ТЕМА 5.3. МЕХАНИЗМЫ БЕЗОПАСНОСТИ В ОПЕРАЦИОННЫХ СИСТЕМАХ СЕМЕЙСТВА LINUX

В данной теме рассматриваются следующие вопросы:

- Типы файлов в ОС Linux.
- Владельцы файлов.
- Управление правами доступа в файловой системе.
- Атрибуты файлов.
- Управление свойствами файлов.

Лекции – 2 часа, лабораторные занятия – 2 часа, самостоятельная работа – 2 часа.

Экзаменационные вопросы по теме:

- Механизмы безопасности в операционных системах семейства Linux.

5.3.1. Типы файлов в ОС Linux

В GNU/Linux как и других Unix-подобных операционных системах понятие типа файла не связано с расширением файла (несколькими буквами после точки в конце имени), как это обстоит в Windows. Поэтому тип файла в Linux – это скорее тип объекта, но не тип данных как в Windows [1].

В операционной системе GNU/Linux существуют следующие типы файлов:

- обычные файлы
- каталог
- символьные ссылки
- блочные устройства
- символьные устройства
- сокеты
- каналы

Каждый тип имеет собственное обозначение одним символом.

Обычные файлы (-)

Сюда относятся все файлы с данными, играющими роль ценной информации сами по себе.

Linux не различает текстовые файлы, исполняемые или картинки.

В любом случае это будет обычный (regular) файл. Все они обозначаются знаком "-".

Остальные типы файлов считаются специальными (special).

Каталоги (d)

В Linux каталог представляет собой такой тип файла, данными которого является список имен других файлов и каталогов, вложенных в данный каталог. Напрямую, то есть через какой-либо редактор, пользователь не может редактировать данные файла-каталога. Редактированием занимается ядро операционной системы, получая, в том числе от пользователя, команды создания файла, удаления и др.

В файле каталога осуществляется связь между именами файлов (словесного обозначения для людей) и их индексными дескрипторами (истинным именем-числом, которым оперирует ОС).

В Unix-подобных системах один и тот же файл может существовать под разными именами и/или в разных каталогах: все имена будут связаны с одним и тем же индексным дескриптором (механизм жестких ссылок).

Также следует, что файлы всегда должны содержаться в каталогах, иначе станут недоступны, так как нигде не будет содержаться записи о них.

Символьные ссылки (l)

Символьная ссылка – это файл, в данных которого содержится адрес другого файла по его имени (а не индексному дескриптору).

Выполнение символьной ссылки приводит к открытию файла, на который она указывает. Это аналог ярлыков в операционной системе Windows.

Если удалить исходный файл, то символьная ссылка продолжит существовать. Она по-прежнему будет указывать на файл, которого уже нет.

Символьные ссылки не содержат атрибутов файлов, на которые они указывают. У них есть собственные атрибуты (свое время создания, размер, права доступа).

Ссылки на файлы и директории

В системе Linux есть два типа ссылок:

Символьные ссылки (symbolic links, или soft links) указывают путь к другому файлу. Если удалить файл, на который указывает ссылка (называемая target), ссылка все равно будет существовать, но перестанет «работать», поскольку теперь она указывает на несуществующий файл. Может указывать на файл в другой файловой системе.

Жесткие ссылки (hard links) указываюn на то же место (inode) на диске, где находится исходный файл. После создания жесткой ссылки она становится неотличимой от оригинального имени файла, то есть как бы становится дополнительным именем файла. После удаления исходного файла жесткая ссылка продолжает существовать. Файл удаляется, когда количество жестких ссылок на него становится равным нулю. Должны указывать на файл в этой же файловой системе.

Команда **ln** по умолчанию создает жесткие ссылки

ln [OPTION]... [-T] TARGET LINK_NAME

Вывод **ls -l** показывает количество жестких ссылок на файл. Опция **-i** отображает номер inode, где находится файл/

Специальные директории `.` и `..` являются жесткими ссылками

В примере на рис. 5.3.1 специальная директория `.` в директории `docs/subfolder` имеет две ссылки (сама на себя и запись в родительской директории `docs`). Специальная директория `..` имеет три ссылки (сама на себя, запись в родительской директории `~` и ссылка `..` в поддиректории `subfolder`).

```
[me@nnk4 ~]$ tree docs
docs
└── subfolder

1 directory, 0 files
[me@nnk4 ~]$ cd docs/subfolder/
[me@nnk4 subfolder]$ ls -la
total 0
drwxrwxr-x. 2 me me  6 Mar 27 09:31 .
drwxrwxr-x. 3 me me 23 Mar 27 09:47 ..
```

Рис. 5.3.1. Специальные директории . и ..

