ЛАБОРАТОРНАЯ РАБОТА №4

Изучение процессов в ОС Linux

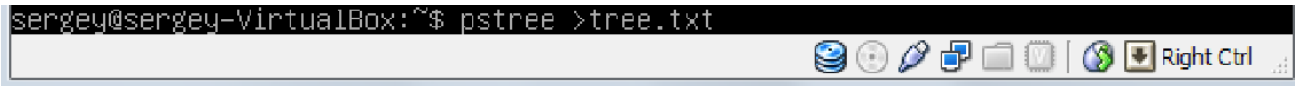
1. Понятие процесса в ОС Linux. После загрузки Linux на всех виртуальных консолях выводится приглашение пользователю начать работу. Для успешной авторизации необходимо ввести логин и пароль, которые служат аргументами программы *login.* Если *login* приходит к выводу, что работать можно, она запускает стартовый командный интерпретатор, посредством которой пользователь и управляет системой.

Программа в стадии выполнения называется *процессом*. Все процессы система регистрирует в *таблице процессов*, в которой каждому присваивается уникальный номер – *идентификатор процесса (PID)*. Манипулируя с процессами, система имеет дело именно с их идентификаторами. Для просмотра всех запущенных процессов пользователя можно воспользоваться командой *ps –all* (модификатор *all* позволяет вывести большее количество информации о процессах):

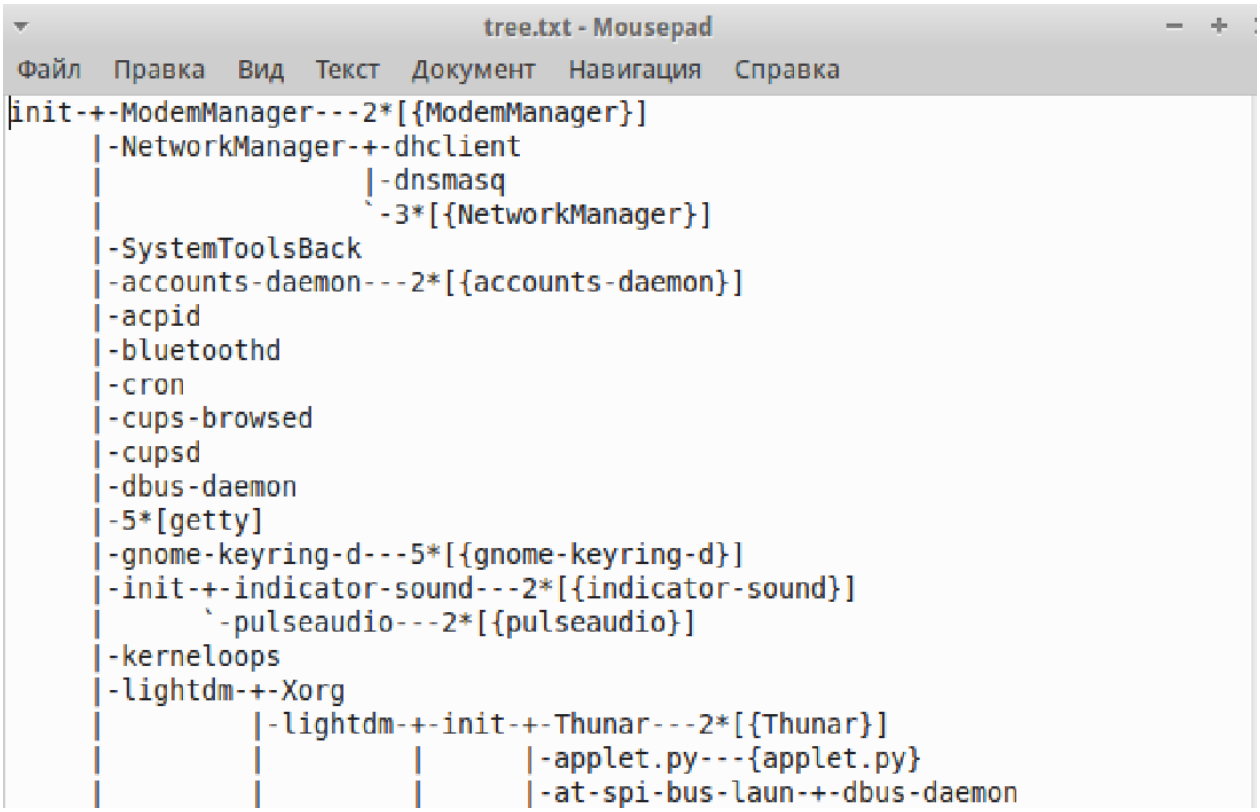
```
sergey@sergey-VirtualBox:~$ ps -all
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S  1000  2298  1070  0  80   0 -  2115 wait   tty1     00:00:14 bash
0 R  1000  3694  2298  0  80   0 -  1617 -      tty1     00:00:00 ps
sergey@sergey-VirtualBox:~$
```

