Евгений Коноплев

Базовый курс Linux

Печатное руководство к видеокурсу

СОДЕРЖАНИЕ

Модуль 1. Общие сведения об ОС Linux

Модуль 2. Установка ОС Linux

Модуль 3. Базовое понимание Linux

Модуль 4. Файловая система

Модуль 5. Терминал и командный интерпретатор BASH

Модуль 6. Пользователи и группы

Модуль 7. Управление процессами

Модуль 8. Управление программным обеспечением

Модуль 9. Текстовый редактор VIM

Модуль 10. Базовая настройка Linux

Модуль 11. Сетевая файловая система

Модуль 12. Справочная система

Модуль 13. Работа в ОС Linux

Модуль 14. Мониторинг системы

Модуль 15. Ядро операционной системы

Модуль 16. Итоговая информация по Linux

Модуль 1. Общие сведения об ОС Linux

Open Source

Термин Open Source появился в 1998 году, благодаря Эрику Реймонду и Брюсу Перенсу, основателям OSI (Open Source Initiative), организации занимающейся продвижением Open Source.

Open Source – это программное обеспечение с открытым исходным кодом и доступным для всех:

– свобода в распространение программного обеспечения и исходного кода;

– возможность модификации кода;

– целостность авторского кода.

GNU Project

В 1984 году Ричард Столлман уволился из Массачусетского технологического института чтобы посвятить себя написанию свободной операционной системы – GNU system (GNU – это рекурсивный акроним от англ.– **G**NU`s **N**ot **U**nix – "GNU – не UNIX!"). В рамках проекта GNU было создано многое программное обеспечение (tar, sed, make, bash, …) и библиотеки. Но ядро операционной системы так и не было завершено.

В 1991 году финский программист Линус Торвальдс выкладывает первую версию Linux-ядра в общий доступ. Открытость исходных кодов позволила проекту GNU использовать его в своей операционной системе, так как на тот момент своего ядра у ней не было. Так появился GNU/Linux – набор программного обеспечения проекта GNU и ядра Линуса Торвальдса. Однако название "Linux" получило большее распространение. На данный момент Linux является самой распространенной бесплатной операционной системой. Существует большое количество версий Linux – так называемых дистрибутивов.

Принципы Linux

1. Все в системе представлено в виде файлов;

2. Каждая программа в Linux решает узкоспециализированную задачу;

3. Возможность объединения программ в цепочку для выполнения сложных задач;

4. Конфигурационные файлы представлены в текстовом формате;

5. Два вида объектов в Linux: файлы и процессы.

Особенности Linux

1. Быстродействие – Linux имеет гибкие настройки и отключение не нужных компонентов (например, X Window) позволяет запускать операционную систему на оборудование с небольшой производительностью;

2. Надежность – Linux не восприимчив к компьютерным вирусам. Система надежно защищена от обычных пользователей. Именно из под обычного пользователя и рекомендуется работать в системе если не требуется что-то большее;

3. Многозадачность – одновременно могут выполняться сотни программ;

4. Многопользовательский режим – в одной и той же системе одновременно могут работать десятки пользователей;

5. Виртуальные консоли – каждый пользователь одновременно может иметь несколько сеансов работы в системе. Переключение между сеансами осуществляется с помощью клавиатуры;

6. Защита памяти процесса – зависший процесс не может вызвать зависания всей системы;

7. Полная поддержка сетей TCP/IP;

8. Наличие средств для взаимодействия с ОС Windows:

– Возможность подключения разделов DOS, FAT, NTFS;

– SAMBA для общего доступа к файлам и принтерам;

– Wine для запуска Windows-программ в Linux;

– Поддержка стека протоколов TCP/IP предоставляет доступ к стандартным протоколом обмена данными, такими как FTP.

9. Хорошая документация.

Область применения Linux

ОС Linux может применяться где угодно. По возможностям она не уступает ОС Windows, а как явный плюс – полностью бесплатна. Выбор дистрибутива

для решения ваших задач целиком зависит от вас. Но обычно выбирают тот дистрибутив, который лучше всего знают. Или который лучше всего знает сосед по лестничной площадке.

Дистрибутивы Linux

Существует большое количество дистрибутивов Linux (больше 100), как для массового пользователя, так и узкоспециализированных, например для использования на маршрутизаторах.

Дистрибутивы делятся на группы:

1. Основанные на Debian или использующие формат пакетов deb (Debian, Knoppix, Ubuntu…);

2. Дистрибутивы основанные на Ubuntu, который в свою очередь происходит от Debian (Kubuntu, Xubuntu, Mint…);

3. Дистрибутивы основанные на Red Hat или использующие формат пакетов RPM (Red Hat, CentOS, Fedora Core, Suse…);

4. Дистрибутивы основанные на Slackware (Zenwalk, MOPSLinux, VectorLinux…);

5. Дистрибутивы использующие другие пакетные системы (Gentoo, Arch Linux…).

Какой дистрибутив использовать – личное дело каждого.

Модуль 2. Установка ОС Linux

Существует несколько способов установки ОС Linux:

1. Установка с CD/DVD-диска. Это самый простой и распространенный способ;

2. Установка с раздела жесткого диска;

3. Установка по сети посредством FTP, HTTP или NFS.

Установка с CD/DVD-диска

Шаг 1

После того как вы выставите в BIOS загрузку с CD/DVD-диска, сохраните настройки и выйдите из BIOS, начнется процесс установки ОС Linux. Первым экраном будет предложение выбрать способ установки. Есть два режима установки ОС Linux:

1. Установка в графическом режиме. Для выбора этого режима установки нажмите **<ENTER>**;

2. Установка в текстовом режиме. Для выбора этого режима установки введите команду **linux text** и нажмите **<ENTER>**.

Мы будет производить установку в текстовом режиме, поэтому вводим команду linux text и нажимаем клавишу **<ENTER>**.

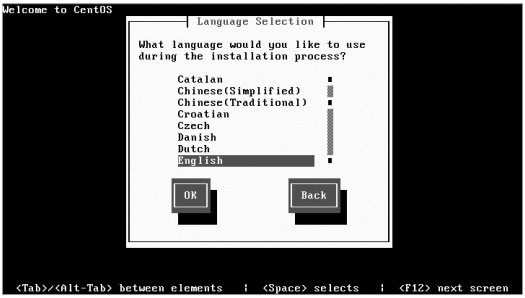
Шаг 2

В этом диалоговом окне нам предлагают протестировать компакт-диск на ошибки, прежде чем мы продолжим. Тестирование может занять продолжительное время, поэтому я обычно пропускаю его. Нажимаем кнопку **<Skip>** чтобы пропустить тестирование.

Шаг 3

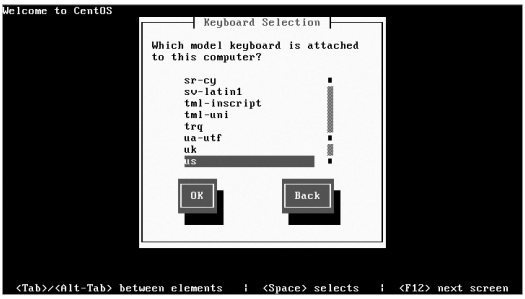
Программа инсталляции приветствует нас. Нажимаем **<OK>**.

Шаг 4



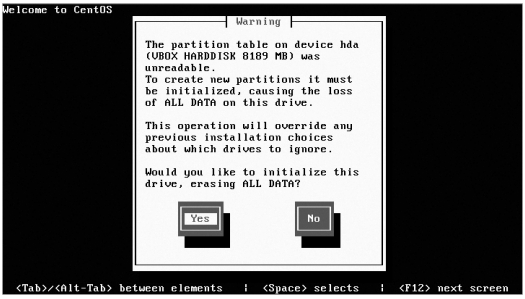
В этом диалоговом окне выбираем язык, на котором будет проходить процесс установки.

Шаг 5



В этом диалоговом окне нам предлагают выбрать модель клавиатуры. Оставьте по умолчанию если не знаете что выбрать.

Шаг 6



В этом диалоговом окне нам сообщают, что жесткий диск не имеет разделов и предлагают их создать.

Шаг 7

В этом диалоговом окне нам предлагают сделать выбор, каким образом разбивать жесткий диск на разделы. Варианты есть следующие:

Remove all partitions on selected drives and create default layout – удалить ВСЕ разделы на выбранном жестком диске и разбить его по умолчанию.

Remove linux partitions on selected drives and create default layout – удалить linux разделы на выбранном жестком диске и разбить его по умолчанию.

Use free space on selected drives and create default layout – использовать свободное место на выбранном жестком диске и разбить его по умолчанию.

Create custom layout – сделать ручное разбиение жесткого диска.

Разбиение по умолчанию спасет новичков от многих ошибок. В конкретном примере я выбрал второй вариант и нажал **<OK>**.

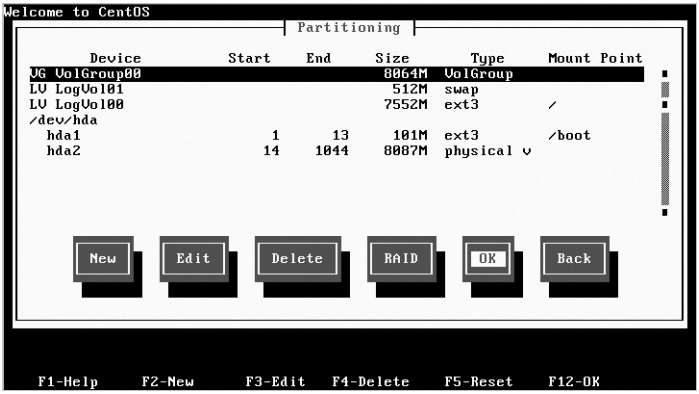
Шаг 8

Выводится предупреждение, что данные на выбранном жестком диске будут удалены. Убедитесь, что важные данные сохранены в надежном месте (если диск не был пустым) и нажимайте **<Yes>**.

Шаг 9

В данном диалоговом окне нам предлагают просмотреть и при желание модифицировать таблицу разделов. Нажимаем **<Yes>** чтобы вывести таблицу разделов.

Шаг 10



В данном диалоговом окне отображена таблица разделов по умолчанию. Именно такой она будет если мы не внесем изменений. Если вы новичок то определенно не стоит что либо здесь менять, так как значения по умолчанию очень даже хорошие. При ручном создание разделов нужно соблюдать несколько условий:

1. Раздел swap должен быть в два раза больше объема RAM, то есть если на компьютере 1 Гб оперативной памяти, swap будет 2 Гб. Это не обязательное условие, но очень рекомендуется.

2. В зависимости от назначения будущей системы, разбейте диск на разделы с некоторым запасом места.

Разбиение жесткого диска на несколько разделов имеет явные преимущества:

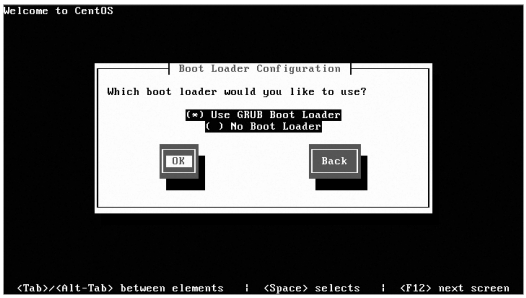
1. Защищенность пользовательских данных. Например каталог/home на отдельном разделе будет защищен при переустановке ОС;

2. Переполнение диска. Если какая либо программа начнет активно заполнять дисковое пространство то это будет происходить только в одном разделе. Достаточно часто происходит проблема когда все место в разделе заполняется и программы его использующие перестают функционировать правильно. Этого можно избежать если правильно разбить жесткий диск и регулярно следить за свободным местом;

3. На одном жестком диске можно установить несколько ОС.

После того как таблица разделов готова нажимаем **<OK>**.

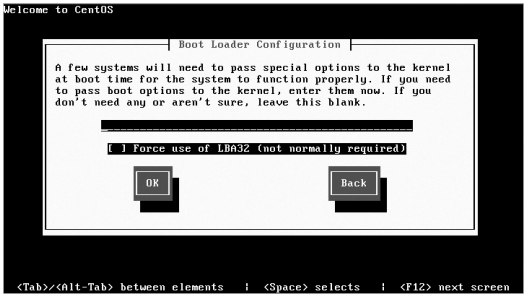
Шаг 11



На этом шаге необходимо сделать выбор – использовать загрузчик или нет. lilo ушел в историю и на его место пришел GRUB.

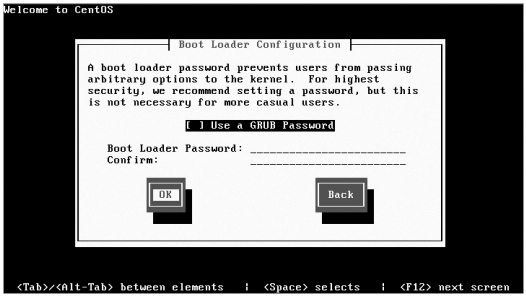
Выбираем пункт Use GRUB Boot Loader и нажимаем **<OK>**.

Шаг 12



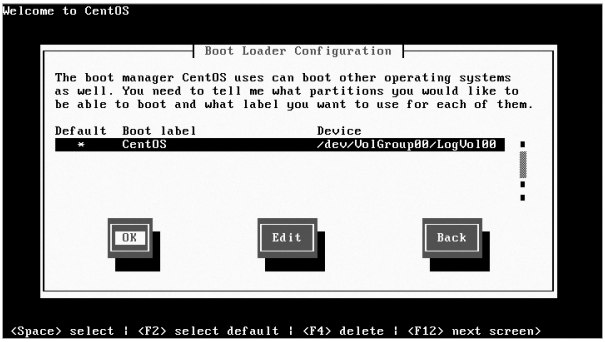
В этом диалоговом окне можно задать специальные опции ядра во время загрузки. Нажимаем **<OK>**.

Шаг 13



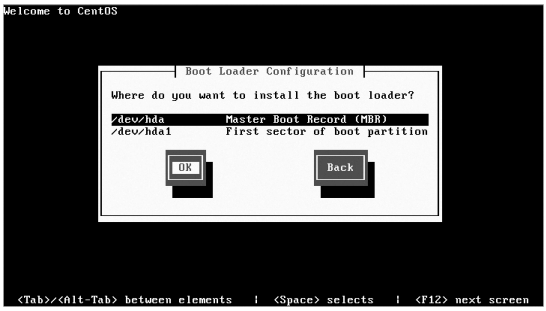
В этом диалоговом окне предлагают установить пароль на загрузчик. В большинстве случаев вам это не понадобится. Нажимаем **<OK>**.

Шаг 14



В этом диалоговом окне можно задать другие разделы для загрузки, в случае если на компьютере несколько ОС. Нажимаем **<OK>**.

Шаг 15

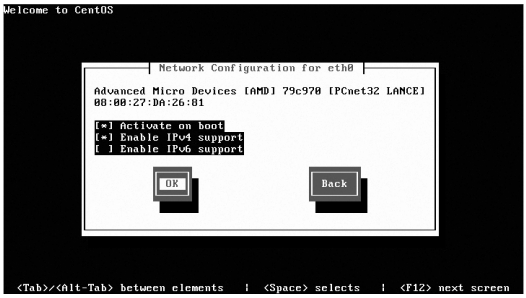


В этом диалоговом окне нас спрашивают куда устанавливать загрузчик. Мы выбираем Master Boot Record и нажимаем **<OK>**.

Шаг 16

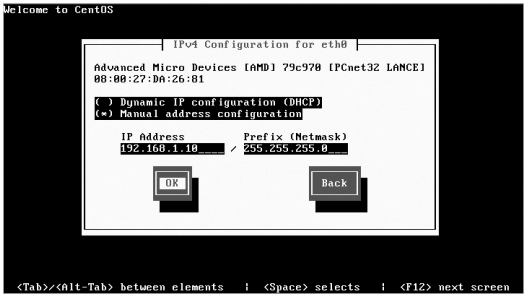
Нам предстоит решить – будем ли мы использовать сетевой интерфейс или нет. Думаю в 99 % случаев ответ будет положительным. Нажимаем **<Yes>**.

Шаг 17



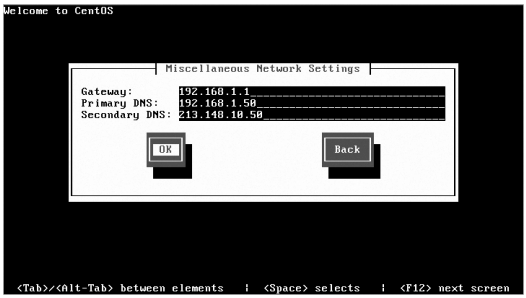
Выбрав пункт “Activate on boot” наш сетевой интерфейс будет включаться при загрузке ОС. Также выбираем пункт “Enable IPv4 support”, IPv6 пока не набрал обороты и он нам не понадобится. Нажимаем **<OK>**.

Шаг 18



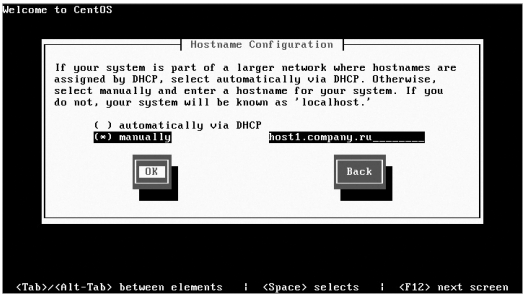
В данном диалоговом окне необходимо выбрать способ получения сетевых настроек. Это может быть динамически при загрузке ОС благодаря протоколу DHCP, или сетевые настройки могут быть прописаны вручную, как показано на рисунке.

Шаг 19



В данном диалоговом окне вводим шлюз (Gateway), первичный DNS-сервер (Primary DNS) и вторичный DNS-сервер (Secondary DNS).

Шаг 20

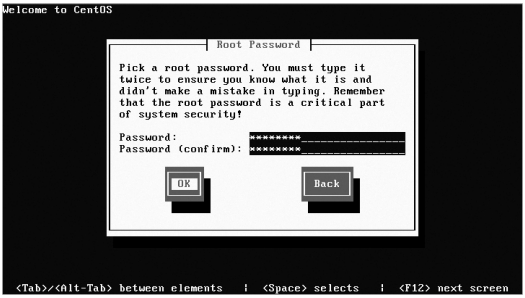


Указываем способ получения hostname (имени хоста). Это может быть динамически благодаря протоколу DHCP, или hostname может быть указан вручную.

Шаг 21

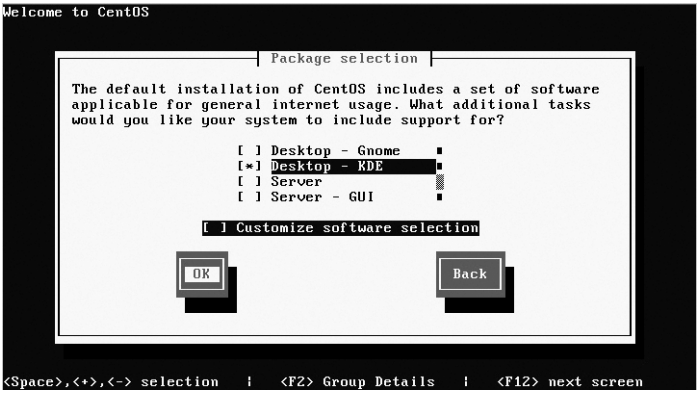
Выбираем timezone (временную зону) и нажимаем **<OK>**.

Шаг 22



Указываем пароль root`а (администратора). Пароль не может быть меньше 6 символов.

Шаг 23



В данном диалоговом окне мы выбираем пакеты для установки. Можно выбрать готовую конфигурацию, например “Desktop – KDE” для настольной системы или “Server” для сервера. А можно вручную составить список программ которые будут установлены, для этого нужно поставить галочку на пункте Customize software selection. Я этого делать не буду, так как в рамках обучения установки ОС Linux это не имеет принципиального значения.

Шаг 24

Здесь нам сообщают что сейчас начнется инсталляция и что после установки будет создан файл install.log

Нажимаем кнопку **<OK>**.

Начинается процесс установки…

Шаг 25

После того как установка завершится нажимаем кнопку **<Reboot>**.

Система перезагружается…

Шаг 26

В процессе первой загрузки ОС перед нами возникнет конфигурационное окно показанное на рисунке. Здесь необходимо сделать пару вещей:

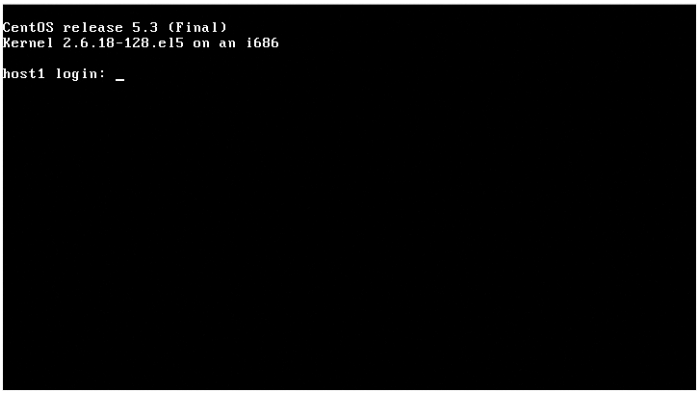
1. Отключаем SELinux: Firewall Configuration -> SELinux: Disabled

2. Firewall Configuration -> Customize – здесь вы настраиваете какие протоколы будут работать через iptables (Firewall).

Вернуться к этим настройкам можно с помощью команды **system-config-securitylevel.**

Далее нажимаете **<Exit>** и продолжаем загрузку ОС.

Шаг 27



Операционная система Linux установлена!

Модуль 3. Базовое понимание Linux

Этапы загрузки ОС

1. BIOS проверяет систему и запускает first stage boot loader (загрузчик первой стадии);

2. first stage boot loader загружает себя в оперативную память и запускает second stage boot loader (загрузчик второй стадии) из /boot раздела;

3. second stage boot loader загружает ядро Linux в оперативную память, который в свою очередь загружает все необходимые модули и монтирует корневой раздел только на чтение;

4. Ядро передает контроль по загрузке ОС программе **/sbin/init**;

5. Программа **/sbin/init** загружает все сервисы, user-space программы и монтирует все разделы перечисленные в **/etc/fstab**;

6. Пользователю выводится приглашение входа в систему.

Программа init

init – (сокращение от, initialization инициализация) это программа которая запускает другие программы. init можно представить как ствол дерева, от которого идут ветви – другие процессы. init в системе представлен как процесс, который обычно имеет PID (Process ID, уникальный номер процесса) равный 1.

На данный момент широко используются два вида init:

1. SysV-style. В этом случае процесс init проверяет запись initdefault в файле /etc/inittab, чтобы выяснить runlevel по умолчанию. Если в этой записи не указан runlevel то пользователю будет выведена консоль, в которой он должен явно указать его. SysV init применяется в Red Hat и соответственно в CentOS.

2. BSD-style. В этом случае запускается скрипт инициализации расположенный в /etc/rc, далее запускается программа getty (сокращенно от get teletype), которая обслуживает физические или виртуальные терминалы. Когда происходит подключение к системе то программа login выводит приглашение ввода логина и пароля для аутентификации. В этом виде init понятие runlevel отсутствует, и как будет запущен init определяет файл /etc/rc.

Есть и другие виды init, такие как upstart в Ubuntu Linux, или своя редакция init в Gentoo Linux.

Программа login

Программа которая отвечает за регистрацию пользователя в системе называется login. Именно она выводит на экран приглашение ввода имени пользователя и пароля: login:

Этапы загрузки login:

1. init запускает программу mingetty;

2. mingetty запускает программу login;

3. Если указанные логин и пароль правильные то запускается shell;

4. Если логин и пароль не правильные то init перезапускает mingetty;

5. При выходе из системы shell завершается и мы возвращаемся к шагу 1.

Если в процессе установки ОС вы не создавали другие учетные записи то доступ в систему будет возможен только пользователем root.

После аутентификации в системе вы попадаете в свой домашний каталог, именно в этом каталоге будут храниться все ваши личные файлы.

Регистрация в системе

В ходе загрузки ОС создаются несколько виртуальных консолей – виртуальных устройств ввода-вывода, которые могут использоваться различными процессами/программами. Если быть более точным то 6 виртуальных консолей (text-based) и одна графическая (для X Window System).

В ОС Linux есть два режима работы:

1. Текстовый (для подключения используются виртуальные консоли)

2. Графический (для подключения используются display managers)

Для серверов обычно применяется только текстовый режим работы. Для настольных систем применяются оба режима.

В Red Hat и соответственно в CentOS применяется бесплатная реализация X Window System – Xorg.

Переключение между виртуальными консолями происходит с помощью сочетания клавиш **Ctrl+Alt+F [1–6]**. Если вы захотите перейти на консоль номер 3 нажмите сочетание клавиш **Ctrl+Alt+F3**, если потребуется консоль номер 5 то нажимаем сочетание клавиш **Ctrl+Alt+F5** и т. д.

Переключение в графическую консоль осуществляется комбинацией клавиш Ctrl+Alt+F7

После загрузки ОС перед вами появится приглашение ввода логина и пароля. ОС будет работать в текстовом режиме, но режим можно изменить в любой момент (файл/etc/inittab).

Уровни выполнения ОС

В ОС Linux существует несколько уровней выполнения, так называемых runlevels. Суть runlevels заключается в том, что разные системы могут использоваться по разному. К примеру сервера будут работать более эффективно без графического интерфейса, который потребляет много ресурсов но в случае сервера практически бесполезен. Или случаются ситуации когда администратору необходимо переключиться на другой уровень выполнения чтобы выполнить специфичные задачи, например диагностика поврежденной файловой системы.

Каждый уровень выполнения определяет, какие сервисы будут запущены или наоборот отключены процессом init. Например на runlevel 1 (single user mode) будет отключена сеть, в то время как на runlevel 3 она будет включена.

На данный момент в Red Hat и соответственно во всех клонах определенны следующие уровни:

Уровень	Название	Описание
0	Halt	Выключенная ОС
1	Single-user text mode	Однопользовательский режим работы с отключенным сетевым интерфейсом
2	Not used (user-definable)	Промежуточный уровень
3	Full multi-user text mode	Многопользовательский режим работы с поддержкой сети
4	Not used (user-definable)	Промежуточный уровень
5	Full multi-user graphical mode (with an X-based login screen)	Многопользовательский режим работы с графическим интерфейсом и поддержкой сети
6	Reboot	Перезагрузка ОС
S или s	Scripts	Используется скриптами исполняемыми на уровне 1

В большинстве случаев используются два runlevel`a, это 3 для серверов и 5 для настольных систем.

