Администрирование Linux

Ivan M. Zhdanov

13 августа 2014 г.

Содержание

1 Начало работы с Linux **4**

2 Файлы в Linux **4**

2.1 Типы файлов в Linux . . . 4
2.2 Подробная информация о типах файлов . . . 5
2.2.1 Обычный файл . . . 6
2.2.2 Каталог . . . 6
2.2.3 Символьная ссылка . . . 7
2.2.4 Жёсткая ссылка . . . 8
2.2.5 Символьный специальный файл . . . 9
2.2.6 Блочный специальный файл . . . 10
2.2.7 Файл-очередь . . . 11
2.2.8 Сокет . . . 12
2.3 Описание некоторых файлов из каталога /dev . . . 13
2.3.1 /dev/null, /dev/zero . . . 13
2.3.2 /dev/random, /dev/urandom . . . 13
2.3.3 /dev/ttyS?, /dev/ttyUSB? . . . 14
2.3.4 /dev/hd?, /dev/sd? . . . 14
2.3.5 /dev/md?, /dev/dm? . . . 14
2.3.6 /dev/loop? . . . 14
2.3.7 /dev/ram? . . . 15
2.4 Примеры работы с файлами . . . 15
2.4.1 less more . . . 15
2.4.2 cat tac . . . 16
2.4.3 head tail . . . 17
2.4.4 hexdump . . . 17
2.4.5 dd . . . 17
2.4.6 touch . . . 18
2.4.7 sed . . . 18
2.4.8 vi . . . 18
2.5 Точки монтирования . . . 18
2.5.1 Команда mount . . . 18
2.5.2 Команда lsof . . . 19
2.6 Пространства имён . . . 20

3 Управление пользователями и правами доступа **21**

3.1 Работа с локальными пользователями . . . 21
3.1.1 Добавление локальных пользователей . . . 21
3.1.2 Изменение локальный пользователей . . . 22
3.1.3 Удаление локального пользователя . . . 22
3.2 Работа с локальными группами . . . 22
3.2.1 Добавление локальной группы . . . 23

3.2.2 Изменение локальной группы . . . 23
3.2.3 Удаление локальной группы . . . 23
3.2.4 Управление паролями . . . 23
3.3 Права доступа к файлам . . . 23
3.3.1 Описание POSIX прав . . . 24
3.3.2 Управление POSIX правами . . . 25
3.4 Утилита sudo . . . 26
3.4.1 Использование утилиты sudo . . . 26
3.4.2 Настройка утилиты sudo . . . 27
3.5 SELinux . . . 27
3.5.1 Проверка . . . 27
3.5.2 Отключение . . . 28

Введение

Данный текст предназначен для ознакомления сотрудников компании Paragon с основами работы в ОС Linux. Эти знания необходимы сотрудникам команды UFSD т.к. основная система для тестов или обслуживания тестовых серверов для нашей команды является Linux. Многое из описанного в отношении линукса можно так-же применить при работе с Mac OS, QNX или Windows через cygwin. Данный текст составлялся на основе лекций и семинаров проведённых для новых сотрудников компании, в случае нахождения ошибок в тексте можно о них сообщить по адресу ivan.zhdanov@paragon-software.com .

1 Начало работы с Linux

ОС Linux изначально состояла только из ядра, а все сопутствующие утилиты не входили в состав ОС. Сейчас под словосочетанием ОС Linux подразумевается ядро и минимальный набор утилит, с помощью которых возможна работа в ОС. Под работой в нашем случае нужно понимать операции с файлами, передача данных по сети и автоматизация выше перечисленного для выполнения тестирования драйверов файловых систем.
Многое из описанного дальше можно найти в документации поставляемой в составе дистрибутивов Linux. Для вызова справки по любой команде нужно выполнить команду:

man command

где «command» - имя команды по которой нужно получить справку.

2 Файлы в Linux

2.1 Типы файлов в Linux

В линуксе есть несколько типов файлов, которые могут быть созданы на файловых системах. Кроме стандартных файлов и каталогов есть ещё файлы ссылок и специальные файлы. Некоторые файловые системы могут не поддерживать работы со специальными файлами, но тогда пользователь узнает об этом при попытке создать специальный файл.
Специальные файлы бывают следующих типов:

- Обычный файл — Все файлы программ, данные, логи и т.д.(2.2.1)
- Каталог — Каталоги содержащие другие каталоги или файлы. (2.2.2)
- Символьная ссылка — Файл, указывающий на другой файл. (2.2.3)
- Жёсткая ссылка — Ещё одно имя для имеющегося файла. (2.2.4)
- Символьный специальный файл — файл символьного устройства. (2.2.5)

- Блочный специальный файл — файл блочного устройства. (2.2.6)
- Файл-очередь — файл pipe. (2.2.7)
- Сокет — файл сокета. (2.2.8)

2.2 Подробная информация о типах файлов

Для определения типа файла мы будем использовать утилиты ls и stat. Это не единственные варианты определения типа файла, но другие способы обычно используются при написании скриптов или поиске файла и будут рассматриваться отдельно. Перед началом работы с файлами, кратко разберём формат вывода нужных нам команд.

