Санкт-Петербургский государственный политехнический университет

Факультет технической кибернетики

Кафедра компьютерных систем и программных технологий

Отчёт по лабораторной работе №2

«Файловая система ОС UNIX»

Работу выполнил студент группы № 4081/12

Дорофеев Юрий Владимирович

Работу принял преподаватель _____

Малышев Игорь Алексеевич

г. Санкт-Петербург

2012

1. Цель работы

Изучение принципов организации файловой системы ОС UNIX на примере Linux.

2. Программа работы

2.1. Иерархия каталогов

Содержимое корня:

```
dorofeev@dorofeev-VirtualBox:~$ ls -l /
total 112
drwxr-xr-x   2 root root  4096 Oct  6 19:16 bin
drwxr-xr-x   3 root root  4096 Oct  6 19:19 boot
drwxr-xr-x   2 root root  4096 Oct  6 18:55 cdrom
drwxr-xr-x  15 root root  4080 Nov  8 22:31 dev
drwxr-xr-x 140 root root 12288 Nov  8 22:31 etc
drwxr-xr-x   3 root root  4096 Oct  6 18:59 home
lrwxrwxrwx   1 root root    36 Oct  6 19:13 initrd.img -> boot/initrd.img-
3.2.0-29-generic-pae
lrwxrwxrwx   1 root root    37 Oct  6 18:33 initrd.img.old ->
/boot/initrd.img-3.2.0-29-generic-pae
drwxr-xr-x  22 root root  4096 Oct  6 19:16 lib
drwx------   2 root root 16384 Oct  6 18:33 lost+found
drwxr-xr-x   4 root root  4096 Nov  8 22:31 media
drwxr-xr-x   2 root root  4096 Apr 19  2012 mnt
drwxr-xr-x   3 root root  4096 Oct  6 19:27 opt
dr-xr-xr-x 147 root root     0 Nov  8 22:30 proc
-rw-r--r--   1 root root 23300 Sep 30 08:48 rez
drwx------   7 root root  4096 Nov  2 21:38 root
drwxr-xr-x  21 root root   780 Nov  8 22:31 run
drwxr-xr-x   2 root root  4096 Oct  6 19:29 sbin
drwxr-xr-x   2 root root  4096 Mar  5  2012 selinux
drwxr-xr-x   2 root root  4096 Aug 18 02:05 srv
drwxr-xr-x  13 root root     0 Nov  8 22:30 sys
drwxrwxrwt  10 root root  4096 Nov  8 22:31 tmp
drwxr-xr-x  10 root root  4096 Aug 18 02:05 usr
drwxr-xr-x  13 root root  4096 Oct 25 12:35 var
lrwxrwxrwx   1 root root    33 Oct  6 19:13 vmlinuz -> boot/vmlinuz-3.2.0-29-
generic-pae
```

Для получения информации о файловой системе воспользуемся утилитой df:

```
dorofeev@dorofeev-VirtualBox:~$ df
Filesystem     1K-blocks      Used Available Use% Mounted on
/dev/sda1        7739864   3814260   3532440  52% /
udev              243120         4    243116   1% /dev
tmpfs             101512       912    100600   1% /run
none                5120         0      5120   0% /run/lock
none              253772       808    252964   1% /run/shm
123            204697596 155362020  49335576  76% /media/sf_123
/dev/sr0           53914     53914         0 100%
/media/VBOXADDITIONS_4.2.0_80737
```

Размер блока равен:

```
dorofeev@dorofeev-VirtualBox:~$ stat -c "%s" /
4096
```

Для определения размера блока использовалась команда stat, которая позволяет получить информацию о файловой системе или файлах. Размер блока равен 4096 Кб.

Создаем пустую папку и узнаем размер в блоках:

```
dorofeev@dorofeev-VirtualBox:~$ mkdir new
dorofeev@dorofeev-VirtualBox:~$ ls -sd new
4 new
```

Определим связь логической структуры файловой системы и физической структурой, используя команду df:

```
dorofeev@dorofeev-VirtualBox:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1       7.4G  3.6G  3.5G  51% /
udev            238M  4.0K  238M   1% /dev
tmpfs           100M  920K   99M   1% /run
none            5.0M     0  5.0M   0% /run/lock
none            248M  856K  247M   1% /run/shm
123             196G  149G   47G  77% /media/sf_123
/dev/sr0         53M   53M     0 100% /media/VBOXADDITIONS_4.2.0_80737
```

Команда df предназначена для получения информации о свободном дисковом пространстве, а так же выводит информацию о файловых системах. Ключ -h говорит выводить информацию в формате удобном для человека (указание размеров дискового пространства в Мб, Кб...).

