Лекция 2. Работа с файлами в терминале

Цель лекции

1. Познакомимся с оболочкой bash
2. Научимся работать с файлами
3. Изучим типы файлов
4. Научимся работать с текстовыми редакторами
5. Узнаем, как можно получать справку по командам

План лекции

1. Что такое оболочка?
2. Приглашение оболочки (bash)
3. Структура файловой системы в Linux, домашняя директория
4. Специальные файловые системы. Принцип “всё есть файл”
5. Типы файлов
6. Работа с файлами: ls, pwd, cd, mkdir, cp, rm, mv, touch, cat
7. Относительные и абсолютные пути
8. Жесткие ссылки
9. Символические ссылки
10. Текстовые редакторы vim, mcedit, nano
11. Терминальные пейджеры less и more
12. Просмотр начала и конца файла: head и tail

Текст лекции

Что такое оболочка?

Начнём с важного вопроса: что же такое на самом деле “оболочка” и зачем она нужна. Оболочка (shell) – это промежуточное звено между пользователем и операционной системой, основное назначение – интерпретация команд и передача результатов их выполнения пользователю. Мы вводим команды в оболочке и ОС их выполняет, возвращая некий результат. Так как мы работаем в текстовом интерфейсе, то команды и их результат также являются текстом.
В нашей системе Linux чаще всего встречается оболочка bash (Bourne again shell), хотя существуют и другие: zsh, sh и т.д. Bash это продвинутая версия классической оболочки sh (Bourne shell), которая использовалась еще в UNIX-системах с 1977 года. Зная особенности работы с bash, мы сможем эффективно решать задачи в любой Linux-системе. Перейдём к практике.

Приглашение оболочки (bash)

Первое, что мы видим в терминале после логина – приглашение оболочки bash. Стандартная настройка в Ubuntu включает в себя следующие элементы:

- логин (имя) пользователя;
- хост (имя компьютера);
- текущий каталог (~ – это домашняя директория);
- указатель типа пользователя (обычный – $ или root – #).

Все эти элементы важны при работе в системе. Мы понимаем, от имени какого пользователя будет выполнена команда, на какой системе мы работаем, какой текущий каталог и являемся мы администратором или нет.
Формат приглашения bash можно настроить для себя индивидуально, например, включить дополнительные элементы или функциональность.
Теперь пора разобраться, что такое домашняя директория, в которой мы оказались сразу после логина в терминале.

Структура файловой системы в Linux, домашняя директория

Как мы говорили на первой лекции, структура файловой системы Linux построена на базе стандарта. В нем описаны функции всех основных каталогов. Всё начинается с корня (**/**), это верхний уровень иерархии каталогов в Linux. Попасть в него можно с помощью команды **cd** (change directory): **cd /**.
Сейчас нас интересует каталог **/home**, именно в нём положено создавать домашние каталоги пользователей.
Наш домашний каталог находится по адресу **/home/db**, где **db** – это логин нашего пользователя. Именно этот каталог отображается как тильда в приглашении **bash**.

В домашнем каталоге находятся файлы пользователя: документы, рабочие файлы, настройки программ. В домашнем каталоге пользователь имеет полные права на управления файлами и не будет иметь проблем с доступом.
Важно понимать, что в домашнем каталоге хранятся настройки только для пользователя, системные настройки приложений находятся в каталоге **/etc**.
Домашний каталог содержит обычные файлы и директории, но существуют специальные файловые системы, в которых всё по-другому.

Специальные файловые системы. Принцип “всё есть файл”

Для упрощения доступа к данным и настройкам в Linux используется принцип “всё есть файл”. Это значит, что в виде файлов могут быть представлены не только привычные файлы на диске, но и структуры данных из ядра операционной системы, данные состояния ОС и даже настройки системы или сетевые интерфейсы и устройства.

Например, если мы зайдём в директорию **/proc**, то увидим специальные файлы текущего состояния системы и информацию о процессах. То есть, на диске таких файлов нет, это всего лишь особая структура в оперативной памяти.
В директории **/dev** находится информация об устройствах, **/dev/sda** – это специальный файл, который отображается на наш первый диск в системе.
Благодаря такому подходу мы можем легко ссылаться на устройства, считывать и изменять настройки системы, пользуясь стандартными инструментами оболочки, текстовыми редакторами, командами, так же, как и с обычными файлами.
Реализация этого принципа требует внедрения специальных типов файлов, о которых мы сейчас и поговорим.