Команда **ln** с опцией **-s** используется для создания символических ссылок

В выводе команды **ls -l** символические ссылки обозначаются буквой **l** в блоке разрешений, а после имени файла ссылки можно увидеть и целевой файл, на который указывает ссылка (рис. 5.3.2).

```
[me@nnk5 docs]$ echo Hello everybody > folder1/hello.txt
[me@nnk5 docs]$ ln -s folder1/hello.txt rellink.txt
[me@nnk5 docs]$ ln -s /home/me/docs/folder1/hello.txt abslink.txt
[me@nnk5 docs]$ ls -l
total 0
lrwxrwxrwx. 1 me me 31 Mar 27 21:33 abslink.txt -> /home/me/docs/folder1/hello.t
xt
lrwxrwxrwx. 1 me me 17 Mar 27 21:33 rellink.txt -> folder1/hello.txt
drwxrwxr-x. 2 me me  6 Mar 27 09:31 subfolder
[me@nnk5 docs]$ cat abslink.txt
Hello everybody
[me@nnk5 docs]$ cat rellink.txt
Hello everybody
```

Рис. 5.3.2. Символьные ссылки

Если путь к целевому файлу не указан полностью, его местоположение интерпретируется как относительное местоположение ссылки. Это может создать проблемы при перемещении ссылки или файла, на который она указывает.

Символьные (c) и блочные устройства (b)

Файлы устройств предназначены для обращения к аппаратному обеспечению компьютера (дискам, принтерам, терминалам и др.). Когда происходит обращение к файлу устройства, то ядро операционной системы передает запрос драйверу этого устройства.

К символьным устройствам обращение происходит последовательно (символ за символом). Примером символьного устройства может служить терминал.

Считывать и записывать информацию на блочные устройства можно в произвольном порядке, причем блоками определенного размера. Пример: жесткий диск.

Сокеты (s) и каналы (p)

Чтобы понять, что такое канал и сокет, необходимо понимание процессов в операционной системе. И каналы, и сокеты организуют их взаимодействие. Пользователь с данными типами файлов почти не сталкивается.

Ключевым отличием канала от сокета является то, что канал однонаправлен. Через канал один процесс всегда передает данные второму, но не наоборот. Сокеты позволяют передавать данные в разных направлениях, т. е. осуществляют связь.

Также следует отметить, что канал представлен в структуре каталогов файлом, только если он именован. Когда возникают безымянные каналы, то они существуют только внутри ядра Linux.

Команда file

Линукс не делает предположение о типе данных в обычном файле, но есть специальная утилита, которая выполняет эту задачу, – программа **`file`**. Для этого она анализирует начало содержимого файла и находит в нем специальные "сигналы", характерные для определенного типа – бинарного файла, текстового, изображения и др.

Временные файлы

Временные файлы — это файлы, используемые программами для хранения данных, которые нужны только в течение короткого времени. Это могут быть данные о запущенных процессах, журналы ошибок, промежуточные файлы, используемые при преобразовании данных, файлы кэша и т.д.

Filesystem Hierarchy Standard (FHS) определяет стандартные расположения временных файлов в системах Linux:

`/tmp`	Рекомендуется очистить этот каталог (все файлы будут удалены) во время загрузки системы
`/var/tmp`	Этот каталог не должен быть очищен во время загрузки системы, то есть файлы, хранящиеся здесь, обычно сохраняются между перезагрузками
`/run`	Этот каталог должен быть очищен во время загрузки системы, содержит данные, используемые запущенными процессами, программы могут создавать здесь подкаталоги

5.3.2. Владельцы файлов

В Linux у каждого файла и каждого каталога есть два владельца: пользователь и группа.

Эти владельцы устанавливаются при создании файла или каталога. Пользователь, который создаёт файл становится владельцем этого файла, а первичная группа, в которую входит этот же пользователь, так же становится владельцем этого файла. Чтобы

определить, есть ли у вас как у пользователя права доступа к файлу или каталогу, оболочка проверяет владение ими [2].

