Занятие 3

QA.

START UP

Коммандная строка UNIX

Темы занятия

1. Работа с файлами и каталогами
2. Другие утилиты по работе с файлами
3. Права и безопасность в UNIX
4. Процессы
5. Vi – текстовый редактор

Работа с файлами

- **touch**
- Команда **touch** позволяет создавать файлы. Её применение наиболее просто:

touch <имя файла>

Если файл с заданным именем существует в текущей директории, команда touch обновит его время создания на текущее.

•*ln*: Ссылки

В UNIX допускается, чтобы один и тот же файл существовал в системе под разными именами. Доступ к одному и тому же файлу при помощи нескольких имён может понадобиться в следующих случаях:

- Одна и та же программа известна под несколькими именами.
- Доступ пользователей к некоторым каталогам в системе может быть ограничен из соображений безопасности. Однако если всё же нужно организовать доступ пользователей к файлу, который находится в таком каталоге, можно создать жёсткую ссылку на этот файл в другом каталоге.
- Современные файловые системы даже на домашних персональных компьютерах могут насчитывать до нескольких десятков тысяч файлов и тысячи каталогов. Обычно у таких файловых систем сложная многоуровневая иерархическая организация — в результате пути ко многим файлам становятся очень длинными. Чтобы организовать более удобный доступ к файлу, который находится очень «глубоко» в иерархии каталогов, также можно использовать жёсткую ссылку в более доступном каталоге.
- Полное имя некоторых программ может быть весьма длинным (например, i586-alt-linux-gcc-3.3), к таким программам удобнее обрщаться при помощи сокращённого имени (жёсткой ссылки) — gcc-3.3.

Работа с файлами

- ***ln*: создание ссылок**

- Ссылки бывают двух типов: жесткие и символические (soft или Symbolic). Для создания ссылок используется команда ln.
  В качестве первого параметра пишется абсолютный адрес и имя исходного файла, в качестве второго – адрес и имя ссылки. Например:

ln file.txt link2 создание жесткой ссылки

ln -s file.txt link1 создание символической ссылки

Команда с ключем -s создает символическую (soft) ссылку link1, которая ссылается на текстовый файл file.txt.

Символьная ссылка (также симлинк от англ. Symbolic link, символическая ссылка) — специальный файл в файловой системе, для которого не формируются никакие данные, кроме одной текстовой строки с указателем. Эта строка трактуется как путь к файлу, который должен быть открыт при попытке обратиться к данной ссылке (файлу).

- Целью ссылки может быть любой объект — например, другая ссылка, файл, папка, или даже несуществующий файл (в последнем случае при попытке открыть его должно выдаваться сообщение об отсутствии файла). Ссылка, указывающая на несуществующий файл, называется висячей. В отличие от жестких ссылок, символьные ссылки можно создавать и на каталоги.

- Практически символьные ссылки используются для более удобной организации структуры файлов на компьютере, так как позволяют одному файлу или каталогу иметь несколько имён, различных атрибутов и свободны от некоторых ограничений, присущих жёстким ссылкам (последние действуют только в пределах одного раздела и не могут ссылаться на каталоги).

Модифицируя ссылку вы автоматически модифицируете исходный файл — file.txt

• *ln*: Удаление ссылок

Посмотреть, куда ведет символьная ссылка можно командой:

ls -l linkname

Висячие ссылки (которые указывают куда-то, где нет файла) ls с цветной схемой отображения показывает красным.

- **Удаление символьной ссылки**
- Удалять символьную ссылку нужно как обычный файл, при удалении она не затрагивает то, куда ссылается. Если символическая ссылка указывает на файл, то ее можно просто удалить с помощью команды rm.

rm linkname

- При удалении символьной ссылки на каталог следует учитывать, что записывать команду без слеша в конце, иначе bash выдаст ошибку "невозможно удалить ссылку: Это не каталог".
- Корректная запись:

unlink linkname

rm -r linkname

Работа с файлами

- **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.

Работа с файлами и каталогами

- **rm**
- Команда **rm** используется для удаления файлов. Основные параметры, используемые с командой rm

-f	Удаляет без запросов подтверждения все файлы, независимо от прав доступа к ним, если имеется право записи для каталога.
-i	Запрашивает подтверждения, прежде чем удалить файл. Опция -i отменяет действие опции -f; она действует даже тогда, когда стандартный входной поток не связан с терминалом.
-r	Рекурсивное удаление, с подкаталогами, в том числе, не пустыми.
-R	То же, что и опция -r.

Команда **rm** без опций рекурсивного удаления не удаляет каталоги. Для удаления **пустых** каталогов предназначена команда **rmdir**. Если в каталоге есть другие файлы, кроме ссылок на текущий и родительский каталог, команда **rmdir** его не удаляет.