Типы файлов

Увидеть обозначение типа файла можно с помощью команды ls -l. Первый символ первого столбца будет символизировать тип файла.
Самый простой тип файла это **обычный файл (-)**. То есть область данных на диске, которая имеет имя и расположение в глобальной структуре директорий.
Директория (d) это особый тип файла, который также вполне привычен. В директории содержатся данные о файлах, которые там расположены.
Файл блочного устройства (b) – это отображение диска или раздела на диске, адресация здесь может осуществляться только на уровне блоков (например, секторов диска). Такие файлы нужны для указания в качестве аргументов в утилитах разбивки диска или форматирования разделов.
Файл символьного устройства (c) – это устройство с посимвольной адресацией. Типичным примером здесь будет терминал – программы могут выводить текст такие устройства посимвольно.
Файл-сокет (s – UNIX-сокет) – специальный файл, который заменяет сетевое соединение. Например, две программы могут использовать такой сокет для обмена данными в рамках одной машины. В настройках подключения вместо IP-адреса и порта указывается путь к файлу-сокету.
Именованный канал (p – pipe) – специальный файл, который создан для обмена программ данными по принципу FIFO (first in, first out). Используется для передачи данных между программами.
Символическая ссылка (l – link) – указатель на другой файл или директорию, содержит путь к файлу или каталогу.
Некоторые типы файлов нам пока не очень интересны, о некоторых мы поговорим подробнее на этом занятии.
Пока научиться навигации в файловой системе и базовым командам работы с файлами.

Работа с файлами: ls, pwd, cd, mkdir, cp, rm, mv, touch, cat

Давайте определим, где мы находимся в иерархии файлов. Для этого отлично подходит команда **pwd**, сокращенно print working directory. Многие команды в Linux составлены из сокращений слов, так их легче запомнить и быстрее набирать. Кстати, если встречаете короткую команду (две или три буквы), скорее всего это важная и часто используемая утилита, её стоит запомнить.
Вторая полезная команда – **ls**, сокращенное list. ls позволяет показать список файлов в директории. Формат вывода может быть разным. Самые популярные параметры это **-l**, длинный вывод списка и **-a** – показ всех элементов, включая скрытые файлы (название которых начинается с точки). Эти параметры можно объединять: **ls -al**.
Чтобы перемещаться по дереву каталогов создана команда **cd** (change directory). У неё есть единственный параметр: путь. Подробнее о вариантах указания путей поговорим чуть позже.
Для создания директорий мы используем команду **mkdir (make directory)**. Если нужно сделать сразу несколько уровней, добавляем опцию **-p**.
Копированием файлов и директорий занимается команда **cp** (copy). В большинстве команд, где есть источник и назначения они указываются именно в таком порядке: откуда-куда. Два основных аргумента команды: источник (откуда) и назначение (куда). Если мы копируем директорию, нужно указать опцию **-r** (recursive). Без этой опции мы не сможем копировать директорию.
Удалением файлов и директорий заведует команда **rm** (remove). Для удаления без подтверждения мы можем использовать опцию **-f** (force, в Ubuntu она не нужна), рекурсивное удаление (включая вложенные элементы) – это опция **-r,** часто они используются вместе: **rm -rf**. Будьте осторожны при использовании этой команды, восстановить файлы после удаления будет сложно, никакой “корзины” здесь не будет.
Перенос (переименование) файлов это команда **mv** (move). Если источник и назначение находятся в рамках одной файловой системы, то операция мгновенна и по сути является переименованием объекта. А если вы переносите с одного диска на другой, тогда это будет полноценное копирование данных и удаление на источнике. Команда mv работает как cp, только здесь не нужен ключ для рекурсии, даже при работе с директориями.
Команда **touch** позволяет быстро создать пустой файл. Например, **touch test** создаст файл test в текущей директории. У этой команды есть еще множество функций, их можно посмотреть в справке. Напомню, что справку обычно можно вызвать через ключи **-h** или **--help**, а более подробное руководство через команду **man touch**.
Команда **cat** (catenate) также имеет множество применений. Например cat > testfile создаст новый файл и будет записывать текст из консоли в него. Закончить ввод можно сочетанием Ctrl+D. Если использовать несколько файлов в качестве источников, можно склеить их содержимое и сохранить в новый файл: **cat test test2 > test3**. В этих примерах используется перенаправление ввода-вывода (символ **>**), которое мы будем подробнее рассматривать в следующих лекциях. Можно просто распечатать файл в консоль: **cat testfile**.

Для применения всех перечисленных выше команд нужно корректно указывать пути к файлам и директориям. Что такое относительные и абсолютные пути, как их использовать мы разберемся далее.