Какой runlevel будет загружаться при старте системы определяется конфигурационным файлом **/etc/inittab** а именно строкой:

```
id:5: initdefault:
```

Цифра после id: и есть runlevel. При редактирование этого файла соблюдайте осторожность, даже самая небольшая опечатка может привести систему в неработоспособность.

Конфигурационные файлы для SysV init расположены в директории **/etc/rc.d/**

Помимо файлов **rc, rc.local, rc.sysinit** и опционально **rc.serial** здесь находятся субкаталоги, которые сообщают системе, какие программы необходимо запустить или остановить на конкретном уровне выполнения.

```
init.d/rc0.d/rc1.d/rc2.d/rc3.d/rc4.d/rc5.d/rc6.d/
```

Буква K в начале имени символической ссылки говорит о том, что произойдет выключение сервиса (Kill), буква S в начале имени говорит о том, что

сервис будет запущен (Start). Число в имени символической ссылки задает последовательность выполнения скриптов.

```
lrwxrwxrwx 1 root root 19 Sep 15 19:43 K35 vncserver ->../init.d/vncserver

...

lrwxrwxrwx 1 root root 19 Sep 15 19:38 S99 firstboot ->../init.d/firstboot

lrwxrwxrwx 1 root root 11 Sep 15 19:33 S99 local ->../rc.local

lrwxrwxrwx 1 root root 16 Sep 15 19:34 S99 smartd ->../init.d/smartd
```

Каталог **init.d** содержит скрипты используемые **/sbin/init** при контроле сервисов. Остальные каталоги содержат символические ссылки для каждого из уровней выполнения.

Узнать текущий runlevel в системе можно следующими командами:

```
[root@linuxbox ~]# who -r
```

или

```
[root@linuxbox ~]# runlevel
```

Переключение между уровнями выполнения может происходить ниже указанными командами:

```
[root@linuxbox ~]# telinit
```

или

```
[root@linuxbox ~]# init
```

Скрипт **/etc/rc.d/rc.local** выполняется программой **/sbin/init** во время загрузки или при переключение runlevel`ов. Добавление нужных команд в этот файл позволит выполнять дополнительные команды/скрипты. Этот файл будет полезен в простых случаях. Для удобного управления программным обеспечением стоит использовать **chkconfig**.

Для работы с runlevels различных сервисов есть несколько программ:

1. **/sbin/chkconfig** – простая и очень удобная текстовая программа;

2. **/usr/sbin/ntsysv** – основанная на ncurses (псевдографика, обычно на синем фоне) утилита предоставляющая интерактивный интерфейс, более легкая в использование чем chkconfig но не позволяющая указывать конкретные runlevels;

3. **system-config-services** – Services Configuration Tool, графическая программа для управления уровнем выполнения. Требуется X Window.

runlevel операционной системы можно изменять на стадии загрузки. Для этого находясь в меню GRUB выберите нужное ядро и нажмите клавишу a, в конце текущей строки добавьте через пробел номер нужного runlevel, например, если мы хотим попасть в single mode то это будет выглядеть вот так:

```
grub append> ro root=/dev/VolGroup00/LogVol00 rhgb quiet 1
```

Загрузчик GRUB

При включение компьютера, ОС загружается в оперативную память с помощью специальной программы – загрузчика (boot loader). Обычно загрузчик располагается на первичном жестком диске и обеспечивает выбор и загрузку ядра с необходимыми ему файлами или позволяет загрузить другую операционную систему. При включение компьютера будет отображено меню обхода загрузчика GRUB. Если не будет нажата любая клавиша в течение 3 секунд то начнется загрузка ОС или ядра по умолчанию.

В конце 20/начале 21 века популярным загрузчиком был lilo, который использовался по умолчанию в большинстве дистрибутивов. Сейчас он практически не используется а на его смену пришел более функциональный загрузчик – GRUB (The GNU GRand Unified Boot loader).

> Программа установки Microsoft Windows перезаписывает MBR, удаляя все загрузчики. Поэтому в случае необходимости иметь две ОС на одном компьютере, Microsoft Windows лучше устанавливать в первую очередь.

Если GRUB не устанавливается во время установки ОС то это можно сделать потом с помощью команды **/sbin/grub-install** например вот так:

```
[root@linuxbox ~]#/sbin/grub-install /dev/hda
```

где /dev/hda – master IDE устройство на primary IDE шине.

Интерфейсы работы с GRUB

У GRUB есть 3 интерфейса:

1. Меню. Это интерфейс по умолчанию. Перемещаться по списку доступных ОС и ядер можно стрелками вверх и вниз, нажатие **<Enter>** позволит загрузить выбранную ОС или ядро. Если до истечения таймаута выбор

не сделан то будет загружено ОС/ядро по умолчанию. Нажатие **e** позволит попасть в редактор а нажатие **c** в интерфейс командной строки.

2. Интерфейс редактора записей. Чтобы сюда попасть нажмите **e** в меню GRUB. В этом интерфейсе будут отображены команды GRUB которые можно модифицировать (нажмите **o** для вставки новой строки после текущей, нажмите **O** для вставки новой строки перед текущей, нажмите **e** для редактирования или **d** для удаления строки). После завершения редактирования нажмите **b** для исполнения команд и загрузки ОС. Для отмены изменения и возврата в меню нажмите **<ESC>**.

3. Интерфейс командной строки. Чтобы сюда попасть нажмите **c** в меню GRUB. Этот интерфейс предоставляет больше возможностей чем два других. Командный интерфейс позволяет ввести любую команду GRUB и выполнить её нажатием **<Enter>**.

Список самых распространенных команд указан в таблице ниже.

Команда	Описание
boot	Загружает ОС или последний загруженный последовательный загрузчик
chainloader </путь/к/файлу>	Передает управление файлу. Если файл расположен в первом секторе раздела, то можно просто использовать chainloader +1
displaymem	Отображает статус занятой памяти на основе полученной из BIOS информации. Может пригодиться для определения доступной в системе памяти до начала загрузки
initrd </путь/к/initrd>	Задает положение исходного RAM диска. Может оказаться необходимым в случае, если для корректной загрузки ядра требуются определенные модули (например, когда корневой раздел имеет тип ext3)
install <стадия-1><install-disk><стадия-2>pconfig-file	Устанавливает GRUB в MBR. Команда install переписывает все текущие данные в MBR
kernel </путь/к/ядруl><параметр-1><параметр-N>	Задает файл ядра для загрузки. </путь/к/ядру> – абсолютный путь от раздела, заданного командой «root». <параметр-1> – передаваемый ядру параметр, указывающий на устройство, где расположен раздел «root», например, root=/dev/VolGroup00/LogVol00. Допускается использование нескольких параметров, разделенных пробелом

root (<тип_ устройства><номер_ устройства>, <раздел>)	Настраивает и монтирует корневой раздел для GRUB, например: root (hd0,0)
rootnoverify (<тип_ устройства><номер_ устройства>, <раздел>)	Настраивает корневой раздел аналогично команде root, но не монтирует его
reboot	Перезагрузка системы

Получить список всех доступных команд можно командой **help –all**

Файл конфигурации GRUB

Конфигурационный файл GRUB – это **/boot/grub/grub.conf**. Команды глобальных настроек расположены в начале файла, затем идут секции для каждой ОС или ядра (секция title).

Пример простого файла конфигурации позволяющего загружать Linux и Microsoft Windows 2000 показан ниже.

```
default=0

timeout=5

splashimage= (hd0,0)/grub/splash.xpm.gz

hiddenmenu

title CentOS (2.6.18–128.el5)

root (hd0,0)

kernel/vmlinuz-2.6.18–128.el5 ro root=/dev/VolGroup00/LogVol00 rhgb quiet

initrd/initrd-2.6.18–128.el5.img

title Windows

rootnoverify (hd0,0)

chainloader +1
```

В этом примере Linux будет загружаться по умолчанию, автоматическая загрузка которого начнется через 5 секунд.

Корневая файловая система GRUB не имеет нечего общего с корневой

файловой системой Linux. Корневая файловая система GRUB сопоставлена верхнему уровню заданного устройства. Например, файл (hd0,0)/grub/splash.xpm.gz расположен в каталоге /grub/верхнего уровня раздела (hd0,0), что в действительности является разделом /boot/системы.

Наиболее часто используемые директивы конфигурационного файла представлены в таблице ниже.

Директива	Описание
default=<число>	Замените <число> номером записи «title» загружаемой по умолчанию операционной системы
timeout=<число>	Задает интервал в секундах, по истечению которого будет выполнена запись “title”, определенная командой default
splashimage=<путь_к_изображению>	Директива определяет расположение заставки GRUB
hiddenmenu	Предотвращает отображение меню GRUB. По истечении периода, заданного в timeout, будет загружена операционная система или ядро по умолчанию (запись default)
title <название_группы>	Директива задает название группы набора команд для загрузки конкретного ядра или операционной системы
root (<тип_устройства><номер_устройства>,<раздел>)	Настраивает и монтирует корневой раздел для GRUB, например: root (hd0,0)
kernel </путь/к/ядруl><параметр-1><параметр-N>	Директива задает файл ядра для загрузки
initrd </путь/к/initrd>	Задает положение исходного RAM диска. Может оказаться необходимым в случае, если для корректной загрузки ядра требуются определенные модули (например, когда корневой раздел имеет тип ext3)
rootnoverify (<тип_устройства><номер_устройства>,<раздел>)	Настраивает корневой раздел аналогично команде root, но не монтирует его

chainloader </путь/к/файлу>	Передает управление файлу. Если файл расположен в первом секторе раздела, то можно просто использовать chainloader +1
password=<пароль>	Предотвращает неавторизованное редактирование конфигурационного файла GRUB
color <обычный_цвет><цвет_выбора>	Позволяет изменить цвета переднего и заднего плана меню. Используйте названия цветов, например: color red/black green/blue
fallback=<число>	Замените <число> номером записи «title» операционной системы, загружаемой в случае, если первая попытка окажется неудачной

Комментарии в файле начинаются с символа #

Дополнительная информация по GRUB

```
[root@linuxbox ~]# info grub
```

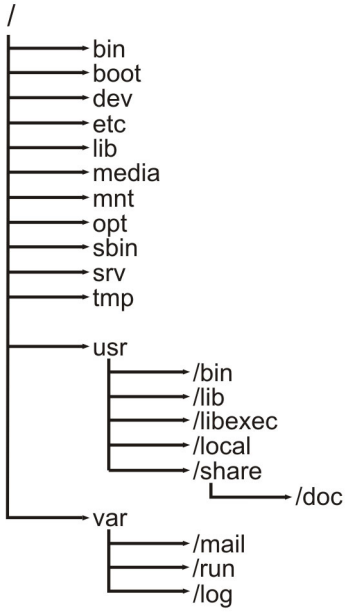
Модуль 4. Файловая система

Иерархия файловой системы

Файловая система – это основная часть любой операционной системы. Она отвечает за организацию хранимых ресурсов (файлов). По сути – это часть ядра управляющая доступом к файлам и каталогам.

Файловая система представляет из себя единую иерархическую структуру которая начинается с каталога /, его еще называют root-каталогом (не путать с /root, который является домашним каталогом для пользователя root) и разветвляется на произвольное число подкаталогов. Каталог самого верхнего уровня называется корневым.

Замечу, что в UNIX-like системах используется слеш (“slash”) – символ / в путях файлов и каталогов, в то время как в Windows системах бекслеш (“backslash”) – символ \



В UNIX/Linux-системах регистр играет роль (В отличие от Windows). То есть файлы:

```
file.txt
```

и

```
FILE.txt
```

будут совершенно разными файлами, и они спокойно могут находиться в одном каталоге.

Имя каталога в файловой системе ограничено 255 символами а отдельно взятое путевое имя 1023 символами.

Filesystem Hierarchy Standard

В CentOS используется Filesystem Hierarchy Standard (FHS) для структуры файловой системы. Он определяет имена, расположение и права доступа для многих типов файлов и каталогов. Файловая система в Linux никогда не была хорошо упорядочена так как использовалось большое количество правил именования файлов. Filesystem Hierarchy Standard эту проблему решает.

Согласно этому стандарту в корне файловой системы должны присутствовать следующие директории:

Директория	Описание
bin	Основные бинарные программы
boot	Статичные файлы загрузчика
dev	Файлы устройств
etc	Системные конфигурационные файлы
lib	Основные разделяемые библиотеки и модули ядра
media	Точка монтирования для извлекаемых устройств
mnt	Временная точка монтирования для файловых систем
opt	Каталог дополнений для программ (add-on`s)
sbin	Основные системные бинарные программы
srv	Данные для сервисов предоставляемых системой
tmp	Временные файлы
usr	Вторая файловая иерархия
var	Изменяемые данные
home	Домашние каталоги пользователей (опционально)
lib<qual>	Альтернативные базовые разделяемые библиотеки (опционально)
root	Домашняя директория пользователя root (опционально)

Файлы в файловой системе разбиты на две логические категории:

1. Доступные для общего доступа и не доступные (shareable и unshareable);

Файлы с общим доступом доступны как с локальной машины так и по сети. Файлы не доступные для общего доступа, доступны только локально.

2. Изменяемые и статические файлы.

Изменяемые файлы – это различные документы, конфигурационные файлы, то есть файлы которые могут изменяться в процессе работы ОС. Статические файлы – это в основном бинарные файлы, изменение которых на уровне файловой системы не предусмотрено. Ниже в таблице показаны примеры каталогов различных категорий.

	shareable	unshareable
static	/usr	/etc
	/opt	/boot
variable	/var/mail	/var/run
	/var/spool/news	/var/lock

Такое разделение файлов позволяет соотнести функции файлов с правами на директории, которые эти файлы хранят. То как пользователи будут взаимодействовать с файлами определяется каталогами, смонтированы ли они в read only режиме или в read write, и какие права доступа у пользователя на этот файл.

Файловая система ext3

ext3 – это журналируемая файловая система. Файловая система ext3 интегрируется в ядро Linux начиная с версии 2.4.16, и используется по умолчанию в CentOS. Для увеличения целостности системы, журналируемая файловая система хранит список изменений, которые она будет проводить с файловой системой перед фактической записью изменений. Эти записи хранятся в отдельной части файловой системы, называемой «журналом». Как только эти записи журнала безопасно записаны, журналируемая файловая система вносит эти изменения в файловую систему и затем удаляет эти записи из журнала.

ext3 пришла на смену ext2 и обладает массой преимуществ:

1. Доступность;

В случае крушения системы (это также называется unclean system shutdown) в ext2, при загрузке ОС необходимо было проверять целостность файловых систем программой e2 fsck а это могло занять большое количество времени. Журналирование которое предоставляется ext3 позволяет забыть про эти

проверки при загрузке ОС. Проверка целостности может понадобиться только в редких случаях, таких как отказ жесткого диска, что происходит не часто. К тому же время восстановления файловой системы ext3 никак не связано с ее размером или количеством файлов. Это напрямую зависит от размера файла журнала который необходим чтобы поддерживать целостность файловой системы.

2. Целостность данных;

ext3 обеспечивает целостность данных при крушение системы (например потеря электропитания). ФС ext3 позволяет выбирать уровень защиты данных. По умолчанию используется высокий уровень обеспечения целостности данных.

3. Скорость;

В большинстве случаев пропускная способность у ext3 больше чем у ext2.

4. Легкий переход.

Файловую систему ext2 очень легко конвертировать в ext3 без переформатирования, для этого достаточно воспользоваться утилитой tune2 fs.

Файловая система ext3 поддерживает файлы до 1 Тб и блочные устройства до 4 Тб.

В ext3 есть три режима журналирования:

1. writeback: в журнал записываются только метаданные файловой системы, то есть информация о её изменении. Не гарантирует целостности данных но работает быстрее чем ext2.

2. ordered: то же, что и writeback, но запись данных в файл производится гарантированно до записи информации о изменении этого файла. Не гарантирует целостности данных но увеличивает вероятность их сохранения.

3. journal: полное журналирование как метаданных ФС, так и пользовательских данных. Самый медленный, но и самый безопасный режим.

Режим журналирования можно задать при монтирование раздела с помощью команды mount, например:

```
[root@linuxbox ~]# mount/dev/sda4/mnt/sda4-t ext3-o data=<режим>
```

Создание новой файловой системы ext3

Для создания файловой системы ext3 нужно проделать следующее:

1. Создать раздел с помощью parted или fdisk;
2. Отформатировать раздел в файловой системе ext3 с помощью mkfs -t ext3;
3. Назначить разделу метку с помощью e2label;
4. Создать точку монтирования (директория в которой будет доступ к разделу, директорию можно создать командой mkdir);
5. Добавить раздел в файл/etc/fstab, если хотите чтобы он монтировался во время загрузки ОС.

Суперблок

Суперблок – это запись, содержащая характеристики файловой системы (метаданные). В суперблоке хранится следующая информация:

– размер файловой системы;

– количество свободных блоков в файловой системе;

– индекс следующего свободного блока в списке свободных блоков;

– размер списка индексных дескрипторов;

– количество свободных индексных дескрипторов в файловой системе;

- следующий свободный индексный дескриптор в списке свободных;

- заблокированные поля для списка свободных блоков и свободных индексных дескрипторов;

- флаг показывающий, что в суперблок были внесены изменения.

Повреждение суперблока приводит к стиранию исключительно важной информации, поэтому в системе создается несколько копий суперблока, которые размещаются в разных местах.

Посмотреть список запасных суперблоков можно командой dumpe2fs, например:

```
[root@linuxbox ~]# dumpe2fs /dev/sda1 | grep superblock
```

```
dumpe2fs 1.39 (29-May-2006)
Primary superblock at 1, Group descriptors at 2-2
Backup superblock at 8193, Group descriptors at 8194-8194
Backup superblock at 24577, Group descriptors at 24578-24578
Backup superblock at 40961, Group descriptors at 40962-40962
Backup superblock at 57345, Group descriptors at 57346-57346
Backup superblock at 73729, Group descriptors at 73730-73730
```

Использовать запасной блок можно вот так:

1. Загружаемся с Rescue disk / Live CD (установочный диск CentOS подойдет). Файловая система должна быть отмонтирована.

2. Проверяем файловую систему с использованием запасного суперблока:

```
[root@linuxbox ~]# fsck -y -b 8193 /dev/sda1
```

3. Пробуем смонтировать файловую систему обычным способом. Если не получается то передаем это явно через опцию sb, например вот так:

```
[root@linuxbox ~]# mount sb=8193 /dev/sda1 /mnt
```

Естественно, точка монтирования будет зависит от файловой системы.

4. Добавляем в /etc/fstab опцию sb (sb=8193) к нашей файловой системе для использования ее во время автозагрузки ОС (если без этой опции не работает).

Inode

Помимо имени с которым обычно работают пользователи, у файла есть так называемый индексный дескриптор или просто – inode (сокр. от Information NODE – информационный узел). Inode – это структура данных обладающая индивидуальным номером в рамках файловой системы. Каждый inode содержит около 40 информационных полей (“метаданные”), большинство из которых используется только ядром. Для конечных пользователей полезной информацией будет следующее:

– владелец файла;

– группа которой принадлежит файл;

– режим доступа к файлу (чтение, запись, выполнение);

– тип файла;

– длина файла в байтах;

– дата последнего изменение (ctime), последней модификации (mtime) и последнего доступа (atime);

– счетчик жестких ссылок на файл.

Каждый файл характеризуется одним inode но может быть связан с несколькими именами, в *nix это называется жесткими ссылками, речь о которых пойдет дальше. В случае жестких ссылок, файл удаляется только тогда, когда удаляется последняя жесткая ссылка на него в файловой системе.

Вывести список индексных дескрипторов для всех файлов в каталоге можно командой ls -i а посмотреть основное содержимое командой ls –l.

Типы файлов

В большинстве файловых систем поддерживается семь типов файлов:

1. Обычные файлы;

Это просто последовательность байтов. К обычным файлам возможен как прямой так и последовательный доступ. Файл можно создать текстовым редактором или перенаправлением вывода, а удалить – командой rm.

2. Каталоги;

Каталоги содержат именованные ссылки на другие файлы. Помимо самих каталогов, также существуют ссылки «.» и «..», которые обозначают текущий и родительский каталог соответственно. Удалить их нельзя. Поскольку у корня нет родителя то ссылка «..» в нем эквивалентна «.». Каталоги создаются командой mkdir, а удаляются командой rm -rf (полные каталоги) или rmdir (пустые каталоги).

3. Файлы байт-ориентированных (символьных) устройств;

Байт-ориентированные устройства, например, принтер и модем, передают данные посимвольно, не как отдельные блоки, а как непрерывный поток байтов. Файлы устройств можно создавать командой mknod, а удалять –

командой rm.

4. Файлы блок-ориентированных (блочных) устройств;

Блок-ориентированные устройства, например жесткий диск, передают данные блоками. Файлы устройств можно создавать командой mknod, а удалять – командой rm.

5. Сокеты;

Сокеты инкапсулируют соединения между процессами, позволяя им взаимодействовать, не подвергаясь влиянию других процессов. Сокеты создаются с помощью системного вызова socket. Когда с обеих сторон соединение закрыто, сокет можно удалить командой rm или системным вызовом unlink.

6. Именованные каналы (named pipe);

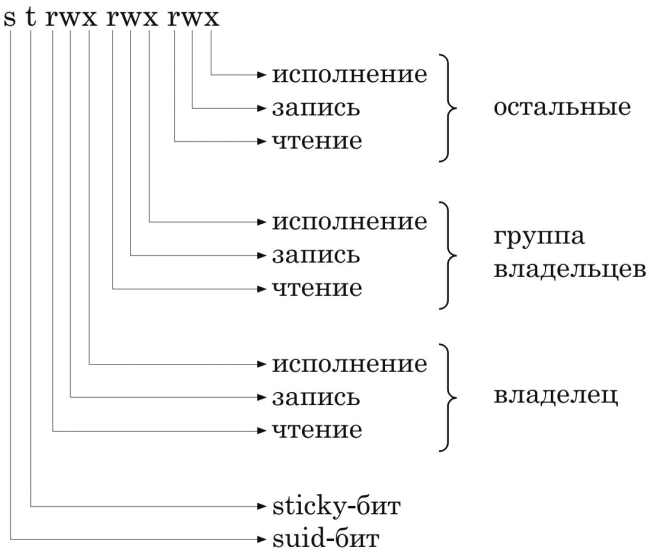
Подобно сокетам, именованные каналы обеспечивают взаимодействие двух процессов, выполняемых на одном компьютере. Именованные каналы можно создавать командой mknod, а удалять – командой rm.

7. Ссылки.

Символическая, или "мягкая" ссылка обеспечивает возможность вместо путевого имени файла указывать псевдоним. Когда ядро при поиске файла сталкивается с символической ссылкой то оно извлекает из нее путевое имя. Для пользователя такой файл в большинстве ситуаций неотличим от того, на который он ссылается: операции чтения, записи и пр. над символьной ссылкой работают так, как если бы они производились непосредственно над тем файлом, на который указывает ссылка. Другой тип ссылки – "жесткий". Разница между "мягкими" и "жесткими" ссылками состоит в том, что "жесткая" ссылка является прямой, то есть указывает непосредственно на индексный дескриптор файла в то время как "мягкая" указывает на файл по имени. Жесткие ссылки создаются командой ln, мягкие (символические ссылки) создаются с помощью команды ln -s. Удалить ссылки можно командой unlink или rm.

Права доступа к файлам

Каждому файлу в файловой системе соответствует набор прав доступа. Они определяют каким пользователям можно получать доступ на чтение, запись или выполнение файла. Права доступа описываются 9 битами. Еще 3 бита (SUID, SGID, Sticky) отвечают за то, как будет запускаться файл. Все 12 бит образуют так называемый код режима доступа к файлу. Изменять права доступа можно с помощью команды chmod.



Бит SUID

Если установлен бит SUID (в восьмеричной форме – 4000) и файл исполняемый, то при запуске на выполнение, файл получает не права запустившего его, а права владельца файла. Где это может быть необходимо? Ну например такая ситуация. Обычному пользователю в системе разрешено самостоятельно изменять пароль при помощи утилиты passwd. passwd работает с файлом /etc/passwd изменять который может только пользователь root. Что бы непривилигированый пользователь мог работать с системным файлом (т.е. изменить свой пароль) на утилите passwd должен быть установлен бит SUID.

Бит **s** на утилите passwd означает, что на ней установлен бит SUID.

```
[root@linuxbox ~]# ls -l /usr/bin/passwd

-rwsr-xr-x 1 root root 22984 Jan  6  2007 /usr/bin/passwd
```

Установить SUID на файл можно вот так (бит SUID + права 775):

```
[root@linuxbox ~]# chmod 4775 file.txt

[root@linuxbox ~]# ls -l file.txt

-rwsrwxr-x 1 user user 0 Aug  2 22:00 file.txt
```

Бит SGID

Бит SGID (в восьмеричной форме – 2000) равнозначен SUID, только наследуются права не владельца файла а группы владельца. Если бит SGID установлен для каталога, то создаваемые в нем файлы при запуске будут принимать идентификатор группы каталога, а не группы в которую входит владелец файла. Бит SGID полезен при коллективной работе пользователей. Например, пользователи входящие в группу developers смогут получать полный доступ к файлам в каталоге /dir1 созданными другими пользователями также входящими в группу developers. Без бита SGID файлы созданные другими пользователями принадлежали бы им и их основной группе, что в итоге не позволило бы участникам общей группы получать к ним полноценный доступ.

Установить SGID на каталог можно вот так (бит SGID + права 775):

```
[root@linuxbox ~]# mkdir /dir1

[root@linuxbox ~]# chmod 2775 /dir1

[root@linuxbox ~]# chown root:developers /dir1

[root@linuxbox ~]# ls -ld /dir1

drwxrwsr-x 2 root developers 4096 Oct 11 03:54 /dir1/
```

Теперь пользователи входящие в группу developers, будут иметь полный доступ к файлам созданным другими участниками группы developers – все создаваемые файлы будут иметь группу владельцев developers.

Бит Sticky

Если бит Sticky (в восьмеричной форме – 1000) устанавливается на каталог то удалять и переименовывать файлы в этой каталоге может только владелец этого каталога, владелец файла или пользователь root. Иметь только разрешение на запись в этот каталог недостаточно. Sticky бит помогает каталогам вроде /tmp стать более защищенными. Бит установленный на файл запрещает ему выгрузку из памяти. Это было полезно для часто запускаемых программ в 80-90-ые годы. В настоящее время почти не используется.

