Министерство образования Республики Беларусь

ПОЛОЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра технологий программирования

Методические указания для выполнения лабораторной работы № 1 по курсу «Операционные системы и системное программирование»

«Файловая система: поиск файлов в директории, права доступа.»

Полоцк, 2019

ЦЕЛЬ РАБОТЫ

Работа в командной строке Linux и оболочке "bash". Работа с файловой системой, предоставление прав доступа.

Краткие теоретические сведения

Файловая система - это структура, с помощью которой ядро операционной системы предоставляет пользователям (и процессам) ресурсы долговременной памяти системы, т. е. памяти на различного вида долговременных носителях информации - жестких дисках, магнитных лентах, CD-ROM и т. п.

С точки зрения пользователя, файловая система — это логическая структура каталогов и файлов. В отличие от Windows, где каждый логический диск хранит отдельное дерево каталогов, во всех UNIX-подобных системах эта древовидная структура растет из одного корня: она начинается с корневого каталога, родительского по отношению ко всем остальным, а физические файловые системы разного типа, находящиеся на разных разделах и даженаудаленныхмашинах, представляютсякакветвиэтогодерева.

Имена файлов и каталогов могут иметь длину до 255 символов. Символы «/» (слэш) и символ с кодом 0 запрещены. Кроме того, ряд символов имеет специальное значение для командногоинтерпретатора, иихиспользованиенерекомендуется. Этосимволы:

~ ! @ # $ & % * ( ) [ ] { } ' " \ : ; > < пробел Заметьте, что точки среди специальных символов нет, и имена вроде

this.is.a.text.file.containing.the.famous.string.hello.world допустимы и широко распространены. Часто последняя отделенная точкой часть имени используется подобно «расширению имени» в Windows, обозначая файл определенного типа, но это обозначение несет смысл только для человека. Так, человеку имя файла ivan_home_tar.gz подсказывает, что это домашний каталог пользователя ivan, упакованный архиватором tar и сжатый компрессором gzip.

Если имя файла начинается с точки, то этот файл считается скрытым: некоторые команды его «не видят». Например, введя в своем домашнем каталоге команду просмотра содержимого каталога ls с ключом -a, означающим «показывать скрытые файлы», вы увидите больше файлов, чем введя ту же команду без ключей.

Linux различает регистр символов в именах файлов: так, в одном каталоге могут находиться два разных файла README и Readme.

Имена каталогов строятся по точно тем же правилам, что и имена файлов.

Полным именем файла (или путем к файлу) называется список вложенных друг в друга каталогов, заканчивающийся собственно именем файла. Начинаться он может с любого каталога, потому что в древовидной структуре между любыми двумя узлами существует путь. Если этот список начинается с корневого каталога, то путь называется абсолютным. Если с любого другого — то относительным (по отношению к этому каталогу).

Корневой каталог обозначается символом «/» (слэш), и этим же символом разделяются имена каталогов в списке. Таким образом, абсолютным именем файла README в домашнем каталоге пользователя ivanov будет /home/petrov/README.

В каждом каталоге существуют два особых «подкаталога» с именами «две точки» и «точка». Первый из них служит указанием на однозначно определенный родительский каталог, а второй — на сам данный каталог. Для корневого каталога, у которого нет родителя, оба эти «подкаталога» указывают на корневой каталог. С помощью этих имен образуются относительные имена файлов. Так, именем вышеупомянутого файла README относительно домашнего каталога /home/ivanov

пользователя ivan будет ../petrov/README.

Жесткая ссылка является просто другим именем для исходного файла. После создания такой ссылки ее невозможно отличить от исходного имени файла. «Настоящего» имени у файла нет, точнее, все такие имена будут настоящими. Команда ls показывает количество именно таких жестких ссылок. Удаление файла по любому из его имен уменьшает на единицу количество ссылок, и окончательно файл будет удален только тогда, когда это количество станет равным нулю. Поэтому удобно использовать жесткие ссылки для того, чтобы предотвратить случайное удаление важного файла.

Создадим жесткую ссылку на файл README и посмотрим, что изменилось в его свойствах:

```
$l
    /home/ivanov/README /home/ivanov/readme_too
$l
    -1 /home/ivanov/README    0    Feb    14    19:08
-rwxr-xr-- 2 ivanov    users    e/ivanov/README
```

Жесткую ссылку можно создавать в любом каталоге, но обязательно на том же физическом носителе (то есть в той же файловой системе), что и исходный файл. О причине этого будет сказано позже.

Другой тип ссылок представляют собой символические ссылки. По назначению они аналогичны ярлыкам в ОС Windows: указывают на файл, расположенный где угодно (например, на съемном носителе), и после удаления такого файла или размонтирования съемного носителя становятся бесполезны.