2.2. Невидимые символы

Создаем файл prog.c с текстом программы:

```
#include <stdio.h>
main ()
{
printf ("Hello, everybody ! ");
}
```

Применив команду ls , определяем владельца и группу файла, права доступа, размер файла, время последнего изменения:

```
dorofeev@dorofeev-VirtualBox:~$ ls -l
total 64…
-rw-rw-r-- 1 dorofeev dorofeev   65 Nov  8 22:50 prog.c
```

Владелец и группа – dorofeev

Права доступа:

- владелец - запись и чтение;
- группа – запись и чтение;
- остальные - чтение.

Размер файла = 65 байт
Время последнего изменения: 2012-11-8 22:50

С помощью команды od (octal dump - восьмеричный дамп) выведим побайтную распечатку prog.c:

```
dorofeev@dorofeev-VirtualBox:~$ od prog.c
0000000 064443 061556 072554 062544 036040 072163 064544 027157
0000020 037150 066412 064541 020156 024450 075412 070012 064562
0000040 072156 020146 021050 062510 066154 026157 062440 062566
0000060 074562 067542 074544 020440 021040 035451 076412 005040
0000100 000012
0000101
```

od -c выводит ASCII-символы.

```
dorofeev@dorofeev-VirtualBox:~$ od -c prog.c
0000000   #   i   n   c   l   u   d   e       <   s   t   d   i   o   .
0000020   h   >  \n   m   a   i   n       (   )  \n   {  \n   p   r   i
0000040   n   t   f       (   "   H   e   l   l   o   ,       e   v   e
0000060   r   y   b   o   d   y       !       "   )   ;  \n   }      \n
0000100  \n
0000101
```

od -b выводит побайтно коды символов в 8-миричной системе.

```
dorofeev@dorofeev-VirtualBox:~$ od -b prog.c
0000000 043 151 156 143 154 165 144 145 040 074 163 164 144 151 157 056
0000020 150 076 012 155 141 151 156 040 050 051 012 173 012 160 162 151
0000040 156 164 146 040 050 042 110 145 154 154 157 054 040 145 166 145
0000060 162 171 142 157 144 171 040 041 040 042 051 073 012 175 040 012
0000100 012
0000101
```

od –cb вывод ASCII-символы и символы 8-миричной виде.

```
dorofeev@dorofeev-VirtualBox:~$ od -cb prog.c
0000000   #   i   n   c   l   u   d   e       <   s   t   d   i   o   .
        043 151 156 143 154 165 144 145 040 074 163 164 144 151 157 056
0000020   h   >  \n   m   a   i   n       (   )  \n   {  \n   p   r   i
        150 076 012 155 141 151 156 040 050 051 012 173 012 160 162 151
0000040   n   t   f       (   "   H   e   l   l   o   ,       e   v   e
        156 164 146 040 050 042 110 145 154 154 157 054 040 145 166 145
0000060   r   y   b   o   d   y       !       "   )   ;  \n   }      \n
        162 171 142 157 144 171 040 041 040 042 051 073 012 175 040 012
0000100  \n
        012
0000101
```

Перевод строки - \n (код 012)
Пробел - код 040
Табуляция - \t (код 011)

2.3. Ввод-вывод для файлов-терминалов

С помощью tty определяем полное имя нашего файла-терминала:

```
dorofeev@dorofeev-VirtualBox:~$ tty
/dev/pts/0
```

Выведем информацию посредством echo, переназначив вывод на этот терминал:

```
dorofeev@dorofeev-VirtualBox:~$ echo test >>/dev/pts/0
test
```

Переназначим вывод на другой терминал:

```
dorofeev@dorofeev-VirtualBox:~$ echo test >>/dev/pts/1
```

На консоли терминала pts/1 появляется `dorofeev@dorofeev-VirtualBox:~$ test` Файл-терминал — находится в каталоге /dev и воспринимается системой как файл-устройство. В результате, при выводе непосредственно в файл-терминал информация отображается в консоле.