Установить бит Sticky на файл можно вот так (бит Sticky + права 775):

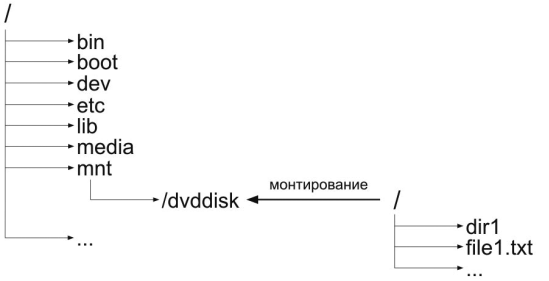
```
[root@linuxbox ~]# chmod 1775 file.txt

[root@linuxbox ~]# ls -l file.txt

-rwxrwxr-t 1 user user 0 Aug  2 22:00 file.txt
```

Монтирование и размонтирование файловых систем

Особенностью операционных систем семейства UNIX является объединение файловых систем в единое разветвленное дерево с единым корневым каталогом. Процесс объединения называется монтированием, процесс отключения от единого дерева каталогов называется размонтированием.



Для монтирования и размонтирования файловых систем есть команды mount и umount соответственно.

Команда mount

Команда mount предназначена для монтирования файловой системы. Команда вызванная без аргументов выводит список текущих примонтированных файловых систем. После того как файловая система примонтирована, соответствующая запись добавляется в файл /etc/mtab

mount [ОПЦИИ] [УСТРОЙСТВО] [ТОЧКА_МОНТИРОВАНИЯ]

mount – название команды для монтирования ФС

ОПЦИИ – различные опции команды mount

УСТРОЙСТВО – это название устройства которое мы хотим примонтировать. В данном случае это /dev/cdrom (симлинк на /dev/hdc)

ТОЧКА_МОНТИРОВАНИЯ – это название каталога, куда мы монтируем устройство

Смонтируем DVD-диск в каталог /mnt (каталог должен существовать):

```
[root@linuxbox ~]# mount -t iso9660 /dev/cdrom /mnt/
mount: block device /dev/cdrom is write-protected, mounting read-only
[root@linuxbox ~]# ls -Fl /mnt/
total 472
drwxr-xr-x 2 user user 407552 Mar 21 18:05 CentOS/
```

```
...
drwxr-xr-x 4 root root   2048 Mar 21 18:04 images/
drwxr-xr-x 2 root root   2048 Mar 21 17:46 isolinux/
drwxr-xr-x 2 root root  12288 Mar 17 04:30 NOTES/
...
drwxr-xr-x 2 root root   2048 Mar 21 17:35 repodata/
```

Монтируем раздел жесткого диска (команда fdisk -l поможет выяснить какие разделы есть):

```
[root@linuxbox ~]# mount -t ext3 /dev/hdb1 /mnt
```

Благодаря опции -o loop смонтируем ISO-образ:

```
[root@linuxbox ~]# mount -o loop -t iso9660 ~/dvddisk.iso /mnt/iso
```

Ключ -t задает тип файловой системы, есть несколько десятком различных типов, но самые распространенные:

Linux: ext3, reiserfs

CD/DVD: iso9660

Сетевая файловая система (NFS): nfs

Windows: vfat, ntfs

SAMBA: smbfs/cifs

Семейство BSD-систем (FreeBSD, OpenBSD, NetBSD..., а также Solaris): ufs

В таблице ниже перечислены популярные ключи команды mount:

Ключ	Описание
-a	Монтировать все файловые системы перечисленные в файле /etc/fstab
-r	Монтировать файловую систему в режиме read
-w	Монтировать файловую систему в режиме read/write
-t	Указание типа файловой системы
-o	Позволяет задать дополнительные опции монтирования (об опциях можно почитать в справке по команде mount – man mount)

Команда umount

Команда umount используется для размонтирования файловой системы. После того как размонтирование успешно завершено, из файла /etc/mtab удаляется запись о соответствующей ФС.

Формат команды:

umount [ОПЦИИ] [УСТРОЙСТВО] или [ТОЧКА_МОНТИРОВАНИЯ]

umount – название команды для размонтирования ФС

ОПЦИИ – различные опции команды umount

УСТРОЙСТВО – название устройства которое примонтировано

ТОЧКА_МОНТИРОВАНИЯ – название каталога куда примонтировано устройство

В большинстве случаев достаточно указать точку монтирования или устройство чтобы размонтировать ФС, например:

```
[root@linuxbox ~]# umount /mnt
```

В примере выше мы размонтировали устройство /dev/hdb1 (было примонтировано командой – mount -t ext3 /dev/hdb1 /mnt)

Популярные ключи команды umount:

Ключ	Описание
-n	Размонтирование без удаления записи из /etc/mtab
-a	Размонтирование всех файловых систем перечисленных в /etc/mtab
-f	Принудительное размонтирование устройства
-l	Lazy unmount. Отключение файловой системы от файловой иерархии и удаление всех ссылок на нее, как будто она больше не занята

В работе команд mount/umount используется несколько файлов:

/etc/fstab, /etc/mtab, /etc/filesystems

Файл /etc/fstab

В этом файле описываются все файловые системы которые будут автоматически монтироваться во время загрузки ОС и по ключу -a команды mount. Программы только читают этот файл не записывая в него что либо. Это обязанность пользователя – вносить изменения в /etc/fstab. /etc/fstab — это конфигурационный файл программы mount и все недостающие параметры программа берет из него.

Запись в файле состоит из 6 полей, каждая запись описывает одну файловую систему.

Формат записей в файле:

[устройство] [точка монтирования] [тип ФС] [опции] [флаг dump] [флаг fsck]

Пример:

```
/dev/hdb1          /mnt              ext3   defaults      0 0
```

/dev/hdb1 – устройство которое будем монтировать

/mnt – каталог куда мы будем монтировать устройство

ext3 – тип файловой системы

defaults – опции по умолчанию (rw, suid, dev, exec, auto, nouser и async)

0 – флаг программы dump. Она на данный момент практически не используется в Linux, поэтому флаг равен нулю. Оставлено для совместимости.

0 – флаг программы fsck. Если флаг равен нулю то файловая система при запуске программы fsck с параметром -A не будет автоматически проверяться, в том числе и при запуске системы. Если флаг равен единице, значит проверка будет производиться. Файловые системы проверяются в том порядке, в котором они описаны в этом файле. Если Вы хотите изменить порядок проверки, вместо единицы используйте двойку. Тогда сначала проверяются все файловые системы с единицей, а затем с двойкой.

Если вы хотите чтобы файловая система монтировалась автоматически при загрузке ОС то добавьте ее в этот файл.

Файл /etc/mtab

Когда программа mount подключает файловую систему, она дописывает соответствующую строку в /etc/mtab. Когда umount отключает файловую систему, из этого файла соответствующая строка удаляется. Вручную редактировать этот файл не требуется.

Пример файла /etc/mtab:

```
[root@linuxbox ~]# cat /etc/mtab
/dev/mapper/VolGroup00-LogVol00 / ext3 rw 0 0
proc /proc proc rw 0 0
```

```
sysfs /sys sysfs rw 0 0
devpts /dev/pts devpts rw,gid=5,mode=620 0 0
/dev/hda1 /boot ext3 rw 0 0
tmpfs /dev/shm tmpfs rw 0 0
/dev/hdb1 /mnt ext3 rw 0 0
none /proc/sys/fs/binfmt_misc binfmt_misc rw 0 0
sunrpc /var/lib/nfs/rpc_pipefs rpc_pipefs rw 0 0
```

Файл /etc/filesystems

В случае, если тип файловой системы не задан ключом -t, будет попытка его выяснить автоматически перебирая список ФС из этого файла.

```
[root@linuxbox ~]# cat /etc/filesystems
ext3
ext2
nodev proc
nodev devpts
iso9660
vfat
hfs
hfsplus
```

Файловая система /proc

/proc – это псевдо файловая система которая используется как интерфейс к структурам данных ядра. Она является виртуальной, загружаясь в оперативную память при старте системы и не занимает места на жестком диске. Многие файлы в этой файловой системе можно изменять, тем самым изменяя параметры ядра в режиме реального времени. /proc это также хороший источник информации о системе.

Некоторые важные файлы:

/proc/cpuinfo - информация о процессоре (модель, семейство, размер КЭШа, частота и т.д.)

/proc/meminfo - информация о RAM, размере свопа и т.д.

/proc/mounts - список подмонтированных файловых систем

/proc/devices - список устройств

/proc/filesystems - поддерживаемые файловые системы

/proc/modules - список загружаемых модулей

/proc/version - версия ядра

/proc/cmdline - список параметров, передаваемых ядру при загрузке.

Это обычные текстовые файлы и просмотреть их можно командой cat, например:

```
[root@linuxbox ~]# cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 15
model name      : Intel(R) Core(TM)2 Duo CPU    E6750  @ 2.66GHz
...
[вывод урезан]
```

Многие параметры ядра можно изменять в режиме реального времени, например поменяем имя компьютера:

```
[root@linuxbox ~]# hostname
localhost.localdomain
[root@linuxbox ~]# echo 'new-hostname' > /proc/sys/kernel/hostname
[root@linuxbox ~]# hostname
new-hostname
```

Но /proc не сохраняет своего текущего состояние после перезагрузки системы. Чтобы параметры внесенные в /proc сохранились, их необходимо зафиксировать в соответствующих конфигурационных файлах. Тот же hostname нужно прописать в /etc/sysconfig/network

Работа с файлами: создание, просмотр, редактирование, удаление

Создать текстовый файл можно разными способами:

1. Текстовый редактор. По умолчанию в CentOS всегда есть как минимум 1 текстовый редактор – это vi/vim.

2. С помощью перенаправления вывода команды echo (или cat):

В файл file.txt записываем строку "text":

```
[root@linuxbox ~]# echo 'text' > file.txt
```

Содержимое file.txt будет удалено и в нем будет только строка "text". Если файла не существовало то он будет создан.

Но удалять содержимое файла совсем не обязательно. Можно добавить строку в конец файла, для этого надо использовать двойную стрелку вправо:

```
[root@linuxbox ~]# echo 'text' >> file.txt
```

В этом случае строка "text" будет добавлена в конец файла file.txt не затрагивая другие данные файла. Если файла не существовало то он будет создан.

3. Если файл существует то можно сделает его копию командой cp:

```
[root@linuxbox ~]# cp file.txt myfiles/file2.txt
```

В первом аргументе команды cp файл который мы хотим скопировать, во втором аргументе – куда мы хотим его скопировать и под каким именем.

4. Команда touch позволяет создать пустой файл:

```
[root@linuxbox ~]# touch file.txt
```

Просмотреть текстовый файл можно разными способами:

1. Текстовый редактор. По умолчанию в CentOS всегда есть как минимум 1 текстовый редактор – это vi/vim.

2. Полностью вывести на консоль с помощью команды cat:

```
[root@linuxbox ~]# cat file.txt
text…
```

Файл полностью выводится на консоль. Этот способ не очень удобен в случае с большими файлами.

3. Постраничный/построчный вывод с помощью команды less. Если файл нужно только просмотреть то эта команда лучший кандидат. Схожая по работе с less – команда more. Но less гораздо удобнее в использование.

4. Команда head по умолчанию выводит первые 10 строчек файла. С помощью ключа -n можно вывести произвольное количество. Например:

```
[root@linuxbox ~]# head -n 20 /etc/login.defs
```

5. Команда tail по умолчанию выводит последние 10 строчек файла. С помощью ключа -n можно вывести произвольное количество. Например:

```
[root@linuxbox ~]# tail -n 20 /etc/login.defs
```

6. Команда nl направляет содержимое файла на стандартный вывод (консоль) и нумерует строки. Например:

```
[root@linuxbox ~]# nl /etc/login.defs
     1  # *REQUIRED*
     2  #   Directory where mailboxes reside, _or_ name of file, relative to the
     3  #   home directory.  If you _do_ define both, MAIL_DIR takes precedence.
     4  #   QMAIL_DIR is for Qmail
     5  #
     6  #QMAIL_DIR     Maildir
     7  MAIL_DIR       /var/spool/mail
```

```
   8  #MAIL_FILE     .mail
…
[вывод урезан]
```

Редактировать файл можно любым текстовым редактором. В Linux как минимум есть vi/vim, но также популярны: emacs, nano, ee и встроенный редактор файлового менеджера mc.

Удалить файл можно командой rm.

Работа с каталогами: создание, просмотр, переименование, удаление

Создание каталога происходит командой mkdir, например:

```
[root@linuxbox ~]# mkdir /files
```

Ключ -p будет полезен в ситуации, когда нужно создать родительский каталог если его не существует.

```
[root@linuxbox ~]# mkdir -p /files/documents/fromwork
```

В случае отсутствия какого либо каталога до fromwork – documents или / files/documents, они будут автоматически созданы.

Просмотр каталога происходит командой ls, например:

```
[root@linuxbox ~]# ls -l /root
total 64
-rw-------  1 root root  1119 Jul 28 02:21 anaconda-ks.cfg
-rw-r--r--  1 user user     5 Aug  5 03:34 ile.txt
-rw-r--r--  1 root root 30116 Jul 28 02:21 install.log
-rw-r--r--  1 root root  4286 Jul 28 02:19 install.log.syslog
drwxr-xr-x  2 root root  4096 Aug  5 01:17 tes
```

Переименование каталога происходит командой mv, например:

```
[root@linuxbox ~]# mv tes tes2
```

Первый аргумент – каталог который хотим переименовать, второй аргумент – новое название каталога и его расположение. В данном примере каталог останется на старом месте.

Удаление каталога происходит командами rm -rf (занятые каталоги) и rmdir (пустые каталоги).

```
[root@linuxbox ~]# rm -rf /root/tes2
```

или, если каталог пустой, то есть не содержит файлы и подкаталоги, то его можно удалить командой rmdir:

```
[root@linuxbox ~]# rmdir /root/tes2
```

Команда ls: вывод содержимого каталога

Команда ls позволяет вывести содержимое каталога (по умолчанию текущего) и информацию о файлах в нём.

Формат команды: ls [опции]… [файл]…

Пример вывода команды ls:

```
[root@linuxbox ~]# ls

anaconda-ks.cfg  dir  file.txt  install.log  install.log.syslog
```

Команда ls в большинстве случаев будет использоваться с какими либо ключами, так как они добавляют в вывод большое количество нужной информации.

Вывод подробной информации о файлах:

```
[root@linuxbox ~]# ls -l /root

total 60

-rw-------   1 root root   1119  Jul  28  02:21  anaconda-ks.cfg

drwxr-xr-x   3 user user   4096  Aug   3  20:08  dir

-rwxrwxr-x   1 user user      0  Aug   2  21:59  file.txt

-rwxrwxr-x   1 root root  30116  Jul  28  02:21  install.log
```

```
-rw-r--r--    1 root root    4286  Jul  28  02:19  install.log.syslog
```

Каждая строка описывает один файл. Первая колонка – это права доступа, самый первый символ в первой колонке – тип файла. Вторая колонка – количество жестких ссылок на этот файл или каталог. Третья колонка – владелец. Четвертая колонка – группа владелец. Пятая колонка – размер файла или каталога. Шестая колонка – время последней модификации. Седьмая колонка – название файла или каталога.

Иногда нужно вывести список всех файлов в каталоге. В этом поможет опция -a:

```
[root@linuxbox ~]# ls -la /root
total 148
drwxr-x---   5 root root     4096  Aug   3   20:08  .
drwxr-xr-x  23 root root     4096  Aug   4   18:02  ..
-rw-------    1 root root     1119   Jul  28   02:21    anaconda-ks.cfg
-rw-------    1 root root      555   Aug   1   23:34    .bash_history
-rw-r--r--    1 root root       24   Jan   6   2007     .bash_logout
-rw-r--r--    1 root root      191   Jan   6   2007     .bash_profile
-rw-r--r--    1 root root      176   Jan   6   2007     .bashrc
-rw-r--r--    1 root root      100   Jan   6   2007     .cshrc
drwxr-xr-x   3 user user     4096   Aug   3   20:08    dir
```

В каждом каталоге будут ссылки “.” и “..”. Они обозначают текущий каталог и родительский соответственно. В корневом каталоге ссылка “..” соответствует “.” поскольку у него нет родительского каталога.

Популярные ключи команды ls:

Ключ	Описание
-a	Показывать все файлы, включая скрытые (начинаются с точки)
-h	human readable format, отображать размер файлов в более удобном формате (например в Мб или Гб)
-i	Выводить inode каждого файла
-l	Выводить больше информации по файлам
-R	Рекурсивный режим. Выводить листинги подкаталогов

Типы файлов в выводе команды ls:

Символ	Тип файла	Команда создания	Команда удаления
-	Обычный файл	Текстовый редактор, cp и др.	rm
d	Каталог	mkdir	rmdir, rm -r
c	Файлы байт-ориентированных (символьных) устройств	mknod	rm
b	Файлы блок-ориентированных (блочных) устройств	mknod	rm
s	Сокет	socket(2)	rm
p	Именованный канал	mknod	rm
l	Символическая ссылка	ln -s	rm

Выяснить тип файла можно командой file или ls -l, например:

```
[root@linuxbox ~]# file /dev/sda

/dev/sda: block special (8/0)

[root@linuxbox ~]# ls -l /dev/sda

brw-r----- 1 root disk 8, 0 Oct 11 00:53 /dev/sda
```

Команда cd: смена каталога

Команда cd (change directory) предназначена для смены текущего каталога. В качестве аргумента передается целевой каталог (пункт назначения).

Путь может быть абсолютный, то есть начинающийся с корня файловой системы и относительный, то есть относительный текущего каталога.

Пример перехода с использованием абсолютного пути:

```
[root@linuxbox ~]# cd /usr/share/doc/
```

Пример перехода с использованием относительного пути. Предположим, мы находимся в каталоге /usr/share/yum-cli/ и нам нужно попасть в /usr/share/doc/

```
[root@linuxbox ~]# cd ../doc/
```

Символ ~ означает домашний каталог пользователя. Для администратора системы под этим символом скрывается /root. Это можно использовать при указание путей к каталогам и файлам в домашнем каталоге, например:

```
[root@linuxbox ~]# cd ~/myfiles/docs/
```

Это команда соответствует команде: cd /root/myfiles/docs/

Команда pwd: просмотр рабочего каталога

Команда pwd (print working directory) выдает полный путь к текущему каталогу.

Формат команды: pwd [опции]

```
[root@linuxbox ~]# pwd

/root
```

Команда cat: слияние файлов и чтение на стандартный вывод

Команда cat по очереди читает указанные файлы и выдает их содержимое на стандартный вывод. Если не указан ни один файл или среди аргументов встретился символ "-", то команда cat читает данные со стандартного ввода.

Формат команды: cat [опции] [файл]...

Вывести содержимое файла file.txt на стандартный вывод (при добавление ключа -n строки будут пронумерованы)

```
[root@linuxbox ~]# cat file.txt
```

Содержимое файлов file1.txt и file2.txt будет сохранено в file3.txt

```
[root@linuxbox ~]# cat file1.txt file2.txt > file3.txt
```

Добавление содержимого file1.txt к содержимому file2.txt

```
[root@linuxbox ~]# cat file1.txt >> file2.txt
```

Добавляем данные в файл file.txt со стандартного ввода:

```
[root@linuxbox ~]# cat - >> file.txt
<вводим текст>
<нажимаем Enter>
<нажимаем Ctrl+C>
```

Команда grep: поиск по шаблону

Команда grep используется для поиска конкретных текстовых строк в файлах.

Формат команды: grep [опции] шаблон [файл...]

Команда grep в файле /etc/resolv.conf найдет строки содержащие "nameserver" и выведет их на экран.

```
[root@linuxbox ~]# grep nameserver /etc/resolv.conf
nameserver 192.168.79.2
[root@linuxbox ~]#
```

Команда grep часто используется для поиска в выводе другой команды.

```
[root@linuxbox ~]# rpm -qa | grep chk
chkconfig-1.3.30.1-2
chkfontpath-1.10.1-1.1
[root@linuxbox ~]#
```

Команда chmod: изменение прав доступа

Программа chmod (change mode) служит для того, чтобы изменять права доступа к файлам.

Формат команды: chmod [опции] [режим] файл

Как уже говорилось выше, режим доступа к файлу описывается 9 битами.

По три бита на каждую из категорий: владелец, группа владелецев и все остальные. В каждой из категорий могут быть следующие права доступа: чтение, запись и выполнение. Чтение обозначается буквой r (сокр. от read), запись w (сокр. от write) и выполнение x (сокр. от e**x**ecutable).

В таблице ниже представлены возможные права, числовое значение и описание:

Буква	Числовое значение	Описание
r	4	Чтение
w	2	Запись
x	1	Выполнение

Для того чтобы выставить права на файл нужно команде chmod в качестве аргумента передать числовое значение и название файла, на который будут установлены права доступа.

Посмотреть какие права на файл установлены сейчас, можно с помощью ls -l

```
[root@linuxbox ~]# ls -l file.txt

-rwxrwxr-x 1 user user 0 Aug  2 22:00 file.txt
```

Как видим на файле file.txt установлены следующие права доступа:

```
-rwxrwxr-x
```

Весь вывод можно разбить на 3 части:

Владелец	Группа владельцев	Все остальные

Первый символ в выводе задает тип файла. В данном случае это -, что означает обычный файл.

Первая категория (первые 3 бита): rwx, это права доступа для владельца. Как видим ему разрешено все (read, write, executable)

Вторая категория: rwx, это права доступа для группы владельцев. Как видим всем пользователям находящимся в группе user разрешено производить с файлом любые манипуляции (read, write, executable).

Третья категория: r-x, это права доступа для всех остальных. Обычные пользователи (это не относится к пользователю user, группе user и пользователю root) в системе могут читать и выполнять этот файл (read, executable).

Установить права доступа можно, например, вот так:

```
[root@linuxbox ~]# chmod 775 file.txt
```

Где первым аргументом идет числовое значение прав доступа а вторым аргументом идет файл, на который эти права доступа нужно установить.

Мы знаем, что числовое значение на чтение – 4, запись – 2, выполнение – 1. Как же посчитать конечную цифру которая должна фигурировать в команде chmod? Все просто. Нужно просто сложить эти цифры. Например если мы хотим на файл установить право на чтение то мы укажем 4, если к этому нужно добавить право на выполнение то 4+1 = 5, если хотим установить все возможные права на файл то 4+2+1 = 7.

Я приведу примеры, чтобы все стало понятнее:

Права доступа	Числовое значение	Описание
-rwxr-x---	750	Владелец – чтение, запись, выполнение. Группа владельцев – чтение, выполнение. Все остальные – полное отсутствие доступа
-rw-------	600	Владелец – чтение, запись. Группа владельцев – полное отсутствие доступа. Все остальные – полное отсутствие доступа
-rwxr-xr-x	755	Владелец – чтение, запись, выполнение. Группа владельцев – чтение, выполнение. Все остальные – чтение, выполнение
-rwsr-xr-x	4755	Установлен бит SUID. Владелец – чтение, запись, выполнение. Группа владельцев – чтение, выполнение с правами владельца. Все остальные – чтение, выполнение с правами владельца

Популярные ключи команды chmod:

Ключ	Описание
-v	Выводит отладочную информацию
-R	Рекурсивный режим. Изменять права доступа для всех подкаталогов и файлов

Команда chown: смена владельца файла и группы

Команда chown предназначена для смены владельца файла и/или группы. Первым аргументом команды является новый владелец, вторым – файл

или каталог, для которого нужно применить изменения. Через двоеточие в первом аргументе можно передать название новой группы.

Формат команды: chown [опции] [владелец]:[группа] файл

Для файла file.txt меняем владельца файла на user2:

```
[root@linuxbox ~]# chown user2 file.txt
```

Для файла file.txt меняем владельца и группу владельцев на user2 и group2 соответственно:

```
[root@linuxbox ~]# chown user2:group2 file.txt
```

Для каталога /dir и всех его подкаталогов и файлов меняем владельца и группу владельцев на user2 и group2 соответственно:

```
[root@linuxbox ~]# chown -R user2:group2 /dir
```

Для файла file.txt меняем только группу владельцев на group2:

```
[root@linuxbox ~]# chown  :group2 file.txt
```

Популярные ключи команды chown:

Ключ	Описание
-v	Выводит отладочную информацию
-R	Рекурсивный режим. Эффект команды chown будет распространяться на все файлы и подкаталоги выбранного каталога.

Команда umask: изменение режима доступа для создаваемых каталогов и файлов

Команда umask позволяет задать с какими правами доступа будет вновь создаваемый файл. Значение umask вычитается из числа 777 для каталогов, и из 666 для файлов, перед тем как быть назначенным. Например команда:

```
[root@linuxbox ~]# umask 022
```

приведет к тому, что созданный каталог будет иметь права 755 (rwxr-xr-x) а файл 644 (rw-r--r--). В таблице представлены возможные значения umask.

umask	Маска режима доступа
0	rwx
1	rw-
2	r-x
3	r--
4	-wx
5	-w-
6	--x
7	---

По умолчанию umask = 022, то есть на все создаваемые вами каталоги права доступа будут rwxr-xr-x а на файлы rw-r--r--

Посмотреть текущее значение umask можно вызвав команду без аргументов:

```
[root@linuxbox ~]# umask

0022
```

Если для создаваемых файлов вы хотите задать другой umask то просто отредактируйте ~/.bash_profile и добавьте туда соответствующую строку, например: “umask 077”.

Команда mkdir: создание каталогов

Команда mkdir (make directory) предназначена для создания каталогов.

Формат команды: mkdir [опции] каталог…

В качестве аргумента, команде mkdir передается имя каталога который нужно создать, например:

```
[root@linuxbox ~]# mkdir testdir
```

У команды mkdir есть пара полезных ключей. Ключ -m позволяет задать права доступа на каталог при его создание, например:

```
[root@linuxbox ~]# mkdir -m 700 testdir
```

Ключ -p будет полезен в ситуации, когда нужно создать родительский каталог если его не существует, например создаем следующею структуру каталогов одной командой:

```
[root@linuxbox ~]# mkdir -p /root/documents/fromwork
```

Команда rmdir: удаление каталога

Команда rmdir может удалить только пустой каталог.

Формат команды: rmdir [опции] каталог…

В качестве аргумента передается каталог который нужно удалить.

Пример использования:

```
[root@linuxbox ~]# rmdir ~/emptydir
```

В большинстве случаев вы будете использовать команду rm -rf

Команда rm: удаление файлов и каталогов

Команда rm позволяет удалять файлы и каталоги. В подавляющем большинстве ситуаций вы будете использовать именно её.

Формат команды: rm [опции] файл…

В качестве аргумента передается файл или каталог который нужно удалить. Благодаря ключу -r можно рекурсивно удалить все содержимое каталога. Ключ -f отключает постоянные запросы на подтверждение действий пользователя.