Это происходит в следующем порядке:

1. Оболочка проверяет, являетесь ли вы владельцем файла, к которому вы хотите получить доступ. Если вы являетесь этим владельцем, вы получаете разрешения и оболочка прекращает проверку.
2. Если вы не являетесь владельцем файла, оболочка проверит, являетесь ли вы участником группы, у которой есть разрешения на этот файл. Если вы являетесь участником этой группы, вы получаете доступ к файлу с разрешениями, которые для группы установлены, и оболочка прекратит проверку.
3. Если вы не являетесь ни пользователем, ни владельцем группы, вы получаете права других пользователей (Other).

Чтобы увидеть текущие назначения владельца, вы можете использовать команду **ls -l**. Эта команда показывает пользователя и группу-владельца.

Команда **chown** позволяет сменить владельца (команда **chgrp** – только группу-владельца). Имя пользователя-владельца и группы владельца разделяется двоеточием; если указывается только один компонент, двоеточие сохраняется (рис 5.3.3). Опция **–R** служит для рекурсивного изменения владельца во всех вложенных директориях.

```
[me@nnk4 docs]$ ls -l
total 300
-rw-rw-r--. 1 me me 139614 Mar 26 21:34 martin_eden.txt
-rw-rw-r--. 1 me me  34500 Mar 26 21:33 self.txt
-rw-rw-r--. 1 me me 123059 Mar 26 21:34 tom_sowyer.txt
[me@nnk4 docs]$ sudo chown jack:writers martin_eden.txt
[me@nnk4 docs]$ sudo chown mark:writers tom_sowyer.txt
[me@nnk4 docs]$ sudo chown :writers self.txt
[me@nnk4 docs]$ ls -l
total 300
-rw-rw-r--. 1 jack writers 139614 Mar 26 21:34 martin_eden.txt
-rw-rw-r--. 1 me   writers  34500 Mar 26 21:33 self.txt
-rw-rw-r--. 1 mark writers 123059 Mar 26 21:34 tom_sowyer.txt
```

Рис. 5.3.3. Изменение владельца файла

Владельцы по умолчанию

Когда пользователь создает файл, применяется владение по умолчанию.

Пользователь, который создает файл, автоматически становится владельцем этого файла, а основная группа этого пользователя автоматически становится владельцем этого файла. Обычно это группа, которая указана в файле `/etc/passwd` в качестве основной группы пользователя. Однако если пользователь является членом нескольких групп, он может изменить эффективную основную группу.

Чтобы показать текущую эффективную первичную группу, пользователь может использовать команду **groups**:

```
[root@server1 ~]# groups lisa
lisa : lisa account sales
```

Если текущий пользователь linda хочет изменить эффективную первичную группу, он будет использовать команду **newgrp**, за которой следует имя группы, которую он хочет установить в качестве новой эффективной первичной группы. После использования команды **newgrp** первичная группа будет активна, пока пользователь не введет команду **exit** или не выйдет из системы.

После изменения действующей основной группы все новые файлы, созданные пользователем, получат эту группу в качестве группы-владельца. Чтобы вернуться к исходной настройке первичной группы, используйте **exit**.

Чтобы иметь возможность использовать команду **newgrp**, пользователь должен быть членом той группы, которую он хочет использовать в качестве первичной. Кроме этого, групповой пароль может быть использован для группы с помощью команды **gpasswd**. Если пользователь использует команду **newgrp**, но не является членом целевой группы, оболочка запрашивает пароль группы. После того, как вы введете правильный групповой пароль, будет установлена новая эффективная первичная группа.

5.3.3. Управление правами доступа в файловой системе

Опция **-l** команды **ls** выводит информацию о файлах и директориях в длинном формате, включая разрешения и владельцев.

```
[me@nnk4 ~]$ ls -al
total 52
drwx------. 16 me   me    4096 Mar 26 20:36 .
drwxr-xr-x.  8 root root    83 Feb 14 20:32 ..
-rw-------.  1 me   me   11689 Feb 22 16:35 .bash_history
-rw-r--r--.  1 me   me      18 Apr  1  2020 .bash_logout
-rw-r--r--.  1 me   me     193 Apr  1  2020 .bash_profile
-rw-r--r--.  1 me   me     231 Apr  1  2020 .bashrc
drwxr-xr-x.  2 me   me       6 Jan 11 14:08 Desktop
drwxr-xr-x.  2 me   me       6 Jan 11 14:08 Documents
drwxr-xr-x.  2 me   me       6 Jan 11 14:08 Downloads
drwxrwxr-x.  9 me   me     106 Mar 26 20:36 edu
```