Запустим cat без аргументов, переназначив ее вывод на файл prog.c и введем строчку:

```
dorofeev@dorofeev-VirtualBox:~$ cat >>prog.c
kolbasa
```

После ввода команды терминал перешел в режим ожидания ввода символов. По нажатию Ctrl+D, ввод символов завершается, и данные записываются в файл. С помощью od убеждаемся, что в конец файла prog.c записывается введеная строчка (и символ перевода строки \n):

```
dorofeev@dorofeev-VirtualBox:~$ od -cb prog.c
0000000   #   i   n   c   l   u   d   e       <   s   t   d   i   o   .
        043 151 156 143 154 165 144 145 040 074 163 164 144 151 157 056
0000020   h   >  \n   m   a   i   n       (   )  \n   {  \n   p   r   i
        150 076 012 155 141 151 156 040 050 051 012 173 012 160 162 151
0000040   n   t   f       (   "   H   e   l   l   o   ,       e   v   e
        156 164 146 040 050 042 110 145 154 154 157 054 040 145 166 145
0000060   r   y   b   o   d   y       !       "   )   ;  \n   }      \n
        162 171 142 157 144 171 040 041 040 042 051 073 012 175 040 012
0000100  \n   k   o   l   b   a   s   a  \n
        012 153 157 154 142 141 163 141 012
0000111
```

Запустим cat, наберем несколько символов без перевода строки:

```
dorofeev@dorofeev-VirtualBox:~$ cat >>prog.c
sobaka
```

Приглашения в shell появляется только после второго нажатия Ctrl+D. Это объясняется тем, что операционная система ищет в конце ввода символ перевода строки (\n) или конца потока ввода (символ Ctrl+D конец потока ввода). Если она его находит то по нажатию Ctrl+D завершается ввод, если не находит, то Ctrl+D воспринимается как конец потока ввода.

В файл записалась введенная строчка без символа перевода строки \n:

```
dorofeev@dorofeev-VirtualBox:~$ od -cb prog.c
0000000   #   i   n   c   l   u   d   e       <   s   t   d   i   o   .
        043 151 156 143 154 165 144 145 040 074 163 164 144 151 157 056
0000020   h   >  \n   m   a   i   n       (   )  \n   {  \n   p   r   i
        150 076 012 155 141 151 156 040 050 051 012 173 012 160 162 151
0000040   n   t   f       (   "   H   e   l   l   o   ,       e   v   e
        156 164 146 040 050 042 110 145 154 154 157 054 040 145 166 145
0000060   r   y   b   o   d   y       !       "   )   ;  \n   }      \n
        162 171 142 157 144 171 040 041 040 042 051 073 012 175 040 012
0000100  \n   k   o   l   b   a   s   a  \n   s   o   b   a   k   a
        012 153 157 154 142 141 163 141 012 163 157 142 141 153 141
0000117
```

2.4. Содержимое файлов

С помощью команды file определим содержимое файла prog.c:

```
dorofeev@dorofeev-VirtualBox:~$ file prog.c
prog.c: ASCII text
```

Исследуем с помощью этой команды различные файлы:

1) /bin

```
dorofeev@dorofeev-VirtualBox:~$ file /bin
/bin: directory
```

2) текстовый файл

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

3) объектный файл

```
dorofeev@dorofeev-VirtualBox:~$/lab1$ file prog.o
prog.o: ELF 32-bit LSB relocatable, Intel 80386, version 1 (SYSV), not
stripped
```

4) исполняемый файл

```
dorofeev@dorofeev-VirtualBox:~/lab1$ file  hello.out
hello.out: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked (uses shared libs), for GNU/Linux 2.6.24,
BuildID[sha1]=0xeacc03002882ba7dc767544a94460ca2db81da38, not stripped
```

ELF - формат исполняемых и компонуемых файлов. LSB executable - стандарт на исполняемые файлы. Intel 80386 - процессор для которого собран файл. dynamically linked (uses shared libs) - собран. GNU/Linux 2.6.15 - операционная система. not stripped - включена отладочная информация.