Удаляем выбранный файл без диалога подтверждения на удаление:

```
[root@linuxbox ~]# rm -f badfile.txt
```

Удаляем каталог badfiles в домашнем каталоге пользователя root и все его содержимое.

```
[root@linuxbox ~]# rm -rf ~/badfiles/
```

Команда cp: копирование файлов и каталогов

Команда cp (copy) позволяет скопировать файлы и каталоги.

Формат команды: cp [опции] [источник] [пункт_назначения]

Для работы команды нужно два параметра – “что копируем” и “куда копируем”.

Копируем файл file.txt в каталог /home/user/

```
[root@linuxbox ~]# cp file.txt /home/user/
cp: overwrite `/home/user/file.txt'? y
```

Копируем каталог myfiles со всем содержимым в /home/user/

```
[root@linuxbox ~]# cp -R myfiles/ /home/user/
```

Копируем содержимое каталога myfiles в /home/user

```
[root@linuxbox ~]# cp -R myfiles/* /home/user/
```

Команда mv: перемещение или переименование файлов и каталогов

Команда mv позволяет перемещать или переименовывать файлы и каталоги.

Формат команды: mv [опции] [источник] [пункт_назначения]

Для работы команды нужно два параметра – "что перемещаем" и "куда перемещаем".

Перемещаем файл file.txt в /home/user/

```
[root@linuxbox ~]# mv file.txt /home/user/
```

Перемещаем каталог mydir и переименовываем в newdir

```
[root@linuxbox ~]# mv mydir/ /home/user/newdir
```

Команда ln: создание ссылок между файлами

Команда ln позволяет создавать ссылки между файлами. Ссылки бывают двух видов: "мягкие" и "жесткие". Символическая, или "мягкая" ссылка обеспечивает возможность вместо путевого имени файла указывать псевдоним. Когда ядро при поиске файла сталкивается с символической ссылкой то оно извлекает из нее путевое имя. Для пользователя такой файл в большинстве ситуаций неотличим от того, на который он ссылается: операции чтения, записи и пр. над символьной ссылкой работают так, как если бы они производились непосредственно над тем файлом, на который указывает ссылка. Другой тип ссылки – "жесткий". Разница между "мягкими"

и "жесткими" ссылками состоит в том, что "жесткая" ссылка является прямой, то есть указывает непосредственно на индексный дескриптор файла в то время как "мягкая" указывает на файл по имени.

Формат команды: ln [опции] [цель] [имя_ссылки]

Создание символической ссылки происходит командой ln -s. Создаем ссылку linktotestfile на файл testfile.txt:

```
[root@linuxbox ~]# ln -s testfile.txt linktotestfile
```

Ссылку можно удалить командой unlink:

```
[root@linuxbox ~]# unlink linktotestfile
```

Создание "жесткой" ссылки происходит командой ln без опций. Делаем "жесткую" ссылку на файл file.txt

```
[root@linuxbox ~]# ln file.txt file2.txt
```

А теперь посмотрим на номер inode обоих файлов.

```
[root@linuxbox ~]# ls -li
786434 -rw-r--r-- 2 root root 0 Aug 7 21:20 file2.txt
786434 -rw-r--r-- 2 root root 0 Aug 7 21:20 file.txt
```

Он совпадает.

Команда touch: изменение времени доступа и модификации файла

Команда touch имеет две функции. Первая – это изменение времени доступа к файлу (atime, ключ -a) и времени последней модификации (mtime, ключ -m). Вторая функция – если файла не существует то он будет создан.

Формат команды: touch [опции] файл

Создание пустого файла:

```
[root@linuxbox ~]# touch ~/file.txt
```

Команда df: статистика по использованию файловой системы

Команда df предназначена для отображения статистики занятости файловой

системы. Если ФС не указана явно то статистика отображается для всех ФС в системе.

Формат команды: df [опции] [файл]…

Выводим статистику по всем файловым системам:

```
[root@linuxbox ~]# df -h
Filesystem                      Size  Used Avail Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
                                7.2G  2.5G  4.3G  37% /
/dev/hda1                       99M   12M   82M  13% /boot
tmpfs                           125M     0 125M   0% /dev/shm
```

Самый популярный ключ -h который выводит статистику в удобном для человека формате (Мб, Гб и т.д.)

Команда du: статистика по использованию дискового пространства

Команда du позволяет выяснить сколько места занимает определенный файл или каталог. Самые популярные ключи – это -h который представляет статистику в удобном для человека формате (Мб, Гб и т.д.) и -s который производит суммирование результатов подкаталогов и файлов в них и выводит на экран только конечный результат.

Формат команды: du [опции] [файл]

Выясняем сколько места занимает каталог /usr/share/

```
[root@linuxbox ~]# du -sh /usr/share/
1.4G    /usr/share/
```

Команда echo: отображение строки текста

Команда echo выводит строку текста на стандартный вывод.

Формат команды: echo [опции] [строка]

Выводим строку текста на экран:

```
[root@linuxbox ~]# echo 'blablabla'
blablabla
[root@linuxbox ~]#
```

С помощью ключа -n можно не делать переход на новую строку:

```
[root@linuxbox ~]# echo -n 'blablabla'
blablabla[root@linuxbox ~]#
```

Вывод команды echo можно перенаправить в файл. Записываем строку в файл file.txt:

```
[root@linuxbox ~]# echo 'blablabla' > file.txt
```

Команда wc: статистика по количеству строк, слов и байтов в файле

Команда wc выводит статистику по количеству строк, слов или байтов в файле.

Формат команды: wc [опции] [файл]…

-c – количество байтов в файле

-m – количество символов в файле

-l – количество строк в файле

-L – длина самой длинной строки

-w – количество слов в файле

Пример использования – выводим количество строк в файле:

```
[root@linuxbox ~]# wc -l /etc/login.defs
58 /etc/login.defs
```

Команда head: считывание первых строк файла на стандартный вывод

Команда head позволяет вывести произвольное количество строк с начала файла на стандартный вывод. По умолчанию выводятся первые 10 строк. Ключ -n позволяет задать другое количество.

Формат команды: head [опции] [файл]…

Выводим на экран первые 20 строк файла /etc/login.defs:

```
[root@linuxbox ~]# head -n 20 /etc/login.defs
```

Команда tail: считывание последних строк файла на стандартный вывод

Команда tail позволяет вывести произвольное количество строк с конца файла на стандартный вывод. По умолчанию выводятся последние 10 строк.

Ключ -n позволяет задать другое количество.

Формат команды: tail [опции] [файл]…

Выводим на экран последние 20 строк файла /etc/login.defs:

```
[root@linuxbox ~]# tail -n 20 /etc/login.defs
```

Команда less: построчный или поэкранный просмотр файла

Команда less позволяет просматривать файл по строкам, клавишами вверх-вниз, или переключая экраны, клавишами Page Up-Page Down. Программа пришла на замену команде more и гораздо удобнее в использование.

Формат команды: less файл…

Укажите в качестве аргумента название файла чтобы его просмотреть:

```
[root@linuxbox ~]# less /etc/login.defs
```

Выйти из файла можно нажатием клавиши q

Подробная справка:

```
[root@linuxbox ~]# less --help
```

Команда find: поиск файлов

Команда find служит для поиска файлов и каталогов.

Формат команды:

find [путь_поиска] [опция_поиска] [значение] [опция_действия]

Популярные опции find

Опция	Описание
-name	Имя файла или каталога который необходимо найти
-type	Тип объекта -type f – обычный файл -type d – каталог -type b – блочное устройство -type c – символьное устройство -type p – именованный канал -type l – символьная ссылка -type s – сокет
-size	Использовать размер файлов как критерий поиска. Например -size +1024 – файлы размером больше чем 1024*512 байт
-mtime	Дата модификации
-ctime	Время последнего изменения атрибутов файла
-atime	Время последнего обращения к файлу
-print	Просто напечатать маршрутное имя файла
-exec	Выполнить определенную команду для найденных объектов

Примеры использования find

Найти все файлы с именем passwd. Поиск начинается с корня файловой системы. По умолчанию на экран будет выведено маршрутное имя к файлу, поэтому опцию -print добавлять не требуется.

```
[root@linuxbox ~]# find / -name passwd

/etc/pam.d/passwd

/etc/passwd

/usr/bin/passwd

/usr/share/doc/nss_ldap-253/pam.d/passwd

[root@linuxbox ~]#
```

Поиск с использованием шаблонов (команда find использует стандартные шаблоны shell). Поиск все файлов начинающихся на syslogd

```
[root@linuxbox ~]# find / -name syslogd*

/selinux/booleans/syslogd_disable_trans

/var/run/syslogd.pid

/usr/share/man/man8/syslogd.8.gz

/usr/share/logwatch/default.conf/services/syslogd.conf

/usr/share/logwatch/scripts/services/syslogd

/sbin/syslogd

[root@linuxbox ~]#
```

Поиск только каталогов. Для поиска только каталогов используется опция -type d. Найдем все каталоги с именем pam.

```
[root@linuxbox ~]# find / -name pam -type d

/usr/share/locale/pam

[root@linuxbox ~]#
```

Поиск файлов и удаление. Результат выполнения команды find можно через конвейер передать в другую программу. Все файлы с именем core в каталоге /tmp будут переданы на удаление команде rm. Команда xargs служит для передачи аргументов.

```
[root@linuxbox ~]# find /tmp -name core -type f -print | xargs /bin/rm -f
```

Поиск файлов с определенными правами доступа. В каталоге /etc/ ищем файлы с правами доступа 777.

```
[root@linuxbox ~]# find /etc/ -perm 777 -type f

/etc/badfile

[root@linuxbox ~]#
```

В домашнем каталоге пользователя root найти и удалить файл badfile.

```
[root@linuxbox ~]# find ~/ -name badfile -exec rm -f {} \;
```

Файлы специальных устройств

Файлы /dev/random и /dev/urandom используются для генерации случайных чисел. Они предоставляют интерфейс к системному генератору случайных чисел который выводит шумы из драйверов устройств и других источников в "хаотичный" пул (entropy pool). Генератор сохраняет необходимое количество битов шума в этом пуле и формирует из него случайные числа.

Файл /dev/null является так называемой "черной дырой". Любая информация направленная в этот файл безвозвратно удаляется. Файл используется для поглощения не нужного вывода программ.

```
[root@linuxbox ~]# команда > /dev/null 2>&1
```

Из символьного устройства /dev/zero можно прочитать только нули. Устройство может использоваться для создания файла нужного размера.

```
[root@linuxbox ~]# dd if=/dev/zero of=file.iso bs=1024 count=1024

1024+0 records in

1024+0 records out

1048576 bytes (1.0 MB) copied, 0.00673231 seconds, 156 MB/s

[root@linuxbox ~]# ls -lh

total 1.1M

-rw-r--r-- 1 root root 1.0M Sep 16 17:34 file.iso

[root@linuxbox ~]#
```

Модуль 5. Терминал и командный интерпретатор BASH

Linux является многопользовательской ОС и одновременно в ней могут работать десятки пользователей.

Для организации параллельной работы применяются так называемые консоли, виртуальные и графические. В системе присутствует 6 виртуальных консолей и как минимум 1 графическая.

Виртуальные консоли – это несколько параллельно выполняемых операционной системой программ – getty (от get teletype). В CentOS используется одна из реализаций getty – mingetty.

```
[root@linuxbox ~]# ps aux | grep mingetty
root     2806  0.0  0.1   1656   432 tty1    Ss+  03:44   0:00 /sbin/mingetty tty1
root     2807  0.0  0.1   1656   436 tty2    Ss+  03:44   0:00 /sbin/mingetty tty2
root     2808  0.0  0.1   1656   436 tty3    Ss+  03:44   0:00 /sbin/mingetty tty3
root     2809  0.0  0.1   1656   436 tty4    Ss+  03:44   0:00 /sbin/mingetty tty4
root     2810  0.0  0.1   1656   436 tty5    Ss+  03:44   0:00 /sbin/mingetty tty5
root    17903  0.0  0.2   3912   668 pts/0   R+   04:13   0:00 grep mingetty
```

После того как пользователь подключился к консоли, управление передается программе login, задача которой аутентифицировать пользователя.

Виртуальные консоли обеспечивают текстовый интерфейс и переключаться между ними можно по нажатию клавиш Ctrl+Alt+[F1-F6]. Виртуальные консоли обозначаются как ttyN, где N – это номер консоли. Переключиться в графическую консоль можно нажатием клавиш Ctrl+Alt+F7. Переключиться из графической консоли в одну из текстовых консолей можно нажатием клавиш Ctrl+Alt+[F1-F6].

Терминальный доступ и управляющие символы

Терминал – это устройство последовательного ввода/вывода информации предназначенное для взаимодействия пользователя с системой. Терминал передает последовательно только символы. Существует два типа символов: текстовые – это обычные текстовые символы из которых образуются слова

или команды, и управляющие, которые позволяют отдавать специальные команды. Терминалом может быть внешнее устройство или программное обеспечение, например ssh-клиент, благодаря которому в систему можно подключиться через Интернет. Вообще понятие терминала достаточно широкое и обычно под этим подразумевается любая точка входа в систему пользователя. Любые текстовые символы не будут переданы в систему до тех пор пока не будет нажата клавиша <**Enter**>. Пример текстовых символов переданных в систему показан ниже.

```
[root@linuxbox ~]# whoami

root

[root@linuxbox ~]#
```

Команда whoami благодаря командному интерпретатору сообщила системе, что в ответ на запрос необходимо вернуть текущею учетную запись из под которой происходит работа. Что собственно и было сделано. Пример управляющих символов показан ниже.

```
[root@linuxbox ~]# cat

text1 <Enter>

text1

text2 <Enter>

text2

[Ctrl+C или Ctrl+D]



[root@linuxbox ~]#
```

Команда cat запущенная без аргументов построчно считывает данные со стандартного ввода (клавиатура) и отображает их на стандартном выводе (дисплей). После набора строки и нажатия [Enter] команда cat выводит на консоль строку которая была набрана и предоставляет новую для ввода новых данных. Вот почему строки с текстом “text1” и “text2” были продублированы. Для того чтобы завершить ввод необходимо нажать комбинацию клавиш Ctrl+C или Ctrl+D. Эти комбинации клавиш и являются управляющими, так как диктуют системе сделать определенные действия.

Некоторые управляющие символы (они входят в GNU Readline Library) представлены ниже.

CTRL-команды (комбинации при нажатой клавише Ctrl, принято обозначать символом "^"):

Ctrl + a – переместиться в начало строки
Ctrl + b – переместиться на один символ назад
Ctrl + c – прервать текущий процесс (сигнал SIGINT)
Ctrl + d – выход из командного интерпретатора. Если курсор находится под каким либо символом, то символ будет удален
Ctrl + e – переместиться в конец строки
Ctrl + f – переместиться на один символ вперед
Ctrl + g – выход из режима дополнения
Ctrl + h – удаление символа слева (аналог backspace)
Ctrl + i – аналог Tab
Ctrl + j – сброс параметров терминала
Ctrl + k – удалить все до конца строки
Ctrl + l – очистка экрана
Ctrl + m – аналог клавиши "Enter"
Ctrl + n – следующая команда в истории команд
Ctrl + p – предыдущая команда в истории команд
Ctrl + q – запустить процесс
Ctrl + r – перевести командную строку в режим поиска команды по истории
Ctrl + s – остановить процесс
Ctrl + t – поменять местами два символа
Ctrl + u – удалить все от курсора и до начала строки
Ctrl + v – преобразует следующую клавишу в её символьное обозначение (Ctrl+D – ^D, Ctrl + c – ^C и т.д.)
Crtl + w – удалить последнее слово
Ctrl + x дважды – прыжок между началом строки и текущей позицией курсора
Ctrl + y – вставить все что было удалено с помощью Ctrl + K
Ctrl + z – перевести процесс в фоновый режим

Alt-команды

Alt + < – к первой команде в истории команд
Alt + > – к последней команде в истории команд
Alt + ? – показать весь список вариантов дополнения
Alt + * – вставить все возможные варианты дополнения
Alt + / – попытаться дополнить имя файла (из имеющихся в текущем каталоге)
Alt + . – вставить последний аргумент из предыдущей команды
Alt + b – на одно слово влево
Alt + c – сделать первую букву слова заглавной (и перейти к следующему слову)

Alt + d – удалить все символы от текущей позиции до конца слова
Alt + f – переместиться вправо на слово
Alt + l – сделать первую букву слова строчной и перейти к следующему слову
Alt + n – искать по истории команд после полного ввода и нажатия [Enter]
Alt + p – искать по истории назад
Alt + r – очистить всю строку
Alt + t – поменять слова местами
Alt + u – сделать все буквы заглавными от текущей позиции до конца слова
Alt + BackSpace – удалить все символы от текущей позиции до начала слова

Tab-команды

2T означает двойное нажатие Tab
2T – все доступные команды
net<2T> – все доступные команды начинающиеся на net
/2T – все каталоги. Для текущего необходимо набрать ./2T
*2T – каталоги, кроме скрытых
~2T – все пользователи, присутствующие в /etc/passwd
~uT – все пользователи, присутствующие в /etc/passwd, начинающиеся на u
$2T – все системные переменные
@2T – все записи в /etc/hosts
=2T – вывод наподобие ls или dir

Esc-команды

Esc + d – удалить от курсора до конца слова
Esc + f – вправо на слово
Esc + b – влево на слово
Esc + t – поменять местами слова (слева направо)

Другие команды

!! – выполнить последнюю команду в истории
!com – выполнить последнюю команду в истории, начинающуюся на com
!a:p – напечатать последнюю команду в истории, начинающуюся на a
!n – выполнить n-ную команду в истории
!$ – последний аргумент последней команды
!^ – первый аргумент последней команды
^abc^xyz – заменить abc на xyz в последней команде и выполнить результат

Дополнительная информация: man readline, man bash

Командный интерпретатор BASH

Командный интерпретатор – это программа которая получая команды от пользователя переводит их в понятный для компьютера язык (интерпретирует). Самым распространенным интерпретатором является bash. Командный интерпретатор также известен как командная строка или shell.

Строка приглашения

Командная строка содержит специальный символ по которому можно понять, привилегированный это shell или нет. Привилегированный пользователь – это пользователь с правами администратора, то есть имеющий полный доступ к системе. Обычному пользователю разрешается работать в домашнем каталоге и выполнять распространенные команды без влияния на систему в целом. Ниже показаны два символа которые присутствуют в любой командной строке и позволяют выяснить её тип.

– командная строка имеет права администратора

$ – командная строка обычного пользователя системы

Командная строка обычного пользователя:

```
[user@linuxbox root]$
```

Командная строка привилегированного пользователя:

```
[root@linuxbox ~]#
```

Версия Bash и его расположение

```
[root@linuxbox ~]# echo $BASH_VERSION
3.00.15(1)-release
```

```
[root@linuxbox ~]# echo $BASH
/bin/bash
```

Формат команды

Команды bash обычно имеют следующий формат:

<имя команды> <флаги> <аргумент(ы)>

Например:

ls -la /usr

Где ls – команда, -la – флаги («-» признак того, что сейчас будут перечисляться флаги, далее идут флаги l и a), /usr – аргумент.

Формат выдерживается не всегда, в большинстве порядок перечисления зависит от команды. Опции могут состоять как из одной буквы так и из целого слова.

Выполнение команд

Если программа находится в текущем каталоге то выполнить ее можно с помощью команды source или ".", например:

```
[root@linuxbox ~]#./myscript.sh
```

Если программа находится в одном из каталогов описанных в переменной $PATH то достаточно набрать ее название. Список каталогов можно посмотреть с помощью echo.

```
[root@linuxbox ~]# echo $PATH

/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
```

Конфигурационные файлы

Как и у большинства команд, у bash тоже есть свои конфигурационные файлы. Их два типа — локальные и глобальные. Локальные – это файлы в домашних каталогах пользователей, они могут вносить в них изменения для своих личных целей. Глобальные распространяются на всех пользователей системы.

Локальные файлы конфигурации:

.bash_logout – Сценарий выполняется при выходе из системы. Сюда можно добавить удаление временных файлов или что-то другое.

.bash_history – Файл с историей команд, которые были введены из под этой учетной записи.

.bash_profile – Сценарий выполняется при Вашем подключение к системе. В некоторых системах используется .bash_login

.bashrc – Этот сценарий выполняется каждый раз когда открывается новый shell. Имеется ввиду запуск нового процесса shell. В графическом режиме X Window это почти всегда используется.

Глобальные файлы конфигурации:

/etc/profile

/etc/bashrc

Локальные файлы конфигурации имеют больший приоритет. То есть при описание переменной в /etc/profile и ~/.bash_profile будет использоваться переменная из последнего файла.

Переменные окружения – это опции с которыми оперирует bash и обращающиеся к ним пользователи. Глобальные можно посмотреть командой env, локальные можно увидеть с помощью команды typeset. Помимо значений переменных вы найдете там массу полезного.

История команд

Все вводимые команды сохраняются в историю команд, которую можно просмотреть командой history. Количество строк которое будет сохранено определяется переменной $HISTSIZE и по умолчанию имеет значение 1000.

```
[root@linuxbox ~]# echo $HISTSIZE
1000
[root@linuxbox ~]#
```

Можно вывести произвольное количество строк передав в качестве аргумента нужное число.

```
[root@linuxbox ~]# history 5
 642  netstat -nr
 643  echo $HISTSIZE
 644  cat /etc/resolv.conf
 645  env
```

```
 646  history 5
[root@linuxbox ~]#
```

Также, историю своих команд можно просмотреть в файле ~/.bash_history только нужно учитывать, что команды текущего сеанса работы отобразятся в нем только после завершения работы.

history	Отображает всю историю команд
history N	Отображать последние N строк
history -d N	Удалить строку N, которая к примеру, могла содержать в себе пароль
!!	Выполнить последнею введенную команду
!N	Выполнить команду N
!string	Самая последняя команда, которая начинается со строки string

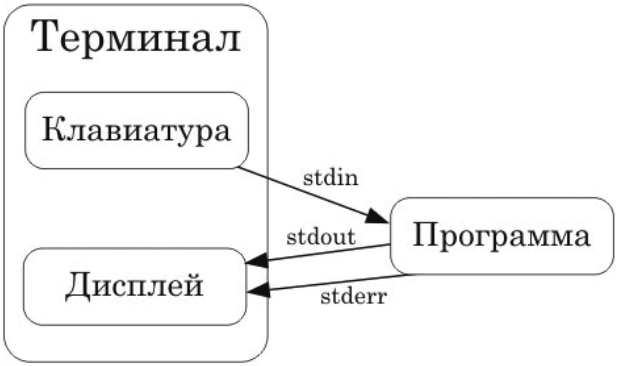
Перенаправление ввода/вывода

Файловый дескриптор – это неотрицательное целое число назначаемое операционной системой открываемому процессом файлу.

В системе постоянно открыто 3 файловых дескриптора и номера их не меняются:

Номер дескриптора	Название	Описание
0	stdin	Через этот дескриптор осуществляется стандартный ввод, яркий пример – ввод данных с клавиатуры.
1	stdout	Стандартный вывод, обычно на дисплей монитора.
2	stderr	Дескриптор используется для вывода сообщений об ошибках и отладочных сообщений. Обычно совмещен с stdout.

Зачем нужны stdin и stdout я думаю понятно из таблицы выше. Про stderr поговорим подробнее. stderr служит для вывода ошибок программ и отладочных сообщений. Как правило все выводится на stdout но для удобства мы поступим иначе. Приведу пример чтобы вы лучше поняли его предназначение.




```
1       [root@linuxbox ~]# ls ~
2       file1 file2
3       [root@linuxbox ~]# cat file1 file3
4       [вывод урезан]
        cat: file3: No such file or directory
5       [root@linuxbox ~]#
```

Строка 1: мы запросили вывод содержимого каталога;

Строка 2: в домашнем каталоге видим файлы file1 и file2;

Строка 3: командой cat пытаемся вывести содержимое файлов на stdout;

Строка 4: помимо вывода содержимого file1 на экран будет выведена ошибка, что file3 не найден, что собственно строка 2 и подтверждает;

Строка 5: после того как вывод на stdout завершен, пользователю предоставляется строка для ввода новых команд.

Теперь перенаправим все сообщения об ошибках в файл. Напомню, номер stderr – 2.

```
[root@linuxbox ~]# cat file1 file3 2> err_file
[вывод урезан]
[root@linuxbox ~]#
```

Теперь на экране мы увидели содержимое файла file1 а вот никаких ошибок нет, потому как их вывод мы перенаправили в файл err_file

```
[root@linuxbox ~]# cat err_file

cat: file3: No such file or directory

[root@linuxbox ~]#
```

Перенаправление ошибок может быть полезно в работе автономных программ. Когда программа работает 24 часа в сутки в отличие от администратора системы. Если ошибки выводить на экран то они быстро затеряются в общем выводе (зависит от размера буфера экрана), но при перенаправление в файл их всегда можно просмотреть.

Возможные перенаправления	
Откуда	Куда
stdout	файл
stdout	stderr
stdout и stderr	stdout
stdout и stderr	stderr
stdout и stderr	файл
stderr	файл
stderr	stdout

Результаты выполнения различных команд (перенаправление вывода) можно направлять в файлы, например:

```
[root@linuxbox ~]# date > date.txt
```

А примере выше если файла date.txt не существовало то он будет создан а если был с данными то они будут замещены новыми. Чтобы добавить данные к существующим необходимо использовать “>>”:

```
[root@linuxbox ~]# ls -l >> file.txt
```

Перенаправление ввода происходит символом “<”, например:

```
[root@linuxbox ~]# vi <(ls /etc/)
```

Также, в командном интерпретаторе возможна подстановка вывода одной команды в качестве аргумента для другой, например:

```
[root@linuxbox ~]# echo -e «`ls -l /etc/[^a-w]*.conf`\n»
```

```
-rw-r--r-- 1 root root 585 Jan 21  2009 /etc/yp.conf

-rw-r--r-- 1 root root 333 Jan 22  2009 /etc/yum.conf


[root@linuxbox ~]#
```

Конвейеры

Средство, объединяющее стандартный выход одной команды со стандартным входом другой, называется конвейером и обозначается вертикальной чертой "|". Давайте посмотрим на примере.

```
[root@linuxbox ~]# ls /usr/share/ | wc -l

191

[root@linuxbox ~]#
```

Командой ls мы запрашиваем список файлов и каталогов, но результат выполнения команды передается не на стандартный вывод, как это обычно бывает, а в качестве аргумента для команды wc -l, которая позволяет собрать различную статистику по файлу. В данном случае мы подсчитываем количество строк вывода команды ls, что позволит выяснить как много файлов в текущем каталоге.

