МДК.02.01 Программное обеспечение компьютерных сетей

3-курс

Занятие 05, 06

Файловая система

Файловая система

Организация данных в операционных системах определяется **структурой файловой системы**.

Файловая система является одной из важнейших функциональных частей UNIX. Она **обеспечивает**:

- **хранение данных**, принадлежащих операционной системе и пользователям, а также обеспечение их целостности;
- **эффективный доступ к данным**, находящимся на запоминающих устройствах длительного хранения (накопителях на жестких, на оптических дисках и т. д.);

Файловая система

Файловая система UNIX также **обеспечивает**:

- **эффективное выполнение операций** восстановления данных в случае их повреждения;
- **единообразный механизм доступа** ко всем объектам файловой системы.

Файловая система

Файловая система UNIX устроена таким образом, чтобы соответствовать одной из основных концепций этой операционной системы — **представлению всех объектов** операционной системы, независимо от их природы, **в виде файлов**.

Здесь привычное для многих пользователей понятие файла приобретает более широкий смысл.

В привычных для многих операционных системах MS-DOS и в ранних версиях Windows термином "файл" обозначался двоичный образ данных, записанных на диск.

Файловая система

В UNIX к файлам относятся:

- дисковые файлы или каталоги,
- программные объекты — именованные и неименованные каналы,
- сокеты (гнёзда) – названия программных интерфейсов для обеспечения обмена данными между процессами,
- терминальные линии,
- а также физические устройства ввода/вывода, такие, например, как накопители на жестких дисках, параллельный и последовательный порты и т. д.

Файловая система

При этом устройства ввода/вывода представлены специальными файлами, которые имеют название ***файлов устройств***.

Подобное представление означает, что ко всем объектам файловой системы можно обращаться, используя стандартный программный интерфейс, предоставляемый UNIX.

Например, к дисковому файлу, именованному каналу или параллельному порту можно обращаться, используя системные вызовы `open ()`, `read ()`, `write ()` и `close ()`.

Файловая система

Сказанное касается всех пользовательских и части системных программ, для которых собственно и создана такая модель файловой системы.

Операционная система UNIX на уровне ядра и драйверов устройств **обрабатывает запросы к разным устройствам**, дифференцируя их типы и используя различные подходы.

Файловая система

Есть еще одна **причина**, по которой выбрана именно такая архитектура файловой системы, — это необходимость обеспечить **надежность системы**.

Вспомним, что в UNIX пользовательские приложения **не могут** обращаться **напрямую** к аппаратным устройствам иначе как через системные вызовы.

Поэтому разработчики операционной системы использовали один и тот же программный интерфейс как для дисковых файлов, так и для устройств ввода/вывода.

Файловая система

Для программы пользователя обращение к разным по природе типам устройств (параллельный порт и жесткий диск) **прозрачно**, т. е. программно они различаются лишь именами устройств.

В то же самое время для операционной системы оба этих системных вызова могут обрабатываться **различными способами**, поскольку оба устройства управляются различными драйверами, оперирующими как различными типами данных (символьными и блочными), так и различными аппаратными интерфейсами.

Файловая система

Подобная структура файловой системы очень удобна для разработчиков программного обеспечения, поскольку она **обеспечивает унифицированный программный интерфейс** для работы с объектами файловой системы.

Рассмотрим более подробно основные типы файлов, используемые в UNIX.

Файловая система

К ним относятся:

- **бинарные файлы**, содержащие двоичные данные (например, программы или текст), записанные на жесткий диск,
- специальные **файлы устройств**,
- **сокеты**,
- **именованные каналы**,
- а также **символические и жесткие ссылки**.

Файловая система

Бинарные файлы содержат наборы двоичных битов, в которых в закодированном виде находится та или иная информация, например, текст, рисунки, программы, аудиоданные и т. д.

В большинстве случаев, когда используют термин **"файл",** то имеют в виду **именно такие файлы**.

Данные, записанные в такие файлы, должны интерпретироваться:

- или **операционной системой**, если это программный файл,
- или другими **приложениями**.

Файловая система

Специальные ***файлы устройств*** позволяют операционной системе UNIX и другим программам **взаимодействовать** с аппаратными средствами и периферийными устройствами системы.

Файлы устройств **не эквивалентны драйверам** устройств — драйверы обеспечивают доступ к устройству на уровне аппаратно-программного интерфейса, преобразуя пакеты запросов, поступающие от ядра в соответствующие инструкции процессора.