Символическая ссылка создается той же командой ln с ключом -s: $ln -s /home/ivanov/README /home/ivanov/do.not.readme $ls -1 /home/ivanov/do.not.readme

lrwxrwxrwx 1 ivanov users 16 Feb 14 19:17 /home/ivanov/do.not. readme -> /home/ivanov/README

В поле имени файла после стрелки указано его настоящее имя. Права доступа у всех символических ссылок одинаковы и не значат ничего: возможность доступа к файлу определяется правами исходного файла. Заметьте, что в отличие от файлаоригинала файл-ссылка имеет ненулевую длину: в нем хранится абсолютное имя исходного файла. Попытайтесь вывести файл-ссылку на экран с помощью команды cat, и вы увидите содержание исходного, пустого, файла:

```
$ cat /home/ivanov/do.not.readme
$
```

Значение самой ссылки, то есть имя файла, на который она ссылается, можно узнать с помощью команды readlink.

Права доступа к файлам и каталогам

Поскольку Linux - система многопользовательская, вопрос об организации разграничения доступа к файлам и каталогам является одним из существенных вопросов, которые должна решать операционная система. Механизмы разграничения доступа, разработанные для системы UNIX в 70-х годах (возможно, впрочем, они предлагались кем-то и раньше), очень просты, но они оказались настолько эффективными, что просуществовали уже более 30 лет и по сей день успешно выполняют стоящие перед ними задачи.

Воснове механизмов разграничения доступа лежат имена пользователей и имена групп пользователей. Вы уже знаете, что в Linux каждый пользователь имеет уникальное имя, под которым он входит в систему (логируется). Кроме того, в системе создается некоторое число групп пользователей, причем каждый пользователь может быть включен в одну или несколько групп. Создает и удаляет группы суперпользователь, он же может изменять состав участников той или иной группы. Члены разных групп могут иметь разные права по доступу к файлам, например, группа администраторов может иметь больше прав, чем группа программистов.

Виндексном дескрипторе каждого файла записаны имя так называемого владельца файла и группы, которая имеет права на этот файл. Первоначально, при создании файла его владельцем объявляется тот пользователь, который этот файл создал. Точнее - тот пользователь, от чьего имени запущен процесс, создающий файл. Группа тоже назначается при создании файла - по идентификатору группы процесса, создающего файл. Владельца и группу файла можно поменять в ходе дальнейшей работы с помощью команд chown и chgrp (подробнее о них будет сказано чуть позже).

Теперь давайте еще раз выполним команду ls -l. Но зададим ей в качестве дополнительного параметра имя конкретного файла, например, файла, задающего саму команду ls. (Обратите, кстати, внимание на эту возможность команды ls -l - получить информацию о конкретном файле, а не о всех файлах каталога сразу):

```
[user]$ ls -l /bin/ls
-rwxr-xr-x 1 root root 49940 Sep 12 1999 /bin/ls
```

Вы видите, что в данном случае владельцем файла является пользователь root и группа root. Но нас сейчас в выводе этой команды больше интересует первое поле, определяющее тип файла и права доступа к файлу. Это поле в приведенном примере представлено цепочкой символов -rwxr-xr-x. Эти символы можно условно разделить на 4 группы.

Первая группа, состоящая из единственного символа, определяет тип файла. Этот символ в соответствии с возможными типами файлов, рассмотренными в предыдущем разделе, может принимать такие значения:

- - = - обычный файл;
- d = - каталог;
- b = - файл блочного устройства;

•c = - файл символьного устройства;
•s = - доменное гнездо (socket);
•p = - именованный канал (pipe);
•l = - символическая ссылка (link).

Далее следуют три группы по три символа, которые и определяют права доступа к файлу соответственно для владельца файла, для группы пользователей, которая сопоставлена данному файлу, и для всех остальных пользователей системы. В нашем примере права доступа для владельца определены как rwx, что означает, что владелец (root) имеет право читать файл (r), производить запись в этот файл (w), и запускать файл на выполнение (x). Замена любого из этих символов прочерком будет означать, что пользователь лишается соответствующего права. В том же примере мы видим, что все остальные пользователи (включая и тех, которые вошли в группу root) лишены права записи в этот файл, т. е. не могут файл редактировать и вообще как-то изменять.

Вообще говоря, права доступа и информация о типе файла в UNIX-системах хранятся в индексных дескрипторах в отдельной структуре, состоящей из двух байтов, т. е. из 16 бит (это естественно, ведь компьютер оперирует битами, а не символами r, w, x). Четыре бита из этих 16-ти отведены для кодированной записи о типе файла. Следующие три бита задают особые свойства исполняемых файлов, о которых мы скажем чуть позже. И, наконец, оставшиеся 9 бит определяют права доступа к файлу. Эти 9 бит разделяются на 3 группы по три бита. Первые три бита задают права пользователя, следующие три бита - права группы, последние 3 бита определяют права всех остальных пользователей (т. е. всех пользователей, за исключением владельца файла и группы файла).

При этом, если соответствующий бит имеет значение 1, то право предоставляется, а если он равен 0, то право не предоставляется. В символьной форме записи прав единица заменяется соответствующим символом (r, w или x), а 0 представляется прочерком.