- Вызвав команду ls

```
ls -l /etc/krb5.conf

-rw-r--r--. 1 root wheel 432 May 14  2013 /etc/krb5.conf
```

-rw-r–r–. – группа показывающая тип и права доступа к файлу.
1 – количество ссылок на файл.
root – владелец файла
wheel – группа файла
432 – размер файла
May 14 2013 – Дата изменения файла
/etc/krb5.conf – Имя файла

- Вызвав команду stat

```
stat /etc/krb5.conf
  File: '/etc/krb5.conf'
  Size: 432               Blocks: 8     IO Block: 4096   regular file
Device: fd01h/64769d        Inode: 399647      Links: 1
Access: (0644/-rw-r--r--)  Uid: (   0/   root)   Gid: (  10/   wheel)
Context: system_u:object_r:krb5_conf_t:s0
Access: 2014-06-29 14:35:26.897744651 +0400
Modify: 2013-05-14 02:47:43.000000000 +0400
Change: 2013-05-25 12:47:09.245682485 +0400
```

File – Имя файла
Size – Размер файла
Blocks – Количество блоков занимаемых на диске
IO Block – Размер блока на диске.
regular file — Тип выбранного в данный момент файла.
Device – Устройство файла (в зависимости от типа файла и реализации может

не нести информации)
Inode – inode файла на файловой системе
Links – количество жёстких ссылок на файл
Access – группа прав доступа
Uid – владелец файла
Gid – группа файла
Context – контекст прав для SELinux
Access – Время последнего доступа к файлу
Modify – Время изменения данных файла
Change – Время изменения метаинформации (права доступа, имя) о файле

2.2.1 Обычный файл

К этому типу файлов относится большинство файлов в ОС. Такие файлы как конфигурационные, файлы с данными, скрипты, бинарные утилиты и файлы журналов все они являются обычными файлами. Для работы с обычными файлами используются утилиты для чтения файлов и модификации файлов о которых будет сказано ниже. Определить что файл относится к типу обычных можно следующим образом:

- Вызвав команду ls

```
ls -l /etc/krb5.conf

-rw-r--r--. 1 root root 432 May 14  2013 /etc/krb5.conf
```

- Вызвав команду stat

```
stat /etc/krb5.conf
  File: '/etc/krb5.conf'
  Size: 432               Blocks: 8     IO Block: 4096   regular file
Device: fd01h/64769d        Inode: 399647      Links: 1
Access: (0644/-rw-r--r--)  Uid: (   0/   root)   Gid: (   0/   root)
Context: system_u:object_r:krb5_conf_t:s0
Access: 2014-06-29 14:35:26.897744651 +0400
Modify: 2013-05-14 02:47:43.000000000 +0400
Change: 2013-05-25 12:47:09.245682485 +0400
```

Обычный файл обозначает символ «-» группы прав доступа, или текстовое название типа файла в выводе stat.

2.2.2 Каталог

Каталог используется для структуризации хранения файлов и других каталогов. Создать каталог можно командой «mkdir», удалить «rm -R» или «rmdir» для пустого каталога.
Определить каталог можно следующим образом:

- Вызвав команду ls

```
ls -l /

drwxr-xr-x. 170 root root 12288 Aug  6 21:16 etc
```

- Вызвав команду stat

```
stat /etc

  File: '/etc'
  Size: 12288             Blocks: 24         IO Block: 4096   directory
Device: fd01h/64769d        Inode: 393217      Links: 170
Access: (0755/drwxr-xr-x)  Uid: (    0/   root)   Gid: (    0/   root)
Context: system_u:object_r:etc_t:s0
Access: 2014-08-06 21:16:21.921278824 +0400
Modify: 2014-08-06 21:16:03.471648715 +0400
Change: 2014-08-06 21:16:03.471648715 +0400
```

Каталог обозначает символ «d» группы прав доступа, или текстовое название типа файла в выводе stat. Для каталога по значению количества жёстких ссылок можно определить сколько подкаталогов находится в этом каталоге.

2.2.3 Символьная ссылка

Символьная ссылка специальный файл, который позволяет получить доступ к файлу находящемуся в другом месте. Символьная ссылка содержит в себе путь к файлу на который она ссылается, поэтому существует возможность что ссылка будет указывать в пустоту или сама на себя. При обращении к ссылке указывающей в пустоту или на саму себя операционная система будет выдавать ошибку доступа. Пусть указанный в ссылке может быть абсолютный или относительный. При обращению к файлу символической ссылки из обычных программ, система перенаправляет запрос на файл цель этой ссылки. Поэтому для программ символическая ссылка выглядит как файл цель. Некоторые программы при открытии файла выставляют флаг «O_NOFOLLOW», для предотвращения перехода по символической ссылке. Обычно этот режим используют в сетевых серверах, для предотвращения выхода из разрешённой области файловой системы.
Для создания символьною ссылки нужно выполнить команду:

```
ln -s "Значение ссылки" "Имя ссылки"
```

Удалить символическую ссылку можно командой:

```
rm "Имя ссылки"
```

Определить символическую ссылку можно следующим образом:

- Вызвав команду ls

```
ls -l symbolic_link

lrwxrwxrwx. 1 user group 4 Aug  6 22:39 symbolic_link -> link_target
```

- Вызвав команду stat

```
stat symbolic_link
  File: 'symbolic_link' -> 'link_target'
  Size: 11                Blocks: 0   IO Block: 4096   symbolic link
Device: fd02h/64770d        Inode: 12321070    Links: 1
Access: (0777/lrwxrwxrwx)  Uid: ( 1000/user)   Gid: ( 1000/group)
Context: unconfined_u:object_r:user_home_t:s0
Access: 2014-08-06 22:39:10.376588576 +0400
Modify: 2014-08-06 22:39:08.568602905 +0400
Change: 2014-08-06 22:39:08.568602905 +0400
```

В поле имени файла, кроме самого имени так-же указывается путь на который указывает символическая ссылка.