Файловая система

Файлы устройств можно представить как шлюзы, через которые драйвер получает запросы. Когда ядро получает запрос к файлу устройства, оно просто передает этот запрос соответствующему драйверу.

Структура файла устройства **отличается** от той, которую имеет файл данных.

Сами файлы устройств обрабатываются базовыми средствами файловой системы, а их характеристики записываются на диск.

Взаимодействие пользовательской программы, файла устройства и драйвера было показано на рисунке (обра щение к параллельному порту, которому соответствует файл устройства `/dev/lp0`).

Файловая система

Еще один тип файла UNIX — ***сокет*** (гнездо).

Чаще всего сокеты используются для взаимодействия между независимыми процессами, выполняющимися на одной и той же (локальные сокеты) или на разных системах (сокеты протокола TCP).

Сокеты TCP позволяют взаимодействовать процессам, выполняющимся на разных машинах в сетях TCP/IP, хотя могут применяться для обмена данными между процессами, работающими на одном и том же хосте (в этом случае используется так называемый ***интерфейс обратной связи***, который часто называют ***loopback inter face***, он имеет IP-адрес 127.0.0.1).

Файловая система

К объектам файловой системы UNIX относят **именованные каналы**.

Так же как и локальные сокеты, именованные каналы позволяют взаимодействовать процессам, выполняющимся на одной машине.

Именованные каналы создаются командой `mknod`, а удаляются командой `rm`.

К отдельному типу объектов файловой системы можно отнести ***символические*** и ***жесткие ссылки***.

Это такие объекты файловой системы, которые служат для обеспечения **альтернативных** способов обращения к файлам.

Файловая система

Файловые системы UNIX располагаются, как правило, на жестких дисках, каждый из которых состоит из одной или нескольких логически связанных групп цилиндров, называемых **разделами** (partitions).

Физическое **расположение и размер** раздела устанавливаются **при форматировании** диска.

В UNIX-системах разделы являются **независимыми**, доступ к ним осуществляется как к различным носителям данных, при этом один раздел содержит, как правило, **только одну** физическую файловую систему.

Файловая система

Файловую систему можно рассматривать, с одной стороны, как логическую структуру **в виде дерева каталогов и файлов** с четко установленной иерархией.

Именно в таком виде и представляется файловая система UNIX пользователю.

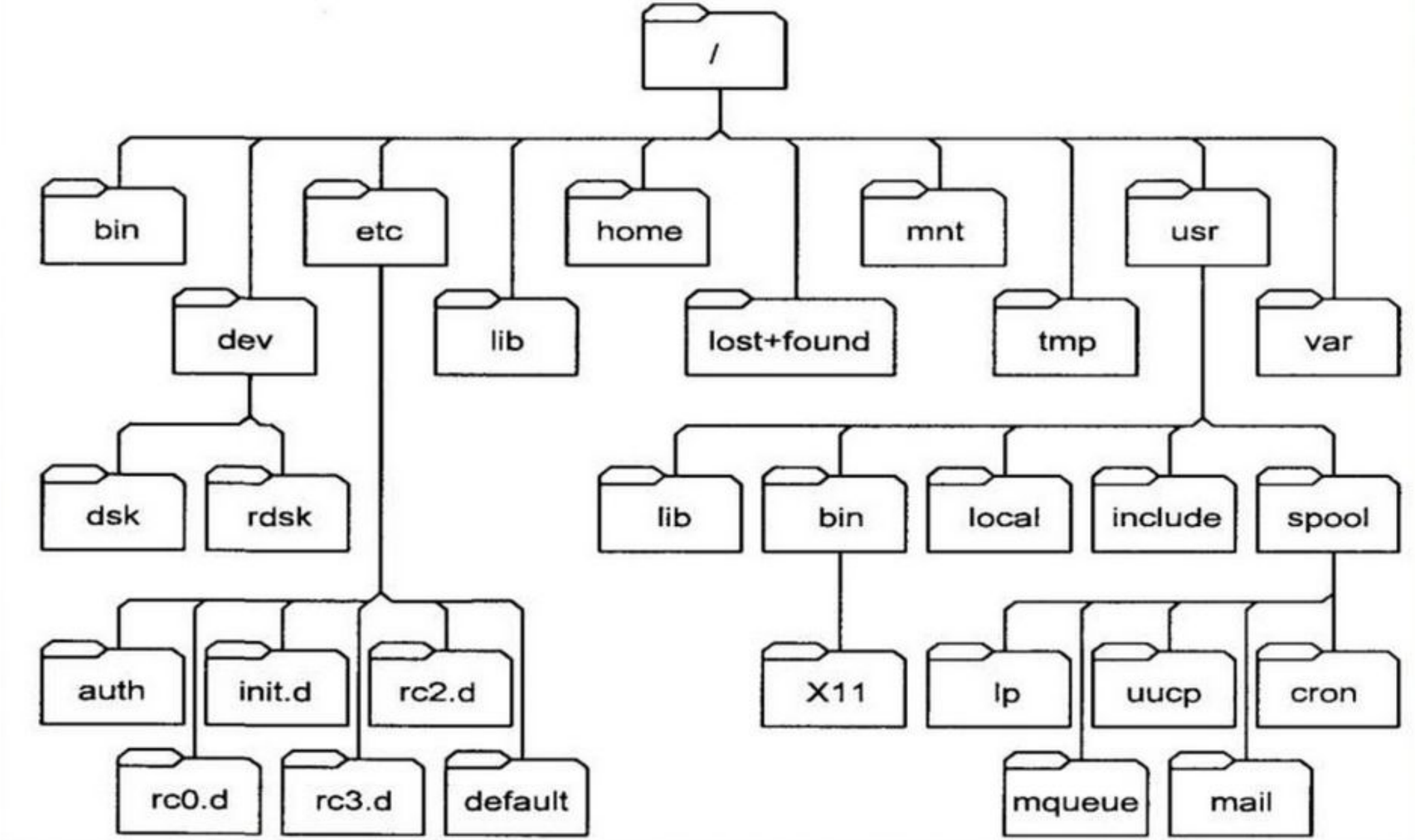
С другой стороны, файловая система — это **совокупность** расположенных на физическом носителе упорядоченных и неупорядоченных **двоичных данных**.

