Файлы и пользователи в LINUX

Учётные записи

Linux — система многопользовательская, а потому пользователь — ключевое понятие для организации всей системы доступа в Linux. Когда пользователь регистрируется в системе (проходит процедуру авторизации, например, вводя системное имя и пароль), он идентифицируется с **учётной записью**, в которой система хранит информацию о каждом пользователе: его системное имя и некоторые другие сведения, необходимые для работы с ним. Именно с учётными записями, а не с самими пользователями, и работает система. Ниже приведён список этих сведений.

Системное имя (user name)

Это то имя, которое вводит пользователь в ответ на приглашение login:. Оно может содержать только латинские буквы и знак «_». Это имя используется также в качестве имени учётной записи.

Идентификатор пользователя (UID)

Linux связывает **системное имя** с **идентификатором пользователя** в системе — **UID** (User ID). **UID** — это положительное целое число, по которому система и отслеживает пользователей. Обычно это число выбирается автоматически при регистрации учётной записи, однако оно не может быть совершенно произвольным. В Linux есть некоторые соглашения относительно того, каким типам пользователей могут быть выданы идентификаторы из того или иного диапазона. В частности, **UID** от «0» до «100» зарезервированы для **псевдопользователей.**

Идентификатор группы (GID)

Кроме идентификационного номера пользователя с учётной записью связан **идентификатор группы**. **Группы пользователей** применяются для организации доступа нескольких пользователей к некоторым ресурсам. У группы, так же, как и у пользователя, есть имя и идентификационный номер — **GID** (Group ID). В Linux каждый пользователь должен принадлежать как минимум к одной группе — **группе по умолчанию.** При создании учётной записи пользователя обычно создаётся и группа, имя которой совпадает с **системным именем**, именно эта группа будет использоваться как группа по умолчанию для этого пользователя. Пользователь может входить более чем в одну группу, но в учётной записи указывается только номер группы по умолчанию. Группы позволяют регулировать доступ нескольких пользователей к различным ресурсам.

Полное имя (full name)

Помимо **системного имени** в учётной записи содержится и **полное имя** (имя и фамилия) использующего данную учётную запись человека. Конечно, пользователь может указать что угодно в качестве своего имени и фамилии. Полное имя необходимо не столько системе, сколько людям — чтобы иметь возможность определить, кому принадлежит учётная запись.

Домашний каталог (home directory)

Файлы всех пользователей в Linux хранятся раздельно, у каждого пользователя есть собственный **домашний каталог**, в котором он может хранить свои данные. Доступ других пользователей к домашнему каталогу пользователя может быть ограничен. Информация о домашнем каталоге обязательно должна присутствовать в учётной записи, потому что именно с него начинает работу пользователь, зарегистрировавшийся в системе.

Начальная оболочка (login shell)

Важнейший способ взаимодействовать с системой Linux — **командная строка**, которая позволяет пользователю вести «диалог» с системой: передавать ей команды и получать её ответы. Для этой цели служит специальная программа — **командная оболочка** (или **интерпретатор командной строки**), по-английски — **shell.** Начальная оболочка (login shell) запускается при входе пользователя в систему в текстовом режиме (например, на виртуальной консоли). Поскольку в Linux доступно несколько разных командных оболочек, в учётной записи указано, какую из командных оболочек нужно запустить для данного пользователя. Если специально не указывать начальную оболочку при создании учётной записи, она будет назначена по умолчанию, вероятнее всего это будет **bash**.

Зачем необходимо знание языка Shell?

Shell-скрипты очень хорошо подходят для быстрого создания прототипов сложных приложений, даже не смотря на ограниченный набор языковых конструкций и определенную

"медлительность". Такая метода позволяет детально проработать структуру будущего приложения, обнаружить возможные "ловушки" и лишь затем приступить к кодированию на C, C++, Java, или Perl.

Скрипты возвращают нас к классической философии UNIX -- "разделяй и влавствуй" т.е. разделение сложного проекта на ряд простых подзадач. Многие считают такой подход наилучшим или, по меньшей мере, наиболее эстетичным способом решения возникающих проблем, нежели использование нового поколения языков -- "все-в-одном", таких как Perl.