Работа с файлами и каталогами

•cp

- Команда **cp** копирует исходный файл в целевой файл или каталог. Она имеет следующий синтаксис:

cp [-p] **исходный целевой**

cp [-r] [-p] **исходный**... **каталог**

Исходный файл не должен совпадать с целевым. Если целевой файл является каталогом, то исходные файлы копируются в него под теми же именами. Только в этом случае можно указывать несколько исходных файлов. Если целевой файл существует и не является каталогом, его старое содержимое теряется. Права доступа, владелец и группа целевого файла при этом не меняются.

Если целевой файл не существует или является каталогом, новые файлы создаются с теми же правами доступа, что и исходные. Время последнего изменения целевого файла (последнего доступа, если он не существовал), а также время последнего доступа к исходным файлам устанавливается равным времени копирования. Если целевой файл был связью на другой файл, все связи сохраняются, а содержимое файла изменяется.

Команда **cp** поддерживает следующие основные опции:

-p сохраняет информацию о владельце, по возможности - права доступа и времена доступа для нового файла;

-r копирует рекурсивно, включая подкаталоги.

Работа с каталогами

- **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);
- -v, --verbose печатать сообщение о каждом созданном каталоге;
- -p — создавать указанные промежуточные каталоги (если они не существуют).

Работа с файлами и каталогами

- **Команды *rm* и *rmdir***
- Для удаления ненужных файлов и каталогов в Linux служат команды rm (удаляет файлы) и rmdir (удаляет **пустой** каталог). 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.

Работа с файлами и каталогами

- **Команда *mv***
- mv [-f][-i] **исходный_файл целевой_файл**
  mv [-f][-i] **исходный_файл** ... **Каталог**
- 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.
- 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.
- **-f** Принудительное перемещение - если целевой файл уже существует, то он удаляется.

Поиск

- **Команда *find* и символы шаблонов для имен файлов**
- 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. 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

[user]$ find / -name '*.rpm -print'

[user]$ find / -ctime 5 -print, где -ctime <сколько дней файлу>

[user]$ find / -size 1024k -print, где -size <размер файла>

Опция -print используется для отображения результатов поиска в некоторых Unix системах

- https://ru.wikipedia.org/wiki/Find

Поиск файлов

- 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.

Поиск файлов

- Команда **which**
- Еще одной полезной командой является which. Эта команда помогает находить файлы, указанные в переменной среды PATH. Это означает, что данные могут быть найдены по команде which лишь в тех местах (каталогах), которые указаны в переменной среды PATH.
- Команду which нецелесообразно использовать в сценарии, но это совсем не означает, что она не приносит никакой пользы. С ее помощью можно, например, очень быстро найти файл для последующей правки, если точно известно, что его местоположение указано в переменной среды PATH.