Файловая система

Основой любой файловой системы является корневой каталог (обозначается как /).

Корневой каталог, как и другие каталоги, является **обычным файлом**, содержащим информацию о других файлах, позволяя четко структурировать объекты файловой системы.

Все остальные каталоги и файлы располагаются в рамках структуры, порожденной корневым каталогом (в нем и в его подкаталогах), независимо от их физического местонахождения.


/
bin
etc
home
mnt
usr
dev
lib
lost+found
tmp
var
dsk
rdsk
lib
bin
local
include
spool
auth
init.d
rc2.d
X11
lp
uucp
cron
rc0.d
rc3.d
default
mqueue
mail

Файловая система

Совокупность имен каталогов, через которые проходит путь к заданному файлу, образует, вместе с именем этого файла включительно, ***путевое имя***.

Путевые имена могут быть:

- ***абсолютными*** (например, `/tmp/myfile`) или
- ***относительными*** (например, `local/filesystem`),

при этом относительное имя указывается по отношению к текущему каталогу.

Максимальный размер имени каталога не должен превышать 255 символов, а отдельное путевое имя не должно быть более 1023 символов.

Файловая система

В операционной системе UNIX путевое имя файла принято называть ***жесткой ссылкой***.

Причем для большинства файлов имеется всего лишь одна жесткая ссылка, хотя при помощи команды `ln` пользователь может создать дополнительные жесткие ссылки для каждого из файлов.

Например, для файла с путевым именем `/home/user1/text` можно создать жесткую ссылку `/home/user1/text.new`, выполнив команду

```
ln /home/user1/text /home/user1/text.new
```

После этого при любых операциях с данным файлом к нему можно обращаться по этой ссылке.

Файловая система

Другой тип ссылки — ***символическая***, или, как ее называют по-другому, ***"мягкая"*** ссылка — **позволяет** вместо путевого имени файла **указывать псевдоним**.

При обнаружении символической ссылки, например, во время поиска файла, ядро извлекает из нее путевое имя.

Жесткая ссылка отличается от символической тем, что она указывает непосредственно на индексный дескрип тор файла, т. е. на **физическое расположение** первого блока файла, в то время как символическая ссылка указывает на файл по **его имени**.

Файловая система

Файл, адресуемый символической ссылкой, и сама ссылка являются физически **разными объектами** файловой системы.

Символическая ссылка создается командой `ln -s`, а удаляется командой `rm`. Такая ссылка может указывать на файл, находящийся в другой файловой системе, или даже на несуществующий файл.

В рассмотренном ранее примере можно жесткую ссылку заменить символической:

```
ln -s /home/user1/text /home/user1/text.new
```

Файловая система

В операционной системе UNIX все объекты файловой системы имеют **определенные права доступа**, которые **позволяют или запрещают** выполнять те или иные операции как отдельным пользователям, так и группам пользователей.