5) /bin/sh - ссылка на dash (интерпретатор)

```
dorofeev@dorofeev-VirtualBox:~$ file /bin/sh
/bin/sh: symbolic link to `dash'
```

6) /etc/passwd - файл паролей, по содержимому которого определяется соответствие между номерами и именами владельцев.

```
dorofeev@dorofeev-VirtualBox:~$ file /etc/passwd
/etc/passwd: ASCII text     ;текст в ASCII
```

7) /lib/libc.a - не существует такой библиотеки.

```
dorofeev@dorofeev-VirtualBox:~$ file /lib/libc.a
/lib/libc.a: ERROR: cannot open `/lib/libc.a' (No such file or directory)
```

```
dorofeev@dorofeev-VirtualBox:~$ file /lib/libiw.so.30
/lib/libiw.so.30: ELF 32-bit LSB shared object, Intel 80386, version 1
(SYSV), dynamically linked,
BuildID[sha1]=0x023366d57e7ffdf0b24b418543071be8d2736ad5, stripped
```

С помощью tail запишем хвост /bin/sh (15 последних символов) в файл hhh:

```
dorofeev@dorofeev-VirtualBox:~$ tail -15 /bin/sh >> hhh
```

Исследуем его с помощью file. Так как команда file распознает вид файла по первому блоку, а в файл hhh записан хвост, в данной ситуации вид файла распознать не удается:

```
dorofeev@dorofeev-VirtualBox:~$ file hhh
hhh: data
```

С помощью od определим магическое число (первые два байта) executable файлов:

```
dorofeev@dorofeev-VirtualBox:~/lab1$ od -cb hello.out | more
\0000000 177   E   L   F 001 001 001  \0  \0  \0  \0  \0  \0  \0  \0
         177 105 114 106 001 001 001 000 000 000 000 000 000 000 000
0000020 002  \0 003  \0 001  \0  \0  \0     203 004  \b   4  \0  \0  \0
         002 000 003 000 001 000 000 000 040 203 004 010 064 000 000 000
```

Отсюда видно, что магическое число (первые 2 байта): 177 105
Сравним полученные результаты с содержимым файла /etc/magic, содержащим магические числа системы. В этом файле нет магического числа.

2.5. Права доступа
2.5.1 Права доступа процессов к файлам

Категории пользователей для определения прав доступа: владелец, группа, остальные.
Виды прав доступа: чтение (r), запись (w) и исполнение (x).

Определим права доступа к своему "домашнему" каталогу:

```
dorofeev@dorofeev-VirtualBox:~/Lab2$ ls -ld /home/dorofeev
drwxr-xr-x 33 dorofeev dorofeev 4096 2011-11-12 15:32 /home/dorofeev
```

Владелец - dorofeev, группа - dorofeev.
Для владельца - чтение, запись и исполнение (rwx).
Для группы - чтение и исполнение (r-x).
Для остальных - чтение и исполнение (r-x).

Посредством команды chmod, обеспечим все права себе, возможность чтения и поиска для группы и никаких прав остальным:

```
dorofeev@dorofeev-VirtualBox:~/Lab2$ chmod 750 /home/dorofeev
dorofeev@dorofeev-VirtualBox:~/Lab2$ ls -ld /home/dorofeev
drwxr-x--- 33 dorofeev dorofeev 4096 2011-11-12 15:35 /home/dorofeev
```

Права доступа расположены в следующем порядке: чтение, запись, исполнение. Таким образом мы должны установить единицы в тех разрядах, которые соответствуют нужным нам правам. Первая опция (7=111) задает все права доступа для владельца, вторая (5=101) - права чтения и поиска для группы, а третья(0=000) не обеспечивает правами остальных. Команде chmod можно задавать параметры и другим способом: указанием кому изменить права доступа (классы пользователей — u, g, o, a):

```
dorofeev@dorofeev-VirtualBox:~/Lab2$ chmod u=rwx,g=rx,o-rwx /home/dorofeev
dorofeev@dorofeev-VirtualBox:~/Lab2$ ls -ld /home/dorofeev
drwxr-x--- 33 dorofeev dorofeev 4096 2011-11-12 15:39 /home/dorofeev
```

2.5.2 Файл паролей /etc/passwd

Структура файла паролей /etc/passwd:

```
dorofeev@dorofeev-VirtualBox:~/Dorofeev/Lab2$ cat /etc/passwd | more
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
...
group13:x:1003:1003:group13,,,,:/home/group13:/bin/bash
```

Назначение полей файла /etc/passwd:

регистрационное имя или логин : хэш пароля : идентификатор пользователя : идентификатор группы : информационное поле GECOS : домашний каталог : какой shell использует по умолчанию

Поле GECOS хранит вспомогательную информацию о пользователе (полное имя, адрес, рабочий телефон, домашний телефон и т.д.).