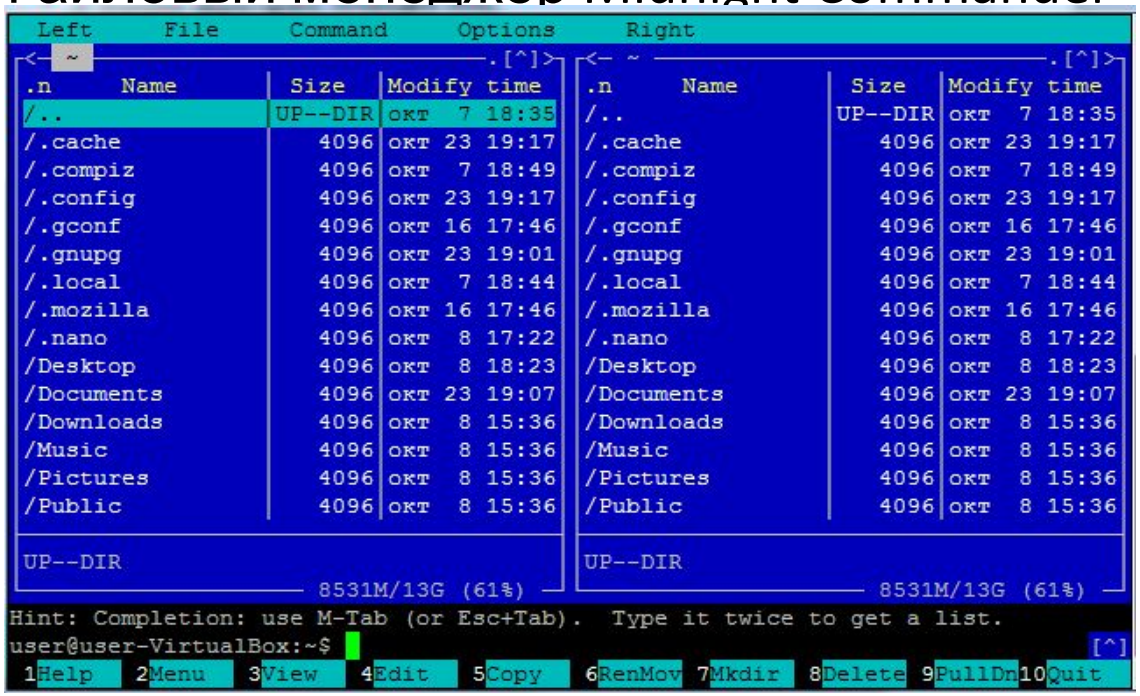
Поиск файлов

- Команда **whereis**
- С помощью команды whereis осуществляется быстрый поиск данных в указанных каталогах. Так, если требуется найти файл с именем test в указанных каталогах, содержащих исходный код программ, двоичные файлы и оперативные страницы руководства, по данной команде будут найдены экземпляры test, встречающиеся в именах файлов. Команда whereis осуществляет быстрый поиск файла по заданному критерию:
- >whereis test

  test: /usr/bin/test /usr/share/man/manl/test.1.gz /usr/share/man/manlp/test.lp.gz
- Возможно, полученный результат — это не совсем то, что вы ищете, но все же команда whereis иногда бывает весьма полезной.

Другие утилиты по работе с файлами

Файловый менеджер Midnight Commander



$ sudo apt-get install mc

$ mc

С помощью mc намного удобнее выполнять рутинные операции: копирование, переименование, архивирование, поиск файлов. К тому же mc оснащен встроенным и очень удобным текстовым редактором, который можно запускать отдельно — командой mcedit. Может, использование mc это и не в стиле UNIX-гуру, но зато удобно.

Другие утилиты по работе с файлами

wc

Команда wc отлично подходит для получения текущих статистических данных о том файле, с которым вам приходится работать. Так, если требуется просмотреть по команде tail конец файла, который может оказаться довольно большим, с помощью команды wc можно очень быстро определить число строк в этом файле. После этого для просмотра конца файла остается лишь указать требуемое число строк в команде tail.

```
user@user-VirtualBox:~$ wc file
  58  519 3261 file
user@user-VirtualBox:~$
```

Если нужно вывести только количество строк, слов или символов в файле, воспользуйтесь ее ключами **-l**, **-w** и **-c** соответственно.

Другие утилиты по работе с файлами

split — команда, копирующая файл и разбивающая его на отдельные файлы заданной длины. В качестве аргументов ей надо указать имя исходного файла и префикс имен выходных файлов. Имена выходных файлов будут составляться из этого префикса и двух дополнительных букв aa, ab, ac и т. д. (без пробелов и точек между префиксом и буквами). Если префикс имен файлов не задан, то по умолчанию используется x, так что выходные файлы будут называться xaa, xab и т. д. По умолчанию размер части равен 1000 строк

split <дополнительные параметры разбиения > <входной файл> <выходной файл >

Параметры разбиения:

-a, --suffix-length=H использовать суффиксы длины H (по умолчанию 2)

-b, --bytes=ЧИСЛО записывать в каждый выходной файл заданное ЧИСЛО байт

-C, --line-bytes=ЧИСЛО записывать не более заданного ЧИСЛА байт из строки

-d, --numeric-suffixes использовать числовые, а не алфавитные суффиксы

-l, --lines=ЧИСЛО записывать в каждый выходной файл заданное ЧИСЛО строк

```
$ split -l 3 file.txt splitfile
$ split -a 1 -d -b 4000M sample.iso sample.iso.part
```

Разобьет исходный файл sample.iso на части по 4 Гбайта (максимальный размер файла в FAT), каждая из которых будет именоваться как sample.iso.partN, где N = 0, 1, 2,… .

Собрать части воедино на целевой системе поможет команда cat:

```
$ cat sample.iso.part* > sample.iso
```

Другие утилиты по работе с файлами

diff сравнения файлов и вывод несоответствий. Команда diff оказывается полезной и в том случае, если требуется установить идентичность двух или более копий одного и того же файла.

diff *<образцовый файл> <обновленный файл>* **>** *<вставной файл>.*

В качестве вставного может быть указан любой файл.

В приведенном ниже примере демонстрируется создание вставного файла:

```
$ diff template.txt update.txt > patch.txt
```

Вставной файл представляет собой разность между образцовым и обновленным файлами.

Если копии не одинаковы, вы сможете выяснить, какие изменения в них произошли. Если же команда diff не дает никакого результата, значит, проверяемые файлы копий ничем не отличаются друг от друга.