Каждый файл имеет **определенную комбинацию** прав доступа, состоящую из девяти битов.

Эта группа битов **определяет**, например, пользователей, имеющих право на **чтение** содержимого файла, на **запись** данных или **выполнение**, если файл является исполняемым.

Файловая система

Биты доступа может **изменить либо владелец файла, либо суперпользователь *root*** при помощи команды UNIX `chmod`.

Эти биты вместе с остальной информацией отображаются на экране командой `ls`.

В коде режима доступа есть биты **специального назначения** — им соответствуют восьмеричные значения 4000 и 2000.

Один из этих битов называется битом **замены идентификатора пользователя** (SUID).

Другой называется битом **замены идентификатора группы** (SGID).

Файловая система

В UNIX-системах нельзя устанавливать биты прав доступа отдельно для каждого пользователя — вместо этого можно применять различные **комбинации из трех битов** (триады) для **владельца** файла, **группы**, которой принадлежит файл, и **прочих** пользователей.

Каждая триада включает **бит чтения, бит записи и бит выполнения** (для каталога последний называется битом поиска).

Очень часто код режима доступа представляют в виде **восьмеричного числа**, при этом каждая цифра в нем представляется тремя битами:

Файловая система

- три старших бита (восьмеричные значения 400, 200 и 100) определяют права доступа к файлу со стороны его **владельца**;
- три средних бита (восьмеричные значения 40, 20 и 10) за дают доступ для пользователей **группы**;
- три младших бита (восьмеричные значения 4, 2 и 1) опреде ляют права доступа к файлу для **остальных** пользователей.

Старший бит каждой триады определяет доступ по **чтению**, средний — по **записи**, и, наконец, младший бит определяет права на **выполнение**.

Файловая система

Для того чтобы удалить или переименовать файл, необходимо, чтобы были установлены соответствующие **биты прав доступа** для каталога, где хранится имя файла.

Установленный бит выполнения, например, **разрешает выполнить файл** как программу или командный сценарий.

Для каталогов этот бит имеет иной смысл.

Если единственным установленным битом является только **бит выполнения**, то разрешается **вход в каталог**, но получить список его содержимого **нельзя**.

Файловая система

Содержимое каталога можно **просмотреть** только при установленных битах **чтения и выполнения**.

Установленные биты **записи и выполнения** позволяют **создавать**, **удалять** и **переименовывать** файлы в данном каталоге.

В операционной системе UNIX есть несколько команд, позволяющих устанавливать права доступа к файлам и каталогам — это команды:

- `chmod` и
- `chown`, причем выполнять их может либо **владелец файла**, либо **суперпользователь *root***.

Файловая система

Команда `chmod` устанавливает права доступа к указанным в командной строке файлам и имеет следующий синтаксис:

`chmod [-fR]` *`абсолютные_права`* *`файл`* `...`

`chmod [-fR]` *`символьные_права`* *`файл`* `...`

Первый параметр команды **указывает права доступа**.

Второй параметр и последующий задают **имена файлов и права доступа, подлежащие изменению**.

Код прав доступа можно задавать в виде **восьмеричного числа**.

В современных версиях поддерживается более наглядная система мнемонических обозначений.

Файловая система

В восьмеричной нотации первая цифра относится к **владельцу**, вторая — к **группе**, а третья — к **остальным пользователям**.

При необходимости задать биты SUID/SGID следует указывать не три, а **четыре восьмеричные цифры**, при этом первая цифра будет соответствовать трем специальным битам.

Опция `-f` команды блокирует вывод сообщения о невозможности установки прав доступа.

А с помощью опции `-R` можно **установить или изменить права доступа** для всех подкаталогов, указанных в списке.

Файловая система

Абсолютные права доступа задаются восьмеричным числом, в то время как символьные права доступа указываются в виде **списка выражений** (через запятую), например:

`[`*`пользователи`*`]  `*`оператор`*`  [`*`права`*`, ...]`

Элемент *`пользователи`* списка определяет, для кого задаются или изменяются права.

Файловая система

Он может принимать значения `u`, `g`, `o` и `a`, относящиеся к владельцу, группе, остальным пользователям, а также ко всем категориям пользователей соответственно.

При этом:

- символ `u` **(user)** обозначает владельца файла,
- символ `g` **(group)** — группу,
- символ `o` **(others)** — других пользователей,
- символ `a` **(all)** — всех пользователей сразу.