Для каких задач неприменимы скрипты

- для ресурсоемких задач, особенно когда важна скорость исполнения (поиск, сортировка и т.п.)
- для задач, связанных с выполнением математических вычислений, особенно это касается вычислений с плавающей запятой, вычислений с повышенной точностью, комплексных чисел (для таких задач лучше использовать C++ или FORTRAN)
- для кросс-платформенного программирования (для этого лучше подходит язык C)
- для сложных приложений, когда структурирование является жизненной необходимостью (контроль за типами переменных, прототипами функций и т.п.)
- для целевых задач, от которых может зависеть успех предприятия.
- когда во главу угла поставлена безопасность системы, когда необходимо обеспечить целостность системы и защитить ее от вторжения, взлома и вандализма.
- для проектов, содержащих компоненты, очень тесно взаимодействующие между собой.
- для задач, выполняющих огромный объем работ с файлами
- для задач, работающих с многомерными массивами
- когда необходимо работать со структурами данных, такими как связанные списки или деревья
- когда необходимо предоставить графический интерфейс с пользователем (GUI)
- когда необходим прямой доступ к аппаратуре компьютера
- когда необходимо выполнять обмен через порты ввода-вывода или сокеты
- когда необходимо использовать внешние библиотеки
- для проприетарных, "закрытых" программ (скрипты представляют из себя исходные тексты программ, доступные для всеобщего обозрения)
- Если выполняется хотя бы одно из вышеперечисленных условий, то вам лучше обратиться к более мощным скриптовым языкам программирования, например Perl, Tcl, Python, Ruby или к высокоуровневым компилирующим языкам -- C, C++ или Java. Но даже в этом случае, создание прототипа приложения на языке shell может существенно облегчить разработку.

Основы работы в bash-е

Название BASH - это аббревиатура от "Bourne-Again Shell" и игра слов от, ставшего уже классикой, "Bourne Shell" Стефена Бурна (Stephen Bourne). Bash - это стандартный интерпретатор команд на большинстве линукс систем. В его обязанности входит обработка и исполнение команд с помощью которых пользователь управляет компьютером. После того как вы завершили работу, можно завершить процесс командного интерпретатора. После нажатия клавиш **Ctrl**–**D**, команд **exit** или **logout** процесс командного интерпретатора будет завершен и на экране снова появится приглашение ввести имя пользователя и пароль.

Использование «cd»

Давайте начнем использовать bash для навигации по файловой системе. Для начала напечатайте следующую команду:

```
$ cd /
```

Этой командой мы указали bash-у что хотим переместиться в корневую директорию — /. Все директории в системе организованы в древовидную структуру и / это её начало (или корень). Команда **cd** служит для изменения текущей рабочей директории.

Пути

Чтобы узнать в каком месте файловой системы в данный момент вы находитесь (текущую рабочую директорию) наберите:

\$ **pwd** /

В приведенном выше примере / — аргумент команды **cd** — называется *путь*. Это место файловой системы, куда мы хотим переместиться. В данном случае / — абсолютный путь, это значит что путь указан относительно корневой директории.

Абсолютные пути

Вот несколько примеров абсолютных путей

- /dev
- /usr
- /usr/bin
- /usr/local/bin

Как вы уже могли заметить, все эти пути объединяет то, что они начинаются с /. Указывая путь /usr/local/bin в качестве аргумента команде **cd** мы говорим ей перейти в корневую директорию /, затем в директорию usr, потом в local и bin. Абсолютные пути всегда начинаются с /

Относительные пути

Второй вид путей называется относительными. **Bash**, команда **cd** и другие команды отсчитывают эти пути относительно текущей директории. Относительные пути никогда не начинаются с /. Например, если мы находимся в /usr

\$ **cd /usr**

Затем мы можем перейти в /usr/local/bin используя относительный путь

- \$ **cd local/bin**
- \$ **pwd**
- /usr/local/bin

Файловая система Linux

Файловая система - часть операционной системы, которая обеспечивает чтение и запись файлов на дисковых носителях информации. Файловая система устанавливает физическую и логическую структуру файлов, правила их создания и управления ими, а также сопутствующие данные файла и идентификацию. Конкретная файловая система определяет размер имени файла, максимальный возможный размер файла. Операционная Система Linux поддерживает множество файловых систем, в том числе и используемые в **Windows** файловые системы **FAT**, **FAT32**, **NTFS**, но при установке **ОС Linux** рекомендуем выбрать родную систему **Extfs**, **Ext2**, **Ext3**, **Ext4**, **ReiserFS**, **XFS**.

В Linux и Unix все - файл. Директории, устройства, сами файлы - все это ФАЙЛЫ. Устройства определяются узлами (Minor node и Major node), но при этом они остаются файлами.

Файловые системы Linux и Unix организованы в виде древовидной иерархической структуры. Самый верхний уровень файловой системы это / или корневой каталог. Все остальные файлы и каталоги находятся в корневом каталоге. Например, /home/jebediah/cheeses.odt показывает правильный полный путь к файлу cheeses.odt, который находится в каталоге jebediah, который находится в каталоге home, который, в свою очередь, находится в корневом каталоге.