Группы тоже имеют имена, информация об этом содержится в файле /etc/group.
Структура файла /etc/group:

```
dorofeev@dorofeev-VirtualBox:~/Dorofeev/Lab2$ cat /etc/group
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
...
group13:x:1003:
```

Назначение полей файла /etc/group:
имя группы : пароль : идентификатор группы : список пользователей в группе

Если в файле /etc/passwd у пользователя указана группа в качестве основной, то в файле /etc/group для этой группы данный пользователь не указывается.

Посредством команды id, определим имена и идентификаторы владельца и группы нашего текущего процесса (shell`a):

```
dorofeev@dorofeev-VirtualBox:~/Dorofeev/Lab2$ id
uid=1003(group13) gid=1003(group13) groups=1003(group13)...

dorofeev@dorofeev-VirtualBox:~/Lab2$ id
uid=1000(dorofeev) gid=1000(dorofeev) группы=1000(dorofeev),...
dorofeev@dorofeev-VirtualBox:~$ id -u dorofeev    ; вывод ID пользователя
      1000
dorofeev@dorofeev-VirtualBox:~$ id -g dorofeev    ; вывод ID группы
      1000
```

2.5.3. Переустановка идентификатора владельца процесса

С помощью ls определим права доступа к файлу /etc/passwd, и к файлу /usr/bin/passwd (программа для смены пароля):

```
dorofeev@dorofeev-VirtualBox:~/Dorofeev/Lab2$ ls -l /etc/passwd
-rw-r--r-- 1 root root 1908 2011-09-14 17:09 /etc/passwd
dorofeev@dorofeev-VirtualBox:~/Dorofeev/Lab2$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 37100 2011-02-15 01:12 /usr/bin/passwd
```

«s» соответствует установленному биту SUID. При запуске обычного исполняемого файла, он получает реальные идентификаторы пользователя, его запустившего. В большинстве случаев эффективные идентификаторы такие же как реальные и по этим идентификаторам определяются права доступа процесса. В результате права доступа процесса чаще всего определяются правами пользователя запустившего процесс.
Если на исполняемый файл установлен бит suid, то при выполнении эта программа автоматически получит эффективные идентификаторы владельца исполняемого файла. То есть, не зависимо от того — кто запускает эту программу, она при выполнении имеет права хозяина этого файла.
У утилиты passwd установлен бит SUID из соображений, что пользователь может менять себе пароль самостоятельно, но чтобы это сделать необходимо обладать правами root (так как право на изменение файла /etc/passwd есть только у root).

Скопируем к себе /usr/bin/passwd с тем же именем. Проверим результат, в т.ч. владельца копии:

```
dorofeev@dorofeev-VirtualBox:~/Dorofeev/Lab2$ cp /usr/bin/passwd .
dorofeev@dorofeev-VirtualBox:~/Dorofeev/Lab2$ ls -l passwd
-rwxr-xr-x 1 group13 group13 37100 2011-10-10 12:19 passwd
```

Сравним оба файла:

```
dorofeev@dorofeev-VirtualBox:~/Dorofeev/Lab2$ cmp /usr/bin/passwd ./passwd
```

Файлы одинаковы.

Попробуем изменить себе пароль с помощью собственной копии passwd:

```
dorofeev@dorofeev-VirtualBox:~/Dorofeev/Lab2$ passwd
Changing password for group13.
current) UNIX password:       ;ввод текущего пароля
Enter new UNIX password:      ;ввод нового пароля
Retype new UNIX password:     ;повторный ввод нового пароля
Password unchanged            ;пароль не изменен
passwd: Authentication token manipulation error
passwd: password unchanged
```

Пароль не изменился, так как у нас нет прав.
Изменить пароль с использованием копии утилиты passwd не получилось, так как идентификаторы владельца у нее group13, group13, а не root root и у нее не установлен бит SUID.
Из всех вышеприведенных операций следует, что права доступа для исполняемых файлов определяются эффективными идентификаторами.