2.2.4 Жёсткая ссылка

Жёсткой ссылкой является другое имя файла в пределах одного раздела. Жёсткая ссылка не может указывать на каталоги. На уровне файловой системы это выглядит как ещё одна запись ссылающаяся на объявленный ранее блок данных на диске.
Создать жёсткую ссылку можно командой:

```
ln "Значение ссылки" "Имя ссылки"
```

Если файл значения и жёсткой ссылки окажутся на разных разделах, или значение будет указывать на каталог, то команда создания завершиться ошибкой.
Удалить жёсткую ссылку можно командой:

```
rm "Имя ссылки"
```

Для демонстрации жёсткой ссылки создадим её на имеющуюся символьную ссылку командой «ln symbolic_link hard_link», а затем посмотрим на результат

- Вызвав команду ls

```
ls -l hard_link

lrwxrwxrwx. 2 user group 4 Aug  6 22:39 hard_link -> link_target
```

- Вызвав команду stat

```
stat hard_link

  File: 'hard_link' -> 'link_target'
  Size: 11                Blocks: 0   IO Block: 4096   symbolic link
Device: fd02h/64770d        Inode: 12321070    Links: 2
Access: (0777/lrwxrwxrwx)  Uid: ( 1000/user)   Gid: ( 1000/group)
Context: unconfined_u:object_r:user_home_t:s0
Access: 2014-08-06 23:12:15.183588378 +0400
Modify: 2014-08-06 22:39:08.568602905 +0400
Change: 2014-08-06 23:12:13.700600392 +0400
```

Отличие от вывода полученного для символической ссылки, для которой мы сделали жёсткую ссылку заключается только в поле Links. Если сейчас посмотреть вывод stat на файле символической ссылки, то значение Links там окажется тоже самое, так как жёсткая ссылка указывает на набор данных на диске и любое имя файла является жёсткой ссылкой.

```
stat symbolic_link

  File: 'symbolic_link' -> 'link_target'
  Size: 11                Blocks: 0   IO Block: 4096   symbolic link
Device: fd02h/64770d        Inode: 12321070    Links: 2
Access: (0777/lrwxrwxrwx)  Uid: ( 1000/user)   Gid: ( 1000/group)
Context: unconfined_u:object_r:user_home_t:s0
Access: 2014-08-06 23:12:15.183588378 +0400
Modify: 2014-08-06 22:39:08.568602905 +0400
Change: 2014-08-06 23:12:13.700600392 +0400
```

При создании жёсткой ссылке у файла изменилось значение «Change», т.к. жёсткая ссылка относится к метаинформации.

2.2.5 Символьный специальный файл

Файл является точкой доступа к символьным устройствам. В такие файлы можно писать и из них можно читать, но производить эти операции можно последовательно. Т.е.произвольного доступа к заданной области получить у такого файла нельзя. Создать файл символьного устройства можно командой:

```
mknod "имя файла" c MAJOR MINOR
```

где MAJOR MINOR – номера устройства.
Удалить файл символьного устройства можно командой:

```
rm "имя файла"
```

Определить символьного устройства можно следующим образом:

- Вызвав команду ls

```
ls -la /dev/null
crw-rw-rw-. 1 root root 1, 3 Aug  6 21:15 /dev/null
```

- Вызвав команду stat

```
stat /dev/null
  File: '/dev/null'
  Size: 0     Blocks: 0      IO Block: 4096    character special file
Device: 5h/5d        Inode: 1039        Links: 1     Device type: 1,3
Access: (0666/crw-rw-rw-)  Uid: (   0/   root)   Gid: (    0/   root)
Context: system_u:object_r:null_device_t:s0
Access: 2014-08-06 21:15:23.236853269 +0400
Modify: 2014-08-06 21:15:23.236853269 +0400
Change: 2014-08-06 21:15:23.236853269 +0400
```

2.2.6 Блочный специальный файл

Файл является точкой доступа к блочному устройству. В такой файл можно писать или читать из него, но эти операции можно производить только блоками определённого размера. Блочное устройство позволяет получить доступ к произвольной области.
Создать файл блочного устройства можно командой:

```
mknod "имя файла" b MAJOR MINOR
```

где MAJOR MINOR – номера устройства.
Удалить файл блочного устройства можно командой:

```
rm "имя файла"
```

Определить блочного устройства можно следующим образом:

- Вызвав команду ls

```
ls -la /dev/sda
brw-rw----. 1 root disk 8, 0 Aug  6 21:15 /dev/sda
```

- Вызвав команду stat

```
stat /dev/sda
  File: '/dev/sda'
  Size: 0      Blocks: 0       IO Block: 4096      block special file
Device: 5h/5d        Inode: 1182        Links: 1     Device type: 8,0
Access: (0660/brw-rw----)  Uid: (   0/   root)   Gid: (    6/   disk)
Context: system_u:object_r:fixed_disk_device_t:s0
Access: 2014-08-06 21:15:32.733734541 +0400
Modify: 2014-08-06 21:15:25.572824065 +0400
Change: 2014-08-06 21:15:25.572824065 +0400
```

2.2.7 Файл-очередь

Файл очередь позволяет передавать данные от одного процесса другому, однако такая передача может быть только в одном направлении. Такой файл работает как pipe созданный добавлением символа «|» между двумя командами. Следующие команды дают один результат:

```
ls | cat
```

и второй вариант с использованием файла очереди «/tmp/pipe»
Консоль 1

```
ls > /tmp/pipe
```

Консоль 2

```
cat /tmp/pipe
```

Создать файл очереди можно командой:

```
mkfifo "имя файла"
```

Удалить файл очереди можно командой:

```
rm "имя файла"
```

Определить очереди можно следующим образом:

- Вызвав команду ls

```
ls -la /tmp/pipe
prw-rw-r--. 1 izhdanov izhdanov 0 Aug  6 21:20 /tmp/pipe
```