Рис. 5.3.4. Просмотр информации о файлах и директориях

При создании файла система сохраняет идентификатор владельца (user) и идентификатор основной группы владельца (user).

```
user@UBUNTU1:~/mydocs$ ls -la
drwxrwxr-x  3 user user  4096 Oct  8 17:18 .
drwxr-xr-x 16 user user  4096 Oct  8 17:18 ..
-rw-rw-r--  1 user user  2451 Oct  8 17:18 ant.txt
drwxrwxr-x  2 user user  4096 Oct  8 17:18 birds
-rw-rw-r--  1 user user 12306 Oct  8 14:45 hare.txt
-rw-rw-r--  1 user user  2351 Oct  8 14:33 sheep.txt
-rw-rw-r--  1 user user  2476 Oct  8 14:30 wolf.txt
```

Рис. 5.3.5. Здесь красным цветом отмечены разрешения для пользователя-владельца, зеленым – для группы-владельца, синим цветом - разрешения для всех остальных пользователей

Первая группа разрешений относится к пользователю-владельцу, вторая – к группе-владельцу, третья – ко всем остальным пользователям. Применяется наиболее точное разрешение.

Значение разрешений для файлов:

- **r** – содержимое файла может быть прочитано
- **w** – содержимое файла может быть изменено
- **x** – файл может быть выполнен как команда

Значение разрешений для каталогов:

- **r** – содержимое каталога (имена файлов) может быть перечислено
- **w** – любой файл в каталоге может быть создан или удален
- **x** – к содержимому каталога возможен доступ (в зависимости от разрешений на файлы)

Числовое обозначение разрешений: rw-rw-r-- = 664

```
---   0        r--   4
--x   1        r-x   5
-w-   2        rw-   6
-wx   3        rwx   7
```

Управление разрешениями

Команда **chmod** позволяет изменять разрешения двумя способами:

символьным

```
chmod a+x cat.txt
```

числовым

```
chmod 750 animals
```

Некоторые опции:

-R, --recursive

изменять файлы и директории рекурсивно

-v, --verbose

выводить диагностику о каждом обрабатываемом файле

-c, --changes

выводить диагностику только для изменяемых файлов

Символьный метод

```
chmod WhoWhatWhich file|directory
```

Who – это u, g, o, a (user, group, other, all)

What – это +, -, = (добавить, удалить, установить)

Which – это r, w, x (Read, Write, eXecute)

```
[me@nnk4 docs]$ ls -l
total 268
-rw-rw-r--. 1 me me   3728 Mar 26 21:43 check.sh
-rw-rw-r--. 1 me me 139614 Mar 26 21:42 martin_eden.txt
-rw-rw-r--. 1 me me 123059 Mar 26 21:42 tom_sowyer.txt
[me@nnk4 docs]$ sudo chmod ug+x check.sh
[me@nnk4 docs]$ sudo chmod g-w *.txt
[me@nnk4 docs]$
[me@nnk4 docs]$ ls -l
total 268
-rwxrwxr--. 1 me me   3728 Mar 26 21:43 check.sh
-rw-r--r--. 1 me me 139614 Mar 26 21:42 martin_eden.txt
-rw-r--r--. 1 me me 123059 Mar 26 21:42 tom_sowyer.txt
```

Рис. 5.3.6. Пример изменения разрешений на файл с помощью символьного метода (предоставление разрешения на выполнение для владельцев и запрет изменения для группы-владельца)

Числовой метод

```
chmod ### file|directory
```

Каждая цифра представляет собой уровень доступа (пользователь, группа, другие).