Видим всего два процесса: это командный интерпретатор *bash* и выполняющийся *ps*. Все они запущены с первой консоли (*tty1*). В столбце *PID* указан идентификатор процесса, в столбце *PPID* – идентификатор родительского процесса, т.е. процесса, породившего данный, в столбце UID – идентификатор пользователя, запустившего данный процесс. Для процесса *ps* родительским является процесс *–bash* (посмотрите на PID), а для *-bash* – процесс *login*, которые не отображается в силу того, что он не является процессом пользователя *sergey*. Столбец *S* отвечает за статус процесса: *S* означает процесс находится в состоянии ожидания, *R* – процесс в состоянии готовности (т.е. процесс готов к выполнению и ожидает, кода освободится процессор для него), *T* (*D*) – состояние блокировки.

Интересной командой является команда *pstree* – выводит дерево всех процессов в системе. Дерево на экране может не поместиться, поэтому перенаправим (знак «>») вывод результатов команды *pstree* в текстовый файл *tree.txt*:

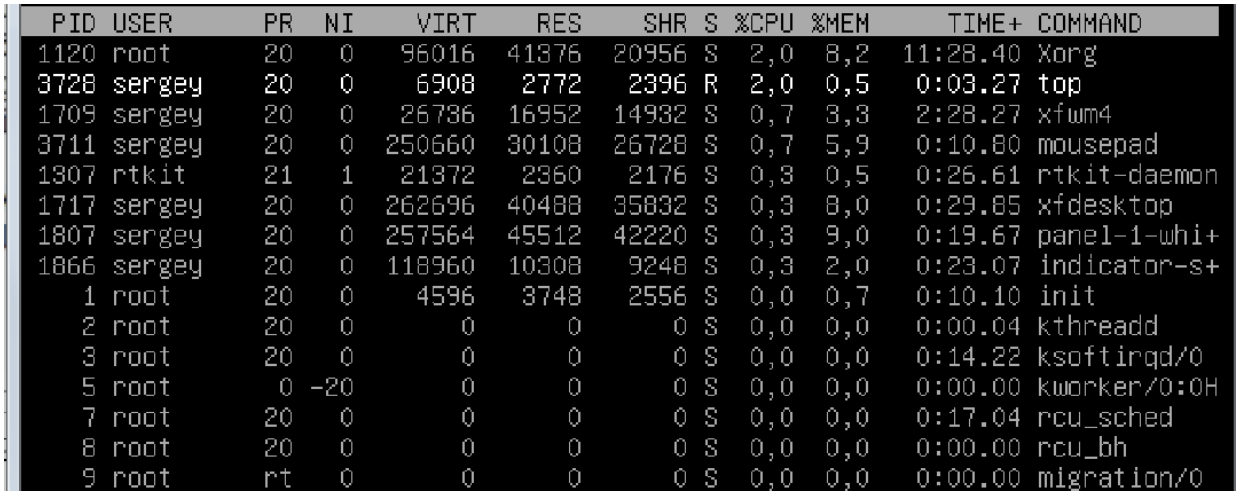


Переключимся в графическую консоль (CTRL+ALT+F7), перейдем в домашний каталог и откроем этот текстовый файл (*tree.txt*):



Видим, процесс *init* является родительским для всех процессов, его *PID*=1.