Файловая система

Если элемент *пользователи* **не указан**, изменения прав действуют для всех категорий пользователей, хотя не переопределяются установки, задаваемые маской создания файлов.

Элемент *оператор* списка может принимать значения +, – или =, означающие **добавление**, **отмену** права доступа и **установку** указанных прав соответственно.

Если после оператора = ничего не указано, то все права доступа для соответствующих категорий пользователей **отменяются**.

Файловая система

Права доступа в команде *chmod*

Восьмеричное число	Двоичное число	Маска режима доступа
0	000	—
1	001	--x
2	010	-w-
3	011	-wx
4	100	r--
5	101	r-x
6	110	rw-
7	111	rwx

Файловая система

Например, команда

```
chmod  744  text
```

предоставляет **владельцу** все права доступа к файлу `text` и право на чтение — **остальным** пользователям.

Несколько примеров мнемонических обозначений приведено в следующей таблице.

Файловая система

Примеры мнемонических спецификаций команды *chmod*

Спецификация	Описание
u+w	Владельцу файла дополнительно разрешается запись в файл
ug=rwx,o=rx	Владелец и группа получают право на чтение/запись и выполнение, остальные пользователи — право чтения и выполнения
a-wx	Все пользователи лишаются права записи и выполнения
ug=srx,o=	Владелец и группа получают право чтения/выполнения, и устанавливается бит SUID; доступ к файлу остальным пользователям запрещен
g=u	Группе назначаются такие же права, что и владельцу

Файловая система

Рассмотрим пример изменения прав доступа.

Предположим, что в текущем каталоге имеется файл ***test***, для которого мы будем изменять права доступа, как показано далее, одновременно наблюдая за результатами с помощью команды `ls -l`:

```
$ chmod +w test
$ ls -l test
-rw-r--r--   1 root 50              0 Sep 21 12:07 test
```

Файловая система

```
$ chmod a+w test
$ ls -l test
-rw-rw-rw-   1 root 50          0 Sep 21 12:10 test
$ chmod u+x,g=x,o= test
$ ls -l test
-rwx--x---   1 root 50          0 Sep 21 12:12 test
```

Файловая система

```
$ chmod ug-x,og+r,u=rwx test
$ ls -l test
-rwxr--r--   1 root 50            0 Sep 21 12:15 test
$ chmod 644 test
$ ls -l test
-rw-r--r--   1 root 50            0 Sep 21 12:18 test
```

Файловая система

Владелец файла, а также суперпользователь `root` могут изменить владельца и группу-владельца файла, используя команду `chown`, имеющую следующий синтаксис:

`chown [-h][-R] `*`владелец`*`[:`*`группа`*`] `*`файл`*` ...`

Для замены группы, владеющей файлом, можно использовать команду `chgrp`:

`chgrp [-h][-R] `*`группа`*` `*`файл`*

Файловая система

В качестве первого параметра обеих команд используется имя нового владельца файла или новой группы соответственно.

Выполнить команду `chgrp` может только владелец файла, пользователь, входящий в назначаемую группу, или суперпользователь `root`.

Опция `-h` требует замены владельца файла, на который указывает символическая ссылка, а не самой ссылки, как это принято по умолчанию.

Файловая система

В большинстве версий команд `chown` и `chgrp` предусмотрен флаг `-R`, который задает смену владельца или группы не только самого каталога, но и всех его подкаталогов и файлов.

В операционных системах UNIX, совместимых с System V, пользователи могут беспрепятственно поменять владельца собственного файла при помощи команды `chown`, тогда как в BSD-совместимых системах эту команду может выполнять только суперпользователь `root`.

Файловая система

Необходимо учитывать, что после смены владельца файла бывший владелец будет иметь права доступа, установленные новым владельцем.

Пусть требуется сменить владельца файла `test`, атрибуты которого показаны далее:

```
$ ls -l

total 2

-rw-r--r--  1 user1  others    6 Sep 10 16:19
test
```

Файловая система

Из результата выполнения команды `ls` видно, что владельцем файла `test` является пользователь `user1`, в сеансе которого и выполняются команды.

Если заменить владельца файла `test` с `user1` на `root`, выполнив команды:

```
$ chown root test
$ ls -l
total 2
-rw-r--r--  1 root others     6 Sep 10 16:19 test
```

Файловая система

то увидим, что после выполнения команды `chown` пользователем файла `test` является `root`.