То есть как мы видим, конвейеры это последовательная обработка данных набором команд прежде чем будет получен результат. Благодаря конвейерам, с данными возможны сложные манипуляции, которых не добиться стандартными средствами.

В конвейер можно объединять произвольное количество команд, главное чтобы пользователь знал, что он делает.

Builtin-команды

Не многие знают что bash поставляется с набором builtin-команд, то есть с командами которые уже встроены в него. Многие из этих команд являются аналогами системных команд поставляемых с ОС. К примеру в bash версии 3.2.25 – 58 команд. Посмотреть их список можно командой enable.

```
[root@linuxbox ~]# enable
```

```
enable .

enable :

enable [

enable alias

enable bg

enable bind

enable break

enable builtin

enable caller

…

[вывод команды урезан]
```

При запуске команды первым делом выполняется builtin-команда и только потом если в builtin-списке она не найдена, происходит её поиск в $PATH, переменной содержащей набор каталогов в которых последовательно ищется исполняемый файл запускаемой пользователем команды. Давайте я приведу пример.

```
[root@linuxbox ~]# pwd --help

bash: pwd: --: invalid option

pwd: usage: pwd [-LP]

[root@linuxbox ~]# /bin/pwd --help

Usage: /bin/pwd [OPTION]

Print the full filename of the current working directory.


      --help display this help and exit

      --version output version information and exit


NOTE: your shell may have its own version of pwd, which usually supersedes
```

```
the version described here. Please refer to your shell's documentation
for details about the options it supports.

Report bugs to <bug-coreutils@gnu.org>.
```

С виду одна и та же команда породила разный вывод, почему ? ответ – это две разные команды. Первая является builtin-командой bash, вторая поддерживается в рамках проекта GNU. Теперь давайте отключим builtin - команду и посмотрим что произойдет.

```
[root@linuxbox ~]# enable -n pwd
[root@linuxbox ~]# pwd --help
Usage: pwd [OPTION]
Print the full filename of the current working directory.
…
[вывод команды урезан]
```

Видите. Сразу начала выполняться “системная” команда. Теперь включим обратно pwd из пакета bash, так как она меня всем устраивает.

```
[root@linuxbox ~]# enable pwd
```

Быстрая справка:

Команда enable без аргументов выводит список включенных команд.

enable -a – выводит список всех команд (включенных и выключенных)

enable -n – выводит список отключенных команд

enable -n <builtin-команда> – отключить команду

enable <builtin-команда> – включить команду

Переменные

Для начала представим схему взаимодействия процессов порождаемых bash в системе:

Родительский процесс --> дочерний процесс --> дочерний процесс

Процессы могут последовательно передавать информацию о переменных на более низкий уровень но не наоборот!

Переменные окружения

Переменные окружения служат для настройки многих программ и системы в целом. Они доступны всем пользователям и в той или иной степени затрагивают их работу. Чтобы создать новую переменную достаточно в командной строке задать ее название и значение, например:

```
[root@linuxbox ~]# value=10
```

Но в этом случае переменная будет доступна только в рамках текущего сеанса. Если вы планируете ее использовать из других программ то нужно сделать так называемое экспортирование. Для этого есть команда export.

```
[root@linuxbox ~]# export value=10
```

Для удаления переменной используется команда unset.

```
[root@linuxbox ~]# unset value
```

Переменные окружения можно посмотреть командой env. Также, командой typeset (или declare) можно посмотреть переменные созданные пользователем в рамках его сеанса работы, но не экспортированных "на вверх". Экспортированная переменная может использоваться в скриптах, например:

```
[root@linuxbox ~]# export value=10
```

Далее пишем такой скрипт и запускаем его.

```
#!/bin/bash

echo $value
```

```
[root@linuxbox ~]# chmod 755 test.sh
[root@linuxbox ~]#./test.sh
10
[root@linuxbox ~]#
```

А теперь создадим переменную но не экспортируем её.

```
[root@linuxbox ~]# unset value

[root@linuxbox ~]# value=10

[root@linuxbox ~]# echo $value

10

[root@linuxbox ~]#./test.sh


[root@linuxbox ~]#
```

Видите, команда выполненная из оболочки где переменная была создана показала значение переменной а скрипт нечего не показал. Наш shell из которого мы набирали команды является родительским процессом для скрипта. Поэтому мы могли разрешить воспользоваться нашей переменной скрипту но мы этого не сделали.

Основные переменные окружения перечислены в таблице.

Переменная	Описание
HOSTNAME	Имя хоста
SHELL	Имя командного интерпретатора
USER	Имя зашедшего в систему пользователя
PATH	В это переменной хранится список каталогов для поиска команд задаваемых в командной строке. Если команда находится в другом месте то необходимо указывать полный путь к ней
PWD	Текущий рабочий каталог
HOME	Домашний каталог пользователя
UID	Цифровой идентификатор текущего пользователя

Локальные переменные

Они доступны в рамках работы блока кода скрипта или функции.

Каждый процесс имеет свою среду исполнения, в рамках этой среды ему доступен некоторый набор переменных и в рамках этой среды он может создавать локальные переменные или наследовать их от родительского процесса.

Дочерний процесс не может влиять на переменные родительского процесса

но наследует их из него.

Скрипт написанный пользователем - это дочерний процесс. Все локальные переменные скрипта могут использоваться только им самим, но он также может пользоваться переменными окружения, хотя и не может влиять на них.

В Bash переменная $0 – содержит имя запускаемого файла, $1 – первый аргумент $2 – второй аргумент и так далее, начиная с 10-го аргумента его необходимо помещать в фигурные скобки {}, пример ${10}.

Скрипт test.sh:

```
#!/bin/bash
echo «Название этого файла - $0»
echo «Первый аргумент - $1»
echo «Второй аргумент - $2»
```

Выведет на экран:

```
[root@linuxbox ~]#./test.sh start stop
Название этого файла ./test.sh
Первый аргумент - start
Второй аргумент - stop
```

$# – содержит в себе информацию о количестве аргументов командной строки

$* – содержит в себе все аргументы командной строки в виде одной строки, переменную необходимо заключать в кавычки

$@ – то же самое что и $*, но каждый параметр представлен отдельной строкой, переменную необходимо заключать в кавычки

$- – показывает список флагов переданных сценарию

$! – показывает PID последнего процесса запущенного в фоновом режиме

$_ – показывает последний аргумент предыдущей команды

$? - показывает код возврата команды, функции или скрипта

$$ – показывает PID процесса нашего сценария

$PS1 – это приглашение командной строки

$PS2 – второй тип приглашения командной строки, выводится когда от пользователя ожидается дополнительный ввод. Отображается как ">"

$PS3 – третий тип приглашения выводится когда пользователь должен сделать выбор в операторе select

$PS4 – приглашение четвертого уровня, выводится в начале каждой строки вывода когда сценарий вызывается с ключом -x. Отображается как "+"

Шаблонные символы

Шаблонные символы позволяют одним словом описывать множество символьных последовательностей, удовлетворяющих определенным требованиям.

* – любая последовательность символов или пустая последовательность

? – любой один символ

[] – символьный класс. Один из символов перечисленных в квадратных скобках. Если первый символ "^" то наоборот, любой символ не перечисленные в квадратных скобах. Через дефис можно указать диапазон символов для поиска совпадений

Выводим список файлов в каталоге /etc начинающихся на host

```
[root@linuxbox ~]# ls -l /etc/host*
-rw-r--r-- 1 root root   17 Jul  23  2000 /etc/host.conf
-rw-r--r-- 1 root root 187 Jul  28  02:10 /etc/hosts
-rw-r--r-- 1 root root 161 Jan  13  2000 /etc/hosts.allow
-rw-r--r-- 1 root root 347 Jan  13  2000 /etc/hosts.deny
[root@linuxbox ~]#
```

Выводим список файлов в каталоге /etc не начинающихся на a-w и имеющих расширение .conf

```
[root@linuxbox ~]# ls -l /etc/[^a-w]*.conf
-rw-r--r-- 1 root root  585 Jan  21  2009 /etc/yp.conf
```

```
-rw-r--r-- 1 root root      333      Jan      22      2009      /etc/yum.conf
[root@linuxbox ~]#
```

Выводим список файлов в каталоге /etc имеющих в своем название 7 любых символов и расширение .conf

```
[root@linuxbox ~]# ls -l /etc/???????.conf
-rw-r--r-- 1 root root    18484      Feb      28      02:00      /etc/dnsmasq.conf
-rw-r--r-- 1 root root      658      Mar       6      02:55      /etc/initlog.conf
-rw-r--r-- 1 root root     2506      May      25      2008       /etc/libuser.conf
-rw-r--r-- 1 root root     4453      May      24      2008       /etc/oddjobd.conf
-rw-r--r-- 1 root root       12      Jan      6       2007       /etc/pam_smb.conf
-rw-r--r-- 1 root root      973      Sep 1    8       2008       /etc/prelink.conf
```

Метасимволы

Метасимволы – это специальные символы, выполняющие роль разделителей слов (аргументы командной строки и названия команд). Некоторые метасимволы также играют роль знаков препинания. К метасимволам относятся:

| & ; ( ) < > пробел табулятор

Иногда требуется включить метасимвол в состав слова, избежав его использования в качестве разделителя слов. Для этого перед метасимволом ставится обратная косая черта (\), которая и означает, что непосредственно следующий за ней специальный символ должен быть лишен своего специального значения и воспринят "буквально". Операция подстановки косой черты называется экранированием.

Символ "|" позволяет организовать конвейер. Средство, объединяющее стандартный выход одной команды со стандартным входом другой.

Для команды или команд помещенных в скобки ("()"), будет создана своя копия командного интерпретатора, в которой они будут выполняться.

Символы "<" и ">" позволяют сделать перенаправления вывода.

Запрашиваем содержимое файла /etc/host.conf но вывод происходит не на монитор пользователя а в файл file.txt

```
[root@linuxbox ~]# cat /etc/host.conf > file.txt
```

В текстовый редактор vi загружаем список файлов и подкаталогов каталога /etc

```
[root@linuxbox ~]# vi <(ls /etc/)
```

Символ ";" используется для перечисления списка команд в командной строке для последовательного выполнения. Команда2 начнет выполняться после завершения команды1.

команда1; команда2

Символ "&" позволяет выполнять несколько команд параллельно. Если используется «&», то команда, сопровождаемая этим знаком, выполняется в фоновом режиме, а выполнение следующей команды начинается немедленно.

команда1& команда2

Комбинация символов "&&" – это условное выполнение команд. Команда2 будет выполнена только в том случае, если команда1 успешно завершила свою работу.

команда1 && команда2

Комбинация символов "||" – это условное выполнение команд. Команда2 будет выполнена только в том случае, если команда1 завершится неудачно.

команда1 || команда2

Путевые символы

Это символ корня "/", откуда начинается путь к любому каталогу или файлу и символ "~" обозначающий домашний каталог (эквивалентен переменной $HOME).

Символы управления переменными

Символ "=" позволяет присвоить значение переменной, символ "$" позволяет подставить значение переменной.

Команда echo и ESC-символы

Команда echo выводит на терминал список своих аргументов, например:

```
[root@linuxbox ~]# echo just text

just text

[root@linuxbox ~]#
```

Также, команда echo может интерпретировать так называемые ESC-символы. Для этого ее надо использовать с ключом -e, например:

```
[root@linuxbox ~]# echo -e «just text\n\n»

just text



[root@linuxbox ~]#
```

Символ \n позволяет добавить новую строку после вывода, что мы и сделали (добавили две пустых строки). Другие ESC-символы перечислены в таблице.

\a	Звуковой сигнал
\b	Удаление последнего символа
\c	Не добавлять символ новой строки
\f	Перевод страницы (очищает экран)
\n	Новая строка
\r	Перевод каретки
\t	Горизонтальная табуляция

Опция -n команды echo позволит не добавлять новую строку во время вывода.

```
[root@linuxbox ~]# echo -n just text

just text[root@linuxbox ~]#
```

Модуль 6. Пользователи и группы

Управление пользователями и группами является ключевым моментом в работе с Linux. Пользователи в системе принадлежат к той или иной группе, задача которой распределять пользователей по тем или иным критериям. Пользователь может принадлежат как минимум к одной группе. Пользователь автоматически получает права владельца тех файлов которые создает. Изменить права доступа может только владелец файла или пользователь root.

В Linux используется механизм user private group (UPG) когда для каждого пользователя автоматически создается группа с таким же именем. Это облегчает управление группами.

Аккаунты могут быть созданы как для физических пользователей так и для определенных программ.

У каждого пользователя и группы есть уникальный 32-х разрядный цифровой номер – UID (user ID) и GID (group ID). По умолчанию UID новых пользователей начинается с 500 (до 500 используются для системных служб) а заканчивается на 60000. Эти опции можно изменить в файле /etc/login.defs

В ОС Linux есть два вида пользователей:

1. Суперпользователь, администратор или просто root. Его UID всегда равен 0. Исторически сложилось, что домашний каталог пользователя root расположен в /root

2. Обычные пользователи. Их UID обычно начинается с 500 а домашний каталог имеет путь вида /home/<имя пользователя>

Работу пользователей в Linux регулируют несколько файлов. В файле /etc/passwd находится список всех пользователей системы. Раньше в нем также хранились пароли, но с целью повышения безопасности сейчас используется файл /etc/shadow и MD5 хеширование. Для хранения списка групп есть файл /etc/group, для паролей используется файл /etc/gshadow. В случае, если учетная запись используется только для приложений работающих в системе, то имеет смысл отключить возможность подключения в систему такой учетной записи. Для этого в качестве shell (7-е поле в файле /etc/passwd) необходимо указать /sbin/nologin.

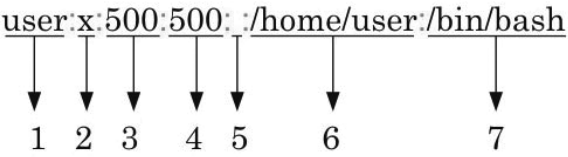
/etc/passwd

Файл является базой всех пользователей в системе. Это обычный текстовый

файл, где одна строка описывает одного пользователя. Каждая строка состоит из 7 полей разделенных символом ":".

```
login:password:UID:GID:GECOS:home:shell
```

Например:

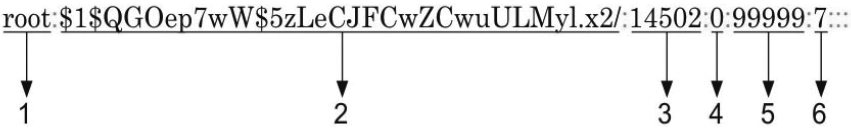


1. Имя пользователя;
2. Символ "x" обозначает, что используется shadow-файл для хранения пароля;
3. UID;
4. GID;
5. Поле GECOS (номер телефона, адрес, полное имя и т.д.);
6. Домашний каталог пользователя;
7. Командный интерпретатор.

В целях повышения безопасности пароли не хранятся в этом файле. Для этого есть файл /etc/shadow

/etc/shadow

В этом файле хранятся пароли пользователей. Для их защиты используется MD5-хеширование. MD5-хеши записываются после набора символов "1"



1. Имя пользователя;
2. Зашифрованный пароль. Рекомендуется использовать не менее 8 символов с использование цифр и спец.символов;

3. Последняя смена пароля. Количество дней с 1-го января 1970 года;

4. Возможный минимальный интервал между сменой пароля;

5. Количество дней которое пароль будет рабочим (по истечение срока пользователю придет запрос на его смену);

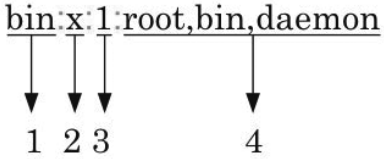
6. За какое количество дней до истечения пароля выводить уведомление, что пароль нужно сменить;

7. Число дней через которое аккаунт с просроченным паролем будет отключен. Не используется;

8. Expire. Дата, при достижение которой учетная запись будет заблокирована. Не используется.

/etc/group

Файл хранящий список групп в системе.



1. Символьное имя группы;

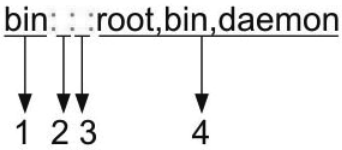
2. Пароль для группы. Устаревшее поле, сейчас не используется. Обычно в нем стоит "x";

3. Идентификатор группы (GID);

4. Список пользователей этой группы.

/etc/gshadow

Файл хранит личные настройки для групп.



1. Имя группы;

2. Зашифрованный пароль;

3. Список администраторов группы через запятую;

4. Список участников группы через запятую.

Обычно пароли на группы не устанавливают.

/sbin/nologin

Файл-заглушка который не позволяет пользователям войти в систему. Используется в ситуациях когда пользователь создается только для работы какой либо программы и давать доступ в систему ему не нужно.

Команды для управления пользователями и группами

Для работы с перечисленными файлами есть набор утилит. Хотя многое можно изменять просто редактируя конфигурационные файлы.

useradd, usermod, userdel – набор команд для добавления, модификации и удаления пользователей.

groupadd, groupmod, groupdel – набор команд для добавления, модификации и удаления групп.

gpasswd – команда для модификации файла /etc/group

pwck, grpck – команды для проверки целостности файлов /etc/passwd и /etc/group

Файл /etc/login.defs

```
# Расположение почтовых ящиков для пользователей

MAIL_DIR       /var/spool/mail

# Password aging controls:
#
#     PASS_MAX_DAYS Максимальное количество дней которое пароль
#                   может использоваться
#     PASS_MIN_DAYS  Минимальное количество дней между
#                   возможностью смены пароля
#     PASS_MIN_LEN    Минимальная длина пароля
#     PASS_WARN_AGE За какое количество дней до истечения пароля
#                   предупреждать пользователя

PASS_MAX_DAYS   99999
PASS_MIN_DAYS   0
PASS_MIN_LEN    5
PASS_WARN_AGE   7

# Минимальное и максимальное значение UID которое будет использоваться
# в программе useradd

UID_MIN                 600
UID_MAX                60000

# Минимальное и максимальное значение GID которое будет использоваться
# в программе groupadd

GID_MIN                 600
GID_MAX                60000

#
# Если опция с командой указана, то она будет запущена во время удаления
# пользователя

#USERDEL_CMD    /usr/sbin/userdel_local

# Должна ли программа useradd создавать домашний каталог для
# пользователя. Эта опции перекрывается ключом -m в командной строке
# программы useradd

CREATE_HOME     yes

# Опция задает значение umask. Если не задано то будет использоваться 022.

UMASK           022

# Опция разрешает программе userdel удалять группы в которых нет
# пользователей

USERGROUPS_ENAB yes

# Какой метод шифрования паролей использовать: MD5 или DES? В CentOS
# по умолчанию используется MD5.

MD5_CRYPT_ENAB yes
```

Команда useradd: добавление новых пользователей

Команда useradd служит для добавления новых пользователей в систему.

Формат команды: useradd [опции] логин

Добавляем нового пользователя:

```
[root@linuxbox ~]# useradd user3
```

Но аккаунт будет заблокирован до тех пор пока на него не поставят пароль. Сделать это можно командой passwd:

```
[root@linuxbox ~]# passwd user3
Changing password for user user3.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
[root@linuxbox ~]#
```

Различные опции команды useradd:

Опция	Описание
-c '<комментарий>'	Опция позволяет добавить комментарий. Обычно это полное имя пользователя
-d<home-dir>	Опция позволяет задать домашний каталог (по умолчанию используется /home/)
-e<date>	Дата, когда этот аккаунт будет отключен, в формате YYYY-MM-DD
-f<days>	Количество дней после истечения срока действия пароля, перед полной блокировкой аккаунта. Значение "0" означает блокировку аккаунта сразу же после истечения срока действия аккаунта. Значение "-1" означает, что аккаунт не будет заблокирован после истечения срока действия пароля
-g<group-name>	Название группы или ее GID. Группа уже должна существовать.
-G<group-list>	Опция позволяет задать целый список дополнительных групп для пользователя. Группы перечисляются через запятую и уже должны существовать

-m	Создать домашний каталог если его не существует
-M	Не создавать домашний каталог
-n	Не создавать индивидуальную группу для пользователя
-r	Создать пользователя с UID меньше чем 500 и без домашнего каталога
-p<password>	Пароль будет зашифрован с помощью crypt
-s	Опция позволяет задать shell (по умолчанию /bin/bash)
-u<uid>	UID пользователя. Он должен быть уникальным и больше чем 499

Команда usermod: изменение информации о пользователе в системе

Команда позволяет модифицировать текущий аккаунт пользователя.

Формат команды: usermod [опции] логин

Изменим shell для пользователя со стандартного /bin/bash на /bin/tcsh

```
[root@linuxbox ~]# usermod -s /bin/tcsh user3
```

Опция	Описание
-d	Изменение домашнего каталога
-s	Изменение shell
-p	Изменение пароля
-g	Изменение первичной группы
-G	Изменение дополнительных групп

Команда userdel: удаление аккаунта пользователя

Команда userdel удаляет аккаунт пользователя и связанные с ним файлы.

Формат команды: userdel [опции] логин

```
[root@linuxbox ~]# userdel user3
```

Опция	Описание
-f	Опция позволяет удалить пользователя пока он находится в системе. Это также позволяет удалить домашний каталог когда его использует другой пользователь или файл почты (mail spool) не принадлежит указанному пользователю. Опцию стоит использовать с осторожностью
-r	Файлы в домашнем каталоге пользователя будут удалены вместе с самим домашним каталогом и почтовым ящиком. Пользовательские файлы, расположенные в других файловых системах, нужно искать и удалять вручную

Команда groupadd: добавление новой группы

Команда groupadd служит для добавления новой группы.

Формат команды: groupadd [опции] группа

Пример добавления новой группы:

```
[root@linuxbox ~]# groupadd newgroup
```

Различные опции команды groupadd:

Опция	Описание
-g<gid>	Group ID для группы. Должен быть уникальным и больше 499
-r	Создать системную группу с GID меньше чем 499
-f	Если при задание GID с помощью опции -g группа уже будет существовать то будет сгенерирован другой уникальный GID
-o	Опция разрешает добавить группу с не уникальным GID
-K KEY=VALUE	Опция позволяет переопределить опции по умолчанию из файла /etc/login.defs (GID_MIN, GID_MAX и т.д.)

Команда groupmod: модификация группы

Команда позволяет изменить название и GID группы.

Формат команды: groupmod [опции] группа

Поменяем название группы justgroup на newgroup и ее GID с 501 на 510:

```
[root@linuxbox ~]# cat /etc/group
...
justgroup:x:501:
[root@linuxbox ~]# groupmod -g 510 -n newgroup justgroup
[root@linuxbox ~]# cat /etc/group
...
newgroup:x:510:
[root@linuxbox ~]#
```

Опции команды groupmod:

Опция	Описание
-g	Числовое значение group ID. Оно должно быть уникальным если только не используется опция -o. Значение не может быть отрицательным числом
-n new_group_name	Новое название для группы

Команда groupdel: удаление группы

Команда groupdel удаляет группу. В качестве единственного аргумента передается название группы.

Формат команды: groupdel группа

```
[root@linuxbox ~]# groupdel justgroup
```

Команда gpasswd: администрирование файлов /etc/group и /etc/gshadow

Команда gpasswd позволяет управлять файлами /etc/group и /etc/gshadow. Устанавливать пароли на группы, добавлять и удалять пользователей.

Формат команды: gpasswd [опции] группа

Опция	Описания
-a	Добавление пользователя в группу

-d	Удаление пользователя из группы
-R	Отключение доступа по паролю к группе через команду newgrp
-r	Удаление пароля группы
-A	Назначения администратора для группы
-M	Добавление множества пользователей в группу

Установим пароль на группу

```
[root@linuxbox ~]# gpasswd newgroup
```

Добавим пользователя user в группу user2

```
[root@linuxbox ~]# gpasswd -a user user2

Adding user user to group user2

[root@linuxbox ~]# cat /etc/group

…

user:x:500:

user2:x:501:user

[root@linuxbox ~]#
```

Команда pwck: проверка целостности файлов паролей

Команда pwck проверяет файлы /etc/passwd и /etc/shadow на наличие ошибок и выводит статистику.

Формат команды: pwck [опции] [passwd shadow]

Пример работы показан ниже:

```
[root@linuxbox ~]# pwck

user adm: directory /var/adm does not exist

user news: directory /etc/news does not exist
```

```
user uucp: directory /var/spool/uucp does not exist

user gopher: directory /var/gopher does not exist

user user2: program /bin/bash1 does not exist

pwck: no changes

[root@linuxbox ~]#
```

Опции команды pwck:

Опция	Описание
-q	Отображать только ошибки
-r	Выполнять pwck в режиме read-only
-s	Сортировать записи в /etc/passwd и /etc/shadow

Команда grpck: проверка целостности файлов групп

Команда grpck проверяет файлы /etc/group и /etc/gshadow на наличие ошибок и выводит статистику.

Формат команды: grpck [-r] [group shadow]

Пример работы показан ниже:

```
[root@linuxbox ~]# grpck

'user2' is a member of the 'user' group in /etc/gshadow but not in /etc/group

'user' is a member of the 'user2' group in /etc/gshadow but not in /etc/group

[root@linuxbox ~]#
```

Чтобы подобных ошибок не возникало, стоит использовать средства администрирования групп и не редактировать файлы напрямую.

Опция -r сообщит команде работать в read-only, это означает отрицательный ответ на все предложения скорректировать файл, в случае нахождения в нем ошибок.

Смена пароля

Для смены пароля используется команда passwd. В качестве аргумента она получает имя пользователя которому необходимо установить новый пароль.

Формат команды: passwd [опции] пользователь

```
[root@linuxbox ~]# passwd user2
Changing password for user user2.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
[root@linuxbox ~]#
```

Если аргумент не передан то будет предложено сменить пароль текущему пользователю:

```
[root@linuxbox ~]# passwd
Changing password for user root.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
```

После добавления в систему нового пользователя его учетная запись находится в отключенном состояние и подключиться в систему он не может. Чтобы ее активировать на нее необходимо установить пароль с помощью команды passwd.

В целях безопасности рекомендуется работать из под обычной учетной записи, переключаясь на учетную запись root только в случае необходимости.

Модуль 7. Управление процессами

Процесс – это системный объект, посредством которого можно контролировать обращения программы к памяти, процессору и ресурсам ввода-вывода. Системные и пользовательские процессы подчиняются одним и тем же правилам, поэтому команды для управления ими – те же самые. Все процессы система регистрирует в таблице процессов, присваивая каждому уникальный номер – идентификатор процесса (process ID, PID).