- это сумма r=4, w=2, x=1

```
[me@nnk4 docs]$ ls -l
total 12
-rw-rw-r--. 1 me me 78 Mar 26 21:48 text1.txt
-rw-rw-r--. 1 me me 78 Mar 26 21:48 text2.txt
-rw-rw-r--. 1 me me 78 Mar 26 21:48 text3.txt
[me@nnk4 docs]$
[me@nnk4 docs]$ sudo chmod 400 text1.txt
[me@nnk4 docs]$ sudo chmod 640 text2.txt
[me@nnk4 docs]$ sudo chmod 777 text3.txt
[me@nnk4 docs]$
[me@nnk4 docs]$ ls -l
total 12
-r--------. 1 me me 78 Mar 26 21:48 text1.txt
-rw-r-----. 1 me me 78 Mar 26 21:48 text2.txt
-rwxrwxrwx. 1 me me 78 Mar 26 21:48 text3.txt
```

Рис. 5.3.7. Пример изменения разрешений с помощью числового метода

Специальные разрешения

Эффект для файлов:

u+s (suid) – файл выполняется от имени владельца файла, а не фактического пользователя

g+s (sgid) – файл выполняется от имени группы-владельца файла

o+t (sticky) – нет эффекта

Эффект для каталогов:

u+s (suid) – нет эффекта

g+s (sgid) – создаваемые в каталоге файлы в качестве группы-владельца получат группу-владельца каталога

o+t (sticky) – пользователи с разрешением write могут удалять только свои собственные файлы

Задание специальных разрешений

Символическое обозначение

```
chmod g+s directory
```

Числовое обозначение

```
chmod 2770 directory
```

Четвертая справа цифра:

4 = setuid

2 = setgid

1 = sticky

```
[me@nnk4 docs]$ sudo chmod 4755 file1
[me@nnk4 docs]$ sudo chmod 4644 file2
[me@nnk4 docs]$
[me@nnk4 docs]$ ls -l
total 8
-rwsr-xr-x. 1 me me 78 Mar 26 22:31 file1
-rwSr--r--. 1 me me 78 Mar 26 22:31 file2
```

Рис. 5.3.8. Установка эффекта suid для файлов

umask

При создании файла или директории, среда операционной системы присваивает им определенные права доступа по умолчанию.

umask - пользовательская маска (user mask), которая используется для определения конечных прав доступа.

umask для всех пользователей по умолчанию устанавливается в файлах `/etc/.bashrc` или `/etc/.profile`.

После процедуры начальной инсталляции Linux по умолчанию она равна 0022 (022) или 0002 (002).

Узнать текущее значение umask:

```
$ umask
0022
```

Изменить значение umask для текущего сеанса

```
$ umask 027
```

umask указывает, какие биты следует сбросить в выставляемых правах на файл.

В двух следующих примерах показано, как маска umask модифицирует устанавливаемые разрешения для файла:

666 rw-rw-rw- базовые права

027 ----▉-▉▉▉ umask

640 rw-r----- результирующие права

777 rwxrwxrwx базовые права

022 ----▉--▉- umask

755 rwxr-xr-x результирующие права

Разрешения по умолчанию

В операционной системе Linux базовые права для директории равны 0777 (rwxrwxrwx), а для файла 0666 (rw-rw-rw)

По умолчанию umask 0002 используется для обычного пользователя. С этой маской права по умолчанию для директории равны 775, а для файла - 664

Для суперпользователя (root) umask по умолчанию равна 0022. С этой маской права по умолчанию для директории равны 755, а для файла - 644

Специальное разрешение «sticky bit»

Специальное разрешение «sticky bit», которое применяется только к каталогам, запрещает пользователям удалять или переименовывать файл в таком каталоге, если они не владеют файлом

В выводе команды **ls -l** каталоги с установленным sticky bit обозначаются буквой t вместо x в последней группе разрешений.

```
[me@nnk4 docs]$ ls -lhd /tmp /var/tmp /run
drwxr-xr-x. 43 root root 1.4K Mar 26 20:30 /run
drwxrwxrwt. 17 root root 4.0K Mar 26 23:03 /tmp
drwxrwxrwt.  8 root root 4.0K Mar 26 23:01 /var/tmp
[me@nnk4 docs]$
[me@nnk4 docs]$ ls -lhd /mytemp/
drwxrwxrwt. 2 root root 39 Mar 26 23:03 /mytemp/
[me@nnk4 docs]$ ls -l /mytemp/*
-rw-rw-r--. 1 jack jack  6 Mar 26 23:02 /mytemp/book1.txt
-rw-rw-r--. 1 me   me   14 Mar 26 23:03 /mytemp/text.txt
[me@nnk4 docs]$ rm -f /mytemp/*
rm: cannot remove '/mytemp/book1.txt': Operation not permitted
[me@nnk4 docs]$ ls -l /mytemp/*
-rw-rw-r--. 1 jack jack 6 Mar 26 23:02 /mytemp/book1.txt
```

Рис. 5.3.9. Демонстрация эффекта sticky bit – пользователь me не может удалить файл book1.txt, принадлежащий пользователю jack

5.3.4. Атрибуты файлов

В Linux атрибуты файла — это свойства метаданных, которые описывают поведение файла. Например, атрибут может указывать, сжат ли файл, или указывать, можно ли удалить файл [3].