Обновить (пропатчить) старый файл (template.txt) патчем изменений (patch.txt):

```
$ patch template.txt patch.txt
```

Другие утилиты по работе с файлами

*Некоторые параметры программы **diff***

-b	Программа будет игнорировать пробельные символы в конце строки
-B	Игнорирует пустые строки
-e	Используется для создания сценария для редактора ed, который будет использоваться для превращения первого файла во второй
-w	Игнорирует пробельные символы
-y	Вывод в два столбца
-r	Используется для сравнения файлов в подкаталогах. Вместо первого файла

http://rus-linux.net/MyLDP/consol/linux-diff-command-file-comparison.html

Сжатие данных

- **Программа tar**
  Программу tar можно использовать для создания архива
  tar [-опции] <имя файла tar> [файлы, которые необходимо поместить в архив при сжатии]
- tar не создаёт сжатых архивов, а использует для сжатия внешние утилиты, такие, как gzip и bzip2
- Основные опции:

-c, --create — создать архив;

-r, --append — добавить файлы в конец существующего архива;

-x, --extract, --get — извлечь файлы из архива;

-f, --file — указать имя архива;

-j, --bzip2 — сжать/распаковать архив дополнительно при помощи bzip2;

-z, --gzip, --gunzip, --ungzip — сжать/распаковать архив дополнительно при помощи gzip;

-v, --verbose — выводить список обработанных файлов

$ man tar | grep '\-C'

tar -cvfj archive.tar.bz2 dir1	*создать архив и сжать его с помощью bzip2*(Прим.переводчика. ключ -j работает не во всех *nix системах)
tar -xvfj archive.tar.bz2	*разжать архив и распаковать его*(Прим.переводчика. ключ -j работает не во всех *nix системах)
tar -cvfz archive.tar.gz dir1	*создать архив и сжать его с помощью gzip*
tar -xvfz archive.tar.gz	*разжать архив и распаковать его*

Права и безопасность UNIX

- ***Авторизация*** – это процесс определения того, имеет или не имеет некоторый субъект доступ к некоторому объекту. Авторизация может быть:
- статической вопрос о доступе к объекту решается один раз, когда права задаются или изменяются, при этом пользователю ставится в соответствие некоторый номинальный субъект системы;
- динамической принятие решения о доступе производится при каждом обращении к объекту, часто это носит характер ограничения возможностей пользователя по объёму памяти и дискового пространства, времени работы и т.п..
- Процессу *авторизации* всегда должен предшествовать процесс *аутентификации*. ***Аутентификация*** – это механизм сопоставления работающего пользователя системы некоторому номинальному субъекту. Как правило, при этом пользователю необходимо ввести пароль или предоставить секретный ключ.

Права и безопасность UNIX

- Для каждого объекта в файловой системе Linux существует набор прав доступа, определяющий взаимодействие пользователя с этим объектом. Такими объектами могут быть файлы, каталоги, процессы, а также специальные файлы (например, устройства). Так у каждого объекта в Linux имеется владелец, то права доступа применяются относительно владельца файла. Они состоят из набора 3 групп по три атрибута:
- **чтение (read, *r*), запись (write, *w*) и выполнение (execution, *x*) для владельца, группы владельца, для всех остальных.**

Права и безопасность UNIX

- **Суперпользователь**
- Пользователь root (он же *суперпользователь*) имеет нулевые UID и GID и играет роль *доверенного субъекта UNIX*. Это значит, что он не подчиняется законам, которые управляют правами доступа, и может по своему усмотрению эти права изменять. Большинство настроек системы доступны для записи только суперпользователю.

Права и безопасность UNIX

- Для распределения прав доступа в Linux существует множество команд. Основные из них - это chmod, chown и chgrp.
- Команда ***chmod*** *(Change MODe - сменить режим)* - изменяет права доступа к файлу. Для использования этой команды также необходимо иметь права владельца файла или права root. Синтаксис команды таков:
- chmod mode filename, где
- filename - имя файла, у которого изменяются права доступа;
- mode - права доступа, устанавливаемые на файл. Права доступа можно записать в 2 вариантах - символьном и абсолютном.

Права и безопасность UNIX

Каждый файл в ОС UNIX содержит набор прав доступа который хранится в индексном дескрипторе данного файла в виде целого значения, из которого обычно используется 12 битов. Причем каждый бит используется как переключатель, разрешая (значение 1) или запрещая (значение 0) тот или иной доступ.

Три первых бита устанавливают различные **виды поведения при выполнении**. Оставшиеся **девять** делятся на три группы по три, определяя права доступа для владельца, группы и остальных пользователей. Каждая группа задает права на чтение, запись и выполнение.