Просмотреть состояния процессов в реальном времени позволяет команда *top:*



Выйти из просмотра процессов можно, нажав клавишу «*q*» на клавиатуре.

Сразу после загрузки ядро монтирует корневую файловую систему и запускает процесс *init*. Теперь процесс *init* ответственен за продолжение загрузки ОС. Помимо этого *init* выполняет еще массу различных операций: проверка и монтирование файловых систем, запуск различных служб (демонов), запуск процедур логирования, оболочек пользователей на различных терминалах и т.д. *Демон* – это процесс, ничего не выводящий на терминал и работающий в фоновом режиме. Демоны обычно используются для выполнения сервисных функций, обслуживания запросов от других процессов и т.д.

Процессы получают доступ к *ресурсам системы* (оперативная память, файлы, внешние устройства и т.д.) и могут изменять их содержимое. Доступ регулируется с помощью идентификатора пользователя PID и идентификатора группы UID, которые система присваивает каждому процессу. Каждый процесс имеет свое *окружение* – набор переменных и их значений. Для каждого пользователя этот набор специфичен. Каждый процесс располагает и свободно

распоряжается своей копией окружения, которую он получает от родителя. Таким образом, если процесс сначала модифицирует свое окружение, а потом породит новый процесс, то потомок получит копию уже измененного окружения.

Переменная среды́ – это текстовая переменная операционной системы, хранящая какую-либо информацию. Например, она может хранить данные о настройках системы. Пользователь может и самостоятельно создавать переменные среды и присваивать им значения. Создадим переменную *a* и присвоим эти значение 123. Посмотрим результат с помощью команды *echo*:

```
sergey@sergey-VirtualBox:~$ a=123
sergey@sergey-VirtualBox:~$ echo $a
123
```

Для удаления переменной выполним команду *unset a*:

```
sergey@sergey-VirtualBox:~$ unset a
sergey@sergey-VirtualBox:~$ echo $a

sergey@sergey-VirtualBox:~$
```

Также имеются много переменных, которые уже созданы системой. Например – переменная *RANDOM*. При ее вызове каждый раз возвращается случайное целое число в диапазоне от 0 до 32768:

```
sergey@sergey-VirtualBox:~$ echo $random

sergey@sergey-VirtualBox:~$ echo $RANDOM
29479
sergey@sergey-VirtualBox:~$ echo $RANDOM
16594
sergey@sergey-VirtualBox:~$ echo $RANDOM
12167
sergey@sergey-VirtualBox:~$
```

Для запуска одного процесса вместо другого служит системный вызов *exec ()*. Старый процесс удаляется из памяти навсегда, вместо него загружается новый, при этом наследует окружения старого процесса (PID также не меняется). Вернуться к выполнению старого процесса невозможно. Единственные вариант – заново запустить с помощью *exec ()*.

Заметим, что имя программы (файла), запустившей процесс и имя процесса (в таблице процессов первая колонка справа) могут не совпадать.

Создадим каталог *lab_4* и в нем – простую программу, которая ничего не выполняет, но процесс для ее реализации будет создан. Ведем в консоль следующий код:

```
sergey@sergey-VirtualBox:~/lab_4$ cat >loop
while true; do true;done
sergey@sergey-VirtualBox:~/lab_4$
```

По окончании ввода нажмем сочетание клавиш *CTRL+C*. В результате наших действий будет создан файл с именем *«loop»*, в котором содержится код (это всего три фразы, введенные нами).