Теперь пользователь `user1` никаким образом не сможет в своем сеансе заменить владельца опять на `user1`:

```
$ chown user1 test
```

```
UX:chown: ERROR: testf: Not privileged
```

Полученное сообщение об ошибке говорит о том, что у пользователя `user1` нет прав привилегированного пользователя для смены владельца файла.

Файловая система

Помимо рассмотренных нами атрибутов доступа, объекты файловой системы операционной системы UNIX имеют и другие атрибуты, которые можно получить, используя команду `ls -l`, как в этом примере:

```
$ ls -l /bin/sh
-rwxr-x--x 1 root  bin  87924 Sep 21 2015 /bin/sh
```

Файловая система

Проанализируем полученный результат.

Здесь в первом поле задается тип файла и маска режима доступа к нему: поскольку первым символом является дефис, то это обычный файл.

Обозначения различных типов файлов представлены кодами, состоящими из одного символа (смотри таблицу).

Файловая система

Обозначение типов файлов в выводе команды `ls -l`

Тип файла	Символ	Создается командой	Удаляется командой
Обычный файл	—	Редакторы, `cp` и др.	—
Каталог	`d`	`mkdir`	`rmdir`, `rm -r`
Файл байт-ориенти-рованного устройства	`c`	`mknod`	`rm`

Файловая система

Вернемся к результату выполнения команды `ls`.

Следующие за дефисом девять символов первого поля представляют триады битов режима, обозначенные литерами `r`, `w` и `x` (чтение, запись и выполнение соответственно).

Для данного примера владелец обладает полным доступом к файлу, пользователи группы `bin` — правом на чтение и выполнение, а остальные пользователи могут только выполнить этот файл.

Файловая система

В следующем поле вывода команды `ls` отображается количество жестких ссылок на файл — оно равно 1, а это значит, что `/bin/sh` — единственное имя, под которым известен данный файл.

Всякий раз при создании жесткой ссылки на файл зна чение счетчика ссылок увеличивается на единицу, при этом символические ссылки в счетчике не учитываются.

Что же касается каталогов, то любой из них имеет, как минимум, две жесткие ссылки: одну из родительского каталога и одну из специального файла внутри самого каталога.

Файловая система

Следующие два поля отображают владельца и группу-владельца файла — в данном примере владельцем файла является `root`, а файл принадлежит группе `bin`.

Ядро UNIX хранит эти данные не в виде строк, а как идентификаторы пользователя и группы.

Если получить символьное представление имен по какой-либо причине невозможно, то в этих полях будут отображаться числа.

Такое случается, если запись пользователя или группы была удалена из файла `/etc/passwd` или `/etc/group` соответственно.

Файловая система

Далее следует поле, отображающее размер файла в байтах: данный файл имеет размер 87 924 байта, т. е. почти 88 Кбайт.

Следующее поле содержит дату последнего изменения: 21 сентября 2015 г., и, наконец, в последнем поле вывода содержится имя файла: `/bin/sh`.

Если мы имеем дело с файлом устройства, то вывод команды `ls` будет другим, например:

```
$ ls -l /dev/ttya

crw-rw-rw-  1 root  daemon 12,  0  Dec  20  2018  /dev/ttya
```

Файловая система

Результат работы этой команды иной, чем в предыдущем примере.

Вместо размера в байтах показаны старший и младший номера устройства, а в первом поле отображается тип устройства (в данном случае литера `c` в первой позиции означает, что устройство имеет символьный тип).

Имя `/dev/ttya` относится к первому устройству, управляемому драйвером устройства 12 (в данной системе это драйвер терминала).

Файловая система

При поиске жестких ссылок часто бывает полезной команда `ls -i`, отображающая для каждого файла номер индексного дескриптора. Жесткие ссылки, указывающие на один и тот же файл, будут иметь один и тот же номер.

Операционная система автоматически отслеживает такие атрибуты, как время изменения, число ссылок и размер файла, автоматически устанавливая корректные значения.

В то же время права доступа и идентификаторы принадлежности файла могут быть модифицированы явным образом с помощью команд `chmod`, `chown` и

Список литературы:

1. Unix и Linux: руководство системного администратора, 4-е издание, 2012, Э. Немет, Г. Снайдер, Т. Хейн, Б. Уэйли
2. Для начинающих работать в UNIX, Ф.И. Торчинский.
3. Организация UNIX систем и ОС Solaris 9, Торчинский Ф.И., 2-е издание, исправленное, 2016.

Благодарю за внимание!

Преподаватель: Солодухин Андрей Геннадьевич

Электронная почта: asoloduhin@kait20.ru