Unix и скриптовые языки

Понятие inode

- **inode** – структура данных, используемая файловой системой для хранения всей информации о файле, за исключением имени и содержимого файла.

- Поскольку файлы используются для представления других типов объектов, например, для устройств, **inod'ы** используются и для их представления.

Понятие inode

inode уникален в пределах файловой системы и содержит следующую информацию:

- владельца объекта ФС
- запись о последнем времени
- размер объекта
- тип файла
- права доступа

Ссылки

- **Жесткая ссылка (англ. hard link)** -- один из путей файла. Операционная система не делает различий между именем, которое было изначально присвоено файлу при его создании, и любыми жесткими ссылками.
- **Символьная (англ. symbolic link) --** это файл UNIX, содержащий в себе лишь текстовую строку - путь к оригинальному файлу, на который, собственно, ссылается.

Ссылки

- **Нельзя** создать жесткую ссылку на директорию
- На директорию всегда указывает хотя бы две ссылки

- Файл в Linux существует, пока на inode существует хотя бы одно указание

Пользователи и права

Поскольку система Linux с самого начала разрабатывалась как многопользовательская, в ней **предусмотрен механизм прав доступа к файлам и каталогам**.

Он позволяет разграничить полномочия пользователей, работающих в системе.

Пользователи и права

Для каждого объекта файловой системы в модели полномочий Linux есть **три типа полномочий**:

- **чтения (r от read),**
- **записи (w от write)**
- **выполнения (x от execution)**

Право выполнения можно установить для любого файла.

Дополнительные атрибуты

В Linux кроме прав чтения, выполнения и записи, есть еще **3 дополнительных атрибута:**

- **Sticky bit.** Если он установлен на директорию, то удалить или переименовать файл в ней может только владелец файла.
- **SUID (Set owner User ID up on execution)** – атрибут исполняемого файла, позволяющий запустить его с правами владельца.
- **SGID (Set owner Group ID up on execution)** – аналогично SUID, но относится к группе

Особенности прав доступа директорий

- Право чтения директории позволяет получить список файлов, находящихся в ней (только их имена и типы).
- Чтобы получить дополнительную информацию о файлах каталога, системе необходимо просмотреть метаданные файлов, что требует права на выполнения для каталога.
- Право на выполнение также потребуется для каталога, в который Вы захотите перейти

Особенности прав доступа символьных ссылок

Если посмотреть на права символьных ссылок, то они всегда выглядят так: **rwxrwxrwx**

Дело в том, что **права на символьную ссылку не имеют особого значения.**

Переменные

- Названия переменных **не могут содержать =**
- Значения переменных могут быть составлены **из любых символов portable character set**
- Некоторые зарезервированы:
  - **HOME** -- домашняя директория
  - **GROUPS** -- массив к групп, в которые входит пользователь
  - **PWD** -- текущий рабочий каталог
  - другие

man bash, man ascii

Позиционные параметры

Позиционные параметры – аргументы командной строки, переданные скрипту при вызове, доступ к ним осуществляется по номеру.

- $0 - имя скрипта
- $1 - первый параметр
- $2 - второй параметр
- …

Раскрытие выражений

Когда оболочка получает какую-то строку на выполнение, она до начала выполнения команды осуществляет грамматический разбор полученной строки.

Одним из этапов разбора является **раскрытие или подстановка выражений**.

Подстановка параметров

${parameter}

$ foo=aaa
$ echo ${foo}bar # не тоже самое, что и $foobar
$ echo $foo

Арифметические подстановки

$((expr))

```
$ foo=32
$ bar=10
$ echo $(($foo + $bar))
42
```

Подстановка команд

$(command)

`command`

```
$ mkdir $(seq 1 3)
$ ls -l
drwxr-xr-x 2 root root 4096 Sep  7 21:54 1
drwxr-xr-x 2 root root 4096 Sep  7 21:54 2
drwxr-xr-x 2 root root 4096 Sep  7 21:54 3
```

Замена знака тильды

Слова, **начинающиеся со знака тильды (~)** раскрываются следующим образом

- все символы до первого / или конца слова воспринимаются как имя пользователя и вместо них подставляется домашняя директория пользователя
- в случае, если имя пользователя отсутствует **(~/foobar)** тильда раскрывается в значение переменной **HOME**.

Шаблоны

Шаблон – это строка, которая может содержать специальные символы, называемые **метасимволами** или wildcards.

Используются как образцы для сопоставления со строками.

Не тоже самое, что и регулярные выражения

Шаблоны

- * -- произвольная строка символов, включая пустую строку
- ? -- любой одиночный символ
- [...] любой символ из числа, указанных в скобках:
  - Пары символов, разделенные знаком минуса, обозначают интервал; любой символ, стоящий лексически между этими двумя символами, включая и символы, задающие интервал, соответствует шаблону
  - Если первым символом является ! или ^, то считается, что шаблону (в данной позиции) соответствуют все символы, не указанные в скобках

Раскрытие шаблонов имен файлов и каталогов

- После выполнения указанных выше раскрытий, shell разбивает строку на токены. **В качестве разделителя используется значение переменной IFS.**

- За этим происходит раскрытие имен путей и файлов. Она используются для того, чтобы с помощью краткого шаблона указать несколько имен файлов, соответствующих данному шаблону.

Раскрытие шаблонов имен файлов и каталогов

Раскрытие происходит следующим образом:

- если найдено слово хотя бы с одним вхождением метасимволов, то это слово рассматривается как шаблон, который должен быть заменен словами из лексикографически упорядоченного списка имен путей, соответствующих шаблону
- если имен, соответствующих шаблону, не найдено, слово не изменяется

Порядок раскрытий

1. замена знака тильды
2. подстановка параметров и переменных
3. подстановка команд
4. арифметические подстановки
5. разделение слов
6. раскрытие шаблонов имен файлов

Кавычки

- Заключение символов в одинарные кавычки **(')** сохраняет буквальное значение каждого символа в кавычках. **Нельзя использовать одинарную кавычку внутри одинарных кавычек, даже если она экранирована обратным слэшем**.

- Заключение символов в двойные кавычки **(")** сохраняет буквальное значение всех символов в кавычках, за исключением $,`, \.

Кавычки

$ cat file file1

...

$ cat 'file file1'

cat: 'file file1': No such file or directory

find

Для поиска можно использовать шаблоны, для этого все метасимволы необходимо экранировать, иначе они будут раскрыты.

$ find /usr/share/doc -name README*
/usr/share/doc/iproute2-2.4.7/README.gz
/usr/share/doc/iproute2-2.4.7/README.iproute2+tc.gz
/usr/share/doc/iproute2-2.4.7/README.decnet.gz

find. exec

С помощью find можно производить любые действия над найденными файлами, используя опцию exec.

$ find /usr/bin -type f -size -50c -exec ls -l '{}' ';'

```
-rwxr-xr-x   1 root   root   27 Oct 28 07:13 /usr/bin/krdb
-rwxr-xr-x   1 root   root   35 Nov 28 18:26 /usr/bin/run-nautilus
-rwxr-xr-x   1 root   root   25 Oct 21 17:51 /usr/bin/sgmlwhich
-rwxr-xr-x   1 root   root   26 Sep 26 08:00 /usr/bin/muttbug
```

$ find /usr/bin -type f -size -50c -exec ls -l '{}' +

find. exec

- при ; команда исполняется для каждого файла
- при + все имена файлов разом подаются на вход команде