Запустим этот файл, выполнив команду:

```
sergey@sergey-VirtualBox:~/lab_4$ bash loop
```

Переключимся в графическую консоль, запустим диспетчер задач и найдем в нем процесс *bash loop*:

Задача	PID	PPID	Резидентная	ЦП
bash loop	2152	2049	2652 КиБ	85%
Диспетчер задач	2160	1891	23 МиБ	9%

Видим, что данный процесс занимает 85% процессорного времени. Переключимся в первую консоль и нажмем сочетание клавиш *CTRL+C*, что приведет к завершению нашего бесполезного процесса.

Мы только что запускали дочерний процесс, который ожидал нажатия клавиш *CTRL+C*. Пока этот процесс работал, пользователь не имел доступа к оболочке (родительский процесс). Т.е. мы не могли печатать никакие команды в командной строке, она была «занята». Запустим теперь тот же процесс, но в параллельном (фоновом) режиме:

```
sergey@sergey-VirtualBox:~/lab_4$ bash loop&
[1] 2057
sergey@sergey-VirtualBox:~/lab_4$
```

Высветился PID процесса (PID=2057). Число в квадратных скобках означает *порядковый номер задания*. При таком запуске пользователь имеет доступ к оболочке и может полноценно в ней работать дальше. Процесс, запускаемый параллельно, выполняется в фоновом режиме. Он не может вводить данные с того же терминала, но может выводить их. Активный процесс – это процесс, имеющий возможность вводить данные с терминала.

Введем команду *ps –f*:

```
sergey@sergey-VirtualBox:~/lab_4$ ps -f
UID          PID  PPID  C STIME TTY          TIME CMD
sergey      1496  1457  0 18:35 tty1     00:00:00 -bash
sergey      2057  1496 63 18:39 tty1     00:01:35 bash loop
sergey      2060  1496 29 18:40 tty1     00:00:18 bash loop
sergey      2061  1496 28 18:40 tty1     00:00:17 bash loop
sergey      2063  1496  0 18:41 tty1     00:00:00 ps -f
sergey@sergey-VirtualBox:~/lab_4$
```

Видно, что для процесса *loop* родительским является процесс *-bash*. Для возврата процесса из фонового режима необходима команда *fg <порядковый номер задания>*. Например, команда *fg %1* сделает активным задание с порядковым номером 1.

Запустим еще три раза программу *loop* и посмотрим, как будут распределены ресурсы системы между процессами *loop*. Выполним команду *ps –u:*

```
sergey@sergey-VirtualBox:~/lab_4$ bash loop&
[2] 2060
sergey@sergey-VirtualBox:~/lab_4$ bash loop&
[3] 2061
sergey@sergey-VirtualBox:~/lab_4$ ps -u
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
sergey    1496  0.0  0.9   8392  4680 tty1     S    18:35   0:00 -bash
sergey    2057 82.6  0.5   6756  2648 tty1     R    18:39   1:20 bash loop
sergey    2060 32.2  0.5   6756  2576 tty1     R    18:40   0:02 bash loop
sergey    2061 28.1  0.5   6756  2580 tty1     R    18:40   0:01 bash loop
sergey    2062  0.0  0.4   6704  2348 tty1     R+   18:40   0:00 ps -u
sergey@sergey-VirtualBox:~/lab_4$
```

Для завершения процесса с помощью комбинаций клавиш CTRL+C, его сначала необходимо сделать активным (команда *fg*). Это не всегда удобно, особенно в случае, когда пользователем запущено много процессов. Чтобы остановить процесс, ему надо отправить *сигнал* – короткое сообщение, посылаемое процессу системой или другим процессом. Для отправки сигнала существует команда *kill ()*. Для остановки все процессов, запущенных нами,