Право на чтение (r) файла означает, что пользователь может просматривать содержимое файла с помощью различных команд просмотра, например, командой more или с помощью любого текстового редактора. Но, подредактировав содержимое файла в текстовом редакторе, вы не сможете сохранить изменения в файле на диске, если не имеете права на запись (w) в этот файл. Право на выполнение (x) означает, что вы можете загрузить файл в память и попытаться запустить его на выполнение как исполняемую программу. Конечно, если в действительности файл не является программой (или скриптом shell), то запустить этот файл на выполнение не удастся, но, с другой стороны, даже если файл действительно является программой, но право на выполнение для него не установлено, то он тоже не запустится.

Вот мы и узнали, какие файлы в Linux являются исполняемыми! Как видите, расширение имени файла тут не при чем, все определяется установкой атрибута "исполняемый", причем право на исполнение может быть предоставлено не всем!

Если выполнить ту же команду ls -l, но в качестве последнего аргумента ей указать не имя файла, а имя каталога, мы увидим, что для каталогов тоже определены

права доступа, причем они задаются теми же самыми символами rwx. Например, выполнив команду ls -l /, мы увидим, что каталогу bin соответствует строка:

drwxr-xr-x 2 root root 2048 Jun 21 21:11 bin

Естественно, что по отношению к каталогам трактовка понятий "право на чтение", "право на запись" и "право на выполнение" несколько изменяется. Право на чтение по отношению к каталогам легко понять, если вспомнить, что каталог - это просто файл, содержащий список файлов в данном каталоге. Следовательно, если вы имеете право на чтение каталога, то вы можете просматривать его содержимое (этот самый список файлов в каталоге). Право на запись тоже понятно - имея такое право, вы сможете создавать и удалять файлы в этом каталоге, т. е. просто добавлять в каталог или удалять из него запись, содержащую имя какого-то файла и соответствующие ссылки. Право на выполнение интуитивно менее понятно. Оно в данном случае означает право переходить в этот каталог. Если вы, как владелец, хотите дать доступ другим пользователям на просмотр какого-то файла в своем каталоге, вы должны дать им право доступа в каталог, т. е. дать им "право на выполнение каталога". Более того, надо дать пользователю право на выполнение для всех каталогов, стоящих в дереве выше данного каталога. Поэтому в принципе для всех каталогов по умолчанию устанавливается право на выполнение как для владельца и группы, так и для всех остальных пользователей. И, уж если вы хотите закрыть доступ в каталог, то лишите всех пользователей (включая группу) права входить в этот каталог. Только не лишайте и себя такого права, а то придется обращаться к суперпользователю![3)]

После прочтения предыдущего абзаца может показаться, что право на чтение каталога не дает ничего нового по сравнению с правом на выполнение. Однако разница в этих правах все же есть. Если задать только право на выполнение, вы сможете войти в каталог, но не увидите там ни одного файла (этот эффект особенно наглядно проявляется в том случае, если вы пользуетесь каким-то файловым менеджером, например, программой Midnight Commander). Если вы имеете право доступа в каком-то из подкаталогов этого каталога, то вы можете перейти в него (командой cd), но, как говорится "вслепую", по памяти, потому что списка файлов и подкаталогов текущего каталога вы не увидите.

Алгоритм проверки прав пользователя при обращении к файлу можно описать следующим образом. Система вначале проверяет, совпадает ли имя пользователя с именем владельца файла. Если эти имена совпадают (т. е. владелец обращается к своему файлу), то проверяется, имеет ли владелец соответствующее право доступа: на чтение, на запись или на выполнение (не удивляйтесь, суперпользователь может лишить некоторых прав и владельца файла). Если право такое есть, то соответствующая операция разрешается. Если же нужного права владелец не имеет, то проверка прав, предоставляемых через группу или через группу атрибутов доступа для остальных пользователей, уже даже не проверяются, а пользователю выдается сообщение о невозможности выполнения затребованного действия (обычно что-то вроде "Permission denied").

Если имя пользователя, обращающегося к файлу, не совпадает с именем владельца, то система проверяет, принадлежит ли владелец к группе, которая

сопоставлена данному файлу (далее будем просто называть ее группой файла). Если принадлежит, то для определения возможности доступа к файлу используются атрибуты, относящиеся к группе, а на атрибуты для владельца и всех остальных пользователей внимания не обращается. Если же пользователь не является владельцем файла и не входит в группу файла, то его права определяются атрибутами для остальных пользователей. Таким образом, третья группа атрибутов, определяющих права доступа к файлу, относится ко всем пользователям, кроме владельца файла и пользователей, входящих в группу файла.

Для изменения прав доступа к файлу используется команда chmod. Ее можно использовать в двух вариантах. В первом варианте вы должны явно указать, кому какое право даете или кого этого права лишаете:

[user]$ chmod wXp имя-файла

где вместо символа w подставляется

•либо символ u (т. е. пользователь, который является владельцем);

•либо g (группа);

•либо o (все пользователи, не входящие в группу, которой принадлежит данный файл);

•либо a (все пользователи системы, т. е. и владелец, и группа, и все остальные). Вместо X ставится:

•либо + (предоставляем право);

•либо - (лишаем соответствующего права);

•либо = (установить указанные права вместо имеющихся).