Права и безопасность UNIX

- Символические обозначения, иногда называемые символическими выражениями, используют буквы вместо восьмеричных значений для назначения прав на файлы и каталоги. Символические выражения используют синтаксис:

chmod (кто)(действие)(права)file, где существуют следующие значения:

Опция	Буква	Значение
(кто)	u	Пользователь (User)
	g	Группа (Group)
	o	Другие (Other)
	a	Все (All)
(действие)	+	Добавление прав
	–	Удаление прав
	=	Явная установка прав
(права)	r	Чтение (Read)
	w	Запись (Write)
	x	Выполнение (Execute)
	t	Sticky бит
	s	SUID или SGID

chmod g+rw lesson5.txt

chmod o=u lesson5.txt

chmod o-w lesson5.txt

Права и безопасность UNIX

- **Специальные права доступа (SUID и SGID)**

Мы рассмотрели обычные права доступа к файлам, но в UNIX есть еще так называемые *специальные права доступа*: SUID (Set User ID root) и SGID (Set Group ID root). Эти права доступа позволяют обычным пользователям запускать программы, требующие для своего запуска привилегий пользователя root. Например, демон pppd требует привилегий root, но чтобы каждый раз при установке PPP-соединения (модемное, ADSL-соединение) не входить в систему под именем root, достаточно установить специальные права доступа для демона pppd.

Делается это так:

chmod u+s /путь/pppd

Однако не нужно увлекаться такими решениями, поскольку каждая программа, для которой установлен бит SUID, является потенциальной "дырой" в безопасности вашей системы. Для выполнения программ, требующих прав root, намного рациональнее использовать программы sudo и su (описание которых можно получить по командам man sudo)

Права и безопасность UNIX

- Так уж заведено, что в мире UNIX чаще пользуются абсолютным методом. Для использования абсолютного режима необходимо представить права доступа к файлу в виде 3-х двоичных групп.

chmod 755 lesson5.txt

•Наиболее популярные права доступа:

644 — владельцу можно читать и изменять файл, остальным пользователям — только читать;

666 — читать и изменять файл можно всем пользователям;

777 — всем можно читать, изменять и выполнять файл.

Права и безопасность UNIX

Команда *chown*: смена владельца файла

- Если вы хотите "подарить" кому-то файл, т. е. сделать какого-то пользователя владельцем файла, нужно выполнить команду chown:

  chown пользователь файл

- Возможно, что после изменения владельца файла вы сами не сможете получить к нему доступ, ведь владельцем будете уже не вы.

Управление процессами UNIX

Пользовательские процессы могут выполняться как в интерактивном (приоритетном), так и в фоновом режимах. Интерактивные процессы монопольно владеют терминалом, и пока такой процесс не завершит свое выполнение, пользователь не имеет доступа к командной строке.

Запуск задания в фоновом режиме

Фоновый режим позволяет продолжить использование сеанса работы с терминалом, пока выполняется команда. Для запуска команды в фоновом режиме, достаточно к команде добавить символ амперсанд (**&**). Командный интерпретатор вернет номер задания и идентификатор процесса:

```
$ make &
```

Если задание не требует от пользователя ввода, оно продолжает свою работу до полного завершения. Если команде нужен ввод, она переходит в состояние ожидания, на экран выводится соответствующее уведомление, которое выглядит примерно так:

```
[1] + Suspended (tty input) programm 0
```

В данном случае в ожидании ввода приостановилось выполнение программы **programm**. Пользователю необходимо перевести из фонового режима в привилегированный и выполнить ввод.

Управление процессами UNIX

Просмотр состояния заданий

С помощью команды **jobs** пользователь имеет возможность просмотреть состояние своих заданий и получит список всех заданий запущенных в сеансе работы с терминалом.

```
$ jobs
 [1] Stopped (user) du
 [2]- Stopped (user) du -a /home/intdbi
[3]+ Stopped (user) du -r /home/intdbi
$
```

Команда **jobs** принимает два флага. Флаг **-l** включает идентификатор процесса с номером задания . Флаг **-p** заменяет номер задания на идентификатор процесса.

Номера заданий

Номер задания позволяет командному интерпретатору наблюдать за процессами. Его можно рассматривать как головной элемент группы процессов, поскольку пользовательское задание порождает любые команды либо в конвейере, либо, как подзадания.

Управление процессами UNIX

Перевод задания в привилегированный режим

Команда **fg** переводит задания в привилегированный режим. При наличии приостановленного задания, его можно сделать привилегированным (перевести на передний план) с помощью команды **fg #номер_задания** (или **fg номер_задания** в **bash**). После этого задание либо выведет на экран сообщение о том, что ему нужно от терминала, либо будет принимать ожидаемый ввод. Переведя задание в привилегированный режим, можно приостановить его выполнение, нажав комбинацию клавиш **Ctrl-Z**, и заняться им позже.