перезагрузим систему командой *exit*. Затем заново авторизуемся в системе, перейдем в каталог *lab_4* и запустим процесс *loop* два раза:

```
admin@VirtualBox:~/lab_4$ ps -u
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
admin     2212  0.0  0.9   8392  4632 tty2     S+   11:23   0:00 -bash
admin     2349  0.0  0.9   8392  4620 tty1     S    11:34   0:00 -bash
admin     2374  0.4  0.5   6756  2596 tty1     R    11:43   0:20 bash loop
admin     2380 99.7  0.5   6756  2596 tty1     R    11:55  54:45 bash loop
admin     2387  0.0  0.4   6704  2396 tty1     R+   12:50   0:00 ps -u
admin@VirtualBox:~/lab_4$
```

Остановим процесс с PID=2374 с помощью команды *kill <-модификатор> <PID>*:

```
admin@VirtualBox:~/lab_4$ kill -stop 2374
admin@VirtualBox:~/lab_4$ ps -u
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
admin     2212  0.0  0.9   8392  4632 tty2     S+   11:23   0:00 -bash
admin     2349  0.0  0.9   8396  4624 tty1     S    11:34   0:00 -bash
admin     2374  1.4  0.5   6756  2596 tty1     T    11:43   1:01 bash loop
admin     2380 98.4  0.5   6756  2596 tty1     R    11:55  55:29 bash loop
admin     2388  0.0  0.4   6704  2836 tty1     R+   12:51   0:00 ps -u

[1]+  Stopped                 bash loop
admin@VirtualBox:~/lab_4$ _
```

Модификатор *–stop* означает, что процесс будет остановлен. После выполнения команды в столбце *STAT* процессу с PID=2374 будет соответствовать буква «*T*».

«Убьем» процесс с PID=2380, используя команду *kill* с модификатором *–kill*:

```
admin@VirtualBox:~/lab_4$ kill -kill 2380
admin@VirtualBox:~/lab_4$ ps -u
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
admin     2212  0.0  0.9   8392  4632 tty2     S+   11:23   0:00 -bash
admin     2349  0.0  0.9   8396  4624 tty1     S    11:34   0:00 -bash
admin     2374  1.3  0.5   6756  2596 tty1     T    11:43   1:01 bash loop
admin     2390  0.0  0.4   6704  2400 tty1     R+   12:58   0:00 ps -u
[2]-  Killed                  bash loop
```

Запустим процесс с PID=2374, а потом убьем его:

```
admin@VirtualBox:~/lab_4$ kill -cont 2374
admin@VirtualBox:~/lab_4$ ps -u
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
admin     2212  0.0  0.9   8392  4632 tty2     S+   11:23   0:00 -bash
admin     2349  0.0  0.9   8396  4624 tty1     S    11:34   0:01 -bash
admin     2374  1.2  0.5   6756  2596 tty1     R    11:43   1:23 bash loop
admin     2399  0.0  0.4   6704  2400 tty1     R+   13:32   0:00 ps -u
admin@VirtualBox:~/lab_4$ kill -kill 2374
admin@VirtualBox:~/lab_4$ ps -u
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
admin     2212  0.0  0.9   8392  4632 tty2     S+   11:23   0:00 -bash
admin     2349  0.0  0.9   8396  4624 tty1     S    11:34   0:01 -bash
admin     2400  0.0  0.4   6704  2396 tty1     R+   13:32   0:00 ps -u
[1]+  Killed                  bash loop
admin@VirtualBox:~/lab_4$
```

2. Доступ процессов к файлам и каталогам. В лабораторной работе №3 было уже упомянуто об атрибутах файла. Для каждого файла имеются индивидуальные права доступа, которые разбиты на три группы:

1. доступ для пользователя-владельца файла;
2. доступ для группы-владельца файла;
3. доступ для остальных пользователей.

С точки зрения операционной системы процессы управляют файлами и каталогами, создают и удаляют их. Факт использования файла процессом называется *доступом* к файлу, а способ воспользоваться файлом – *видом* доступа.