Вместо p - символ, обозначающий соответствующее право:

•r (чтение);

•w (запись);

•x (выполнение).

Вот несколько примеров использования команды chmod:

[user]$ chmod a+x file_name

предоставляет всем пользователям системы право на выполнение данного файла.

[user]$ chmod go-rw file_name

удаляет право на чтение и запись для всех, кроме владельца файла.

[user]$ chmod ugo+rwx file_name

дает всем права на чтение, запись и выполнение.

Если опустить указание на то, кому предоставляется данное право, то подразумевается, что речь идет вообще обо всех пользователях, т. е. вместо

[user]$ chmod a+x file_name

можно записать просто

[user]$ chmod +x file_name

Второй вариант задания команды chmod (он используется чаще) основан на цифровом представлении прав. Для этого мы кодируем символ r цифрой 4, символ w - цифрой 2, а символ x - цифрой 1. Для того, чтобы предоставить пользователям какой-то набор прав, надо сложить соответствующие цифры. Получив, таким образом,

нужные цифровые значения для владельца файла, для группы файла и для всех остальных пользователей, задаем эти три цифры в качестве аргумента команды chmod (ставим эти цифры после имени команды перед вторым аргументом, который задает имя файла). Например, если надо дать все права владельцу (4+2+1=7), право на чтение и запись - группе (4+2=6), и не давать никаких прав остальным, то следует дать такую команду:

[user]$ chmod 760 file_name

Цифры после имени команды в этой форме ее представления есть не что иное, как восьмеричная запись тех самых 9 бит, которые задают права для владельца файла, группы файла и для всех пользователей.

Выполнять смену прав доступа к файлу с помощью команды chmod может только сам владелец файла или суперпользователь. Для того, чтобы иметь возможность изменить права группы, владелец должен дополнительно быть членом той группы, которой он хочет дать права на данный файл.

Чтобы завершить рассказ о правах доступа к файлам, надо рассказать еще о трех возможных атрибутах файла, устанавливаемых с помощью той же команды chmod. Это те самые атрибуты для исполняемых файлов, которые в индексном дескрипторе файла в двухбайтовой структуре, определяющей права на файл, занимают позиции 5-7, сразу после кода типа файла.

Первый из этих атрибутов - так называемый "бит смены идентификатора пользователя". Смысл этого бита состоит в следующем.

Обычно, когда пользователь запускает некоторую программу на выполнение, эта программа получает те же права доступа к файлам и каталогам, которые имеет пользователь, запустивший программу. Если же установлен "бит смены идентификатора пользователя", то программа получит права доступа к файлам и каталогам, которые имеет владелец файла программы (таким образом, рассматриваемый атрибут лучше называть "битом смены идентификатора владельца"). Это позволяет решать некоторые задачи, которые иначе было бы трудно выполнить. Самый характерный пример - команда смены пароля passwd. Все пароли пользователей хранятся в файле /etc/passwd, владельцем которого является суперпользователь root. Поэтому программы, запущенные обычными пользователями, в том числе команда passwd, не могут производить запись в этот файл. А, значит, пользователь как бы не может менять свой собственный пароль. Но для файла /usr/bin/passwd установлен "бит смены идентификатора владельца", каковым является пользователь root. Следовательно, программа смены пароля passwd запускается с правами root и получает право записи в файл /etc/passwd (уже средствами самой программы обеспечивается то, что пользователь может изменить только одну строку в этом файле).

Установить "бит смены идентификатора владельца" может суперпользователь с помощью команды

[root]# chmod +s file_name

Аналогичным образом работает "бит смены идентификатора группы".

Еще один возможный атрибут исполняемого файла - это "бит сохранения задачи" или "sticky bit" (дословно - "бит прилипчивости"). Этот бит указывает

системе, что после завершения программы надо сохранить ее в оперативной памяти. Удобно включить этот бит для задач, которые часто вызываются на выполнение, так как в этом случае экономится время на загрузку программы при каждом новом запуске. Этот атрибут был необходим на старых моделях компьютеров. На современных быстродействующих системах он используется редко.

Если используется цифровой вариант задания атрибутов в команде chmod, то цифровое значение этих атрибутов должно предшествовать цифрам, задающим права пользователя:

[root]# chmod 4775 file_name

При этом веса этих битов для получения нужного суммарного результата задаются следующим образом:

•4 - "бит смены идентификатора пользователя",

•2 - "бит смены идентификатора группы",

•1 - "бит сохранения задачи (sticky bit)".