Для просмотра процессов можно воспользоваться утилитой ps:

```
[root@linuxbox ~]# ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 16:19 ?        00:00:01 init [3]
root         2     1  0 16:19 ?        00:00:00 [migration/0]
root         3     1  0 16:19 ?        00:00:00 [ksoftirqd/0]
root         4     1  0 16:19 ?        00:00:02 [watchdog/0]
root         5     1  0 16:19 ?        00:00:00 [events/0]
root         6     1  0 16:19 ?        00:00:00 [khelper]
...
```

При работе с процессами система использует PID для того чтобы различать процессы между собой. Уникальные идентификаторы бывают различных видов а именно:

Идентификатор процесса (PID)

Идентификатор процесса присваивается каждому новому процессу созданному ядром. Большинство команд и системных вызовов работающих с процессами требуют указания конкретного идентификатора процесса, чтобы можно было выяснить контекст операции. Идентификаторы присваиваются по порядку по мере их создания.

Идентификатор родительского процесса (PPID)

В *nix нет системного вызова, который бы создавал новый процесс для выполнения конкретной программы. Вместо этого существующий процесс

должен клонировать сам себя, чтобы породить новый процесс. Исходный процесс называется родительским процессом, а его клон – дочерним. Помимо PID каждый процесс имеет PPID который содержит идентификатор родительского процесса, породившего данный процесс. Процесс init является родительским процессом для всех других процессов в системе.

Идентификатор пользователя (UID)

UID (User ID) – это идентификатор пользователя создавшего данный процесс, точнее, это копия значения EUID родительского процесса.

Эффективный идентификатор пользователя (EUID)

EUID – это "эффективный" пользовательский идентификатор процесса. Он используется для того чтобы определить к каким ресурсам и файлам у процесса есть право доступа в данный конкретный момент. У большинства процессов UID и EUID будут одинаковыми (исключение составляют программы с установленным битом смены идентификатора пользователя, SUID). Эти два идентификатора используются для того чтобы разделить персонификацию и права доступа между собой.

Идентификатор группы (GID)

GID – это идентификатор группы к которой относится владелец процесса.

Эффективный идентификатор группы (EGID)

EGID связан с GID точно также как и EUID с UID.

Приоритет процессов

От приоритета процесса зависит какую долю процессорного времени он получит. Приоритет процесса рассчитывает динамически, в зависимости от того сколько времени он уже получил и сколько ему еще необходимо. Также здесь играет роль так называемый уровень уступчивости. Этот уровень устанавливается с помощью команды nice и задает в какой степени программа может делиться процессорным временем с другими программами. Чем выше значение nice тем "уступчивее" программа.

Состояние процесса

Есть несколько возможных состояний процесса.

1. Состояние выполнения;
2. Состояние ожидания;
3. Состояние готовности;
4. Состояние зомби.

Состояние выполнение – это активное состояние, когда процесс обладает всеми необходимыми ресурсами и выполняется процессором.

Состояние ожидания – это пассивное состояние в котором процесс находится в заблокированном состояние и ожидает какого либо события (например освобождения устройства, ввода данных и т.п.)

Состояние готовности – это пассивное состояние в котором процесс находится в заблокированном состояние. Но в отличие от состояния ожидания он заблокирован не по внутренним причинам, например ожидание ввода а по внешним, например, окончился квант времени на его выполнение.

Состояние зомби – процесс освободил адресное пространство но еще находится в таблице процессов ядра. Процесс пытается завершиться.

Жизненный цикл процесса

Все процессы в системе кроме init создаются с помощью системного вызова fork (процесс init создается во время загрузки системы). Результатом выполнения fork является копия процесса но имеющая некоторые отличия. В частности новому процессу присваивается новый идентификатор и учет ресурсов для него ведется в независимости от родительского процесса (предка). PPID дочернего процесса равен PID родительского. После выполнения fork новый процесс обычно запускает новую программу с помощью одного из системных вызовов семейства exec*. Когда процесс завершается он вызывает функцию _exit() чтобы сообщить ядру что он готов к завершению. В качестве параметра _exit() передает так называемый код завершения – целое число которое указывает на причину завершения процесса. По соглашению нулевой код завершения означает, что процесс завершился успешно. Прежде чем процесс полностью будет удален необходимо подтверждение на это родительским процессом с помощью системного вызова wait. Данная функция возвращает код завершения потомка и если требуется статистику использования ресурсов. По этой причине ядро должно хранить код завершения процесса пока его не попросит родительский процесс. По окончание дочернего процесса его адресное пространство освобождается но запись о нем в таблице процессов ядра сохраняется. Процесс в этом состояние называется зомби. Механизм завершения процессов работает нормально если родительский процесс завершается после завершения дочерних и выполняет вызов wait чтобы уничтожить процессы-зомби. Если родительский процесс завершается

раньше дочерних то ядро понимает, что вызова wait не последует и переназначает все процессы-зомби процессу init, который выполняет для всех этих процессов вызов wait.

Сигналы

Сигналы – это запросы на прерывание на уровне процессов. В основном они используются для завершения процесса. Сигналы могут посылаться процессам различными способами:

- сигналы могут посылаться от одного процесса другому как средство межзадачного взаимодействия;
- сигналы могут посылаться драйвером терминала для завершения процесса при нажатие клавиш Ctrl+C и Ctrl+Z;
- сигналы могут посылаться командой kill;
- сигналы могут посылаться ядром если процесс выполняет некорректные инструкции.

В программах на языке C к ним добавляется префикс SIG, например SIGHUP, в то время как в команде kill можно просто использовать HUP, например kill -HUP [PID]

Просмотреть список возможных сигналов в Linux можно командой kill -l:

```
[root@linuxbox ~]# kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
 5) SIGTRAP      6) SIGABRT      7) SIGBUS       8) SIGFPE
 9) SIGKILL     10) SIGUSR1     11) SIGSEGV     12) SIGUSR2
13) SIGPIPE     14) SIGALRM     15) SIGTERM     16) SIGSTKFLT
17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU
25) SIGXFSZ     26) SIGVTALRM   27) SIGPROF     28) SIGWINCH
29) SIGIO       30) SIGPWR      31) SIGSYS      34) SIGRTMIN
35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3  38) SIGRTMIN+4
39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12
47) SIGRTMIN+13 48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14
51) SIGRTMAX-13 52) SIGRTMAX-12 53) SIGRTMAX-11 54) SIGRTMAX-10
55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7  58) SIGRTMAX-6
59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
[root@linuxbox ~]#
```

Сигнал посланный процессу может выполняться двумя различными способами: процесс может назначить специальный обработчик для данного сигнала или если обработчика нет то сигнал будет обработан ядром. Обычно это ведет к завершению процесса. Если есть необходимость, то определенные сигналы можно заблокировать или игнорировать. В случае блокирования, сигнал ставится в очередь на обработку и процесс его принимает только когда разблокирует прием сигнала данного типа. В случае игнорирования процесс просто пропускает сигнал и никак на него не реагирует. Но есть сигналы которые программа не может игнорировать и всегда сразу же выполняет. Это сигнал SIGKILL (который можно передать командой kill -9 [PID]) и сигнал SIGSTOP.

Описание всех сигналов можно получить в справочной документации (команда: man 7 signal), мы разберем лишь самые часто используемые.

Сигнал	Номер	Описание
SIGHUP	1	Изначально предназначался для информирования программы о потери связи с управляющим терминалом. Сейчас в основном применяется для того, чтобы заставить программу перечитать свой конфигурационный файл (чтобы изменения вступили в силу).
SIGINT	2	Прерывание с клавиатуры. Соответствует комбинации клавиш CTRL+C
SIGQUIT	3	Сигнал для останова процесса пользователем по нажатию клавиши “quit” на клавиатуре
SIGKILL	9	Принудительное завершение программы. Этот сигнал нельзя обработать или игнорировать.
SIGTERM	15	Аккуратное завершение программы. Она может выполнить все необходимые операции перед своим завершением.
SIGCONT	18	Возобновление процесса остановленного сигналом SIGSTOP
SIGSTOP	19	Приостанавливает выполнение процесса. Этот сигнал нельзя обработать или игнорировать
SIGTSTP	20	Приостанавливает процесс по команде пользователя. Соответствует комбинации клавиш CTRL+Z

Команда kill: передача сигнала процессу

Команда kill позволяет передать сигнал процессу.

Формат команды: kill [сигнал] [PID]

Для передачи нужного сигнала можно использовать как его символьное имя так и его номер. PID – это идентификатор процесса, который можно узнать с помощью команды ps. Давайте создадим свою программу которая по сути нечего не делая будет работать в бесконечном цикле и прекратим ее работу с помощью команды kill.

Содержимое файла loop.sh:

```
#!/bin/bash

while true; do true; done
```

Устанавливаем право выполнения на файл loop.sh

```
[root@linuxbox ~]# chmod +x loop.sh
```

Запускаем программу loop.sh в параллельном режиме, чтобы наш терминал не заблокировался и мы могли продолжить вводить на нем команды.

```
[root@linuxbox ~]#./loop.sh &

[1] 3444

[root@linuxbox ~]#
```

Наша программа запустилась и её процессу был присвоен PID 3444.

```
[root@linuxbox ~]# ps -f

UID        PID  PPID  C  STIME  TTY          TIME  CMD

root     3068  3066  0   20:40  pts/0   00:00:00  -bash

root     3444  3068 96   22:16  pts/0   00:01:10  /bin/bash ./loop.sh

root     3471  3068  0   22:17  pts/0   00:00:00  ps -f

[root@linuxbox ~]#
```

Наша программа представляет собой бесконечный цикл, поэтому она не остановится, пока мы не сделаем это вручную.

```
[root@linuxbox ~]# kill 3444

[root@linuxbox ~]# ps -f

UID        PID  PPID  C STIME TTY          TIME  CMD

root     3068  3066  0  20:40  pts/0    00:00:00  -bash
```

```
root     3478  3068 0  22:20 pts/0    00:00:00 ps -f
[1]+ Terminated              ./loop.sh
```

Если номер сигнала не передан в команду kill то по умолчанию будет передан TERM (SIGTERM) который "просит" процесс корректно завершиться.

Команда killall: передача сигнала процессу по его имени

Формат команды: killall [сигнал] [имя_процесса]

Команда killall позволяет передать сигнал процессу используя его имя. Если имя сигнала не указано то будет передан сигнал SIGTERM который "просит" процесс корректно завершиться. Список доступных сигналов можно посмотреть командой killall -l:

```
[root@linuxbox ~]# killall -l
HUP INT QUIT ILL TRAP ABRT IOT BUS FPE KILL USR1 SEGV USR2 PIPE ALRM
TERM STKFLT CHLD CONT STOP TSTP TTIN TTOU URG XCPU XFSZ VTALRM
PROF WINCH IO PWR SYS UNUSED
[root@linuxbox ~]#
```

Останавливаем работу процесса с именем process1.

```
[root@linuxbox ~]# killall process1
```

Изменение приоритета выполнения. Команды nice и renice

Значение nice (степень уступчивости) сообщает ядру как стоит относится к данному процессу по сравнению с другими во время их конкуренции за процессорное время. Чем ниже значение nice, тем выше приоритет процесса. Приоритет может принимать значения от -20 (самый высокий приоритет) до 19 (самый низкий приоритет) и обычно дочерний процесс наследует его от родительского. Для изменения этого параметра есть две программы: nice и renice. Команда nice позволяет задать приоритет выполнения при запуске программы. Команда renice позволяет его изменить во время выполнения программы. Давайте это проверим на практике. По умолчанию новый процесс получает значение nice равным нулю.

```
[root@linuxbox ~]#./loop.sh &
```

```
[1] 4379
[root@linuxbox ~]# ps -C loop.sh -o ni=
 0
[root@linuxbox ~]#
```

Допустим, мы не хотим чтобы эта программы получала очень много процессорного времени, поэтому понизим ее приоритет до 15 (из максимальных 19). Стартуем программу с нужным нам приоритетом.

```
[root@linuxbox ~]# nice -n 15 ./loop.sh &
[1] 4347
[root@linuxbox ~]#  ps -C loop.sh -o ni=
 15
[root@linuxbox ~]#
```

В процессе работы программы ее приоритет также можно изменить, для этого используется команда renice. В качестве аргумента нужно передать PID нужного нам процесса.

```
[root@linuxbox ~]#  ps -C loop.sh -o pid=
 4379
[root@linuxbox ~]# renice -10 4379
4379: old priority 0, new priority -10
[root@linuxbox ~]#  ps -C loop.sh -o ni=
-10
[root@linuxbox ~]#
```

Ручное выставление приоритетов использовалось в 80-90-е годы и сейчас используется не очень часто, так как обычно самым узким местом является система ввода-вывода.

Команды fg, bg и jobs: управление заданиями

В командном интерпретаторе есть так называемый контроль за работой

(job control), который позволяет одновременно запускать и управлять несколькими процессами на одном терминале. Для этого используются команды fg, bg и jobs которые встроены в bash (shell builtins).

Формат команд:

bg [номер задания]

fg [номер задания]

Команда fg (foreground) позволяет сделать задачу активной, команда bg (background) позволяет переключить задачу в фоновый режим, команда jobs позволяет просмотреть текущий список задач и их номера. Отличие заданий от процессов заключается в том, что каждый процесс при запуске получает свой PID, в то время как задание получает уникальный номер который начинается с 1. То есть второй задание получит номер 2 и т.д.

Разберем все на примере, для этого запустим в активном режиме, то есть выполняющуюся на нашем терминале программу loop.sh

```
[root@linuxbox ~]#./loop.sh
```

Программа loop.sh при своем запуске блокирует наш терминал и мы больше не можем вводить команды, что не очень удобно. Будет лучше если это программа будет выполняться в фоновом режиме а мы продолжим вводить свои команды. Для этого нажимаем комбинацию клавиш CTRL+Z которая посылает сигнал STOP процессу и приостанавливает его работу.

```
[root@linuxbox ~]#./loop.sh


[1]+  Stopped                 ./loop.sh
[root@linuxbox ~]#
```

С помощью команды jobs просмотрим текущие задания.

```
[root@linuxbox ~]# jobs
[1]+  Stopped                 ./loop.sh
[root@linuxbox ~]#
```

Программа loop.sh приостановлена, что можно наблюдать в таблице процессов.

```
[root@linuxbox ~]# ps -C loop.sh -o stat=
T
[root@linuxbox ~]#
```

Буква T в колонке STAT означает, что процесс приостановлен (о команде ps будет ниже).

Переведем нашу программу в фоновый режим с помощью команды bg, чтобы она продолжила выполняться. Если выполняется только одна задача то ее номер указывать не требуется.

```
[root@linuxbox ~]# jobs
[1]+  Stopped                 ./loop.sh
[root@linuxbox ~]# bg
[1]+ ./loop.sh &
[root@linuxbox ~]# jobs
[1]+  Running                 ./loop.sh &
[root@linuxbox ~]#
```

Теперь наша программа выполняется в фоновом (параллельном режиме) и мы можем продолжить вводить команды. Если появится необходимость переключения задачи в активный режим то для этого нужно использовать команду fg. Если выполняется только одна задача то ее номер указывать не обязательно.

```
[root@linuxbox ~]# fg
./loop.sh
```

После работы с программой можно её приостановить чтобы получить доступ к командной строке нажав CTRL+Z и выполнить команду bg, чтобы перевести выполнение программы в фоновый режим.

Команда ps: просмотр текущих процессов

Команда ps позволяет получить список всех текущих процессов в системе. Есть несколько вариаций это команды, но в целом они выдают одну и ту же

информацию.

Формат команды: ps [опции]

Получить список всех выполняющихся в системе процессов можно командой ps aux:

```
USER       PID  %CPU  %MEM    VSZ   RSS  TTY   STAT  START   TIME  COMMAND
root         1   0.3   0.2   2064   652  ?     Ss    21:54   0:05  init [3]
root         2   0.0   0.0      0     0  ?     S<    21:54   0:00  [migration/0]
root         3   0.0   0.0      0     0  ?     SN    21:54   0:00  [ksoftirqd/0]
root         4   0.0   0.0      0     0  ?     S<    21:54   0:00  [watchdog/0]
root         5   0.0   0.0      0     0  ?     S<    21:54   0:00  [events/0]
...
```

Опций у команды ps огромное множество, я покажу лишь самые популярные комбинации этой команды.

ps aux – команда выводит список всех процессов (BSD-style)

ps lax – команда выполняется быстрее чем ps aux, так как она не сопоставляет идентификаторы процессов с именами пользователей

ps -ejH – вывести дерево процессов

ps -C <программа> -o <колонка> – вывести определенную колонку конкретной программы

ps aux | grep <имя программы> – в общем списке процессов найти информацию о конкретном процессе, например:

```
[root@linuxbox ~]# ps aux | grep syslogd
root     1939  0.0  0.2   1820   696 ?        Ss   Aug25 0:00 syslogd -m 0
root     6107  0.0  0.2   3912   668 pts/0    R+   00:38   0:00 grep syslogd
[root@linuxbox ~]#
```

Описание выходной информации команды ps aux:

Параметр	Описание
USER	Имя владельца процесса
PID	Идентификатор процесса
%CPU	Доля времени центрального процессора (в процентах), выделенная данном процессу
%MEM	Часть реальной памяти (в процентах), используемая данным процессом
VSZ	Виртуальный размер процесса в килобайтах
RSS	Размер резидентного набора. Сколько физической памяти процесс использует (в килобайтах)
TTY	Идентификатор управляющего терминала
STAT	Текущий статус процесса: D – не прерывистый сон, обычно I/O R – процесс выполняется S – прерывистый сон, обычно это ожидание события чтобы закончить T – процесс остановлен W – процесс выгружен на диск (не используется начиная с ядра 2.6.xx) X – процесс уничтожен (не должен быть виден) Z – процесс в состояние зомби (defunct) Для BSD-формата команды ps могут отображаться дополнительные статусы: < – процесс имеет высокий приоритет N – процесс имеет низкий приоритет L – некоторые страницы заблокированы в оперативной памяти s – процесс является лидером сеанса (владельцем управляющего терминала) l – мульти-поточный процесс + – процесс находится в группе активных процессов
START	Время запуска процесса
TIME	Время центрального процессора, затребованное процессом
COMMAND	Имя и аргументы исполняемой команды

Команда nohup: запуск программы с иммунитетом к сигналу HUP

Команда nohup позволяет запустить программу с иммунитетом к потери связи (HANGUP). Это позволит программе продолжить выполняться даже после того как пользователь выйдет из системы. В противном случае программа завершила бы свою работу.

Формат команды: nohup команда [аргумент]

Запустим программу loop.sh с иммунитетом к hangup.

```
[root@linuxbox ~]# nohup ./loop.sh &

[1] 6322

[root@linuxbox ~]# nohup: appending output to `nohup.out'


[root@linuxbox ~]#
```

Теперь программа будет выполняться, даже если мы выйдем из системы.

Команда top: просмотр текущих процессов в реальном времени

Команда ps позволяет получить только "снимок" процессов в системе. Для просмотра текущих процессов в реальном времени есть команда top. Самые активные процессы будут в вверху списка. Обновление экрана происходит каждые 3 секунды, но с помощью опции -d интервал можно задать вручную или просто нажать пробел.

Формат команды: top [опции]

Вывод команды top может быть примерно таким:

```
top - 01:50:55 up 12 min,  2 users,  load average: 0.00, 0.15, 0.18

Tasks:  76 total,   2 running,  74 sleeping,   0 stopped,   0 zombie

Cpu(s):  0.0%us,  0.0%sy,  0.0%ni, 98.7%id,  0.0%wa,  0.0%hi,  1.3%si,  0.0%st

Mem:    255556k total,   156840k used,    98716k free,    54352k buffers

Swap:   524280k total,          0k used,   524280k free,    58544k cached
```

```
  PID USER      PR  NI  VIRT  RES  SHR S  %CPU %MEM    TIME+  COMMAND
 2913 root      15   0  2196  992  800 R   0.3  0.4   0:00.02 top
    1 root      15   0  2064  624  536 S   0.0  0.2   0:04.11 init
    2 root      RT  -5     0    0    0 S   0.0  0.0   0:00.00 migration/0
    3 root      34  19     0    0    0 S   0.0  0.0   0:00.00 ksoftirqd/0
    4 root      RT  -5     0    0    0 S   0.0  0.0   0:00.55 watchdog/0
    5 root      10  -5     0    0    0 S   0.0  0.0   0:00.10 events/0
    6 root      10  -5     0    0    0 S   0.0  0.0   0:01.05 khelper
...
```

Первая строка в выводе top соответствует выводу команды uptime, которая показывает время в системе, как долго она работает, количество пользователей в системе и среднюю нагрузку на систему за 1, 5 и 15 минут.

Вторая строка отображает статистику по процессам: сколько всего процессов запущено, сколько выполняется, сколько в режиме ожидания, сколько остановлено, сколько процессов-зомби.

Третья строка показывает долю в процентах для каждого состояния процессора. Как долго он был в различных состояниях:

us – сколько времени было затрачено на пользовательские процессы (включая nice);

sy – сколько времени было затрачено на процессы ядра;

id – процент простоя системы;

wa – процент времени потраченный на ожидание I/O (ввод-вывод);

st – процент времени заимствованный у виртуальной машины.

Четвертая строка показывает статистику по оперативной памяти: сколько всего RAM, сколько занято, сколько свободно, размер буферов ядра.

Пятая строка показывает статистику swap`a: общий объем, сколько использовано, сколько свободно и размер буфера кэша.

Интерактивный режим

В процессе выполнения программы top с ней можно взаимодействовать в интерактивном режиме, изменяя и сортирую выводимую информацию. Для этого используются специальные клавиши. Наиболее популярные

представлены в таблице ниже.

Клавиша	Описание
h	Вывести справку о программе
k	Уничтожить процесс. Будет запрошен PID процесса и номер сигнала который будет ему послан
n	Изменить число отображаемых процессов. Необходимо будет ввести число
u	Сортировать по имени пользователя
M	Сортировать по объему используемой памяти
P	Сортировать по загрузке процессора
1	Просмотр загрузки CPU в SMP-системах
r	Изменить приоритет выполнения
q	Выход из top

Команда top -b -n1 позволяет сделать "снимок" команды top со списком всех процессов. Это удобно когда, к примеру, необходимо вывод команды направить в файл.

У команды top есть аналог с более красивым представлением статистики – htop. Эту программу можно установить из репозитория EPEL:

```
[root@linuxbox ~]# rpm -Uvh http://download.fedora.redhat.com/pub/epel/5/i386/epel-release-5-3.noarch.rpm

[root@linuxbox ~]# yum -y install htop
```

Команда vmstat: статистика виртуальной памяти

Команда vmstat выводит статистику о процессах, памяти, замещение страниц, вводу-выводу, ловушках и активности центрального процессора.

Формат команды: vmstat [опции]

Пример вывода команды vmstat.

```
[root@linuxbox ~]# vmstat
procs -----------memory---------- --swap-- -----io---- --system-- -------cpu-------
 r  b  swpd      free      buff  cache     si  so      bi  bo      in   cs  us  sy  id  wa  st
 0  0     0  115556  44112  58300      0   0      45  10    1008   25   0   1  98   1   0
[root@linuxbox ~]#
```

В качестве опции для команды vmstat можно передать количество секунд, через которое необходимо обновлять экран. В нашем случае статистика будет обновляться каждые 2 секунды.

```
[root@linuxbox ~]# vmstat 2
```

или

```
[root@linuxbox ~]# watch -n 2 vmstat
```

Описание вывода команды vmstat

Колонка	Возможные значения
procs	r – количество процессов ожидающих запуска b – количество процессов в не прерываемом спящем режиме
memory	swpd – сколько всего используется виртуальной памяти free – сколько свободной памяти buff – сколько памяти используется для буферов cache – сколько памяти используется для кэша inact – количество неактивной памяти active – количество активной памяти
swap	si – количество памяти выгруженной из swap`a so – количество памяти выгруженной в swap
IO	bi – сколько блоков принято от блочного устройства bo – сколько блоков отправлено блочному устройству
system	in – количество прерываний в секунду cs – количество переключений контекста в секунду
cpu	us – процент процессорного времени затраченного на пользовательские процессы sy – процент процессорного времени затраченного на процессы ядра id – процент простоя процессора wa – процент процессорного времени затраченного на ожидание ввода-вывода

Команда free: статистика физической памяти

Команда free показывает различную статистику по физической памяти.

Формат команды: free [опции]

Пример вывода команды free:

```
[root@linuxbox ~]# free
             total       used       free     shared    buffers     cached
Mem:        255556     142296     113260          0      44160      60188
-/+ buffers/cache:      37948     217608
Swap:       524280          0     524280
[root@linuxbox ~]#
```

Строка Mem: показывает сколько оперативной памяти всего, сколько используется, сколько свободно, сколько памяти в буферах и кэше.

Строка -/+ buffers/cache: показывает объем физической памяти выделенной в настоящее время для буферов системы.

Строка Swap: показывает статистику использования пространства подкачки.

Для работы команды free в динамическом режим удобно использовать watch:

```
[root@linuxbox ~]# watch -n 1 -d free
Every 1.0s: free   Wed Aug 26 18:22:27 2009
                     total       used       free     shared    buffers     cached
Mem:                255556     142296     113260          0      44164      60200
-/+ buffers/cache:              37932     217624
Swap:               524280          0     524280
```

Ключ -n сообщает как часто обновлять экран, ключ -d предписывает команде watch подсвечивать изменения по сравнению с последним выводом.

Выйти можно с помощью комбинации клавиш Ctrl+C.

Модуль 8. Управление программным обеспечением

RPM

RPM Package Manager (RPM) – это открытая упаковочная система. Благодаря пакетам в формате rpm возможна легкая установка, обновление и удаление программ. Пакет представляет из себя исполняемый файловый архив, в котором хранится дерево каталогов, файлов и метаданные. RPM создает базу данных установленных пакетов в системе и в любой момент позволяет получить информацию по пакету и его файлам.

Преимущества RPM

1. Легкая установка, обновление и удаление программ;
2. RPM является популярной системой и многие программы поставляются в RPM формате;
3. Не интерактивная установка. Это позволяет автоматизировать процесс установки/удаления пакетов;
4. Проверка целостности пакетов с помощью контрольных сумм и GPG-подписей.

Недостатки RPM

1. Основной недостаток RPM – несовместимость между собой различных версий;
2. Макропакеты между дистрибутивами могут сильно отличаться.

