 байта, а каждая запись занимает 16 байт. Однако это ограничение легко обойти, создав один *Extended* раздел, вместо одного из физических разделов. После этого можно добавить несколько логических разделов. Для примера, создадим несколько разделов, с помощью утилиты *fdisk*:

```
nikita@nikita-pc:~/temp1$ sudo fdisk /dev/sdb

Welcome to fdisk (util-linux 2.27.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Command (m for help): p
Disk /dev/sdb: 1,9 GiB, 2051014656 bytes, 4005888 sectors
(...)
Device     Boot   Start     End Sectors   Size Id Type
/dev/sdb1          2048 4005887 4003840   1,9G 83 Linux

Command (m for help): n
To create more partitions, first replace a primary with an extended partition.

```

```
Command (m for help): d
Selected partition 1
Partition 1 has been deleted.

Command (m for help): n
(...)
Select (default p): e
Partition number (1-4, default 1): 4
First sector (2048-4005887, default 2048): 1000000
Last sector, +sectors or +size{K,M,G,T,P} (1000000-4005887, default 4005887): 4005887
Created a new partition 4 of type 'Extended' and of size 1,4 GiB.

Command (m for help): n
(...)
Select (default p): p
Partition number (1-3, default 1): 1
First sector (2048-4005887, default 2048): 2048
Last sector, +sectors or +size{K,M,G,T,P} (2048-999999, default 999999): 20000
Created a new partition 1 of type 'Linux' and of size 8,8 MiB.

Command (m for help): n
(...)
Select (default p): p
Partition number (2,3, default 2): 2
First sector (20001-4005887, default 20480): 20001
Last sector, +sectors or +size{K,M,G,T,P} (20001-999999, default 999999): 50000
Created a new partition 2 of type 'Linux' and of size 14,7 MiB.

Command (m for help): n
(...)
Select (default p): p
Selected partition 3
First sector (50001-4005887, default 51200): 50001
Last sector, +sectors or +size{K,M,G,T,P} (50001-999999, default 999999): 999999
Created a new partition 3 of type 'Linux' and of size 463,9 MiB.

Command (m for help): n
All primary partitions are in use.
Adding logical partition 5
First sector (1002048-4005887, default 1003520): 1002048
Last sector, +sectors or +size{K,M,G,T,P} (1002048-4005887, default 4005887): 2000000
Created a new partition 5 of type 'Linux' and of size 487,3 MiB.

Command (m for help): n
All primary partitions are in use.
Adding logical partition 6
First sector (2002049-4005887, default 2002944): 2002049
Last sector, +sectors or +size{K,M,G,T,P} (2002049-4005887, default 4005887): 3000000
Created a new partition 6 of type 'Linux' and of size 487,3 MiB.

Command (m for help): n
All primary partitions are in use.
Adding logical partition 7
First sector (3002049-4005887, default 3002368): 3002049
Last sector, +sectors or +size{K,M,G,T,P} (3002049-4005887, default 4005887): 3500000
Created a new partition 7 of type 'Linux' and of size 243,1 MiB.

Command (m for help): n
All primary partitions are in use.
Adding logical partition 8
First sector (3502049-4005887, default 3502080): 3502049
Last sector, +sectors or +size{K,M,G,T,P} (3502049-4005887, default 4005887): 4005887
Created a new partition 8 of type 'Linux' and of size 246 MiB.

Command (m for help): p
(...)
```

```
Device     Boot    Start      End  Sectors    Size Id Type
/dev/sdb1           2048    20000    17953    8,8M 83 Linux
/dev/sdb2          20001    50000    30000   14,7M 83 Linux
/dev/sdb3          50001   999999   949999  463,9M 83 Linux
/dev/sdb4        1000000  4005887  3005888    1,4G  5 Extended
/dev/sdb5        1002048  2000000   997953  487,3M 83 Linux
/dev/sdb6        2002049  3000000   997952  487,3M 83 Linux
/dev/sdb7        3002049  3500000   497952  243,1M 83 Linux
/dev/sdb8        3502049  4005887   503839   246M 83 Linux

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

Смонтируем в корневую файловую систему четыре файловых системы с флеш накопителя. Построим дерево файловых систем и проверим результат монтирования в файле */etc/mtab*:

```
nikita@nikita-pc:/$ sudo mkdir /media/m1
nikita@nikita-pc:/$ sudo mkfs.vfat /dev/sdb1
mkfs.fat 3.0.28 (2015-05-16)
nikita@nikita-pc:/$ sudo mount -t vfat /dev/sdb1 /media/m1/

nikita@nikita-pc:/$ sudo mkdir /media/m1/m2
nikita@nikita-pc:/$ sudo mkfs.vfat /dev/sdb3
mkfs.fat 3.0.28 (2015-05-16)
nikita@nikita-pc:/$ sudo mount -t vfat /dev/sdb3 /media/m1/m2/

nikita@nikita-pc:/$ sudo mkdir /media/m1/m2/m3
nikita@nikita-pc:/$ sudo mkfs.vfat /dev/sdb5
mkfs.fat 3.0.28 (2015-05-16)
nikita@nikita-pc:/$ sudo mount -t vfat /dev/sdb5 /media/m1/m2/m3/

nikita@nikita-pc:/$ sudo mkdir /media/m1/m2/m3/m4
nikita@nikita-pc:/$ sudo mkfs.vfat /dev/sdb7
mkfs.fat 3.0.28 (2015-05-16)
nikita@nikita-pc:/$ sudo mount -t vfat /dev/sdb7 /media/m1/m2/m3/m4/

nikita@nikita-pc:/$ cat /etc/mtab | grep sdb
/dev/sdb1 /media/m1 vfat rw,relatime,fmask=0022,dmask=0022,codepage=437,iocharset=iso8859
    -1,shortname=mixed,errors=remount-ro 0 0
/dev/sdb3 /media/m1/m2 vfat rw,relatime,fmask=0022,dmask=0022,codepage=437,iocharset=
    iso8859-1,shortname=mixed,errors=remount-ro 0 0
/dev/sdb5 /media/m1/m2/m3 vfat rw,relatime,fmask=0022,dmask=0022,codepage=437,iocharset=
    iso8859-1,shortname=mixed,errors=remount-ro 0 0
/dev/sdb7 /media/m1/m2/m3/m4 vfat rw,relatime,fmask=0022,dmask=0022,codepage=437,
    iocharset=iso8859-1,shortname=mixed,errors=remount-ro 0 0
```

Теперь при подключении и отключении флеш накопителя будет происходить автоматическое монтирование и размонтирование всех четырех файловых систем.

После монтирования в проводнике операционной системы *Ubuntu* появились все четыре смонтированных файловых системы:

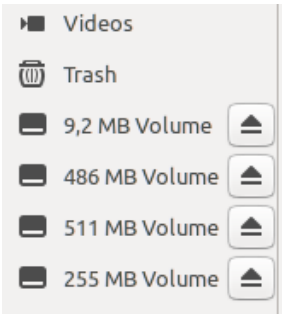

Рис. 1.1

1.4.12 Получение информации о файле

Магические числа, утилита file

file - специальная утилита, выполняющая ряд проверок для указанного файла, пытаясь его классифицировать. В первую очередь происходит тест на файловую систему, после этого тест на магические числа и языковые тесты.

Тестирование на магические числа выполняется, исходя из информации в файлах */usr/share/misc/magic*, */etc/magic* и */usr/lib/magic*.

Рассмотрим заголовок программы-шлюза *10.exe*, найдем его в одном из вышеуказанных файлов и попробуем классифицировать файл утилитой *file*:

```
nikita@nikita-pc:~/temp1$ od 10.exe -cDN25
0000000 177   E   L   F 002 001 001 003  \0  \0  \0  \0  \0  \0  \0  \0
             1179403647        50397442               0               0
0000020 002  \0   >  \0 001  \0  \0  \0 020
                4063234               1              16
0000031

nikita@nikita-pc:~/temp1$ cat /usr/share/mime/magic
(...)
>0=?ELF
1>5=??
2>16=??
>0=?ELF
(...)

nikita@nikita-pc:~/temp1$ file 10.exe
10.exe: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), dynamically linked,
    interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=4313133
    fea619662726492cb4c369c7479552556, not stripped
```

Заголовок файла *10.exe* содержит в заголовке символы "ELF". Эти символы были найдены в файле */usr/share/misc/magic*, на основании чего утилита *file* смогла определить тип исследуемого файла.

Утилита file, примененная к разным типам файлов

Приведем примеры вывода утилиты *file* при применении на различные типы файлов:

```
nikita@nikita-pc:~/temp1$ file -v 10.exe
file-5.25
magic file from /etc/magic:/usr/share/misc/magic

nikita@nikita-pc:~/temp1$ file -l tempfolder/
(...)
Binary patterns:
Strength = 380@6: OpenSSH private key []
Strength = 361@66: EICAR virus test files []
Strength = 340@585: sc68 Atari ST music []
(...)

nikita@nikita-pc:~/temp1$ >tempfile
nikita@nikita-pc:~/temp1$ file tempfile
tempfile: empty

nikita@nikita-pc:~/temp1$ echo "content">tempfile
nikita@nikita-pc:~/temp1$ file tempfile
tempfile: ASCII text

nikita@nikita-pc:~/temp1$ file 10.cpp
10.cpp: C source, ASCII text

nikita@nikita-pc:~/temp1$ file 1.sh
1.sh: Bourne-Again shell script, UTF-8 Unicode text executable
```

Утилита *file* определила пустой файл, файл, наполненный исключительно ASCII символами, файл исходного кода *C++* и bash скрипт, содержащий не только ASCII символы.

Создание собственного типа файлов

Создадим собственный тип файла. Для этого добавим в *etc/magic* следующую строку:

```
0 string MAGIC_HEADER MyType
```

Теперь любой файл, который с нулевого бита содержит строку *MAGIC_HEADER* будет иметь тип *MyType*. Проверим это:

```
1 nikita@nikita-pc:~/temp1$ sudo gedit /etc/magic
2 (*
3   0 string MAGIC_HEADER MyType
4 *)
5 nikita@nikita-pc:~/temp1$ echo "MAGIC_HEADER">tempfile
6 nikita@nikita-pc:~/temp1$ file tempfile
7 tempfile: MyType
```

1.5 Вывод

В данной работе была изучена структура файловой системы ОС Linux. В ней существует несколько типов файлов: обычные, директории, ссылки, сокеты, очереди, блок-ориентированные файлы, байт-ориентированные файлы.У одного файла может быть несколько путей, т.е. несколько файлов в структуре каталогов Linux могут быть физически одним файлом на диске. Это достигается тем, что в файловой системе каждый файл идентифицируется уникальным номером, называемым индексным дескриптором. Каждый файл имеет свой индексный дескриптор, идентифицируемый по уникальному номеру, в файловой системе, в которой располагается сам файл.

Также были изучены и экспериментально проверены следующие утилиты: ls, file, od, hexdump, df, fdisk и др.

Были исследованы способы изменения прав доступа и владельца файла. Были проведены эксперименты с флагом SUID.

1.6 Список литературы

- Мануал awk [Электронный ресурс]. — URL: https://www.opennet.ru/man.shtml?topic=awk (дата обращения 20.10.2016).
- Мануал cp [Электронный ресурс]. — URL: https://www.opennet.ru/man.shtml?topic=cp (дата обращения 20.10.2016).
- Мануал file [Электронный ресурс]. — URL: https://www.opennet.ru/man.shtml?topic=file (дата обращения 20.10.2016).
- Мануал find [Электронный ресурс]. — URL: https://www.opennet.ru/man.shtml?topic=find (дата обращения 20.10.2016).
- Мануал fstab [Электронный ресурс]. — URL: https://www.opennet.ru/man.shtml?topic=fstab (дата обращения 20.10.2016).
- Мануал hexdump [Электронный ресурс]. — URL: https://www.opennet.ru/man.shtml?topic=hexdump (дата обращения 20.10.2016).
- Мануал link [Электронный ресурс]. — URL: https://www.opennet.ru/man.shtml?topic=link (дата обращения 20.10.2016).
- Мануал ln [Электронный ресурс]. — URL: https://www.opennet.ru/man.shtml?topic=ln (дата обращения 20.10.2016).
- Мануал od [Электронный ресурс]. — URL: https://www.opennet.ru/man.shtml?topic=od (дата обращения 20.10.2016).
- Мануал passwd [Электронный ресурс]. — URL: https://www.opennet.ru/man.shtml?topic=passwd (дата обращения 20.10.2016).
- Типы файлов в Linux [Электронный ресурс]. — URL: http://younglinux.info/filestype (дата обращения 20.10.2016).
- Файл /etc/passwd [Электронный ресурс]. — URL: https://ru.wikipedia.org/wiki//etc/passwd (дата обращения 20.10.2016).