Если какие-то из этих трех битов установлены в 1, то несколько изменяется вывод команды ls -l в части отображения установленных атрибутов прав доступа. Если установлен в 1 "бит смены идентификатора пользователя", то символ "x" в группе, определяющей права владельца файла, заменяется символом "s". Причем, если владелец имеет право на выполнение файла, то символ "x" заменяется на маленькое "s", а если владелец не имеет права на выполнение файла (например, файл вообще не исполняемый), то вместо "x" ставится "S". Аналогичные замены имеют место при задании "бита смены идентификатора группы", но заменяется символ "x" в группе атрибутов, задающих права группы. Если равен 1 "бит сохранения задачи (sticky bit)", то заменяется символ "x" в группе атрибутов, определяющей права для всех остальных пользователей, причем "x" заменяется символом "t", если все пользователи могут запускать файл на выполнение, и символом "T", если они такого права не имеют.

Таким образом, хотя в выводе команды ls -l не предусмотрено отдельных позиций для отображения значений битов смены идентификаторов и бита сохранения задачи, соответствующая информация выводится. Вот небольшой пример того, как это будет выглядеть:

[root]# ls -l prim1

-rwSrwsrwT 1 kos root 12 Dec 18 23:17 prim1

Команды для работы с файлами и каталогами

В предыдущих разделах и работах уже упоминали некоторые команды для работы с файлами и каталогами: pwd (имя текущего каталога), cd, ls, ln, chmod. В этом разделе рассмотрим еще несколько часто используемых команд.

Команды chown и chgrp

Эти команды служат для смены владельца файла и группы файла. Выполнять смену владельца может только суперпользователь, смену группы может выполнить сам владелец файла или суперпользователь. Для того, чтобы иметь право сменить группу, владелец должен дополнительно быть членом той группы, которой он хочет дать права на данный файл. Формат этих двух команд аналогичен:

[root]# chown vasja имя-файла [root]# chgrp usersgrp имя-файла

Команда mkdir

Команда mkdir позволяет создать подкаталог в текущем каталоге. В качестве аргумента этой команде надо дать имя создаваемого каталога. Во вновь созданном каталоге автоматически создаются две записи: . (ссылка на этот самый каталог) и .. (ссылка на родительский каталог). Чтобы создать подкаталог, вы должны иметь в текущем каталоге право записи. Можно создать подкаталог не в текущем, а в каком-то другом каталоге, но тогда необходимо указать путь к создаваемому каталогу:

[user]$ mkdir /home/kos/book/glava5/part1

Команда mkdir может использоваться со следующими опциями:

•-m mode - задает режим доступа для нового каталога (например, -m 755);

•-p - создавать указанные промежуточные каталоги (если они не существуют).

Команда touch

Создать пустой файл можно командой touch <имя_файла>. Вообще-то она предназначена для того, чтобы для всех заинтересованных программ (например, утилиты сборки проекта **make)** файл выглядел новее, чем на самом деле: она меняет время последнего изменения файла на текущее время. Но если файла с таким именем не существует, то она его создаст.

Команда cat

Команда cat часто используется для создания файлов (хотя можно воспользоваться и командой touch). По команде cat на стандартный вывод (т. е. на экран) выводится содержимое указанного файла (или нескольких файлов, если их имена последовательно задать в качестве аргументов команды). Если вывод команды cat перенаправить в файл, то можно получить копию какого-то файла:

[user]$ cat file1 > file2

Собственно, первоначальное предназначение команды cat как раз и предполагало перенаправление вывода, так как эта команда создана для конкатенации, т. е. объединения нескольких файлов в один:

[user]$ cat file1 file2 ... fileN > new-file

Именно возможности перенаправления ввода и вывода этой команды и используются для создания новых файлов. Для этого на вход команды cat направляют данные со стандартного ввода (т. е. с клавиатуры), а вывод команды - в новый файл:

[user]$ cat > newfile

После того, как вы напечатаете все, что хотите, нажмите комбинацию клавиш <Ctrl>+<D> или <Ctrl>+<C>, и все, что вы ввели, будет записано в newfile. Конечно, таким образом создаются, в основном, короткие текстовые файлы.

Команда mv

Если вам необходимо не скопировать, а переместить файл из одного каталога в другой, вы можете воспользоваться командой mv. Синтаксис этой команды аналогичен синтаксису команды cp. Более того, она сначала копирует файл (или каталог), а только потом удаляет исходный файл (каталог). И опции у нее такие же, как у cp.**4)**

Команда mv может использоваться не только для перемещения, но и для переименования файлов и каталогов (т. е. перемещения их внутри одного каталога). Для этого надо просто задать в качестве аргументов старое и новое имя файла:

[user]$ mv oldname newname

Но учтите, что команда mv не позволяет переименовать сразу несколько файлов (используя шаблон имени), так что команда mv *.xxx *.yyy не будет работать.

При использовании команды mv, также как и при использовании cp, не забывайте применять опцию -i для того, чтобы получить предупреждение, когда файл будет перезаписываться.