Любое задание из списка, предоставленного командой **jobs**, доступно, если пользователь захочет сделать его привилегированным, даже в том случае, когда оно уже работает в фоновом режиме. Если в этом списке приведено только одно задание, то при использовании команды **fg** пользователю не нужно задавать его номер. Если номер задания не задан, предполагается текущее задание.

Перевод задания в фоновый режим

С помощью команды **bg** можно возобновить в фоновом режиме работу приостановленного или остановленного задания. Для этого нужно указать соответствующий номер задания, после чего оно перейдет в фоновый режим, и будет работать до своего завершения, или пока ему снова не потребуется ввод с терминала.

https://www.youtube.com/watch?v=5RqzpQ9ZlQ4

Управление процессами UNIX

- Первым делом научимся определять, какие процессы в системе запущены. Для этого в Linux (как и во всех UNIX-системах) имеется команда **ps**. Если ее запустить без всяких параметров, то она выдает список процессов, запущенных в текущей сессии. Если вы хотите увидеть список всех процессов, запущенных в системе, надо задать ту же команду с параметром **-ax**.

Управление процессами UNIX

Самые полезные параметры команды `ps`

Параметр	Описание
`-a`	Отображает информацию о процессах всех пользователей, а не только о собственных
`-c`	Вместо полной команды в последнем столбце будет отображено только имя исполнимого файла
`-C`	Изменяет способ вычисления процессорного времени (на результат практически не влияет)
`-e`	Отображает окружение
`-G gid`	Отображает информацию о процессах, принадлежащих пользователям из группы с идентификатором gid (gid — это не имя группы, а число, идентификатор)
`-h`	Если запущено слишком много процессов, и все они не помещаются на одном экране, повторяет заголовок через каждую "страницу"
`-r`	Сортировка по использованию процессора, а не по идентификатору и номеру терминала
`-U пользователь`	Отображает процессы, принадлежащие определенному пользователю, например ps -U den
`-w`	"Широкий" вывод, используются 132 колонки
`-X`	Показывает только процессы, привязанные к терминалу (по умолчанию)
`-x`	Показывает процессы, не привязанные к терминалу (например, сетевые демоны)
`-l`	Показывает дополнительную информацию о процессе: идентификатор родительского процесса (PPID), приоритет, использование памяти и пр.
`-u`	Еще более полезный параметр: позволяет узнать, какой пользователь владеет процессом, просмотреть информацию об использовании оперативной памяти процессом и т. д. Полезен при следующем наборе параметров: -uax

Управление процессами UNIX

- Независимо от наличия опций этой группы команда ps выдает для каждого процесса отдельную строку, но содержимое этой строки может быть разным. В зависимости от заданных опций могут присутствовать следующие поля:
- USER — имя владельца процесса;
- PID — идентификатор процесса в системе;
- PPID — идентификатор родительского процесса;
- %CPU — доля времени центрального процессора (в процентах), выделенного данному процессу;
- %MEM — доля реальной памяти (в процентах), используемая данным процессом;
- VSZ — виртуальный размер процесса (в килобайтах);
- RSS — размер резидентного набора (количество 1K-страниц в памяти);
- STIME — время старта процесса;
- TTY — указание на терминал, с которого запущен процесс;
- S или STAT — статус процесса;
- PRI — приоритет планирования;
- NI — значение nice (см. описание команды nice ниже);
- TIME — сколько времени центрального процессора занял данный процесс;
- CMD или COMMAND — командная строка запуска программы, выполняемой данным процессом;

Управление процессами UNIX

- В поле **Статус процесса**, могут стоять следующие значения:
- R — выполнимый процесс, ожидающий только момента, когда планировщик задач выделит ему очередной квант времени;
- S — процесс "спит";
- D — процесс находится в состоянии подкачки на диске;
- T — остановленный процесс;
- Z — процесс-зомби.

Управление процессами UNIX

- **Команда *top***
- Команда ps позволяет сделать как бы "моментальный снимок" процессов, запущенных в системе. В отличие от ps команда **top** отображает состояние процессов и их активность. Содержимое окна обновляется каждые 5 секунд.
- <Shift>+<N> — сортировка по PID;
- <Shift>+<A> — сортировать процессы по возрасту;
- <Shift>+<P> — сортировать процессы по использованию ЦПУ;
- <Shift>+<M> — сортировать процессы по использованию памяти;
- <Shift>+<T> — сортировка по времени выполнения.
- <U> — показывает только пользовательские процессы (появится небольшое текстовое поле, и вы сможете ввести имя интересующего вас пользователя, для отображения процессов всех пользователей введите плюс: +);
- <H> — получить справку по остальным командам программы top

- **htop** аналог top, намного превосходящий по возможностям
- **iftop** информация об активных сетевых соединениях, скорость сетевой закачки/отдачи
- **iotop** информация о процессах выполняющих активные дисковые операции
- **htop**, **iftop**, **iotop** не входят в стандартный комплект дистрибутива Ubuntu, но легко устанавливаются из стандартных репозиториев.