Некоторые атрибуты, такие как неизменяемость, могут быть установлены или очищены, в то время как другие, такие как шифрование, доступны только для чтения и могут быть только просмотрены. Поддержка определенных атрибутов зависит от используемой файловой системы.

Команда **chattr** принимает следующий общий вид:

```
chattr [OPTIONS] [OPERATOR][ATTRIBUTES] FILE...
```

[OPERATOR] может быть одним из следующих символов:

+ Оператор «плюс» указывает, что **chattr** нужно добавить указанные атрибуты к существующим.

- Оператор «минус» указывает **chattr** удалить указанные атрибуты из существующих.

= Оператор равенства указывает, что **chattr** необходимо установить указанные атрибуты как единственные атрибуты.

За оператором следует один или несколько флагов [ATTRIBUTES], которые вы хотите добавить или удалить из атрибутов файла. Ниже приведен список нескольких общих атрибутов и связанных флагов:

a — Когда этот атрибут установлен, файл можно открыть только в режиме добавления для записи.

A — Когда файл с этим установленным атрибутом открыт, его временная запись не изменяется. atime (время доступа) — это время последнего доступа/открытия файла какой-либо командой или приложением.

e — Этот атрибут означает, что файл использует экстенты для отображения блоков на диске. Атрибут e не может быть изменен с **chattr**.

i — Этот атрибут указывает, что файл является неизменяемым, что означает, что файл нельзя удалить или переименовать.

Чтобы получить полный список всех атрибутов файла и флагов, введите man chattr в свой терминал.

По умолчанию атрибуты файла не сохраняются при копировании файла с помощью таких команд, как **cp** или **rsync**.

Примеры chattr

Одно из распространенных применений **chattr** — установка неизменяемого флага для файла или каталога, чтобы пользователи не могли удалить или переименовать файл.

Вы можете просмотреть атрибуты файла с помощью команды **lsattr**:

```
lsattr todo.txt
```

Вывод ниже показывает, что установлен только флаг e:

```
--------------e----- todo.txt
```

Чтобы сделать файл неизменяемым, добавьте флаг i с оператором + к существующим атрибутам:

```
sudo chattr +i todo.txt
```

Мы используем **sudo**, потому что только root может изменить неизменяемый флаг.

Подтвердите, что атрибут добавлен:

```
lsattr todo.txt
```

```
----i---------e----- todo.txt
```

Чтобы отменить изменения и удалить неизменяемый флаг, используйте оператор **-** :

```
sudo chattr -i todo.txt
```

С помощью **chattr** вы можете добавить или удалить сразу несколько атрибутов. Например, чтобы сделать файл неизменяемым и указать ядру не отслеживать время последнего доступа, вы должны использовать:

```
sudo chattr +iA todo.txt
```

Последний оператор, который вы можете использовать, — это оператор =. Например, чтобы установить атрибут **e** как единственный атрибут, вы должны запустить:

```
sudo chattr "=e" todo.txt
```

Обратите внимание, что оператор и флаг заключены в кавычки, чтобы избежать интерпретации символа + оболочкой.

Перечень атрибутов

A - (no atime updates) не изменять время последнего обращения, что может благоприятно повлиять на производительность файловой системы, если обращение происходит очень часто.

a - (append only) в файл можно только дописывать, но нельзя удалять/переименовывать (удобно для логов). Если установлено на каталог, то находящиеся там файлы удалять нельзя, но можно создавать новые и модифицировать существующие.

c - (compressed) производится прозрачное сжатие на диске информации файла ядром, а при доступе возвращаются несжатые данные.

D - (synchronous directory updates) при модификации директории изменения синхронно записываются на диск.

d - (no dump) игнорировать при создании резервной копии программой **dump**.

i - (immutable) пожалуй самый используемый и полезный бит, который запрещает любые изменения файла (нельзя удалять, переименовывать и модифицировать файл). Для директории данный флаг позволяет модифицировать в ней файлы, но нельзя удалять или создавать новые.

j - (data journalling) если файловая система смонтирована с параметрами «data=ordered» или «data=writeback», данные файла с этим атрибутом сохраняются сначала в журнал файловой системы, и только потом в файл. При монтировании с параметром «data=journal» данные и так сохраняются сначала в журнал, поэтому атрибут не действует. Этот атрибут может снять и установить только root.

s - (secure deletion) полное удаление файла (место на диске, где он находился, после заполняется нулями) .