Команды rm и rmdir

Для удаления ненужных файлов и каталогов в Linux служат команды rm (удаляет файлы) и rmdir (удаляет пустой каталог). Для того, чтобы воспользовался этими командами, вы должны иметь право записи в каталоге, в котором расположены удаляемые файлы или каталоги. При этом полномочия на изменение самих файлов не обязательны. Если хотите перед удалением файла получить дополнительный запрос на подтверждение операции, используйте опцию -i.

Если вы попытаетесь использовать команду rm (без всяких опций) для удаления каталога, то будет выдано сообщение, что это каталог, и удаления не произойдет. Для удаления каталога надо удалить в нем все файлы, после чего удалить сам каталог с помощью команды rmdir. Однако можно удалить и непустой каталог со всеми входящими в него подкаталогами и файлами, если использовать команду rm с опцией -r.

Если вы дадите команду rm *, то удалите все файлы в текущем каталоге. Подкаталоги при этом не удалятся. Для удаления как файлов, так и подкаталогов текущего каталога надо тоже воспользоваться опцией -r. Однако всегда помните, что в Linux нет команды восстановления файлов после их удаления (даже если вы спохватились сразу же после ошибочного удаления файла или каталога)!

Так что дважды подумайте до удаления чего-либо и не пренебрегайте опцией -i.

Команды more и less

Команда cat позволяет вывести на стандартный вывод (на экран) содержимое любого файла, однако она используется для этих целей очень редко, разве что для вывода очень небольших по объему файлов. Дело в том, что содержимое большого файла мгновенно проскакивает на экране, и пользователь видит только последние строки файла. Поэтому cat используется в основном по ее прямому назначению - для конкатенации файлов, а для просмотра содержимого файлов (конечно, текстовых) используются команды more и less (или текстовые редакторы).

Команда-фильтр more выводит содержимое файла на экран отдельными страницами, размером как раз в целый экран. Для того, чтобы увидеть следующую страницу, надо нажать на клавишу пробела. Нажатие на клавишу <Enter> приводит к смещению на одну строку. Кроме клавиш пробела и <Enter> в режиме паузы еще некоторые клавиши действуют как управляющие (например, клавиша <B> возвращает вас на один экран назад), но мы здесь не будем приводить полного их перечня, как и перечня опций команды. Вам для начала надо еще только запомнить, что выйти из режима просмотра можно с помощью клавиши <Q>, так как если вы этого не знаете, то вам придется долго и нудно нажимать пробел, пока вы не

доберетесь до конца длинного файла. Обо всех опциях команды more вы можете прочитать в интерактивном руководстве man или info.

Утилита less, разработанная в рамках проекта GNU, содержит все функции и команды управления выводом, имеющиеся в программе more, и некоторые дополнительные, например, позволяет использовать клавиши управления курсором (<Стрелка вверх>, <Стрелка вниз>, <PgUp>, <PgDown>) для перемещения по тексту. Вспомните, мы уже говорили об этом, когда рассматривали интерактивную подсказку man.

Команды more и less позволяют производить поиск подстроки в просматриваемом файле, причем команда less позволяет производить поиск как в прямом, так и в обратном направлении. Для организации поиска строки символов string надо набрать в командной строке программы в нижней части экрана (там, где двоеточие) /string. Если искомая строка будет найдена, будет отображен соответствующий кусок текста, причем найденная строка будет находиться в самом верху экрана.

Команда find и символы шаблонов для имен файлов

Еще одной часто используемой командой для работы с файлами в Linux является команда поиска нужного файла find. Команда find может искать файлы по имени, размеру, дате создания или модификации и некоторым другим критериям.

Общий синтаксис команды find имеет следующий вид: find [список_каталогов] критерий_поиска

Параметр "список_каталогов" определяет, где искать нужный файл. Проще всего задать в качестве начального каталога поиска корневой каталог /, однако, в таком случае поиск может затянуться очень надолго, так как будет просматриваться вся структура каталогов, включая смонтированные файловые системы (в том числе сетевые, если таковые есть). Можно сократить объем поиска, если задать вместо одного корневого каталога список из нескольких каталогов (естественно, тех, в которых может находиться искомый файл):

```
[user]$ find /usr/share/doc /usr/doc /usr/locale/doc -name instr.txt
```

Началом "критерия_поиска", определяющего, что именно должна искать программа find, считается первый аргумент, начинающийся на "-", "(", ")"', "," или "!". Все аргументы, предшествующие "критерию_поиска", трактуются как имена каталогов, в которых надо производить поиск. Если не указано ни одного пути, поиск производится только в текущем каталоге и его подкаталогах.

Чаще всего поиск проводится по именам файлов, как это показано в предыдущем примере, т. е. "критерий_поиска" задается как "-name имя_файла". Вместо опции -name можно использовать опцию -path, тогда команда будет искать совпадения в полном имени файла, с указанием пути. Например, команда

```
[user]$ find . -path './sr*sc'
```

найдет в текущем каталоге подкаталог './src/misc'. Вместо полного имени файла или каталога в этом примере использован так называемый "шаблон имени". Поскольку шаблоны имен файлов могут использоваться не только с командой find, но и со многими другими командами (включая уже рассмотренные

команды chmod, chown, chgrp, cp, rm, cat, mv), то правилам составления шаблонов стоит уделить некоторое внимание.