Создадим С-программу, которая выводит данные в файл:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
uid_t getuid(void);
uid_t geteuid(void);


int main(void)
{
  FILE *fp;

  if((fp=fopen("test", "wb"))==NULL) {
    printf("Error!\n");
    exit(1);
  }

  fprintf(fp, "Real ID of user:%d \n Real ID of group:%d \n Effective ID of
user: %d \n Effective ID      of group: %d \n", getuid(), getgid(),geteuid(),
getegid());
  fclose(fp);
  system ("cat test");
  return 0;
}
```

С помощью системных вызовов getuid(), getgid(), getegid(), geteuid() занесем в файл информацию о реальных и эффективных идентификаторах пользователя. Создадим исполняемый файл и выполним программу:

```
dorofeev@dorofeev-VirtualBox:~/Lab2$ gcc main.c
dorofeev@dorofeev-VirtualBox:~/Lab2$ ./a.out
```

В выходной файл test записалась информация об ID:

```
dorofeev@dorofeev-VirtualBox:~/Lab2$ cat test
Real ID of user:1000
Real ID of group:1000
Effective ID of user: 1000
Effective ID of group: 1000
```

С помощью назначения соответствующих прав доступа обеспечим режим обращения к файлу с данными только через нашу программу:

```
dorofeev@dorofeev-VirtualBox:~/Lab2$ chmod 641 test
dorofeev@dorofeev-VirtualBox:~/Lab2$ ls -l test
-rw-r--r-- 1 dorofeev dorofeev 105 2011-10-22 18:16 test
dorofeev@dorofeev-VirtualBox:~/Lab2$ chmod u+s a.out ; установка SUID
dorofeev@dorofeev-VirtualBox:~/Lab2$ ls -l a.out
-rwsr-xr-x 1 dorofeev dorofeev  7458 2011-10-22 18:16 a.out
```

```
dorofeev@dorofeev-VirtualBox:/home/dorofeev/Lab3$ ./a.out
Real ID of user:1001
Real ID of group:1000
Effective ID of user: 1000
Effective ID       of group: 1000
```

Устанавливаем SUID. Записывать информацию в test может только владелец dorofeev. Владельцем программы a.out так же является dorofeev. Таким образом, если другой пользователь запустит программу a.out, то увидит информацию записываемую в test, при этом, не имея права, что либо записывать в этот файл и читать его. Обеспечивается режим обращения к файлу с данными только через нашу программу:

2.5.4. Права доступа к каталогам

Определим права доступа к своему «домашнему» каталогу. Обеспечим посредством команды chmod все права себе, возможность чтения и поиска для нашей группы и никаких прав остальным:

```
dorofeev@dorofeev-VirtualBox:~$ ls -l
drwxr-xr-x 2 dorofeev dorofeev 4096 2011-10-22 18:16 Lab2
dorofeev@dorofeev-VirtualBox:~$ chmod 750 Lab2
dorofeev@dorofeev-VirtualBox:~$ ls -l
drwxr-x--- 2 dorofeev dorofeev 4096 2011-10-22 18:16 Lab2
```

Попытаемся поменять владельца файла report.txt с помощью chown:

```
dorofeev@dorofeev-VirtualBox:~/Lab2$ chown root report.txt
```

chown: изменение владельца «report.txt»: Операция не позволяется

Попытка не удалась, так как администратором не предоставлено прав на изменение владельцев файлов.

2.6. Содержимое каталогов

С помощью команды df определим занятое, свободное и общее пространство на дисках:

```
dorofeev@dorofeev-VirtualBox:~/lab1$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1       7.4G  3.5G  3.6G  50% /
udev            238M  4.0K  238M   1% /dev
tmpfs           100M  920K   99M   1% /run
none            5.0M     0  5.0M   0% /run/lock
none            248M  856K  247M   1% /run/shm
123             196G  169G   28G  87% /media/sf_123
/dev/sr0         53M   53M     0 100% /media/VBOXADDITIONS_4.2.0_80737
```

Посредством od прочтем содержимое нашего каталога как файла (od -cb):

```
dorofeev@dorofeev-VirtualBox:~$ od -cb Lab2
od: Lab2: ошибка чтения: Это каталог
```

Удалим файл pri и посмотрим содержимое каталога с помощью ls -a:

```
dorofeev@dorofeev-VirtualBox:~/Lab2$ ls -a
.  ..  a.out  Lab2  main.c  passwd  pri  prog.c  test
dorofeev@dorofeev-VirtualBox:~/Lab2$ rm pri
dorofeev@dorofeev-VirtualBox:~/Lab2$ ls -a
.  ..  a.out  Lab2  main.c  passwd  prog.c  test
```

Удалим файл pri и посмотрим содержимое каталога с помощью ls -i:

```
dorofeev@dorofeev-VirtualBox:~/Lab2$ ls -i
175769 a.out  184868 main.c  183414 pri     184037 test
183411 Lab2   179753 passwd  185280 prog.c
dorofeev@dorofeev-VirtualBox:~/Lab2$ rm pri
dorofeev@dorofeev-VirtualBox:~/Lab2$ ls -i
175769 a.out  184868 main.c  185280 prog.c
183411 Lab2   179753 passwd  184037 test
```

i-node файлов не изменились.

Переместим файл pri:

```
dorofeev@dorofeev-VirtualBox:~/Lab2$ ls -i pri
183441 pri
dorofeev@dorofeev-VirtualBox:~/Lab2$ mv pri ..
dorofeev@dorofeev-VirtualBox:~$ ls -i pri
183441 pri
```

I-node файла pri не изменился.

```
dorofeev@dorofeev-VirtualBox:~/Lab2$ ls -ld
drwxr-xr-x 2 dorofeev dorofeev 4096 2011-11-12 18:53 .
dorofeev@dorofeev-VirtualBox:~/Lab2$ mv pri ..
dorofeev@dorofeev-VirtualBox:~/Lab2$ ls -ld
drwxr-xr-x 2 dorofeev dorofeev 4096 2011-11-12 18:55 .
```

Размер файла-каталога не изменился. Система не сжимает каталог после уменьшения в нем числа файлов.

Создадим жесткую ссылку на файл prog.c:

```
dorofeev@dorofeev-VirtualBox:~/Lab2$ ln prog.c prog1.c
dorofeev@dorofeev-VirtualBox:~/Lab2$ ls -l prog.c
-rw-r--r-- 2 dorofeev dorofeev 70 2011-11-12 14:45 prog.c
```

ls показывает, что на файл prog.c есть две жесткие ссылки.

```
dorofeev@dorofeev-VirtualBox:~/Lab2$ ls -i
175769 a.out  184868 main.c  185280 prog1.c  184037 test
183411 Lab2   179753 passwd  185280 prog.c
```

Оба файла (prog.c и prog1.c) имеют один i-node (185280). Это объясняется тем, что на самом деле существует всего один файл, но две ссылки, которые отображаются как отдельные файлы.

Найдем в /bin или /usr/bin редактор vi и определим количество связей к нему:

```
dorofeev@dorofeev-VirtualBox:~/Lab2$ ls -l '/usr/bin/vi'
lrwxrwxrwx 1 root root 20 2011-10-14 23:54 /usr/bin/vi ->
/etc/alternatives/vi
```

Файл /usr/bin/vi — мягкая ссылка на /etc/alternatives/vi

```
dorofeev@dorofeev-VirtualBox:~/Lab2$ ls -l '/etc/alternatives/vi'
lrwxrwxrwx 1 root root 17 2011-10-14 23:54 /etc/alternatives/vi ->
/usr/bin/vim.tiny
```

Файл /etc/alternatives/vi — мягкая ссылка на /usr/bin/vim.tiny

```
dorofeev@dorofeev-VirtualBox:~/Lab2$ ls -l '/usr/bin/vim.tiny'
-rwxr-xr-x 1 root root 634356 2010-09-28 11:17 /usr/bin/vim.tiny
```

На этот файл существует только одна ссылка. Аналогично файл /usr/bin/ex ссылается на /usr/bin/vim.tiny. Файлы /usr/bin/vi и /usr/bin/ex занимают 20 байт.

Переименуем файл prog1.c (mv):

```
dorofeev@dorofeev-VirtualBox:~/Lab2$ ls -i
175769 a.out  184868 main.c  185280 prog1.c  184037 test
183411 Lab2   179753 passwd
dorofeev@dorofeev-VirtualBox:~/Lab2$ mv prog1.c pr.c
dorofeev@dorofeev-VirtualBox:~/Lab2$ ls -i
175769 a.out  184868 main.c  185280 pr.c
183411 Lab2   179753 passwd  184037 test
```

Номер описателя файла pr.c не изменился (185280). Это объясняется тем, что во время переименования не было создано никаких новых файлов, и они не перемещались, а изменился лишь атрибут файла — имя файла.

Точно так же дескриптор файла не меняется при перемещении файлов между каталогами. При копировании файлов он изменяется.

Удалим дополнительную связь к prog.c и посмотрим, как изменится его i-node:

```
dorofeev@dorofeev-VirtualBox:~/Lab2$ ls -l prog.c
-rw-r--r-- 2 dorofeev dorofeev 64 2011-11-12 19:23 prog.c
dorofeev@dorofeev-VirtualBox:~/Lab2$ ls -i
175769 a.out  184868 main.c  184877 prog1.c  184037 test
183411 Lab2   179753 passwd  184877 prog.c
dorofeev@dorofeev-VirtualBox:~/Lab2$ rm prog1.c
dorofeev@dorofeev-VirtualBox:~/Lab2$ ls -l prog.c
-rw-r--r-- 1 dorofeev dorofeev 64 2011-11-12 19:23 prog.c
dorofeev@dorofeev-VirtualBox:~/Lab2$ ls -i prog.c
184877 prog.c
```

i-node остался прежним, а количество ссылок уменьшилось.

```
dorofeev@dorofeev-VirtualBox:~/Lab2/ttt$ ls -a
.  ..  passwd
dorofeev@dorofeev-VirtualBox:~/Lab2/ttt$ ls -a ..
.  ..  a.out  Lab2  main.c  prog.c  test  ttt
```

Во всех каталогах существуют имена . и .., которые обозначают сам каталог и родительский каталог.

2.7. Специальные файлы

```
dorofeev@dorofeev-VirtualBox:~/Lab2$ ls -l /dev/tty
crw-rw-rw- 1 root tty 5, 0 2011-11-12 14:44 /dev/tty
dorofeev@dorofeev-VirtualBox:~/Lab2$ file /dev/tty
/dev/tty: character special   ; специальный файл-устройство

dorofeev@dorofeev-VirtualBox:~/Lab2$ ls -l /dev/sda
brw-rw---- 1 root disk 8, 0 2011-11-12 14:44 /dev/sda
dorofeev@dorofeev-VirtualBox:~/Lab2$ file /dev/sda
/dev/sda: block special
```

При выводе посредством команды ls -l для файлов указывается тип (первый символ). Соответственно для специальных файлов используются свои символы.

3. Выводы

При выполнении данной лабораторной работы были изучены принципы организации файловой системы ОС UNIX на примере Linux:

- Классическая файловая система представляет данные в виде вложенных друг в друга **каталогов** (их ещё называют папками), в которых содержатся **файлы**. Один из каталогов является «вершиной» файловой системы («корнем»), в нём содержатся все остальные каталоги и файлы.
- В Linux корневой каталог называется «/». Полные имена всех остальных каталогов получаются из «/», к которому дописываются справа имена последовательно вложенных друг в друга каталогов.
- Корневой каталог в Linux всегда только *один*, а все остальные каталоги в него вложены, т. е. для пользователя файловая система представляет собой единое целое. В действительности, разные части файловой системы могут находиться на совершенно разных устройствах: разных разделах жёсткого диска, на разнообразных съёмных носителях. Для того чтобы соорудить единое дерево с одним корнем, используется процедура **монтирования.**
- Монтирование — это подключение в один из каталогов целой файловой системы, находящейся где-то на другом устройстве. Эту операцию можно представить как «прививание» ветки к дереву. Для монтирования необходим пустой каталог — он называется **точкой монтирования**. Происходит объявление, что в данном *каталоге* (пока пустом) нужно отображать файловую систему, доступную на таком-то *устройстве* или же по сети. После этой операции в каталоге (точке монтирования) появятся все те файлы и каталоги, которые находятся на соответствующем устройстве.
- Для Linux самой важной является **корневая файловая система** (root filesystem). Именно к ней затем будут подключаться все остальные файловые системы на других устройствах. Точкой монтирования служит «/».

Так же были получены сведения об индексном дескрипторе (inode). Это структура данных в файловых системах, в которой хранится вся информация о стандартных файлах, каталогах или других объектах файловой системы, кроме непосредственно данных и имени.

При создании файловой системы создаются также и структуры данных, содержащие информацию о файлах. Каждый файл имеет свой индексный дескриптор, идентифицируемый по уникальному номеру, в файловой системе, в которой располагается сам файл. Индексные дескрипторы хранят информацию о файлах, такую как принадлежность владельцу (пользователю и группе), режим доступа (чтение, запись, запуск на выполнение) и тип файла.

Подобная концепция играет важную роль при восстановлении поврежденных файловых систем.

- Номер индексного дескриптора заносится в таблицу индексных дескрипторов в определенном месте устройства; по номеру индексного дескриптора ядро системы может считать содержимое инода.
- Номер индексного дескриптора файла можно посмотреть используя команду ls –i.