Названия пакетов

Собранный пакет обычно именуется в таком формате:

<название>-<версия>-<релиз>.<архитектура>.rpm

Например:

vim-6.3-1.i386.rpm

Если в пакете содержатся исходные коды (SRPM), то вместо архитектуры

будет src:

vim-6.3-1.src.rpm

После установки в систему такого пакета в его конфигурацию можно внести изменения (файл .spec) и собрать rpm-пакет с помощью команды rpmbuild для конечной установки. Пример показан ниже.

```
# mkdir -p /usr/src/redhat/{SOURCES,BUILD,SRPMS,RPMS/i386}

# rpm -Uvh http://mirror.corbina.net/pub/Linux/centos/5.2/updates/SRPMS/
postfix-2.3.3-2.1.el5_2.src.rpm

# cd /usr/src/redhat/SPECS/

# rpmbuild -ba postfix.spec

# rpm -Uvh /usr/src/redhat/RPMS/i386/postfix-2.3.3-2.1.i386.rpm
```

Команда rpm: RPM Package Manager

Команда rpm позволяет устанавливать, получать информацию, обновлять и удалять rpm-пакеты.

Формат команды: rpm [опции] [пакет]

Опция	Описание
-v	Выводить больше информации
-i	Установка пакета
-U	Обновление пакета. Если пакета нет в системе то он будет установлен
-F	Освежить пакет. Отличие от обновления в том, что будет обновлен только уже установленный пакет
-e	Удалить пакет
-q	Получить информацию
-p <package>	Запросить <package>
-a	Запросить все установленные пакеты

-h	Напечатать 50 отметок при распаковке пакета. Обычно используется с опцией -v чтобы придать выводу команды rpm красивый вид
-f <file>	Запросить пакет содержащий <file>
-i	Запросить информацию о пакете, которая включает имя пакета, версию и описание
-l	Запросить список файлов в пакете
-R	Запросить список пакетов, от которых зависит этот пакет

Установка rpm-пакетов

Пример установки пакета показан ниже.

```
[root@linuxbox ~]# rpm -ivh nginx-0.6.38-1.el5.i386.rpm
```

Часто вместо опции -i используют -U. Она также позволяет установить пакет, но если он уже установлен то попытается обновить его.

Удаление rpm-пакетов

Пример удаления пакета показан ниже.

```
[root@linuxbox ~]# rpm -e synaptics-0.14.4-8.fc6
```

При удаление пакета может возникнуть ошибка: error: Failed dependencies: которая сообщает о нарушение зависимости. Она означает, что от пакета который вы хотите удалить зависит работа другого пакета. Если вы все равно хотите удалить пакет то используйте опцию --nodeps.

Получение информации о rpm-пакете

Команда rpm с опциями -qi позволяет получить общую информацию о пакете.

```
[root@linuxbox ~]# rpm -qi openssh-4.3p2-29.el5
Name        : openssh                       Relocations: (not relocatable)
Version     : 4.3p2                         Vendor: CentOS
Release     : 29.el5                        Build Date: Tue 03 Mar 2009 09:08:12 PM MSK
Install Date: Tue 28 Jul 2009 02:13:17 AM MSD      Build Host: chamkaur.karan.org
Group       : Applications/Internet         Source RPM: openssh-4.3p2-29.el5.src.rpm
```

```
Size      : 743486                  License: BSD
Signature  : DSA/SHA1, Mon 09 Mar 2009 04:48:50 AM MSK, Key ID a8a447dce8562897
Packager   : Karanbir Singh <kbsingh@karan.org>
URL       : http://www.openssh.com/portable.html
Summary    : The OpenSSH implementation of SSH protocol versions 1 and 2
Description :
SSH (Secure SHell) is a program for logging into and executing
commands on a remote machine. SSH is intended to replace rlogin and
rsh, and to provide secure encrypted communications between two
untrusted hosts over an insecure network. X11 connections and
arbitrary TCP/IP ports can also be forwarded over the secure channel.

OpenSSH is OpenBSD's version of the last free version of SSH, bringing
it up to date in terms of security and features, as well as removing
all patented algorithms to separate libraries.

This package includes the core files necessary for both the OpenSSH
client and server. To make this package useful, you should also
install openssh-clients, openssh-server, or both.
[root@linuxbox ~]#
```

Просмотр всех установленных RPM-пакетов в системе

```
[root@linuxbox ~]# rpm -qa
basesystem-8.0-5.1.1.el5.centos
cracklib-dicts-2.8.9-3.3
iso-codes-0.53-1
termcap-5.5-1.20060701.1
gnome-backgrounds-2.15.92-1.fc6
```

```
...
[root@linuxbox ~]#
```

Список будет большим. Поскольку на каждой строке указывается по одному пакету то подсчитав количество строк мы выясним количество установленных пакетов. В этом нам поможет команды wc с ключом -l.

```
[root@linuxbox ~]# rpm -qa | wc -l
743
[root@linuxbox ~]#
```

По файлу узнать принадлежность к пакету

С помощью команды rpm -qf можно узнать к какому пакету относится выбранный файл.

```
[root@linuxbox ~]# rpm -qf /usr/bin/passwd
passwd-0.73-1
[root@linuxbox ~]#
```

Расположение всех файлов установленного RPM-пакета

```
[root@linuxbox ~]# rpm -ql iproute-2.6.18-9.el5
/etc/iproute2
/etc/iproute2/ematch_map
/etc/iproute2/rt_dsfield
/etc/iproute2/rt_protos
/etc/iproute2/rt_realms
/etc/iproute2/rt_scopes
/etc/iproute2/rt_tables
/etc/sysconfig/cbq
...
```

Для rpm-пакетов в файловой системе добавляем ключ -p.

```
[root@linuxbox ~]# rpm -qpl /home/datastore/ispell-3.1.20-5.i386.rpm
```

Yellowdog Updater Modified (YUM)

yum – это консольный менеджер RPM-пакетов. Он позволяет облегчить обновление системы с отслеживанием взаимосвязей RPM-пакетов. Установка пакетов происходит из специальных источников – репозиториев. Менеджеры пакетов обеспечивают следующее:

1. Контроль целостности пакетов;

2. Поддержку установки, обновления и удаления пакетов;

3. Контроль зависимостей;

4. Реализацию поиска по доступным/установленным пакетам.

Для работы с менеджером есть одноименная команда – yum.

Формат команды: yum <опции> <команда> <пакет …>

В таблице ниже перечислены самые распространенные опции этой команды.

Опция	Описание
install	Установка пакета в систему
update	Если опция используется без указания пакета то будут обновлены все установленные пакеты в системе. Если указан один или несколько пакетов то будут обновлены только они
check-update	Получения списка пакетов которые можно обновить
upgrade	Тоже самое что и update только с опцией --obsoletes. upgrade может применяться для обновления с одной версии Linux на другую
remove	Удаление пакета из системы
list	Вывести список пакетов yum list – вывести список всех пакетов yum list installed – вывести список установленных пакетов yum list available – вывести список пакетов доступных для установки

provides	Опция позволяет выяснить какой пакет предоставляет указанный файл или особенность
search	Поиск ключевого слова в описание пакета. Полезно когда вы не знаете точное название пакета но знаете, что он должен делать
info	Получение информации о пакете
clean	Очистка кеша yum
repolist	Вывести список репозиториев в системе. По умолчанию отображаются только включенные
-y	На все диалоги отвечать – да

Установка пакета

Для установки пакета используется опция install и имя пакета, который мы хотим установить. Если необходимо установить сразу несколько пакетов, то их можно перечислить через пробел.

```
[root@linuxbox ~]# yum -y install spamassassin
```

Удаление пакета

Для удаления пакета используется опция remove и название пакета, который мы хотим удалить. Если необходимо удалить сразу несколько пакетов, то их можно перечислить через пробел.

```
[root@linuxbox ~]# yum -y remove spamassassin
```

Просмотр списка всех доступных пакетов

Опция list позволяет вывести список всех доступных пакетов. Каждая строка вывода соответствует одному пакету. Строка разбита на три колонки. Первая колонка – это название пакета, вторая – версия пакета, третья – репозиторий, в котором находится пакет. Если пакет установлен то в ней будет слово installed.

```
[root@linuxbox ~]# yum list
```

Поиск по ключевому слову

Поиск по ключевому слову полезен когда вы не знаете точного названия пакета но понимаете, что он должен делать.

```
[root@linuxbox ~]# yum search captcha

python-tgcaptcha.noarch : A TurboGears CAPTCHA widget for forms

perl-Authen-Captcha.noarch : Perl extension for creating captchas

[root@linuxbox ~]#
```

Просмотр информации о пакете

Опция info позволяет получить полную информацию о пакете. Включая точное название пакета, архитектуру, Epoch (число, которое помогает RPM определить последовательность нумерации версий пакета), версию, релиз, размер, репозиторий и описание.

```
[root@linuxbox ~]# yum info squid

Installed Packages

Name   : squid

Arch   : i386

Epoch  : 7

Version: 2.6.STABLE6

Release: 5.el5_1.3

Size   : 3.3 M

Repo   : installed

Summary: The Squid proxy caching server.

...
```

К какому пакету принадлежит файл

Опция provides позволяет выяснить к какому пакету принадлежит указанный файл.

```
[root@linuxbox ~]# yum provides /usr/bin/passwd
```

```
passwd.i386 : The passwd utility for setting/changing passwords using PAM
passwd.i386 : The passwd utility for setting/changing passwords using PAM
[root@linuxbox ~]#
```

Подключение дополнительных репозиториев

Для работы с пакетами, yum использует репозитории. Репозиторий – это файловый архив расположенный на сервере в локальной сети или в Интернет. При подключение дополнительных репозиториев в системе, количество возможных пакетов для установки увеличивается. Обычно репозитории для разных дистрибутивов и версий не совместимы между собой. Посмотреть список репозиториев в системе можно командой yum repolist.

```
[root@linuxbox ~]# yum repolist
repo id          repo name                    status
addons           CentOS-5 - Addons            enabled
base             CentOS-5 - Base              enabled
extras           CentOS-5 - Extras            enabled
updates          CentOS-5 - Updates           enabled
[root@linuxbox ~]#
```

В базовых репозиториях CentOS находится около 3000 пакетов.

```
[root@linuxbox ~]# yum list | wc -l
2925
[root@linuxbox ~]#
```

Многих важных пакетов в этих репозиториях нет, поэтому будет не лишним подключить дополнительные репозитории, которые поддерживаются добровольцами со всего мира.

```
rpm -Uvh http://download.fedora.redhat.com/pub/epel/5/i386/epel-release-5-3.noarch.rpm
rpm -Uvh http://apt.sw.be/redhat/el5/en/i386/RPMS.dag/rpmforge-release-0.3.6-1.el5.
rf.i386.rpm
```

```
rpm -Uvh http://download1.rpmfusion.org/free/el/updates/testing/5/i386/rpmfusion-free-
release-5-0.1.noarch.rpm
rpm -Uvh http://download1.rpmfusion.org/nonfree/el/updates/testing/5/i386/rpmfusion-
nonfree-release-5-0.1.noarch.rpm
rpm -Uvh http://repo.redhat-club.org/redhat/5/i386/redhatclub-repository-release-5-4.el5.
rhc.noarch.rpm
```

После подключения дополнительных репозиториев, количество доступных пакетов для установки сильно возрастет.

```
[root@linuxbox ~]# yum list | wc -l
10077
[root@linuxbox ~]#
```

Но также нужно учитывать, что пакеты могут повторяться, поэтому реальных пакетов будет поменьше.

Конфигурационный файл yum.conf

Настройки yum задаются в его конфигурационном файле yum.conf. В файле используется два типа секций, первая это [main] для определения общих настроек, вторая это [server] для определения настроек для каждого сервера. Секций [server] может быть несколько.

Секция [main]

Опция	Описание
cachedir	Каталог для хранения кэша и баз данных
keepcache	Использовать кэширование (1) или нет (0)
debuglevel	Уровень отладочной информации. По умолчанию 2
logfile	Расположение журнального файла
distroverpkg	Пакет по которому определяется версия дистрибутива
tolerant	Прощать ли пользователю некоторые ошибки (1) или нет (0)
exactarch	Строгое соблюдение целевой архитектуры пакета при обновлении (1) или нет (0)
obsoletes	Опция полезна при удаление пакетов, признанных устаревшими при смене версии дистрибутива
gpgcheck	Проверять ли GPG подписи пакетов (1) или нет (0)

plugins	Разрешить подключаемые модули (1) или нет (0)
metadata_expire	Не проверять обновление метаданных указанный промежуток времени
installonly_limit	Не хранить installonlypkgs-пакетов больше указанного числа. В эту категорию попадают пакеты которые нельзя обновлять а только устанавливать в систему к старым версиям (kernel, kernel-smp, kernel-bigmem и т.д.)

Секция [server]

В этой секции описываются репозитории к которым будет происходить запрос при работе с пакетами.

Формат секции [server] можно представить вот так:

```
[serverid]
name=<уникальное имя>
baseurl=<путь к репозиторию>
gpgcheck=[1|0]
gpgkey=<путь к ключу>
```

Основные опции секции [server]

Опции	Описание
serverid	Уникальное имя сервера
name	Описание репозитория
baseurl	url к репозиторию
gpgcheck	Проверять сигнатуры GPG для пакетов (1) или нет (0)
mirrorlist	URL файла, содержащего список baseurl
gpgkey	Расположение GPG-ключа
enabled	Использовать (1) или нет (0) этот репозиторий

chkconfig: управление сервисами

Команда chkconfig позволяет задать уровни выполнения для сервисов. То есть на каком runlevel они должны включаться а на каком выключаться.

Формат команды: chkconfig [опции] [имя_сервиса] [on|off|reset]

Список всех сервисов в системе и их уровни выполнения можно получить с помощью опции --list

```
[root@linuxbox ~]# chkconfig --list
```

Включение сервиса winbind на уровнях выполнения 2-5.

```
[root@linuxbox ~]# chkconfig winbind on
```

Если мы хотим чтобы сервис выполнялся только на конкретном уровне выполнения то указываем это с помощью опции --level

```
[root@linuxbox ~]# chkconfig winbind off

[root@linuxbox ~]# chkconfig --levels 23 winbind on
```

Полное отключение сервиса. Во время старта системы он включаться не будет.

```
[root@linuxbox ~]# chkconfig winbind off
```

Установка программ из исходных кодов

Бывают ситуации когда нужной программы нет в RPM-пакете или необходимо скомпилировать программу с определенными опциями. В этом случае придется собирать программу из исходных кодов. Но навыки программиста здесь вовсе не нужны. В большинстве случаев исходные коды будут поставляться с нужными для сборки скриптами. Вам остается задать параметры с которыми нужно скомпилировать программу и запустить скрипт. Ниже приведена классическая ситуация, с которой вы будете сталкиваться как минимум в 90% случаев, когда возникнет необходимость установки программы из исходных кодов.

```
1    wget http://www.sai.msu.su/apache/httpd/httpd-2.2.13.tar.gz
2    tar xzvf httpd-2.2.13.tar.gz
3    cd httpd-2.2.13
4    less README или INSTALL
5    ./configure
6    make
7    make install
```

1. Первый этап: скачивание архива исходных кодов программы из Интернета. В большинстве случаев архив будет в формате tar.gz;

2. Второй этап: разархивирование архива;

3. Третий этап: переход в каталог с исходными кодами;

4. Четвертый этап: чтение таких важных файлов как README и INSTALL. Могут присутствовать как оба файла, так и любой один из них. В файлах можно узнать основную информацию о программе и методы ее установки;

5. Пятый этап: процесс конфигурирования программы перед установкой с помощью скрипта configure. Чтобы получить подробную информацию о скрипте выполните его с опцией --help: ./configure --help. Скрипт позволяет задать с какими возможностями будет будущая программа и поддержка каких технологий в ней будет присутствовать;

6. Шестой этап: компиляция программы. На этом этапе нужно просто запустить команды make находясь в каталоге с исходными кодами программы. Процесс компиляции может занять продолжительное время;

7. Седьмой этап: установка скомпилированной программы в систему. Файлы программы будут скопированы в нужные каталоги и на них будут выставлены правильные права доступа. Если каких-то каталогов не существует то они будут созданы.

Модуль 9. Текстовый редактор VIM

vim (**V**i **im**proved) – это популярный текстовый редактор, который пришел на смену vi и имеет обратную совместимость с ним. Он является одним из самых мощных текстовых редакторов благодаря своим возможностям и настройкам.

Режимы работы vim

vim может работать в трех режимах:

1. Режим ввода команд (Command mode). В этом режиме вводятся различные команды по управлению редактором и текстом – copy-paste, изменение режима;

2. Режим ввода текста (Insert mode). В этом режиме происходит работа с текстом. Перемещение по тексту происходит с помощью стрелок вверх, вниз, влево и вправо;

3. Ex режим. В этот режим можно попасть нажатием символа ":". Загрузка файла, сохранение и выход происходят в этом режиме.

По умолчанию мы всегда в командном режиме. Отсюда можно попасть как в Insert режим нажав "i" или Insert так и в Ex режим нажав ":" (Shift+:).

ESC – выход из текущего режима

ESC ESC – выход в командный режим

Преимущества vim

Скорость: начать редактировать документ можно буквально за пару секунд;

Удобство: не нужна мышь или графический интерфейс;

Доступность: vim есть в большинстве дистрибутивов Linux.

Недостатки vim

Сложность: освоение vim может занять продолжительное время.

Команда vim

Исполняемый файл имеет такое же название, как и сам редактор и он входит в пакет vim-enhanced.

```
[root@linuxbox ~]# which vim
/usr/bin/vim
[root@linuxbox ~]# yum provides /usr/bin/vim
vim-enhanced.i386 : A version of the VIM editor which includes recent
enhancements.
vim-enhanced.i386 : A version of the VIM editor which includes recent
enhancements.
[root@linuxbox ~]#
```

Формат команды: vim <опции> <файл>

Справка по vim

Чтобы получить справку по vim находясь в текстовом редакторе, набираем в Ex режиме команду :help

```
:help <Enter>
```

Для вывода справки по конкретной теме:

```
:help <слово > <Enter>
```

Для вывода всех совпадений нажмите Ctrl+D

```
:help <слово> <Ctrl+D>
```

В командном интерпретаторе есть команда vimtutor

```
[root@linuxbox ~]# vimtutor
```

Или просто man vim

```
[root@linuxbox ~]# man vim
```

Простой сеанс работы в vim

Открываем файл на редактирование

Для того чтобы начать редактировать файл с помощью vim, находясь в командном интерпретаторе набираем vim <название файла>. Это может быть как существующий текстовый файл так и новый.

```
[root@linuxbox ~]# vim textfile
```

После того как запустится vim и загрузит выбранный файл, мы попадем в режим ввода команд. Для того чтобы начать вносить изменения в файл мы должны перейти в Insert режим, для этого нажимаем клавишу "i" или Insert.

Открываем файл на редактирование из vim

Команде vim не обязательно передавать название файла на редактирование. Редактор можно запустить командой vim и начать новый документ, или открыть существующий с помощью команды :e из Ex режима.

```
:e anotherfile <Enter>
```

Сохранение документа и выход

Когда редактирование документа завершено, возвращаемся в режим ввода команд – нажимаем клавишу ESC далее ":", вводим wq и нажимаем <Enter>. w – означает, что файл необходимо сохранить, символ q означает выход из текстового редактора. Измененный файл можно сохранить под другим именем, для этого после :w указываем название нового файла.

```
:w textfile2 <Enter>
```

Подтверждение

При диалогах в которых просят подтвердить ваши действия используется символ !, например :wq! Комбинация клавиш сообщает vim чтобы он сохранил файл и закончил свою работу несмотря ни на что. Например после редактирования файла и попытки из него выйти, будет отображена такая ошибка:

```
:q

E37: No write since last change (add ! to override)
```

Она сообщает о том, что файл был отредактирован и если мы выйдем

то изменения будут потеряны. Для того чтобы выйти несмотря ни на что используем символ !

```
:q! <Enter>
```

Добавление содержимого другого файла в текущий

Иногда требуется вставить содержимое другого файла в текущий. Для этого необходимо использовать команду :r <другой файл>

```
:r textfile3 <Enter>
```

Удаление текста

Для удаления одного символа в командном режиме наведите на него курсор и нажмите x. Для удаления строки, поставьте курсор на нужную строку и находясь в командном режиме нажмите dd. Для удаления большего блока текста ставим курсор на первую линию блока, переходим в командный режим нажатием ESC, нажимаем v для перехода в визуальный режим, выделяем блок текста на удаление и нажимаем d.

Перемещение текста (cut-paste)

Есть два способа перемещения текста.

Первый способ – это перемещение в стиле vi.

1. Перемещаем курсор на начало блока текста, который хотим переместить;
2. Нажимаем по очереди клавиши “m” и “a”. На экран нечего не поменяется и это нормально;
3. Перемещаем курсор в конец блока текста, который хотим переместить;
4. Нажимаем по очереди клавиши “d”, “‘” и “a”. Это удалит метку “a” и переместит блок текста в буфер обмена. Вторая клавиша – это ординарная кавычка (‘) которая обычно находится на клавише “э”, без кавычек эта комбинация будет выглядеть вот так d’a;
5. Перемещаем курсор на строку куда следует вставить текст из буфера обмена и нажимаем p. Текст будет вставлен на следующею строку после курсора.

Второй способ – визуальный способ.

1. Перемещаем курсор на начало блока текста, который хотим переместить;
2. Нажимаем по очереди клавиши ESC и v. Так мы перейдем в визуальный режим;
3. Стрелкой вниз мы выделяем участок текста, который хотим переместить Выделение текста можно будет наблюдать в визуальном режиме;
4. Нажимаем клавишу d, это удалит выделенный текст;
5. Перемещаем курсор на строку куда следует вставить текст из буфера обмена и нажимаем p. Текст будет вставлен под курсор.

Копирование текста (copy-paste)

Есть два способа копирования текста.

Первый способ – копирование в стиле vi.

1. Перемещаемся на начало блока текста, который хотим скопировать;
2. Нажимаем по очереди клавиши "m" и "a", тем самым поставив метку "a";
3. Стрелкой вниз перемещаемся на последнею строку блока текста который хотим скопировать;
4. Нажимаем клавиши "y", "'" и "a". Тем самым копируем текст с текущей позиции до метки "a";
5. Перемещаемся на строку, куда мы хотим вставить блок текста и нажимаем p. Текст будет вставлен на следующею строку после курсора.

Второй способ – визуальный режим.

1. Перемещаемся на начало блока текста, который хотим скопировать;
2. Нажимаем клавиши ESC и v чтобы перейти в визуальный режим;
3. Стрелкой вниз мы выделяем участок текста, который хотим переместить. Выделение текста можно будет наблюдать в визуальном режиме;
4. Нажимаем клавишу y чтобы скопировать текст;
5. Перемещаем курсор на строку куда следует вставить текст из буфера обмена и нажимаем p. Текст будет вставлен под курсор.

Перестановка символов

В случае если вы допустили ошибку в написание слова и спутали местами две буквы, текстовый редактор позволяет их поменять местами. Например вы напечатали слово Woosd вместо Woods. Чтобы исправить ошибку нажимаем ESC, ставим курсор на символ s и набираем на клавиатуре xp. Символы s и d поменяются местами и в результате получится слово Woods.

Поиск

Для поиска слова в тексте нажимаем ESC, вводим символ "/" и слово которое необходимо найти – /woods <Enter>. Поиск происходит до первого совпадения, чтобы продолжит поиск по тексту набираем "/" и нажимаем <Enter>.

Поиск с заменой

При поиске какого либо слова его сразу же можно заменить на другое.

Поиск и замена всех найденных слов woods на текущей строке на woodz:

```
:s/woods/woodz/g <Enter>
```

: – командный режим

s – сокращенная форма слова substitute (заменять)

/woods – какой слово искать

/woodz – на что менять найденное слово

g – глобальный флаг, который предписывает сделать замену для каждого слова на строке

Поиск и замена всех найденных слов woods с первой по третью строку на woodz:

```
:1,3s/woods/woodz/g <Enter>
```

Поиск и замена всех найденных слов woods во всем документе на woodz:

```
:%s/woods/woodz/g <Enter>
```

Переход на нужную строку

В большом документе для быстрого перехода на нужную строку нажимаем ESC, набираем ":" и номер нужной строки. Пример перехода на десятую строку показан ниже.

```
:10 <Enter>
```

Для перехода в начало документа в командном режиме нажимаем g g

Отмена последнего действия

Для отмены последнего действия, нажимаем ESC далее клавишу "u".

Отмена последней отмены

Ctrl+R

Сортировка блока текста

К примеру у нас есть такой файл:

```
b
h
u
t
y
o
a
l
c
i
```

И мы хотим отсортировать все буквы по алфавиту. Для этого делаем следующее:

1. Перемещаем курсор на начало блока текста, который хотим отсортировать;

2. Нажимаем по очереди клавиши "m" и "a", тем самым поставив метку "a" в

начале блока текста;

3. Перемещаемся на конец блока текста, который хотим отсортировать;

4. Набираем следующею последовательность – !'asort. Восклицательный знак говорит vim о том, что нужно указанный блок текста пропустить через UNIX команду. Последовательность 'a говорит о том, что нужно обработать участок текста с текущей строки и до метки "a". Сортировку выполняет UNIX-команда sort. В результате мы получим следующее:

```
a
b
c
h
i
l
o
t
u
y
```

Работа в нескольких окнах

Окно текстового редактора можно разделить на несколько независимых между собой окон.

Ctrl+w, s – разделяет экран по горизонтали

Ctrl+w, v – разделяет экран по вертикали

Ctrl+w, стрелка – перемещение между окнами

Работа в каждом из окон идентична работе в стандартном однооконном режиме.

Настройка vim

Настройки в режиме реального времени

Для настройки vim в режиме реального времени, в Ex режиме набираем :set

```
:set <Enter>
```

Будут показаны текущие опции. Чтобы посмотреть все возможные опции набираем :set all

```
:set all <Enter>
```

Поменяем размер табуляции в символах.

```
:set ts=20 <Enter>
```

Информацию об опциях можно найти с помощью команды :help option-list

```
:help option-list <Enter>
```

Конфигурационный файл .vimrc

Чтобы настройки были постоянно их необходимо внести в конфигурационный файл vim. Откуда они будут считываться каждый раз когда vim загружается.

```
[root@linuxbox ~]# vim ~/.vimrc
```

В конфигурационном файле двоеточие использовать не нужно.

```
set ts=20
```

Модуль 10. Базовая настройка Linux

Базовая настройка системы включает в себя:

1. Настройку сетевого интерфейса;

2. Настройку даты и времени;

3. Настройку имени хоста (hostname).