Чаще всего шаблоны имен файлов строятся с помощью специальных символов "*" и "?". Значок "*" используется для замены произвольной строки символов. В Linux

•"*" - соответствует всем файлам, за исключением скрытых;

•".*" - соответствует всем скрытым файлам (но также текущему каталогу "." и каталогу уровнем выше "..": не забывайте об этом!);

•"*.*" - соответствует только тем файлам и каталогам, которые имеют "." в середине имени, или оканчиваются на точку;

•"p*r" - соответствует и "peter" и "piper";

•"*c*" - соответствует и "picked" и "peck".

Значок ? заменяет один произвольный символ, поэтому index?.htm будет соответствовать именам index0.htm, index5.htm и indexa.htm.

Кроме "*" и "?" в Linux при задании шаблонов имен можно использовать квадратные скобки [], в которых дается либо список возможных символов, либо интервал, в который должны попадать возможные символы. Например, [abc]* соответствует всем именам файлов, начинающимся с a, b, c; *[I-N1-3] соответствует файлам, имена которых оканчиваются на I, J, K, L, M, N, 1, 2, 3.

А теперь вернемся к команде find и расскажем подробнее о том, какие критерии поиска возможны. Несколько примеров простых критериев поиска перечислены в табл. 4.4.

Таблица 4.4. Критерии поиска для команды find

Опция	Значение
-name шаблон	Ищет файлы, имена которых соответствуют шаблону
-group имя	Ищет файлы, принадлежащие указанной группе
-size число[c]	Ищет файлы, размером в число 512-байтных блоков. Если после числа стоит символ c, значит размер указан в байтах (символах)
-mtime число	Ищет файлы, которые в последний раз изменялись указанное число дней назад
-newer образец	Ищет файлы, которые изменялись после изменения файла, указанного в образце
-type тип_файла	Ищет файлы указанного типа. Тип задается одним из символов b (блок-

ориентированные устройства), c (байт-ориентированные устройства), d (файл каталога), f (обычный файл), p (именованный канал) либо l (символическая ссылка)

Другие простые критерии вы можете узнать, если просмотрите man-страницу о команде find. Здесь же надо только сказать, что из простых критериев можно строить более сложные с помощью логических операций and, or или операции отрицания, знаком которой служит восклицательный знак. Например, если вы хотите найти все файлы, имена которых оканчиваются на .txt и .doc, то критерий можно записать как (-name *.txt -or -name *.doc). Можно комбинировать таким образом любое число критериев (и не только простых!). Если операция не указана явно, то подразумевается -and, т. е. вместо (-name *.txt -and -name *.doc) можно записать просто (-name *.txt -name *.doc). Если применяется только одна операция -and или !, то скобки обычно можно опустить, а с операцией -or и в сложных выражениях скобки необходимы. Перед скобкой нужно поставить обратную косую черту, а после скобки - пробел. Например, если вы хотите найти каталог по его имени, то можно сделать это командой

[user]$ find /usr -name doc -type d

или (с соблюдением правил построения сложных критериев)

[user]$ find /usr \(-name doc -and -type d \)

В следующем примере мы ищем файлы по такому критерию: либо имя файла оканчивается на .tmp, либо размер файла больше 100 Кбайт.

[user]$ find /home/kos \(\(-name *.tmp \) -or \(-size +200 \) \)

В последнем примере стоит обратить внимание на то, что перед значением размера стоит знак "+". Такой знак можно использовать с любым числовым параметром в критериях поиска команды find. Он означает, что нужно искать файлы, у которых значение параметра больше заданного. Соответственно, знак "-" означает, что надо искать файлы, у которых значение параметра меньше заданного. Если знаки + или - отсутствует, ищутся файлы, у которых значение параметра равно заданному.

Чтобы закончить рассмотрение команды find, надо сказать еще о том, что после критерия поиска в этой команде можно сразу же задать операцию, которая будет применяться ко всем файлам, найденным по указанному критерию. Простейшим примером использования такой возможности является указание команды -print.

[user]$ find /home/kos -name *.tmp -print

По которой выдается на экран список имен всех найденных файлов с указанием полного пути к файлу. Эта операция применяется по умолчанию, т. е. когда никаких операций вообще не указано (как это было во всех приведенных выше примерах). Другим примером операции, применяемой ко всем найденным файлам, может служить операция - exec cmd {} \;, где cmd - произвольная команда оболочки shell. То есть в этом случае ко всем найденным файлам (их именами заменяются поочередно фигурные скобки) применяется команда cmd. За cmd {} в

этом случае должна следовать точка с запятой, экранированная обратной косой чертой.