S - (synchronous updates) прямая запись на диск, без кэширования (обновления в файле происходит на диске синхронно с приложением, изменяющим данный файл) .

u - (undeletable) при удалении файла с таким атрибутом, его содержимое сохраняется, что позволяет успешно использовать инструменты для восстановления удаленных файлов.

T - каталог с таким атрибутом считается расположенным на вершине иерархии директорий с целью использования метода распределения блоков по Orlov.

t - к файлу с таким атрибутом нельзя присоединить в конец другой файл (tail-merging). На момент написания ext2 и ext3 не поддерживали (не считая очень экспериментальных патчей) tail-merging.

E - показывает, что при сжатии файла были ошибки. Нельзя установить/снять с помощью chattr, можно лишь посмотреть командой **lsattr**.

e - показывает, что файл использует дополнения для размещения блоков на диске. Нельзя установить/снять с помощью **chattr**, можно лишь посмотреть командой **lsattr**.

I - показывает, что каталог был проиндексирован при использовании **htree**. Нельзя установить/снять с помощью **chattr**, можно лишь посмотреть командой **lsattr**.

H - показывает, что файл хранит свои блоки в единицах файловой системы, а не в единицах секторов, это означает, что файл имеет размер более 2TB (или когда-то занимал). Нельзя установить/снять с помощью **chattr**, можно лишь посмотреть командой lsattr.

X - показывает, что к сжатому файлу можно получить прямой непосредственный доступ. Нельзя установить/снять с помощью **chattr**, можно лишь посмотреть командой **lsattr**

Z - показывает, что сжатый файл is dirty (?). Нельзя установить/снять с помощью **chattr**, можно лишь посмотреть командой **lsattr**.

5.3.5. Управление свойствами файлов

У каждого файла имеется определённый набор свойств в файловой системе. Например, это права доступа, владелец, имя, метки времени. В Linux каждый файл имеет довольно много свойств, например, права доступа устанавливаются трижды (для владельца, группы и всех прочих), метки времени также бывают трёх разных видов (время создание, доступа и изменения).

Большинство этих свойств была рассмотрена в предыдущих материалах (права доступа и владелец в этой теме, имя файла – в теме 4.3 (подраздел 4.3.6). Осталось рассмотреть метки времени.

У каждого файла доступны следующие метки времени [4]:

- Доступ
- Модифицирован
- Изменён
- Создан

Временные метки файла являются частью функциональности файловой системы. Следовательно, на различных файловых системах некоторые временные метки могут быть недоступны.

> *Примечание. Кроме как в свойствах файла в файловой системе, временные метки могут храниться в метаданных самого файла. Очень многие форматы имеют метаданные и довольно часто эти метаданные содержат свой собственный набор временных меток, которые отличаются по составу и даже по текущему их значению от меток времени в файловой системе. Например, для документов Word в метаданных кроме даты создания могут также содержаться метки времени о дате печати файла, об общем времени редактировании файла и так далее. Для изображений в EXIF метаданных также могут содержаться независимая от файловой системы информация о времени создания файла. Для доступа/редактирования к метаданным используются различные приложения и эти вопросы здесь не рассматриваются.*

Сразу все временные метки файла можно посмотреть с помощью команды **stat**. Запуск очень простой:

```
stat ИМЯ_ФАЙЛА
```

Например, я хочу узнать информацию о файле prog.txt:

```
stat prog.txt
```

Пример вывода для файловой системы ext4:

```
  Файл: prog.txt
  Размер: 7025          Блоков: 16         Блок В/В: 4096   обычный файл
Устройство: 10302h/66306d   Инода: 3952903     Ссылки: 1
Доступ: (0644/-rw-r--r--)  Uid: ( 1000/    mial)   Gid: (  985/   users)
Доступ:        2019-07-08 11:47:18.805871040 +0300
Модифицирован: 2019-06-29 12:24:38.361677946 +0300
Изменён:       2019-06-29 12:24:38.361677946 +0300
Создан:        2019-06-29 12:24:11.448251338 +0300
```

Отображение даты создания – относительно новая функция команды **stat**. А для файловой системы ext2 дата создания файла недоступна и сейчас.