- Вызвав команду stat

```
stat /tmp/pipe
  File: '/tmp/pipe'
  Size: 0               Blocks: 0          IO Block: 4096   fifo
Device: 20h/32d         Inode: 29050       Links: 1
Access: (0664/prw-rw-r--)  Uid: ( 1000/  user)   Gid: ( 1000/  group)
Context: unconfined_u:object_r:user_tmp_t:s0
Access: 2014-08-06 21:20:06.127456412 +0400
Modify: 2014-08-06 21:20:06.127456412 +0400
Change: 2014-08-06 21:20:06.127456412 +0400
```

В качестве примера реального применения файла очереди, можно привести задачу сбора информации о сетевом траффике с удалённой машины в реальном времени. Для такой задачи нужно запустить команду tcpdump с правами root, для подключения к удалённой машине мы имеем данные пользователя без прав рута, но с возможностью их повышения. Просмотреть данные трафика можно в удобной форме

через программу wireshark на локальной машине. Можно на удалённой машине было собрать данные в файл, потом скопировать этот файл на локальную машину, а затем открыть его в wireshark и изучить, но в таком случае мы не можем изучать данные в реальном времени. Для выполнения последнего условия задачи, можно использовать файл очередь. Ниже приводится пример команд, подробнее их рассмотрим позже.
Консоль 1

```
ssh user@srv-name
sudo -s
mkfifo -m 0666 /tmp/dump-pipe
tcpdump -i eth0 -s0 -U -n -w /tmp/dump-pipe 'not port 22'
```

Консоль 2

```
ssh user@srv-name 'cat /tmp/dump-pipe' | wireshark -k -i -
```

2.2.8 Сокет

Файл сокета в Linux работает как сетевой сокет, но в пределах одного хоста и позволяет передавать данные между процессами не используя сетевую подсистему. Это позволяет ограничить доступ к данным внутреннего сервера, который мы не хотим выставлять в открытый доступ по различным причинам и уменьшает количество одновременно открытых подключений. Для создания и удаления сокетов существует специальный API, можно использовать стандартные функции создания сетевых сокетов или специальные для работы с локальными сокетами.
Определить сокеты можно следующим образом:

- Вызвав команду ls

  ```
  ls -la /tmp/.X11-unix/X0
  srwxrwxrwx. 1 root root 0 Aug  6 21:15 /tmp/.X11-unix/X0
  ```

- Вызвав команду stat

  ```
  stat /tmp/.X11-unix/X0
    File: '/tmp/.X11-unix/X0'
    Size: 0                 Blocks: 0          IO Block: 4096    socket
  Device: 20h/32d        Inode: 21626       Links: 1
  Access: (0777/srwxrwxrwx)  Uid: (   0/    root)   Gid: (  0/    root)
  Context: system_u:object_r:xdm_tmp_t:s0
  Access: 2014-08-06 21:16:03.560511572 +0400
  Modify: 2014-08-06 21:15:47.671582693 +0400
  Change: 2014-08-06 21:15:47.671582693 +0400
  ```

2.3 Описание некоторых файлов из каталога /dev

Каталог /dev содержит специальные файлы устройств, которые ссылаются на физические или виртуальные устройства, некоторые файлы могут пригодится при проведении тестов или других работ в ОС Linux, поэтому будут рассмотрены.
Если необходимо получить информацию у других каталогах в используемом дистрибутиве, то такая информация доступна через команду:

```
man hier
```

2.3.1 /dev/null, /dev/zero

Устройства /dev/null и /dev/zero – символьные виртуальные устройства.
/dev/null – устройство в которое можно производить запись без ограничений. Обычно это устройство используется в качестве конечной точки для перенаправления ненужного потока данных.
Например:
Измерение скорости чтения с диска

```
dd if=test-file of=/dev/null
```

Отключение вывода ошибок

```
test_util 2>/dev/null
```

При попытке чтения из **/dev/null** система сразу вернёт результат аналогичный концу файла.
/dev/zero – устройство в которое тоже можно неограниченно писать, но из него также можно читать без ограничений. При чтении из этого устройства система всегда будет генерировать поток байт с кодом «0x00». Устройство используется если нужно получить поток любых данных.
Например:
Очистка диска

```
dd if=/dev/zero of=/dev/sdb
```

Измерение скорости записи в файл

```
dd if=/dev/zero of=test-file bs=512 count=1MB
```

2.3.2 /dev/random, /dev/urandom

Устройства /dev/random и /dev/urandom – символьные устройства генераторов псевдослучайных чисел. При чтении из этих устройств система выдаёт поток псевдослучайных байт. Различие между /dev/random и /dev/urandom в том, что для генерации псевдослучайных данных система собирает информацию об активности пользователя и на её основе генерирует данные и когда информации получаемой от пользователя не хватает для генерации, то /dev/random останавливает поток выдачи

данных, а /dev/urandom продолжает выдавать данные, но с меньшим коэффициентом случайности.
Эти устройства обычно используют для инициализации последовательности шифров. Их можно использовать для заполнения файлов случайными данными, но нужно учитывать, что даже /dev/urandom, который не блокируется, является очень медленным устройством (современные диски могут писать быстрее чем /dev/urandom генерирует данные).

2.3.3 /dev/ttyS?, /dev/ttyUSB?

Устройства /dev/ttyS? и /dev/ttyUSB? (где ? – число) – символьные устройства доступа к com портам.
/dev/ttyS? – устройства доступа к обычным com портам.
/dev/ttyUSB? – устройства доступа к com портам через USB переходник.
Во многих дистрибутивах у этих устройств есть одна группа доступа, которая имеет права на запись в эти устройства. Если в эту группу добавить своего пользователя, то при следующем логине пользователь сможет подключаться к этим портам через программы для работы с ними без повышения привилегий.

2.3.4 /dev/hd?, /dev/sd?