Например, если вы хотите удалить в текущем каталоге все файлы, к которым пользователи не обращались в течение 30 дней, дайте команду:

[root]# find . -type f -atime +30 -exec rm {} \;

Вместо -exec можно поставить -ok, тогда перед выполнением указанной команды cmd применительно к каждому файлу будет запрашиваться подтверждение.

В общем, команда find является очень мощным, полезным и чрезвычайно адаптируемым инструментом поиска в файловой системе. Все ее возможности здесь не перечислены, изучайте соответствующую man-страницу. И будьте очень осторожны с применением таких возможностей команды, как вызов других команд, применяемых ко всем найденным файлам. Помните, что изменения часто необратимы!

Редактирование текстовых файлов

Мелкие правки конфигурационных файлов — обычное дело для администратора, поэтому средство их внесения присутствовало в UNIX-системах всегда. Наиболее распространенное такое средство, присутствующее в л.бой системе Linux – это консольный полноэкранный редактор **vi.** Как полноэкранный редактор, vi может находиться в одном из двух режимов. В режиме вставки вводимые символы поступают в редактируемый файл, в командном режиме они воспринимаются как команды. Перечислим коротко самые употребительные команды редактора **vi:**

РЕЖИМ ВСТАВКИ. Включение режима вставки:

•i в текущей позиции курсора;

•I перед первым непробельным символом в текущей строке;

•w в новой строке, добавленной после текущей;

•W в новой строке, добавленной перед текущей.

Выключение режима вставки:

•<Esc>

Команды режима вставки:

•Ctrl+a повторить предыдущую вставку;

•Ctrl+y вставить символ, находящийся над курсором (в предыдущей строке);

•Ctrl+e вставить символ, находящийся под курсором (в следующей строке).

КОМАНДНЫЙ РЕЖИМ.

Удаление (здесь и далее N — это число):

•N x N символов под курсором и справа от него;

•N X N символов слева от курсора;

•N dd N строк;

•D до конца текущей строки;

•N D до конца текущей строки и еще N-1 строку.

Копирование и вставка строк:

•N yy взять в буфер N строк от текущей и ниже;

•p вставить содержимое буфера после текущей строки;

•P вставить содержимое буфера перед текущей строкой.

Поиск и переход:

•N G перейти к строке с номером N;

•$ G перейти к последней строке файла;

•/< образец > искать образец вниз от курсора;

•?< образец > искать образец вверх от курсора;

•n повторить поиск в том же направлении;

•N (буквально N): повторить поиск в обратном направлении.

Сохранение и выход:

•:w сохранить текущий файл;

•:w <имя> сохранить под новым именем, если файл <имя> еще не существует;

•:w! <имя> сохранить под новым именем, переписав существующий файл;

•:q выйти;

•:q! принудительно выйти без сохранения;

•:wq сохранить и выйти.

Разное полезное:

- N u отменить последние N изменений;
- N Ctrl+r вернуть последние N отмененных изменений;
- U отменить изменения в последней строке;
- N r < символ > заменить N следующих символов на < символ >;
- N > > добавить отступ (Tab) в N следующих строк;
- N < < удалить один отступ (Tab) из N следующих строк;
- :sh временно выйти в оболочку (вернуться — exit);
- :!<команда> выполнить команду оболочки.

Работа с vi в простых случаях сводится к использованию следующего небольшого набора команд:

vi <имя файла>	# открыли файл для просмотра или редактирования или создания
i	перешли в режим ввода ход в режим ввода текста (если требуется)
ESC	вышли из режима редактирования в режим команд
:перешли из режима команд в режим командной строки	
w	записали изменения (если требуется)
q –> Enter	вышли из редактора (если изменения уже записаны или их не было)
q! - > Enter	вышли из редактора без сохранения изменений (если требуется).

ЗАДАНИЯ ДЛЯ ЛАБОРАТОРНЫХ РАБОТ

1. Изучить все команды работы с фалами, представленные в теоретической части.
2. Выполнить все следующие задания:

№	Задание
1	Поиск файлов по маске в указанном каталоге и его подкаталогах. Маска и каталог задается параметром пакетного файла. Найденные файлы вывести.
2	Удаления файлов по заданному расширения в каталогах и их подкаталогах. Расширение и каталог задается параметром пакетного файла. Удаленные файлы вывести.
3	Копирование группы файлов удовлетворяющих маске из одного каталога в два других (копировать подкаталоги). Маска и два других каталога передаются как параметры пакетного файла.
4	Перемещение файлов, в имении которых содержится больше трех цифр, в другой подкаталог. Искомый каталог и каталог для перемещения передаются параметрами пакетному файлу. Перемещаемые файлы вывести.
6	Запуск всех исполняемых файлов в указанном в каталоге. Каталог вводится в качестве параметра. Учесть возможность рекурсии когда запускаемый сценарий находится в том же каталоге.

СОДЕРЖАНИЕ ОТЧЕТА

1. Ф.И.О., группа, название лабораторной работы.
2. Цель работы.
3. Описание проделанной работы.
4. Результаты выполнения лабораторной работы.
5. Выводы.