Управление процессами UNIX

- **Сигналы и команда *kill***
- Сигналы — это средство, с помощью которого процессам можно передать сообщения о некоторых событиях в системе. Сами процессы тоже могут генерировать сигналы, с помощью которых они передают определенные сообщения ядру и другим процессам. С помощью сигналов можно осуществлять такие акции управления процессами, как приостановка процесса, запуск приостановленного процесса, завершение работы процесса. Всего в Linux существует 63 разных сигнала, их перечень можно посмотреть по команде

  [user]$ kill –l

- Для посылки сигналов из командного интерпретатора используется команда **kill**. Она имеет следующий синтаксис:

  [user]$ kill [-**сигнал**] **pid** ...

Kill –авайрийное завершение процесса

Проще всего вычислить PID процесса с помощью следующей команды:

[user]$ ps –ax | grep <имя>

Например, # ps –ax | grep firefox.

Так что, если вы знаете только имя процесса, гораздо удобнее использовать команду:

[user]$ killall <имя процесса>

•Параметр -u команды killall позволяет "убить" процессы, принадлежащие заданному пользователю, например:

[user]$ killall -u user

Управление процессами UNIX

- Программа nice — она позволяет запустить любую программу с указанным приоритетом. Ясно — чем выше приоритет, тем быстрее будет выполняться программа. Формат вызова команды следующий:

  nice -n <приоритет> команда аргументы

- Максимальный приоритет задается числом −20, а минимальный — числом 19. Приоритет по умолчанию равен 10. Если процесс уже запущен, тогда для изменения его приоритета можно использовать команду renice:

  renice –n <приоритет> -p PID

Редакторы типа vi

- Редактор vi может работать в трех режимах:

-основной (визуальный) режим — в нем и осуществляется редактирование текста;

-командный режим — в нем осуществляется ввод специальных команд для работы с текстом (если сравнивать vi с нормальным редактором, то этот режим ассоциируется с меню редактора, где есть команды типа **Сохранить**, **Выйти** и т. д.);

-режим просмотра — используется только для просмотра файла (если надумаете использовать этот режим, вспомните про команду less).

- После запуска редактора вы можете переключать режимы, но выбрать режим можно и при запуске редактора:

```
vi файл
vi -e файл
vi -R файл
```

Первая команда запускает vi и загружает файл. Вторая команда запускает vi в командном режиме и загружает файл. Третья команда — это режим просмотра файла. Если указанный файл не существует, то он будет создан. По умолчанию активируется именно командный режим, поэтому в ключе -e нет смысла. Возврат к режиму команд выполняется клавишей <Esc>

Редакторы типа vi

Первое, чему Вы должны научиться, это выходить из редактора. Команды, начинающиеся с символа ":", будут отображаться в нижней строке. Остальные выполняются "молча". Символ ":" вызывается SHIFT+:

Команда	Описание
:q!	Выход без сохранения
:w	Сохранить изменения
:w <файл>	Сохранить изменения под именем <файл>
:wq	Сохранить и выйти
:q	Выйти, если нет изменений
i	Перейти в режим вставки символов в позицию курсора
a	Перейти в режим вставки символов в позицию после курсора
o	Вставить строку после текущей
O	Вставить строку над текущей
x	Удалить символ в позицию курсора
dd	Удалить текущую строку
u	Отменить последнее действие

Редакторы типа vi

Нажатие клавиш <i>, <a> и других переключает редактор в режим вставки, когда набираемые символы трактуются именно как символы, а не как команды. Для переключения обратно в командный режим служит клавиша <Esc>. В некоторых случаях (например, когда вы пытаетесь передвинуть курсор левее первого символа в строке) переход в командный режим осуществляется автоматически.

$ vi file.txt

Далее нажмите клавишу <i>, чтобы переключиться в режим вставки. Добавьте текст, затем нажмите клавишу <Esc> и введите :wq. Прочтите файл и убедитесь в изменении с помощью $ cat file.txt

Если нажать не клавишу <i>, а клавишу <a>, то вы тоже перейдете в режим вставки, но с одним отличием — введенный текст будет вставляться не перед символом, в котором находится курсор, а после него. Также в режим вставки можно перейти нажатием клавиш <o> или <O>. В первом случае будет добавлена пустая строка после текущей строки, а во втором — перед текущей строкой, и весь дальнейший ввод будет восприниматься именно как ввод текста, а не команд.