Относительные и абсолютные пути

Для адресации файлов и директорий мы можем использовать либо относительные, либо абсолютные пути. Отличить их легко: абсолютные пути начинаются с корня (**/**), относительные – нет.
Абсолютные пути работают одинаково, независимо от текущего положения в файловой иерархии, относительные зависят от текущей директории (проверить её можно командой **pwd**).
Единого совета, какой тип пути использовать, не существует. Если вы хотите написать максимально надёжную команду, которая будет гарантированно выполнять точные действия, стоит использовать абсолютный путь. Если вы хотите сделать универсальную команду, которая работает в контексте текущей директории и не зависит от вышестоящей иерархии директорий, можно использовать относительные пути.
Например, команда **cd home** зависит от текущей директории и сменит нашу директорию на вложенную home, если такая есть. Это относительный путь.
Наоборот, команда **cd /home** будет работать всегда одинаково и перемещает нас в директорию home в корне файловой иерархии.
Особенно важно учитывать разницу относительных и абсолютных путей при написании скриптов и создании символических ссылок, к которым мы скоро подойдём.

Жесткие ссылки

Начнём изучение ссылок со сложного – жёстких ссылок. По сути, это всего лишь еще одно имя файла. То есть, во многих файловых системах (ext2/3/4/, xfs и других) есть логический элемент индексный дескриптор (inode), в котором хранятся все метаданные файла (время создания, изменения, права доступа, владелец, указатель на область с данными и т.д.) У каждого inode есть уникальный в рамках файловой системы номер (например, 123456). Но мы привыкли обращаться к файлам по имени, поэтому в директории создаётся соответствие между inode и именем файла. Именно такое соответствие и называется **жёсткой ссылкой** (hard link).
При создании файла мы автоматически создаём новый inode и жёсткую ссылку (имя) для него.
Создать жёсткую ссылку можно командой **ln**: **ln file1 file_ln**. При этом мы создали 2 имени файла для одного и того же inode. То есть оба этих имени равнозначны. Пока существует хотя бы одна жёсткая ссылка на inode, он считается занятым и сохраняется на диске, то же относится и к содержимому файла, которое связано с

этим inode. Чтобы удалить файл окончательно, нужно удалить все жесткие ссылки на его inode.
У жёстких ссылок есть ограничения: мы можем делать их только на файлы и только в рамках одной файловой системы (раздела диска). Единственное исключение – жёсткие ссылки “.” и “..” в директориях, которые ведут на текущую и вышестоящую директории.

Область применения жестких ссылок достаточно узкая: резервные копии и каталогизация большого массива файлов.

Символические ссылки

Гораздо проще работают символические ссылки. Символическая ссылка – это файл специального типа, который содержит в себе адрес на файл-источник. То есть, кроме пути файла, на который она ссылается, там ничего нет.
Отметим разницу с жесткими ссылками. Жесткая ссылка это всего лишь еще одна запись соответствия имя файла - **inode** в файле директории. Символическая ссылка это полноценный отдельный файл, который сам по себе занимает inode в файловой системе и имеет своё содержимое (путь к файлу или директории).
Создать символическую ссылку просто: **ln -s file1 file_lns**. Здесь уже важно, какой тип пути вы будете использовать. Если для файла-источника использовать абсолютный путь, то эта ссылка будет работать из любого места в файловой иерархии системы. Например, **ln -s /home/db/testfile test_lns**.
Если использовать относительный путь, то ссылка будет работать с учетом текущей директории (где она находится).
Перейдём к следующей важной теме – редактированию файлов в терминале.

Текстовые редакторы vim, mcedit, nano

Для работы в Linux очень важно уметь работать в консольных текстовых редакторах. Конфигурационные файлы, журналы, исходный текст программ – все это нужно редактировать, часто именно в терминале.
Особенность работы с текстом в консоли в том, что необходимо уметь работать с различными текстовыми редакторами. Мы не можем заранее знать, какой редактор будет установлен в системе и не всегда будут права на установку своего любимого варианта. Поэтому мы рассмотрим три наиболее популярных редактора, научимся выполнять базовые действия. Для ежедневной работы вы можете выбрать любой подходящий вам редактор.
Сначала рассмотрим самый нестандартный редактор **vim**, который является развитием редактора **vi**. В ubuntu его можно установить командой **apt install vim**.
Любой редактор в качестве параметра принимает путь к файлу для редактирования. Обычно, если файла такого нет, то при сохранении он будет создан. Открываем файл: **vim ~/.bashrc**. Если мы попробуем набрать текст или создать новые строки, то быстро

поймём, что редактор не делает ожидаемых изменений в файле. Дело в том, что редактор находится в командном режиме, а не в режиме редактирования. Для перехода в обычный режим редактирования нужно нажать Insert или I. После этого редактируем файл как обычно.
Далее, для выхода из редактора или выхода с сохранением нужно выйти обратно в командный режим: клавиша Esc. После этого набираем двоеточие (:) вводим команду: wq. Здесь q (quit) - выход из редактора, w (write) - запись файла.
Основные варианты команд:

- q - выйти из редактора (если не было изменений в файле);
- w – записать изменения в файл;
- wq - выйти с сохранением результата редактирования в файл;
- q! – выйти без сохранения файла.