Устройства /dev/hd? и /dev/sd? – блочные устройства доступа к жёстким дискам.
/dev/hd? – в старых дистрибутивах использовался для обозначения IDE дисков.
/dev/sd? – используется сейчас для обозначения накопителей информации(жёсткие диски, флешки).

2.3.5 /dev/md?, /dev/dm?

Устройства /dev/md? и /dev/dm? – блочные устройства, которые используются для создания программных RAID и/или шифрования.
/dev/md? – файл создаваемый подсистемой Linux Software Raid.
/dev/dm? – файл создаваемый подсистемой Device Mapper.
Оба устройства являются виртуальными блочными устройствами, которые в зависимости от параметров хранят данные на других устройствах.

2.3.6 /dev/loop?

Файлы /dev/loop? – блочные файлы устройств, которые позволяют работать с обычными файлами на файловой системе как с блочным устройством. Эта возможность позволяет монтировать образы дисков с известными файловыми системами, подготавливать такие образы, просто копируя файлы на подключённый раздел. Для подключения образа диска можно использовать следующие команды:

```
mount -oloop image-file mount-point
```

Если нужно подключить образ нестандартным способом, то существует утилита управления loop устройствами.
Подключение образа к устройству:

```
losetup /dev/loop0 image-file
```

после этого с устройством /dev/loop0 можно работать как с обычным блочным устройством, а все данные будут сохранены в файле image-file.
У команды losetup есть дополнительные опции

- `-o` – смещение от начала файла, после которого он будет доступен как блочное устройство
- `--sizelimit` – разер области в файле, которая будет блочным устройством.

2.3.7 /dev/ram?

Файл /dev/ram? – блочное устройство диска в оперативной памяти.
Параметры устройства задаются при загрузке ядра или через параметры загрузки модуля. В новых дистрибутивах обычно используется второй вариант.

```
modprobe rd rd_nr=1 rd_size=102400 max_part=5
```

- `rd_nr` – Количество устройств (рамдисков).
- `rd_size` – Размер каждого устройства в килобайтах.
- `max_part` – Максимальное количество разделов на рамдиске.

После инициализации рамдиска с ним можно работать как с любым диском, но данные после выгрузки драйвера пропадут. В некоторых версиях драйвера рамдиска были ошибки из-за которых данные на рамдиске стирались в процессе работы. Такие проблемы могут стать причиной падения тестов нашего драйвера, хотя ошибка не в нём.

2.4 Примеры работы с файлами

Ниже представлены краткие примеры работы с файлами, как можно прочитать или изменить данные на системах под управлением Linux.

2.4.1 less more

Эти команды сделаны для постраничного вывода содержимого текстового файла.
Команда `more` позволяет постранично просмотреть файл и при этом поддерживает команды управления терминалом (позволяет отображать выделение цветом).

```
more /var/log/boot.log

[  OK  ] Started Reload Configuration from the Real Root.
[  OK  ] Reached target Initrd File Systems.
[  OK  ] Reached target Initrd Default Target.

Welcome to RFRemix 20 (Heisenbug)!
```

Команда `less` позволяет просматривать файл с построчной навигацией, но обычно игнорирует управляющие команды терминала и текст может выглядеть по другому:

```
less /var/log/boot.log

[ESC[32m  OK  ESC[0m] Started Reload Configuration from the Real Root.
[ESC[32m  OK  ESC[0m] Reached target Initrd File Systems.
[ESC[32m  OK  ESC[0m] Reached target Initrd Default Target.

Welcome to ESC[0;34mRFRemix 20 (Heisenbug)ESC[0m!
```

Обе команды поддерживают поиск по шаблону и при работе с большими файлами загружают из файла только отображаемую часть, поэтому их можно использовать для чтения больших файлов логов.

2.4.2 cat tac

Команды `cat` и `tac` позволяют вывести на стандартный вывод содержимое файлов имена которых были переданы в качестве параметров.
При выводе содержимое файлов выводится одно за другим в порядке передачи имён файлов при запуске команды. Отличие команды `tac` от `cat` в том, что `tac` выводит содержимое файла построчно начиная с конца.

```
cat /var/log/boot.log

[  OK  ] Started Reload Configuration from the Real Root.
[  OK  ] Reached target Initrd File Systems.
[  OK  ] Reached target Initrd Default Target.

Welcome to RFRemix 20 (Heisenbug)!

tac /var/log/boot.log

Welcome to RFRemix 20 (Heisenbug)!

[  OK  ] Reached target Initrd Default Target.
[  OK  ] Reached target Initrd File Systems.
[  OK  ] Started Reload Configuration from the Real Root.
```

2.4.3 head tail

Команды `head` и `tail` позволяют вывести часть содержимого файла.

- `head` – выводит часть начиная с начала файла.
- `tail` – выводит часть начиная с конца файла.

Ещё одна особенность команды `tail` – возможность следить за дополнением файла и выводить добавленное содержимое. Для этого у этой команды есть соответствующие опции,:

- `-f` – Следить за изменение файла и выводить добавленное содержимое.
- `-F` – Следить за изменение файла и выводить добавленное содержимое, а также отслеживать удаление и новое создание файла и продолжать работу после появления файла.