Редакторы типа vi

Чтобы удалить символ, нужно перейти в режим команд и над удаляемым символом нажать клавишу <x>. Клавиши <Backspace> и <Delete> тут не работают. Точнее,<Backspace> работает, но для удаления последней непрерывно введенной последовательности символов. Например, у нас есть текст "vi — текстовый редактор". Вы перейдете в режим вставки и измените текст так "vi — неудобный текстовый редактор". Нажатие клавиши <Backspace> удалит слово "неудобный", но не сможет удалить тире идругие символы. Чтобы удалить строку, в которой находится курсор, нужно нажать клавиши <d>+<d>. Помните, что vi считает строкой не то, что вы видите на экране, а последовательность символов до первого символа новой строки (\n). Если строка длиннее 80 символов, то она переносится на две экранных строки и визуально выглядит как две строки, а не как одна. Чтобы перейти в конец строки (клавиши <Home> и <End> тоже не работают, как вы успели заметить, если уже запускали vi), нужно нажать клавишу <$>. При навигации курсор перемещается не по экранным линиям, а как раз по строкам текста. Для отмены последней операции служит клавиша <u> Вот только истории изменений нет, да и по клавише <u> отменяется вся предыдущая команда целиком. Например, вы создали файл, перешли в режим вставки (клавиша <i>) и ввели весь текст "Длинный длинный текст". Если вы нажмете клавишу <u>, то она отменит всю предыдущую команду, т. е. удалит весь введенный вами текст.

Удаленный доступ

- UNIX — многозадачная многопользовательская операционная система. Это означает, что в один момент с системой могут работать несколько пользователей, и каждый пользователь может запустить несколько приложений. При этом вы можете зайти в систему локально, а кто-то — удаленно, используя один из протоколов удаленного доступа (telnet (устарел и из-за незащищенности практически не используется), SSH) или по FTP. Для работы по SSH в системе должен бить запушен сервер SSH, по умолчанию он принимает запросы на порт 22. В разных unix системах может понадобится дополнительная настройка SSH сервера и фаервола после его установки. Все настройки SSH-сервера хранятся в одном-единственном файле — /etc/ssh/sshd_config, а настройки программы-клиента — в файле /etc/ssh/ssh_config.

Установка SSH сервера:

sudo apt-get install ssh

sudo service sshd start запуск сервера (при необходимости)

Работать с SSH-клиентом очень просто. Для подключения к удаленному компьютеру введите команду:

ssh [опции] <адрес_удаленного_компьютера>

Для подключение в качестве ssh клиента, будем использовать программу putty

Удаленный доступ

Опции программы ssh

Опция	Описание
`-c blowfish|3des|des`	Используется для выбора алгоритма шифрования, при условии, что применяется первая версия протокола SSH (об этом позже). Можно указать `blowfish`, `des` или `3des`
`-c шифр`	Задает список шифров, разделенных запятыми в порядке предпочтения. Опция используется для второй версии SSH. Можно указать `blowfish`, `twofish`, `arcfour`, `cast`, `des` и `3des`
`-f`	Переводит ssh в фоновый режим после аутентификации пользователя. Рекомендуется использовать для запуска программы X11. Например: `ssh -f server xterm`
`-l имя_пользователя`	Указывает имя пользователя, от имени которого нужно зарегистрироваться на удаленном компьютере. Опцию использовать не обязательно, поскольку удаленный компьютер и так запросит имя пользователя и пароль
`-p порт`	Определяет порт SSH-сервера (по умолчанию используется порт 22)
`-q`	"Тихий режим" — будут отображаться только сообщения о фатальных ошибках. Все прочие предупреждающие сообщения в стандартный выходной поток выводиться не будут
`-x`	Отключает перенаправление X11
`-X`	Задействовать перенаправление X11. Полезна при запуске X11-программ
`-1`	Использовать только первую версию протокола SSH
`-2`	Использовать только вторую версию протокола SSH. Вторая версия протокола более безопасна, поэтому при настройке SSH-сервера нужно использовать именно ее

Удаленный доступ

Для открытия соединения с любым FTP-сервером введите команду:

ftp <имя или адрес FTP-сервера>

Можно просто ввести команду ftp, а в ответ на приглашение

ftp>

ввести команду:

open <имя или адрес FTP-сервера>

Подключившись к серверу, вы можете ввести команду help, чтобы просмотреть список доступных команд. Для получения справки по той или иной команде введите help <имя_команды>

Команда	Описание
ls	Вывод содержимого каталога
get	Загрузить файл с сервера
put	Загрузить файл на сервер
mget	Получить несколько файлов с сервера. Допускается использование масок файлов, например, *.tgz
mput	Загрузить несколько файлов на сервер
cd	Изменить каталог
mkdir	Создать каталог
rmdir	Удалить пустой каталог
delete	Удалить файл

Дополнительная информация

- 20 приёмов работы в командной строке Linux, которые сэкономят уйму времени
- Самые полезные приёмы работы в командной строке Linux

СПАСИБО