Метки времени (timestamps)

Доступ (**A**ccess - last access) — время, когда файл был прочитан последний раз. Это время меняется при доступе таких системных вызовов как **mknod**(2), **utimes**(2) и **read**(2).

Если это текстовый файл, то дата последнего доступа обновляется при каждом его открытии. Если это исполнимый файл, то дата доступа обновится при его запуске.

> *При некоторых опциях монтирования диска (***noatime** *или* **relatime***) это значение может быть неточным*

Модифицирован (**M**odify - last modified) — время последнего изменения содержимого файла. То есть если это текстовый файл, то время модификации поменяется когда вы его откроете и удалите какое-то слово или что-то допишите. Меняется системными вызовами **mknod**(2), **utimes**(2) и **write**(2).

Изменён (**C**hange - last changed) — Время последнего изменения метаданных файлов в файловой системе. То есть если в файле изображения вы измените EXIF метаданные — это будет модификация (поскольку по сути поменяется содержимое файла). Примером Изменения файла является смена разрешений доступа к нему (чтение, запись, выполнение), смена владельца, группы и т. д. Меняется с **chmod**(2), **chown**(2), **link**(2), **mknod**.

Метки времени для папок

Посмотреть метки времени папки можно также с помощью команды **stat**:

```
stat /путь/до/папки
```

Например, чтобы посмотреть информацию о текущей папке:

```
stat .
```

Для папок время последнего доступа обновляется при просмотре списка файлов внутри неё. Действуют такие же правила, как и для файлов — зависит от опций, с которыми смонтирована файловая система.

При создании или удалении нового файла внутри директории, при модификации содержимого или изменении свойств файла внутри папки, одновременно обновляется и время изменения, и время модификации данной папки.

Как отредактировать метки времени файла

С помощью команды **touch** можно изменить три метки времени файла или папки:

- время доступа
- время модификации
- время изменения статуса

Если вы хотите изменить все эти значения на текущее время, то достаточно запустить команду вида:

```
touch ФАЙЛ
```

Можно отдельно поменять только время доступа или только время модификации, соответствующие опции:

-a изменить только время доступа

-m изменить только время модификации

С помощью опции **-t** можно установить любое время, на которое мы хотим поменять метки файла. Формат указания времени следующий:

```
[[CC]YY]MMDDhhmm[.ss]
```

В этой строке то, что в фигурных скобках, является необязательным. Значения букв следующее:

CC – Первые две цифры года (от слова century — века)

YY – Вторые две цифры года

MM – Месяц года (01-12)

DD – День месяца (01-31)

hh – Часы дня (00-23)

mm – Минуты часа (00-59)

ss – Секунды (00-59)

Если не указать века или год вообще, то будет использоваться текущий год. Если не указать секунды, то значением по умолчанию является 00.

Пример:

```
touch -t '198306080301.23' file.txt
```

С помощью опции **-t** невозможно указать доли секунды и в выводе команды **stat** на их месте всегда будет 000000000

С помощью опции **-d** можно использовать разные более человечные способы указать время, например, «две недели назад». Это должно работать на английском, не знаю как с национальными языками. Опция **-d** (судя по описанию) понимает много разных вариантов синтаксиса, но самым интересным свойством является возможность установить доли секунды, чтобы метки времени выглядели естественно. Пример установки времени с указанием долей секунды:

```
touch -d '1983-08-06 04:15:34.123456789' files.txt
```

То есть формат строки такой (из неё можно пропустить дефисы и двоеточия — опция **-d** всё равно её поймёт, но они добавлены для наглядности):

```
YYYY-MM-DD HH:MM:SS.mmmmmmmmm
```

Список использованных источников

1. Типы файлов Linux
https://younglinux.info/bash/filestype

2. Права в Linux (chown, chmod, SUID, GUID, sticky bit, ACL, umask)
https://habr.com/ru/articles/469667/

3. Команда Chattr в Linux (атрибуты файлов)
https://andreyex.ru/linux/komanda-chattr-v-linux-atributy-fajlov/

4. Время создания, доступа и изменения файла: что это, как их узнать и изменить. Как найти файлы по их времени создания, изменения или последнему открытию
https://hackware.ru/?p=9186