В Linux имеется три вида доступа. Доступ на *чтение* (бит *r*) разрешает получать информацию из объекта, доступ на *запись* (бит *w*) – изменять информацию в объекте, доступ на *использование* (бит *x*) – выполнять операцию, специфичную для данного типа объектов.

Между файлом и пользователем (процессом) роль распределяется так:

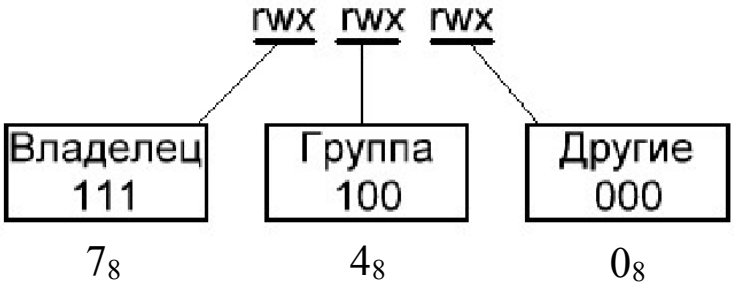
- если UID файла совпадает с UID процесса, его использующего, то пользователь – хозяин файла;
- если GID файла совпадает с GID любой группы, в которую входит пользователь, он – член группы, которой принадлежит файл;
- если ни UID ни GID не пересекаются с UID процесса и списком групп, в которые входит запустивший его пользователь, этот пользователь – посторонний.

Для обычного файла бит чтения (*r*) означает возможность открывать и читать файл. Бит записи (*w*) позволяет изменять содержимое файла, в том числе полностью очищать его. Правом удаления и переименования файла управляют биты, заданные для его родительского каталога, поскольку именно там хранится имя файла.

Бит выполнения *x* служит для разрешения выполнения файла. Существует два типа исполняемых файлов: бинарные фалы, которые непосредственно центральным процессором, и сценарии, обрабатываемые интерпретатором команд или какой-либо другой программной. Для каталога бит выполнения *x* означает возможность переходить в каталог, т.е. делать его текущим (команда *cd*), но без получения списка файлов. Для того, чтобы просмотреть содержимое каталога необходимо установить комбинацию битов чтения и выполнения (*r=1, x=1*). Комбинация битов записи и выполнения (*w=1, x=1*) позволяет создавать, удалять и переименовывать файлы из данного каталога.

Для смены прав доступа к объекту файловой системы существует команда *chmod <права доступа> <имя файла (каталога)>*. Параметр *<права доступа>* выражаются числом в восьмеричной системе счисления.

Например, последовательность 111 100 100 означает, что для владельца разрешено чтение (*r=1*), запись (*w=1*) и выполнение (*x=1*), для группы разрешено чтение (*r=1*), запись и выполнение запрещены (*w=0, x=0*), для всех остальных все запрещено (*r=0, w=0, x=0*) (см. рис. ниже). Для задания таких прав необходимо каждую тройку бит перевести в восьмеричную систему счисления, т.е. $111_2=7_8$, $100_2=4_8$ и $000_2=0_8$. Таким образом, команда примет вид: *chmod 704 <имя файла (каталога)>*.



Вновь обратимся к файлу *loop* и посмотрим его атрибуты, установленные по умолчанию, а также выясним, кто его владелец и какой группе он принадлежит:

```
admin@VirtualBox:~/lab_4$ ls -li
total 4
183781 -rw-rw-r-- 1 admin admin 26 сент. 21 10:13 loop
admin@VirtualBox:~/lab_4$ ~
```

«*-rw-rw-r--*» – файл может читать и редактировать как владелец, так и группа, а остальные пользователи могут только читать этот файл. Файл принадлежит группе *admin* и пользователю *admin*. Для изменения его атрибутов наберем команду *chmod 777 loop*. Тем самым мы разрешили право на запись, чтение и выполнение владельцу, группе и всем остальным пользователям (процессам):

```
admin@VirtualBox:~/lab_4$ chmod 777 loop
admin@VirtualBox:~/lab_4$ ls -li
total 4
183781 -rwxrwxrwx 1 admin admin 26 сент. 21 10:13 loop
admin@VirtualBox:~/lab_4$
```

Доступ к каталогам немного отличается от доступа к файлам. Доступ по чтению – это возможность просмотреть содержимое каталога (список фалов), доступ по записи – возможность изменить содержимое каталога, доступ для использования – возможность воспользоваться этим содержимым: сделать этот каталог текущим и обратиться за доступом к содержащимся в нем файлам.