2.4.4 hexdump

Команда `hexdump` позволяет вывести содержимое файла в шестнадцатеричных кодах. Это бывает полезно при разборе проблем работы драйвера.

```
hexdump -vC /var/log/boot.log
```

```
000 5b 1b 5b 33 32 6d 20 20 4f 4b 20 20 1b 5b 30 6d |[.[32m  OK  .[0m|
010 5d 20 53 74 61 72 74 65 64 20 53 68 6f 77 20 50 |] Started Show P|
020 6c 79 6d 6f 75 74 68 20 42 6f 6f 74 20 53 63 72 |lymouth Boot Scr|
030 65 65 6e 2e 0d 0a 5b 1b 5b 33 32 6d 20 20 4f 4b |een...[.[32m  OK|
040 20 20 1b 5b 30 6d 5d 20 52 65 61 63 68 65 64 20 |  .[0m] Reached |
050 74 61 72 67 65 74 20 50 61 74 68 73 2e 0d 0a 5b |target Paths...[|
```

2.4.5 dd

Команда `dd` позволяет копировать блоки данных из одного файла/потока в другой, с возможностью указания размера копируемого и его смещения. Опции команды dd:

- `if` – файл ввода, если не указан, то данные будут браться с стандартного ввода.
- `of` – файл вывода, если не указан, то данные будут выводиться на стандартный вывод.
- `bs` – размер блока передачи.
- `count` – количество передаваемых блоков.
- `skip` – количество блоков, которые надо пропустить при чтении.
- `seek` – количество блоков, которые надо пропустить при записи.

2.4.6 touch

Команда `touch` позволяет обновить времена файла. Без параметров `touch file-name` обновляет время последнего доступа и изменения у файла, если файла не существует, то он будет создан.
Также команда позволяет задать произвольное время для файла.

2.4.7 sed

Команда `sed` – потоковый редактор, который на основе имеющегося набора команд и регулярных выражений позволяет изменять входящий символьный поток.
Этот раздел будет потом дополнен.

2.4.8 vi

Команда `vi` – текстовый редактор с большим функционалом, но не интуитивно понятным интерфейсом. Обычно присутствует на многих платформах.

- <esc>:q!<enter> – Выход без сохранения.
- <esc>:x!<enter> – Выход с сохранением.

Этот раздел будет потом дополнен.

2.5 Точки монтирования

Так как в Linux практически все проецируется на файлы, то различные тома подключаются как каталоги. Общая структура файловых систем начинается с корня, который обозначается «/». На любой каталог, включая корневой можно примонтировать другую файловую систему.

2.5.1 Команда mount

Если запустить команду `mount`, то она покажет список разделов замонтированных в системе. Для этого команда `mount` использует файл `/etc/mtab`, который должен быть доступен на запись. В современных дистрибутивах вместо него делают символическую ссылку на файл `/proc/mounts`, тогда корень может быть доступен только для чтения.
Если `mount` использовать с параметрами, то она позволит примонтировать новый раздел.

```
mount -t fsname -o fsoptions device_name mount_point
```

где:

- fsname – имя драйвера файловой системы (vfat, ufsd, ntfs-3g)
- fsoptions – опции монтирования файловой системы (noatime)

- device_name – имя устройства на котором хранится файловая система.
- mount_point – каталог в который будет подключена файловая система.

Подробную информацию об опциях файловых систем можно узнать из документации к конкретной файловой системе. Однако можно выделить несколько опций, которые стоит описать отдельно.

- loop – монтирование файловой системы из файла с использованием loop устройства. (2.3.6)
- remount – изменение опций монтирования без отключения текущего раздела.

  ```
  mount -orw,remount /
  ```

  переключение корня в режим записи.
- bind, rbind – отдельная опция позволяющая подключить один каталог поверх другого. В отличии от символической ссылке в этом случае не создаётся нового файла на файловой системе, а производится наложение содержимого на уровне ядра. Отличие опций bind от rbind в том, что bind подключает только одну файловую систему из начального каталога в новую точку монтирования, а rbind подключает все файловые системы замонтированные внутрь начального каталога.

  ```
  mount --bind /etc /tmp/etc
  ```

  После подключения одного каталога в другой, на него можно задать дополнительные параметры, через опцию remount.

  ```
  mount -oro,remount /tmp/etc
  ```

  Переводит новую точку доступа к etc в режим только чтения.

2.5.2 Команда lsof

Иногда при попытке размонтировать файловую систему, ОС сообщает что файловая система занята. Что-бы проверить кто в данный момент использует данный каталог существует команда `lsof`.

```
lsof +d /mnt
```

выводит список программ использующих в данный момент`/mnt`

2.6 Пространства имён

В ядре линукс существует механизм для дополнительной изоляции приложений запущенных в системе, который называется пространство имён. Если посмотреть файл `/proc/mounts`, то можно увидеть список замонтированных разделов. Однако если посмотреть в каталог `/proc/num/`, где num – номер, являющийся PID, то можно увидеть наличия другого файла `mounts`. В обычном режиме работы, этот файл содержит жёсткую ссылку на файл `/proc/mounts` и все приложения видят один на всю систему список разделов.
Механизм пространств имён позволяет создать для каждого запущенного приложения свой список замонтированных разделов.
При запуске приложения с использованием специального API ядра, можно вывести это приложение в отдельное пространство имён. Тогда для такого приложения файл `mounts` будет не жёсткой ссылкой на `/proc/mounts`, а копией этого файла и если изменить такой файл, то это не повлияет на всю систему.
Этот механизм активно используется в Android начиная с версии 4.2. Для каждого запущенного приложения создаётся своё пространство имён, потом в нём размонтируются почти все разделы и перемонтируются новые в соответствии с запрошенными через манифест правами.
Из-за этого может быть следующая проблема: программа запущенная с правами рута, но через приложение установленное как apk, попадает в отдельное пространство имён, потом вызывает монтирование раздела через драйвер FUSE. Замонтированный раздел виден в этой программе и во всех программах запущенных из неё, но не виден в системе т.к. он находится в отдельном пространстве имён и не влияет на остальные приложения.

3 Управление пользователями и правами доступа

Линукс разрабатывалась как копия UNIX, которая изначально была многопользовательской системой. Т.е. система позволяет работать в ней нескольким пользователям и предоставляет возможность разграничить права что-бы пользователи друг другу не мешали.