Для более подробного изучения редактора vim можно использовать программу vimtutor.
Второй редактор **nano** – стандартный редактор в Debian-дистрибутивах (включая Ubuntu). Он проще в работе, сочетания клавиш для работы с ним перечислены в нижней части окна. Знак крышки (^) обозначает Ctrl, знак M – Alt. Например, для получения справки нужно нажать Ctrl+G. Сохранение файла Ctrl+O, также при выходе из редактора он спросит, нужно ли сохранить файл и какое имя использовать. Еще одна полезная возможность: включение номеров строк: Alt+N.
Еще один популярный редактор – **mcedit**, входит в пакет mc (midnight commander). Можно настроить его как редактор по умолчанию в **mc** (редактор открывается клавишей F4). Интерфейс **mcedit** отличается от других, больше похож на **mc**. Быстрые действия в основном используют клавиши Fx, обширное меню можно вызвать через F9. В этом редакторе есть особенности работы с буфером обмена. Вставить содержимое буфера можно как обычно, а вот скопировать в буфер уже не получится. Для выделения и копирования в mcedit используются блоки – начало и конец выделения - F3. Если нужно сохранить выделенный блок во временный файл – Ctrl+F, вставить его можно через Shift+F5.
Выйти из редактора mcedit можно через двойное нажатие Esc или по кнопке F10. При наличии несохранённых изменений появится диалог о сохранении.
Помимо редактирование текста, часто нужно просто посмотреть большой файл. Можно это сделать через известную команду cat, но это не всегда удобно. Здесь на помощь приходят пейджеры.

Терминальные пейджеры less и more

Пейджеры дают возможность просмотра больших текстовых файлов, с возможностью удобной навигации и поиска.
Если есть возможность, лучше использовать пейджер **less**. Он более современный и удобный. Открыть файл через него просто: **less file**. Прокрутка осуществляется кнопками PgUp PgDown или курсором. Поиск вперёд по файлу – / и шаблон поиска, назад – ? и шаблон. Вызвать справку можно клавишей H. Выход из less – q.

Через **less** можно просматривать не только файлы, но и вывод команд. Например: **ps afx | less**. Здесь мы используем команду вывода дерева процессов и конвейер, с которыми подробнее будем разбиться на будущих занятиях.
Второй вариант пейджера **more** – менее удобный. Прокрутка вперёд – пробел или Enter, назад - B. Также есть поиск (/) и справка (H). Полезно получить хотя бы минимальный опыт работы с ним, так как иногда в системе может не оказаться less.
Итак, мы научились просматривать большие файлы в консоли. Но что, если нам нужно только начало или конец файла? Здесь мы можем использовать команды **head** и **tail**.

Просмотр начала и конца файла: head и tail

Для просмотра начала файла можно использовать команду head. Она принимает в качестве параметра путь к файлу, например: **head /var/log/syslog**. Также можно указать количество строк (параметр **-n**), которые мы хотим посмотреть с начала файла: **head -n 20 /var/log/syslog**. Как и пейджеры, head можно использовать в конвейере: **ps afx | head**. Эта команда перенаправляет вывод списка процессов в программу **head**.
Гораздо чаще нам требуется посмотреть окончание файла, например при поиске последних сообщений в файле журнала. Для этого мы используем команду **tail: tail /var/log/syslog**. Также можно задать количество строк: **tail -n 30**. Еще один интересный режим - follow, включается параметром **-f**. В этом режиме команда не завершается, а продолжает работать, отслеживая изменения файле. Если в файле появляются новые строки, мы сразу же увидим их в терминале. Такой режим очень полезен при отладке какой-то проблемы, чтобы сразу видеть сообщения об ошибках. Завершить **tail -f** можно комбинацией клавиш Ctrl+C.

Выводы

- На этой лекции мы освоили основы работы в терминале.
- Разобрались с назначением различных типов файлов.
- Поработали с файлами в консоли.
- Изучили различие символических и жестких ссылок.
- Попрактировались в работе с текстовыми редакторами.
- Научились быстро просматривать текстовые файлы в терминале.