Авторизуемся в консоли №2 под именем *test*. Создадим в домашнем каталоге каталог *dir*. Посмотрим его атрибуты:

```
test@VirtualBox:/home/test$ mkdir dir
test@VirtualBox:/home/test$ ls -li
total 12
183748 drwxrwxr-x 2 test test 4096 сент. 22 08:54 dir
```

Хозяином каталога *dir* является пользователь *test*. У пользователей *admin* и *test* нет общих групп, поэтому пользователь *admin* является сторонним по отношению к каталогу *dir*. По умолчанию почти все права разрешены (кроме записи в каталог сторонними пользователями). Установим запрет на чтение (команда *chmod 740 dir*) для сторонних пользователей, переключимся в консоль №1 и попытаемся от имени учетной записи *admin* войти в каталог *test* командой *cd*:

```
test@VirtualBox:/home/test$ chmod 740 dir
test@VirtualBox:/home/test$
```

```
admin@VirtualBox:/home/test$ cd dir
-bash: cd: dir: Permission denied
admin@VirtualBox:/home/test$
```

Доступ на чтение запрещен, что и следовало ожидать. Установим теперь разрешение на чтение, запись и выполнение для сторонних пользователей и снова попытаемся войти в каталог *dir* посредством команды *cd*, используя учетную запись *admin*:

```
test@VirtualBox:/home/test$ chmod 707 dir
test@VirtualBox:/home/test$ ls -li
total 12
183748 drwx---rwx 2 test test 4096 сент. 22 08:54 dir
```

```
admin@VirtualBox:/home/test$ cd dir
admin@VirtualBox:/home/test/dir$ ls -li
total 0
admin@VirtualBox:/home/test/dir$
```

Теперь пользователь *admin* смог получить доступ к каталогу *dir*.

Скопируем файл *loop* из каталога *admin* в каталог *home/test/dir* и посмотрим его атрибуты:

```
admin@VirtualBox:~$ cp loop /home/test/dir
```

```
test@VirtualBox:/home/test/dir$ ls -li
total 4
184018 -rwxrwxr-x 1 admin admin 25 сент. 22 13:44 loop
```

Видим, что несмотря на то, что файл *loop* теперь находится в каталоге пользователя *test*, его владелец не поменялся.

Все же в Linux имеется возможность изменить владельца файла (каталога) и группу, которой он принадлежит. Для смены владельца файла применим команду *chown <новый_владелец> <файл(каталог)>*. Для смены группы, которой принадлежит файл, требуется команда *chgrp <новая_группа> <файл (каталог)>*. Напомним, что операции по изменению владельцев файлов и каталогов необходимо проводить от имени администратора. Включить режим администратора можно командой *sudo –i* и ввести пароль текущей учетной записи. Также отметим, что запустить пользователя с правами администратора модно лишь в том случае, когда он добавлен в группу администратора. Например, изменим владельца файла *loop:*

```
admin@VirtualBox:~$ sudo -i
root@VirtualBox:~# pwd
/root
root@VirtualBox:~# cd /home/admin
root@VirtualBox:/home/admin# chown test loop
root@VirtualBox:/home/admin# ls -li
total 40
183777 drwxrwxr-x 2 admin admin 4096 сент. 21 14:44 lab_4
183890 -rwxrwxrwx 1 test  admin   25 сент. 21 19:55 loop
131486 drwxr-xr-x 2 admin admin 4096 сент. 19 13:04 Видео
131473 drwxr-xr-x 2 admin admin 4096 сент. 19 13:04 Документы
131438 drwxr-xr-x 2 admin admin 4096 сент. 19 13:04 Загрузки
131485 drwxr-xr-x 2 admin admin 4096 сент. 19 13:04 Изображения
131484 drwxr-xr-x 2 admin admin 4096 сент. 19 13:04 Музыка
131472 drwxr-xr-x 2 admin admin 4096 сент. 19 13:04 Общедоступные
131432 drwxr-x--- 2 admin admin 4096 сент. 19 13:04 Рабочий стол
131447 drwxr-xr-x 2 admin admin 4096 сент. 19 13:04 Шаблоны
root@VirtualBox:/home/admin#
```