3.1 Работа с локальными пользователями

ОС Linux позволяет хранить информацию о пользователях локально на компьютере где она запущена, а так-же получать информацию о пользователях с центрального сервера. Сейчас рассмотрим работу с локальными пользователями.

3.1.1 Добавление локальных пользователей

Добавление локальных пользователей в Linux осуществляется командой `useradd [options] LOGIN`, где options – параметры команды, а LOGIN – логин нового пользователя.
Параметры могут быть следующими:

- `-g` – основная группа пользователя. Если параметр не задан, то по умолчанию будет создана группа с именем пользователя и пользователь будет в неё добавлен.
- `-N` – не создавать группу с именем пользователя.
- `-G` – другие группы в которые будет входить пользователь. Перечисляются через запятую.
- `-u` – задать ID пользователя.|
- `-o` – разрешить создавать пользователя с повторяющимся ID.
- `-s` – указать оболочку по умолчанию.
- `-r` – создать системного пользователя.
- `-b` – задать каталог в котором будет создан домашний каталог, на основе имени пользователя.
- `-c` – комментарий к пользователю.
- `-e` – дата блокировки аккаунта.
- `-f` – дата устаревания пароля.
- `-k` – каталог шаблонов для домашнего каталога пользователя.
- `-m` – создать домашний каталог пользователя.

- `-M` – не создавать домашний каталог пользователя.
- `-p` – задать пароль пользователя.
- `-U` – создать группу пользователя по умолчанию.
- `-Z` – задать пользователя SELinux.

При работе эта команда правит файлы `/etc/passwd`, `/etc/shadow`, `/etc/group`, `/etc/gshadow` и должна запускаться от пользователя с правами на запись в них (например от root), и с корневым разделом доступным на запись.

3.1.2 Изменение локальный пользователей

Для изменения локальных пользователей используется команда `usermod [options] LOGIN`, где options – параметры команды, а LOGIN – логин изменяемого пользователя.
Её параметры в большей степени повторяют параметры `useradd`, но есть отличия:

- `-a` – добавить пользователя в дополнительную группу. Используется вместе с `-G`, без неё дополнительные группы пользователя будут заменены.
- `-m` – перемещает домашний каталог пользователя в новый, указанный через `-d`.
- `-L` – заблокировать пароль пользователя.
- `-U` – разблокировать пароль пользователя.

Эта команда требует доступ к файлам аналогичный `useradd`. Дополнительное условие, для работы этой команды — отсутствие процессов запущенных от имени изменяемого пользователя.

3.1.3 Удаление локального пользователя

Для удаления локального пользователя существует команда `userdel [options] LOGIN`, где options – параметры команды, а LOGIN – логин удаляемого пользователя.
У неё есть следующие параметры:

- `-f` – Удаляет пользователя даже если он сейчас работает в системе. При удалении пользователя удаляется его домашний каталог.
- `-r` – Удалить домашний каталог пользователя вместе с ним.

3.2 Работа с локальными группами

Главная особенность локальных групп Linux, они могут содержать только пользователей. Локальные группы не могут входить в другие локальные группы. Если нужно поддерживать более сложную структуру групп для пользователей, то обычно используют специальные сервисы каталогов (см. **??**).

3.2.1 Добавление локальной группы

Добавление локальной группы производится командой `groupadd [options] GROUP` где options – параметры команды, а GROUP – имя новой группы.
Параметры могут быть следующими:

- `-f` – команда всегда выполняется успешно если группа существует.
- `-g` – задать ID группы.
- `-o` – игнорировать дублирования ID группы.
- `-r` – создать системную группу.

3.2.2 Изменение локальной группы

Изменение локальной группы производится командой `groupmod [options] GROUP` где options – параметры команды, а GROUP – имя изменяемой группы.
Параметры могут быть следующими:

- `-n` – новое имя группы.
- `-g` – задать ID группы.
- `-o` – игнорировать дублирования ID группы.

3.2.3 Удаление локальной группы

Удаление групп производится командой `groupdel GROUP`, где GROUP – имя удаляемой группы.

3.2.4 Управление паролями

Для управления паролями в Linux существует команда `passwd`.
Если её запустить без параметров, то она предложит сменить пароль текущего пользователя.
Если команду запускать от пользователя `root`, то можно указать какому пользователю будет изменён пароль и текущий пароль запрошен не будет.
При вводе пароля он **не отображается** в консоли.

3.3 Права доступа к файлам

В данном разделе будут рассмотрены права доступа к файлам в Linux и инструменты работы с ними. В Linux существует несколько методов определения прав, сейчас мы будем рассматривать POSIX права, которые есть во всех UNIX подобных системах.

3.3.1 Описание POSIX прав

POSIX права обычно обозначаются в виде прав на чтение, запись, выполнение для владельца, группы и остальных. Такие права можно увидеть в свойствах файла в виде набора цифр `0755` или символов `-rwxr-xr-x`. Разберём оба варианта подробнее.

Цифорвая запись прав. В такой записи, каждая цифра обозначает группу суммы прав:
$0_1 7_2 5_3 5_4$

1. Расширенные опции прав:
   - 4 – Set User ID – если выставлен на исполняемом файле, то файл будет запускаться от имени пользователя владельца файла.
   - 2 – Set Group ID – если выставлен на исполняемом файле, то файл будет запускаться от группы — группы файла.
   - 1 – Sticky bit – закрепляющий бит, если установлен на каталоге, то файлы внутри такого каталога могут удалить или пользователи владельцы файла или пользователи владельцы каталога.
2. Права пользователя владельца:
   - 4 – Может читать.
   - 2 – Может писать.
   - 1 – Может исполнять файл или перечислять содержимое каталога.
3. Права группы:
   - 4 – Может читать.
   - 2 – Может писать.
   - 1 – Может исполнять файл или перечислять содержимое каталога.
4. Права остальных:
   - 4 – Может читать.
   - 2 – Может писать.
   - 1 – Может исполнять файл или перечислять содержимое каталога.