В корневом каталоге находится набор важных каталогов, общих для большинства систем Linux. Вот список основных каталогов, которые находятся непосредственно в корневом (/) каталоге:

- /bin - важные *бинарные* (исполняемые) приложения
- /boot - *загрузочные* конфигурационные файлы, ядра и другие файлы, необходимые во время *загрузки* операционной системы
- /dev - файлы *устройств*
- /etc - конфигурационные файлы, стартовые сценарии
- /home - *домашние* каталоги пользователей
- /initrd - используется при создании частного *initrd* процесса загрузки
- /lib - системные *библиотеки*
- /lost+found - предоставляет систему *потерян+найден* для файлов в корневом (/) каталоге
- filename>/media
- /mnt - файловые системы на вашем жестком диске, примонтированные вручную
- /opt - каталог для установки *дополнительных* приложений
- /proc - специальный динамический каталог, содержащий информацию о состоянии системы, включая *процессы*, исполняемые в данный момент
- /root - домашний каталог пользователя *root*, произносится "слэш-рут"
- /sbin - важные *с*истемные *бин*арные приложения

- /srv - может содержать файлы веб-сервера, ftp-сервера и др.
- /sys - *системные* файлы
- /tmp - временные файлы
- /usr - приложения и файлы, наиболее доступные всем пользователям
- /var - изменяемые файлы, такие как логи и базы данных
- Пользователи и группы пользователей
- **UNIX**-подобные операционные системы являются многопользовательскими, предоставляя возможно одновременной работы нескольких пользователей в пределах одной операционной системы. Пользователи Linux могут иметь персональные каталоги с файлами, запускать определенные процессы в системе и выполнять различные действия отдельно от других пользователей.
- Пользователи Linux бывают двух типов — суперпользователь которому позволено делать в системе все и обычный. Первый обладает неограниченными правами, поэтому от имени суперпользователя(root) почти всегда работает администратор, выполняя настройку, установку программного обеспечения и другие системные операции.
- Каждый пользователь имеет определенные параметры, так система может понять какой процесс кто запустил и еще много другой информации.
- Информация о пользователях в системе **Ubuntu** хранится в файле — **/etc/passwd.** Если открыть этот файл редактором, то вы увидите, что информация о пользователе хранится в одной строке с набором определенных параметров разделенных знаком — «**:**«.
- Регистрационное имя(Login);
- Зашифрованный пароль или заполнитель пароля(вместо пароля будет записан символ x);
- Идентификатор пользователя(UID);
- Идентификатор группы по умолчанию(GID);
- Так называемое поле GECOS или комментарий, в котором указывают информацию о пользователе;
- Домашний каталог;
- Оболочка;

ПРАВА ДОСТУПА К ФАЙЛАМ В LINUX

В операционной системе Linux есть много отличных функций безопасности, но она из самых важных - это система прав доступа к файлам. Linux, как последователь идеологии ядра Linux в отличие от Windows, изначально проектировался как многопользовательская система, поэтому права доступа к файлам в linux продуманы очень хорошо.

ОСНОВНЫЕ ПРАВА ДОСТУПА К ФАЙЛАМ В LINUX

- **Чтение** - разрешает получать содержимое файла, но на запись нет. Для каталога позволяет получить список файлов и каталогов, расположенных в нем;
- **Запись** - разрешает записывать новые данные в файл или изменять существующие, а также позволяет создавать и изменять файлы и каталоги;
- **Выполнение** - вы не можете выполнить программу, если у нее нет флага выполнения. Этот атрибут устанавливается для всех программ и скриптов, именно с помощью него система может понять, что этот файл нужно запускать как программу.

Но все эти права были бы бессмысленными, если бы применялись сразу для всех пользователей. Поэтому каждый файл имеет три категории пользователей, для которых можно устанавливать различные сочетания прав доступа:

- **Владелец** - набор прав для владельца файла, пользователя, который его создал или сейчас установлен его владельцем. Обычно владелец имеет все права, чтение, запись и выполнение.
- **Группа** - любая группа пользователей, существующая в системе и привязанная к файлу. Но это может быть только одна группа и обычно это группа владельца, хотя для файла можно назначить и другую группу.
- **Остальные** - все пользователи, кроме владельца и пользователей, входящих в группу файла.

СПЕЦИАЛЬНЫЕ ПРАВА ДОСТУПА К ФАЙЛАМ В LINUX

- **SUID** - если этот бит установлен, то при выполнении программы, id пользователя, от которого она запущена заменяется на id владельца файла. Фактически, это позволяет обычным пользователям запускать программы от имени суперпользователя;

- **SGID** - этот флаг работает аналогичным образом, только разница в том, что пользователь считается членом группы, с которой связан файл, а не групп, к которым он действительно принадлежит. Если SGID флаг установлен на каталог, все файлы, созданные в нем, будут связаны с группой каталога, а не пользователя. Такое поведение используется для организации общих папок;
- **Sticky-bit** - этот бит тоже используется для создания общих папок. Если он установлен, то пользователи могут только создавать, читать и выполнять файлы, но не могут удалять файлы, принадлежащие другим пользователям

Как посмотреть и изменить права доступа

Чтобы узнать права на файл linux выполните такую команду, в папке где находится этот файл:

ls -l

Чтобы изменить права на файл в linux вы можете использовать утилиту chmod. Она позволяет менять все флаги, включая специальные.

$ chmod опции категориядействиефлаг файл

ВЫВОДЫ

Вот и все, теперь вы знаете не только что такое права доступа к файлам в linux, но и как их посмотреть, и даже как их изменить. Это очень важная тема, в которой действительно стоит разобраться новичкам, чтобы использовать свою систему более полноценно.