А теперь изменим группу, в которую входит файл *loop:*

```
root@VirtualBox:/home/admin# chgrp test loop
root@VirtualBox:/home/admin# ls -li
total 40
183777 drwxrwxr-x 2 admin admin 4096 сент. 21 14:44 lab_4
183890 -rwxrwxrwx 1 test  test    25 сент. 21 19:55 loop
131486 drwxr-xr-x 2 admin admin 4096 сент. 19 13:04 Видео
131473 drwxr-xr-x 2 admin admin 4096 сент. 19 13:04 Документы
131438 drwxr-xr-x 2 admin admin 4096 сент. 19 13:04 Загрузки
131485 drwxr-xr-x 2 admin admin 4096 сент. 19 13:04 Изображения
131484 drwxr-xr-x 2 admin admin 4096 сент. 19 13:04 Музыка
131472 drwxr-xr-x 2 admin admin 4096 сент. 19 13:04 Общедоступные
131432 drwxr-x--- 2 admin admin 4096 сент. 19 13:04 Рабочий стол
131447 drwxr-xr-x 2 admin admin 4096 сент. 19 13:04 Шаблоны
root@VirtualBox:/home/admin#
```

Вернем обратно владельца и группу файлу *loop*:

```
root@VirtualBox:/home/admin# chown admin loop
root@VirtualBox:/home/admin# chgrp admin loop
root@VirtualBox:/home/admin# ls -li
total 40
183777 drwxrwxr-x 2 admin admin 4096 сент. 21 14:44 lab_4
183890 -rwxrwxrwx 1 admin admin   25 сент. 21 19:55 loop
```

ЗАДАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ

0. Подробно изучите материал, представленный в данной работе.
1. Создайте каталог с именем *lab_4* в домашнем каталоге.
2. Изучение процессов:
    2.1. Сохраните дерево процессов в каталоге *lab_4*.
    2.2. Выведите список всех процессов на консоль в реальном времени.
    2.3. Создайте файл *loop* в каталоге *lab_4*, запустите его в фоновом и активном режимах от имени администратора.
    2.4. Установите *rwx* биты для файла *loop*, чтобы пользователь *test* не имел права запускать этот файл.
    2.5. Запустите процесс *loop* в фоновом режиме, затем остановите его, затем вновь запустите и в итоге «убейте» командой *kill*.
    2.6. Запустите три процесса *loop* и по очереди «убейте» их.
    2.5. Установите *rwx* биты для каталога *lab_4*
3. Подготовьте ответы на вопросы:
    3.1. Что называют процессом? Таблицей процессов? PID?
    3.2. Что такое демон?
    3.3. Что такое окружение процесса?
    3.4. Назовите процесс, который запускается самым первым при загрузке ОС.
    3.5. Что такое переменная среды?
    3.6. Назовите группы прав доступа к объектам файловой системы.
    3.7. Поясните смысл *rwx* битов. Приведите пример.
    3.8. Как определить владельца файла или каталога?
    3.9. Что называют потоком?
    3.10. В чем отличие процесса и потока?
4. Будьте готовы ответить на дополнительные вопросы преподавателя по данной теме.