В данном примере получается что заданы следующие права: Расширенных прав нет, владелец может все ($4 + 2 + 1 = 7$), группа ($4 + 1 = 5$)
и остальные ($4 + 1 = 5$) могут читать и выполнять.

Символьная запись прав В такой форме записи, каждая буква означает какое-то право доступа к файлу:
-rwx$_1$r-x$_2$r-x$_3$

1. Права владельца
   - - – не задано.
   - r – Чтение.
   - w – Запись.
   - x – Выполнение/перечисление каталога.
   - S – Set User ID бит.
   - s – Set User ID бит + выполнение/перечисление каталога.
2. Права группы
   - - – не задано.
   - r – Чтение.
   - w – Запись.
   - x – Выполнение/перечисление каталога.
   - S – Set User ID бит.
   - s – Set User ID бит + выполнение/перечисление каталога.
3. Права остальных
   - - – не задано.
   - r – Чтение.
   - w – Запись.
   - x – Выполнение/перечисление каталога.
   - T – Sticky bit.
   - t – Sticky bit + выполнение/перечисление каталога.

3.3.2 Управление POSIX правами

Для изменения POSIX прав существуют утилиты: `chmod` и `chown`. Первая утилита позволяет настроить права доступа, вторая изменить пользователя и группу файла. Утилита `chmod` может быть запущена следующим образом: `chmod 0755 file_name`, где 0755 – права в цифровой форме, file_name – имя файла, для которого меняются права.
У `chmod` существуют дополнительные параметры, для указания прав доступа через

символьную форму, но они могут не работать в различных реализациях данной утилиты. Цифровая форма существует во всех версиях.
Для изменения прав доступа, необходимо быть пользователем файла или пользователем `root`.
Для изменения пользователя или группы файла используется команда:
`chown user:group file_name`
где user – новый пользователь файла, group – новая группа файла, file_name – изменяемый файл. Изменить пользователя файла может только `root`, изменить группу файла может или `root`, или пользователь файла, но только на группу в которую он входит.

3.4 Утилита sudo

Так как в ОС Linux существует разделение пользователей, то обычный пользователь часто не имеет прав на выполнение различных операций. Что-бы обойти эту проблему существуют утилиты для повышения привилегий пользователя. К таким утилитам относятся `su` и `sudo`. Утилита `su` позволяет сменить пользователя, на другого зная его пароль, или запустить от его имени определённую команду. Но всегда вводить пароль для выполнения нескольких команд не удобно и раздавать пароль от пользователя `root` не безопасно поэтому была сделана утилита `sudo`, которая позволяет настроить правила проверки пароля и упростить выполнение программ от имени другого пользователя.

3.4.1 Использование утилиты sudo

Команда `sudo` имеет много различных параметров, о которых можно прочитать в документации к ней `man sudo`. Но в этом документе будут перечислены основные варианты её использования:

- `sudo application_name` – запуск программы `application_name` от имени пользователя `root`
- `sudo -u user application_name` – запуск программы `application_name` от имени пользователя `user`
- `sudo -s` – перейти в сеанс пользователя `root`
- `sudo -u user -s` – перейти в сеанс пользователя `user`

Если `sudo` настроена на выполнение команды с запросом пароля, то будет запрошен пароль пользователя который запускает команду`sudo`, а не пользователя на которого нужно переключиться.

3.4.2 Настройка утилиты sudo

Утилита `sudo` позволяет настроить, в каких случаях нужно запросить пароль перед сменой пользователя, в каких случаях его можно не запрашивать или просто запретить выполнение команды от другого пользователя.
За конфигурацию утилиты `sudo`, отвечает файл `/etc/sudoers`. В новых дистрибутивах используют схему с подключаемыми конфигурационными файлами, которые лежат в `/etc/sudoers.d/`.
Основной синтаксис правил конфигурации выглядит так:

```
user MACHINE=COMMANDS
```

- `user` – пользователь или группа к которой будет применено правило.
- `MACHINE` – хост к которому относится правило.
- `CAMMANDS` – команды к которым относится правило.

Группа от пользователя в конфигурационном файле отличается тем, что перед её названием стоит символ «%». Если перед списком команд указать «`NOPASSWD:`», то пароль запрашиваться не будет.
Примеры конфигураций:

```
%wheel ALL=(ALL) ALL
```

пользователи входящие в группу «`wheel`», могут запускать все от имени `root`.

```
%teamcity_cfg ALL=NOPASSWD:/sbin/service testAgents *
```

пользователи входящие в группу «`teamcity_cfg`», могут управлять тестовыми агентами без запроса пароля.
Информацию о дополнительных параметрах файла конфигурации команды `sudo`, можно прочитать в документации `man sudoers`.
Владельцем всех конфигурационных файлов для `sudo`, должен быть `root` и изменять их должен тоже только он, иначе команда `sudo` работать не будет.

3.5 SELinux

SELinux – технология мандатных прав доступа. Сейчас про неё будет сказано только как проверить её наличие и как отключить её. Позже будет добавлена информация о том, как правильно с ней работать.

3.5.1 Проверка

Проверить состояние SELinux можно несколькими способами:

- Командой `cat /selinux/enforce`
- Командой `selinuxenabled && echo "On"`

3.5.2 Отключение

Отключить SELinux можно отредактировав файл `/etc/selinux/config`, заменив значение `SELINUX=` на `SELINUX=disabled` и перезагрузив машину.