Настройка сетевого интерфейса

При настройке сетевого интерфейса необходимо настроить как минимум 3 параметра:

1. IP-адрес;

2. Сетевую маску;

3. Gateway (шлюз).

Для того чтобы была возможность обращаться к другим компьютерам по имени нужно использовать:

файл /etc/hosts – если компьютеров не много перечисляем их в этом файле в формате <IP-адрес> <имя>

или

DNS-сервер – если используется много компьютеров или есть выход в сеть Интернет. DNS-сервера указываются с помощью директивы nameserver в файле /etc/resolv.conf

Файлы влияющие на работу сетевого интерфейса

/etc/modprobe.conf – в этом файле загружаются модули ядра для различных устройств. Обычно вручную редактировать этот файл не требуется.

/etc/hosts – список ip-адресов и назначенных им имен. Удобно в отсутствие DNS-сервера или когда нет необходимости сообщать эти имена на всю сеть.

/etc/resolv.conf – в этом файле с помощью директивы nameserver указываются DNS сервера.

/etc/host.conf – файл задает последовательность использования механизмов разрешения имен. По умолчанию поиск соответствия происходит сначала в

/etc/hosts а только потом происходит обращение к DNS-серверу.

/etc/init.d/network – скрипт останавливающий и запускающий работу сети в Linux. Можно также использовать команду service, например:

```
[root@linuxbox ~]# service network restart
```

/proc/sys/net/ipv4/ip_forward – включение маршрутизации для интерфейсов. Если у вас два или более интерфейсов её нужно включить. Включение осуществляется передачей "1" в этот файл.

```
[root@linuxbox ~]# echo '1' > /proc/sys/net/ipv4/ip_forward
```

Для того чтобы опция не сбросилась после перезагрузки, в файле /etc/sysctl.conf опцию

```
net.ipv4.ip_forward = 0
```

меняем на

```
net.ipv4.ip_forward = 1
```

/etc/sysconfig/network – в этом файле задается:

- Будет ли наша система доступна по сети, если да то по каким протоколам;
- Имя хоста (hostname);
- Default gateway (может быть переназначен в ifcfg-<интерфейс>).

Опция	Описание
NETWORKING=yes	Будет ли наш сервер работать в сети. Это опция для протокола IPv4
NETWORKING_IPV6=no	Аналогичная опция но для IPv6
HOSTNAME= linuxbox.company.ru	Имя хоста
GATEWAY	Шлюз по умолчанию

/etc/sysconfig/network-scripts/ – каталог в котором находятся различные скрипты, влияющие на работу сетевого интерфейса.

/etc/sysconfig/network-scripts/ifcfg-* – в наборе файлов ifcfg-* описываются все сетевые интерфейсы системы. Отчет начинается с eth<x>, где x – это 0 для первого сетевого интерфейса, 1 для второго и т.д. настройки loopback-интерфейса расположены в /etc/sysconfig/network-scripts/ifcfg-lo

Виды сетевых интерфейсов

DHCP iface

Сетевой интерфейс может получать реквизиты динамически с помощью протокола DHCP при загрузке системы, такой интерфейс называется DHCP iface. Отличительной чертой такого интерфейса является наличие опции BOOTPROTO=dhcp в файле конфигурации (ifcfg-<интерфейс>). Пример такого интерфейса показан ниже:

Опция	Описание
DEVICE=eth0	Имя сетевого устройства
BOOTPROTO=dhcp	Использовать протокол DHCP
HWADDR=00:0C:29:43:5B:3D	MAC-адрес сетевой карты
ONBOOT=yes	Загружать ли сетевой интерфейс при загрузке системы ?

Custom iface

Мы рассмотрели DHCP iface, но скорее всего вам придется иметь дело с custom-интерфейсами, то есть настроенными вручную или статически. Пример такого интерфейса показан ниже.

/etc/sysconfig/network-scripts/ifcfg-<имя_интерфейса>

Опция	Описание
DEVICE=eth0	Имя сетевого устройства
HWADDR=00:0C:29:43:5B:3D	MAC-адрес сетевой карты
IPADDR=192.168.146.130	IP-адрес сетевого интерфейса
NETMASK=255.255.255.0	Сетевая маска
BROADCAST=192.168.146.255	Адрес широковещательной рассылки
GATEWAY=192.168.146.2	Сетевой шлюз
ONBOOT=yes	Загружать ли сетевой интерфейс при загрузке системы ?

loopback iface

loopback интерфейс есть в любой системе. Его IP-адрес 127.0.0.1 и он всегда работает. Интерфейс необходим для корректной работы сети.

/etc/sysconfig/network-scripts/ifcfg-lo

Опция	Описание
DEVICE=lo	Имя сетевого устройства
IPADDR=127.0.0.1	IP-адрес сетевого устройства
NETMASK=255.0.0.0	Сетевая маска
NETWORK=127.0.0.0	Сеть в которой находится IP-адрес
BROADCAST=127.255.255.255	Адрес широковещательной рассылки
ONBOOT=yes	Загружать ли сетевой интерфейс при загрузке системы ?
NAME=loopback	Имя сетевого интерфейса. В выводе команды ifconfig будет отображаться имя из DEVICE

Скрипты network, ifdown, ifup

После настройки конфигурационных файлов необходимо перезагрузить сетевой интерфейс чтобы он начал работать по-новому. Для перезагрузки сетевой подсистемы Linux можно использовать скрипт network:

```
[root@linuxbox ~]# /etc/init.d/network restart
```

или

```
[root@linuxbox ~]# service network restart
```

В этом случае будет перезагружена подсистема Linux и все сетевые интерфейсы. Для перезагрузки только одного сетевого интерфейса есть команды ifdown и ifup. Команда ifdown выключает сетевой интерфейс, команда ifup его включает.

```
[root@linuxbox ~]# ifdown eth0

[root@linuxbox ~]# ifup eth0
```

Команда ifconfig

Команда выводит список сетевых интерфейсов в системе, по умолчанию только активных.

```
[root@linuxbox ~]# ifconfig

eth0 Link encap:Ethernet HWaddr 00:0C:29:43:5B:3D

inet addr:192.168.146.130 Bcast:192.168.146.255 Mask:255.255.255.0
```

```
inet6 addr: fe80::20c:29ff:fe43:5b3d/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:1327 errors:0 dropped:0 overruns:0 frame:0
TX packets:1340 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:149547 (146.0 KiB) TX bytes:232486 (227.0 KiB)
Interrupt:59 Base address:0x2000
...
```

MTU:1500 – это Maximum Transfer Unit. Различные сети и каналы передачи данных имеют разные скорости обмена. Это определяет максимальную длину пакета, пересылка которого с высокой вероятностью произойдет без ошибок. Для Ethernet-сетей значение MTU составляет 1500 байт.

Metric:1 – чем меньше это значение тем лучше считается маршрут до этой сети. На серверах скорее всего, этот параметр менять не придется. Он играет большую роль в работе протоколов маршрутизации.

Collisions:0 – нулевое значение говорит о том, что с сетевым интерфейсом все в порядке на физическом уровне.

RX bytes – сколько данных принято

TX bytes – сколько данных отослано

Для вывода информации по конкретному сетевому интерфейсу используем синтаксис: ifconfig <название интерфейса>, например

```
[root@linuxbox ~]# ifconfig eth0
```

Опция -a команды ifconfig позволит вывести информацию о всех сетевых интерфейсах в системе.

```
[root@linuxbox ~]# ifconfig -a
```

Маршрутизация

Маршрутизация — это процесс поиска наилучшего пути от источника к получателю. Это набор правил, по которым будет передаваться информация.

Таблицу маршрутизации можно посмотреть командой netstat -r

Kernel IP routing table							
Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Iface
192.168.146.0	*	255.255.255.0	U	0	0	0	eth0
default	192.168.146.2	0.0.0.0	UG	0	0	0	eth0

В выводе мы увидим список сетей и как до них добраться. То есть какой шлюз использовать для достижения пункта назначения. Поскольку кроме шлюза в файле /etc/sysconfig/network больше никаких нет, то он был помечен как default, то есть шлюз по умолчанию. Весь трафик в сети, которые явно не описаны в таблице маршрутизации, отправляются через шлюз по умолчанию. В нашем случае default gateway – 192.168.146.2

Вывод команды "netstat -r" говорит следующее:

Первая строка. Чтобы попасть в сеть 192.168.146.0 нам не нужен никакой шлюз (*), поскольку мы и так находимся в этой сети.

Вторая строка – это default, то есть маршрут по умолчанию. Весь трафик предназначенный во все другие сети будет уходить через этот шлюз.

Genmask – это сетевая маска, благодаря маске можно отделить сетевую часть адреса от адреса хоста.

Flags: U – это UP, то есть маршрут поднят и функционирует.

Flags: UG – UP, Gateway. Маршрут поднят и использует gateway в своей работе.

MSS – Maximum Segment Size, определяет максимальный размер пакета для этого маршрута.

Window – размер окна. Максимальный размер пакета, который система готова принять.

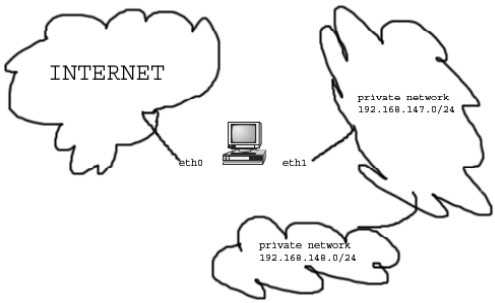
irtt – initial round trip time, задает значение которое используется при установление соединения. Round trip time – представляет из себя отрезок времени, если в течение которого от удаленного хоста не пришло подтверждение о получение пакета, пакет будет послан заново.

Iface – показывает к какому интерфейсу относится маршрут.

Настройка маршрутизации

Если на сервере несколько сетевых интерфейсов то скорее всего понадобится вручную составить таблицу маршрутизации.

Приведу пример. У нас два сетевых интерфейса eth0 и eth1. Через eth0 мы получаем доступ во внешние сети и по умолчанию весь трафик направляется через него. Через eth1 мы получаем доступ во внутреннею сеть 192.168.147.0/24 но что еще важнее, в этой сети есть сервер 192.168.147.1 у которого есть сетевой интерфейс в сеть 192.168.148.0/24 и мы очень хотим туда попасть.



В данном случае таблица маршрутизации выглядит вот так:

```
[root@linuxbox ~]# netstat -r
Kernel IP routing table
```

Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Iface
192.168.147.0	*	255.255.255.0	U	0	0	0	eth1
192.168.146.0	*	255.255.255.0	U	0	0	0	eth0
192.168.148.0	192.168.147.1	255.255.255.0	UG	0	0	0	eth1
default	192.168.146.2	0.0.0.0	UG	0	0	0	eth0

В сеть 192.168.148.0/24 мы попадаем через eth1 -> 192.168.147.1, сервер выполняющий роль шлюза в эту сеть. Все остальное идет через шлюз по умолчанию. Ниже показаны настройки сети.

Настройка шлюза по умолчанию

```
[root@linuxbox ~]# cat /etc/sysconfig/network
...
GATEWAY=192.168.146.2
...
```

Настройка сетевого интерфейса eth0

```
[root@linuxbox ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
HWADDR=00:0C:29:43:5B:3D
IPADDR=192.168.146.130
NETMASK=255.255.255.0
BROADCAST=192.168.146.255
ONBOOT=yes
```

Настройка сетевого интерфейса eth1

```
[root@linuxbox ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth1
DEVICE=eth1
ONBOOT=yes
HWADDR=00:0c:29:43:5b:47
IPADDR=192.168.147.2
NETMASK=255.255.255.0
```

Статический маршрут для eth1

Мы могли бы добавить маршрут в сеть 192.168.148.0/24 командой route:

```
[root@linuxbox ~]# route add -net 192.168.148.0/24 gw 192.168.147.1
```

Но в этом случае маршрут пропадет после перезагрузки ОС. Для постоянного действия статический маршрут для интерфейса прописывается в файл /etc/sysconfig/network-scripts/route-<интерфейс>. Если файла не существует то его необходимо создать. В файл /etc/sysconfig/network-scripts/route-eth1 записываем следующею строку:

```
192.168.148.0/24 via 192.168.147.1
```

В файл записывается сеть и как туда попасть. Каждая новая запись начинается с новой строки.

После редактирования всех файлов необходимо перезагрузить сетевую подсистему:

```
[root@linuxbox ~]# service network restart
```

Типы маршрутов

Динамические – которые динамически назначаются сетевому интерфейсу, например сервером DHCP.

Статические – которые вы вручную прописываете и они остаются в настройках после перезагрузки.

Маршруты по умолчанию – когда никакие другие маршруты не подходят для того, чтобы отослать по ним пакеты.

Полезные опции команды netstat

Опция	Описание
r	Показать таблицу маршрутизации
n	При выводе не пытаться определить имя хоста по IP-адресу. Это работает несколько быстрее и обычно, вывод команды удобнее читать
a	Показывает состояние всех сокетов на сервере. Сокет – это конечная точка сетевых коммуникаций. Каждый сокет имеет тип и ассоциированный с ним процесс.
t	Протокол tcp
u	Протокол udp
i	Отображать таблицу сетевых интерфейсов
l	Отображает сокеты в режиме LISTEN, то есть ожидающие соединения
p	Показать PID (process ID) и имя программы с которой взаимодействует сокет

netstat на практике

Слушает ли какая-нибудь программа 25-й порт?

```
[root@linuxbox ~]# netstat -nlp | grep :25
```

Посмотрим активные подключения

```
[root@linuxbox ~]# netstat -nt
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address Foreign Address State
tcp 0 0 ::ffff:192.168.146.130:22 ::ffff:192.168.146.1:2642 ESTABLISHED
tcp 0 132 ::ffff:192.168.146.130:22 ::ffff:192.168.146.1:2027 ESTABLISHED
```

Выведем список программ/портов ожидающих подключение

```
[root@linuxbox ~]# netstat -ntlp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name
tcp 0 0 0.0.0.0:111 0.0.0.0:* LISTEN 2480/portmap
tcp 0 0 127.0.0.1:631 0.0.0.0:* LISTEN 2762/cupsd
tcp 0 0 127.0.0.1:25 0.0.0.0:* LISTEN 2786/sendmail: acce
tcp 0 0 0.0.0.0:991 0.0.0.0:* LISTEN 2505/rpc.statd
tcp 0 0 :::22 :::* LISTEN 2747/sshd
```

Состояния сокетов

ESTABLISHED – сокет с установленным соединением

SYN_SENT – сокет в процессе установки соединения

SYN_RECV – был принят запрос установки соединения из сети

FIN_WAIT1 – сокет закрыт и соединение закрывается

FIN_WAIT2 – сокет закрыт и ждет закрытия соединения с удаленного хоста

TIME_WAIT – сокет после своего закрытия, еще какое-то время принимает пакеты из сети

CLOSED – сокет не используется

CLOSE_WAIT – удаленный хост отключился, ожидаем закрытие сокета

LAST_ACK – удаленный хост отключился и сокет закрыт. Ожидание подтверждения

LISTEN – сокет ожидает входящие подключения

CLOSING – оба сокета отключились, но еще не все наши данные отправлены

UNKNOWN – статус сокета неизвестен

Команда traceroute

Команда traceroute предназначена для определения маршрута следования пакетов. Отправляя ICMP-пакеты из целевой системы к пункту назначения, она формирует список всех маршрутизаторов которые проходят данные.

```
[root@linuxbox ~]# traceroute www.ru

traceroute to www.ru (194.87.0.50), 30 hops max, 40 byte packets

 1  192.168.79.2 (192.168.79.2)  0.140 ms  0.062 ms  0.114 ms

 2  www.ru (194.87.0.50)  21.104 ms  24.591 ms  26.570 ms

[root@linuxbox ~]#
```

Это бывает полезно при анализе проблем в сети.

Дополнительная информация

Для получения полной информации о возможных опциях при настройке сетевого интерфейса обратитесь к справочному файлу:

```
[root@linuxbox ~]# less /usr/share/doc/initscripts-*/sysconfig.txt
```

Этот файл содержит информацию не только по настройке сети, здесь можно найти описание всех файлов в каталоге /etc/sysconfig/

Настройка даты и времени

Для настройки даты и времени в системе есть команда date. Если команду выполнить без аргументов то будет показана текущая дата и время.

```
[root@linuxbox ~]# date

Wed Sep  2 15:48:01 MSD 2009

[root@linuxbox ~]#
```

Для настройки даты и времени, команде date необходимо передать в качестве аргумента текущий месяц, день, час, минуты, год, секунды в формате [ммддччмм[гг]].сс

Пример 1. Устанавливаем 14 часов 10 минут 30 секунд 15 декабря 2010 год

```
[root@linuxbox ~]# date 121514102010.30

Wed Dec 15 14:10:30 MSK 2010

[root@linuxbox ~]#
```

Пример 2. Устанавливаем 19 часов 35 минут 25 секунд 2 сентября 2009 год

```
[root@linuxbox ~]# date 090219352009.25

Wed Sep  2 19:35:25 MSD 2009

[root@linuxbox ~]#
```

Настройка имени хоста (hostname)

Для настройки имени хоста необходимо отредактировать файл /etc/sysconfig/network внеся в директиву HOSTNAME желаемое имя.

```
[root@linuxbox ~]# cat /etc/sysconfig/network

NETWORKING=yes

NETWORKING_IPV6=yes

HOSTNAME=linuxbox.company.ru

[root@linuxbox ~]#
```

Семейство команд system-config-* – настройка системы

В семейство команд system-config-* входит около 30 различных команд. Каждая из которых позволяет настроить тот или иной функционал в системе. Команда system-config-* появились в Red Hat и используются в дистрибутивах построенных на его основе. Половина команд обладает интерфейсом разработанным на базе библиотеки ncurses, которая позволяет создавать интерфейсы псевдографики. Псевдографика не требует X Window и представляет данные в удобном виде, что полезно для начинающих пользователей. Другая половина команд в своей работе использует X Window и без него отказывается работать. Ниже перечислены основные команды.

Команда	Описание
setup	Основной конфигуратор системы. Здесь собран функционал многих других команд о которых написано ниже. Настройка аутентификации, firewall, клавиатуры, сети, timezone, X Window – все это можно сделать здесь.
authconfig	Настройка аутентификации в консоли
authconfig-tui	Настройка аутентификации с использованием интерфейса основанного на библиотеке ncurses
system-config-securitylevel	Настройка firewall/SELinux
system-config-date	Настройка timezone
system-config-keyboard	Выбор раскладки клавиатуры
system-config-language	Настройка языка для системы
system-config-network	Настройка сетевых интерфейсов и DNS
system-config-printer	Настройка принтера. Требуется X Window

Для получения полного списка возможных команд можно воспользоваться командой yum:

```
[root@linuxbox ~]# yum list | grep system-config
```

И в случае отсутствия команды в системе, проинсталлировать её.

```
[root@linuxbox ~]# yum -y install system-config-keyboard
```

Семейство команд adsl-* – настройка ADSL-модема

Набор команд adsl-* используются для настройки ADSL-модема.

Команда	Описание
adsl-setup	Настройка аккаунта. Логин/пароль, параметры соединения – все это задается здесь. В процессе конфигурирования будет создан файл конфигурации – /etc/sysconfig/network-scripts/ifcfg-ppp0
adsl-start	Запуск PPPoE-клиента и подключение к провайдеру
adsl-status	Запрос статуса соединения
adsl-stop	Завершение соединения

Настройки в команде adsl-setup

LOGIN NAME – логин для подключения

INTERFACE – интерфейс к которому подключен ADSL-модем

Do you want the link to come up on demand, or stay up continuously? – здесь нужно сделать выбор – подключение будет постоянным или по запросу. Время когда за Интернет оплачивали по минутному тарифу практически прошло. Поэтому логичнее выбрать подключение на постоянной основе просто нажав **<Enter>**.

DNS – задаем DNS-сервера. Если они назначаются провайдером динамически то в этом поле необходимо написать слово server

PASSWORD – пароль для подключения

USERCTL – разрешить ли обычным пользователям запускать и останавливать ADSL-подключение?

FIREWALLING – настройки брандмауэра для подключения:

NONE – не использовать брандмауэр. В целях безопасности не рекомендуется.

STANDALONE – настройки для обычного компьютера

MASQUERADE – настройки для шлюза

Do you want to start this connection at boot time? – подключаться на стадии загрузке системы или нет?

Модуль 11. Сетевая файловая система

Network File System — сетевая файловая система разработанная компанией Sun Microsystems в 1984 году. NFS позволяет использовать каталоги и файлы совместно с другими системами, посредством сети. Благодаря NFS пользователи и программы могут получать доступ к файлам на удалённых системах точно так же, как если бы эти файлы располагались в локальной системе. Предоставление доступа к своим файловым системам удаленным клиентам называется экспортирование.

Включение NFS

Скорее всего NFS уже есть в системе. Это можно проверить следующей командой.

```
[root@linuxbox ~]# chkconfig --list nfs

nfs 0:off 1:off 2:off 3:off 4:off 5:off 6:off
```

Добавляем в автозагрузку

```
[root@linuxbox ~]# chkconfig nfs on
```

и запускаем

```
[root@linuxbox ~]# service nfs start
```

Если nfs в системе не обнаружен то устанавливаем его из репозитория

```
[root@linuxbox ~]# yum -y install nfs-utils
```

В своей работе NFS использует RPC-вызовы а значит работоспособность службы можно проверить с помощью portmapper (он тоже должен быть запущен на сервере где выполняются программы использующие RPC Calls).

```
[root@linuxbox ~]# rpcinfo -p localhost | grep nfs
   100003   2   udp   2049   nfs
   100003   3   udp   2049   nfs
   100003   4   udp   2049   nfs
   100003   2   tcp   2049   nfs
   100003   3   tcp   2049   nfs
   100003   4   tcp   2049   nfs
[root@linuxbox ~]#
```

rpcinfo показывает, что NFS-сервер обслуживает несколько версий NFS – 2, 3 и 4 (последнею).

Экспортирование файловых систем

Файл конфигурации /etc/exports содержит список файловых систем которые мы экспортируем, то есть разрешаем монтировать по протоколу NFS в удаленных системах.

Каждая строка это файловая система которую мы экспортируем и режим доступа к ней.

Формат записи в файле /etc/exports показан ниже.

[файловая система] [кому разрешено получать доступ] [опции]

Пример /etc/exports:

```
[root@linuxbox ~]# cat /etc/exports
/home 192.168.79.130(rw,no_root_squash)
```

Мы разрешаем монтировать /home на сервере 192.168.79.130 в режиме rw

Внимание! Аккуратнее с пробелами в файле. Если написать /home 192.168.79.130 (rw) то сервер 192.168.79.130 получит доступ к /home в режиме ro, все остальные в rw. Если 192.168.79.130(rw) то сервер 192.168.79.130 получит доступ в режиме rw, всем остальным доступ будет запрещен.

Перечитываем файл чтобы внесенные изменения вступили в действие.

```
[root@linuxbox ~]# exportfs -r
```

Проверяем список экспортированных ФС

```
[root@linuxbox ~]# exportfs
/home           192.168.79.130
```

Монтирование NFS-ресурсов

Теперь на сервере 192.168.79.130 попробуем примонтировать ФС /home экспортированную на сервере linuxbox.

Создаем каталог куда будем монтировать

```
[root@system5 ~]# mkdir /mnt/nfs
```

Монтирования файловой системы происходит с помощью команды mount. Ключом -t задается тип файловой системы. Полный формат команды показан ниже.

mount -t nfs <сервер>:<имя_NFS_ресурса> <точка_монтирования>

сервер – IP-адрес или hostname системы, откуда монтируется NFS-ресурс

имя_NFS_ресурса – в нашем случае это /home

точка монтирования – название каталога куда примонтировать NFS-ресурс

```
[root@system5 ~]# mount -t nfs 192.168.79.132:/home /mnt/nfs
```

По умолчанию монтирование происходит в режиме rw.

Если ФС экспортированы на сервере который работает 24 часа в сутки то монтирование разделов можно добавить в /etc/fstab тем самым автоматизировав процедуру монтирования при старте компьютера.

Строчка в /etc/fstab в нашем случае будет такая:

```
192.168.79.132:/home /mnt/nfs nfs defaults
```

Опции конфигурационного файла /etc/exports

Опция	Описание
ro	Только чтение
rw	Чтение и запись
root_squash	Не разрешает пользователю root получать root-привелегии в удаленной файловой системе, все действия будут сделаны от лица пользователя nobody
no_root_squash	Пользователь root в локальной системе получает такие же права в удаленной. Стоит использовать эту опцию только в случае острой необходимости. Используется для бездисковых клиентов
all_squash	Все запросы происходят от анонимного пользователя, что способствует повышению безопасности. Актуально для публичных разделов
anonuid/anongid	Позволяет задать UID и GID пользователя от лица которого будут выполняться все запросы
sync	Синхронный режим работы, ответы на запросы происходят только после того, как данные будут надежно записаны на диск. Надежность выше, производительность меньше
async	Асинхронный режим работы, ответы на запросы происходят сразу, не дожидаясь записи на диск. Надежность ниже, производительность выше
insecure	Разрешать запросы с портов более чем 1024
no_subtree_check	Если экспортируется подкаталог файловой системы, но не вся файловая система, сервер, проверяет, находится ли запрошенный файл в экспортированном подкаталоге. Эта проверка называется проверкой подкаталога. Отключение проверки уменьшает безопасность, но увеличивает скорость передачи данных

Способы указания хостов которым разрешено монтировать ресурсы

1. Указать hostname или просто IP-адрес;

2. Указать «*», что означает разрешено всем. Если указать *.fasttech.ru то будет разрешено testers.fasttech.ru но запрещено alex.testers.fasttech.ru Чтобы этого избежать нужно написать *.*.fasttech.ru;

3. В нужных местах использовать «?», что заменяет любой символ (не применимо к IP-адресам);

4. Использовать маски подсетей, например 192.168.146.32/27;

5. Использовать NIS-группы, например @nisgroup2.

Дополнительная информация

```
[root@linuxbox ~]# man nfs
```

```
[root@linuxbox ~]# man exports
```

```
[root@linuxbox ~]# man exportfs
```

Не путайте exports и exportfs. exports – это конфигурационный файл, exportfs – программа для работы с ним.

Модуль 12. Справочная система

man

В любом дистрибутиве Linux содержатся тысячи страниц справочной документации, которые охватывают практически все команды в системе, многие функции и файлы. Самой популярной справочной системой является man (on-line manual pages). Для получения man-информации о нужной команде/файле, необходимо в качестве аргумента команды man передать соответствующее имя.

Формат команды: man [имя команды или файла]

Например: man cat

Все справочные страницы man принято оформлять по определенному стандарту. man-страницы разных команд могут содержать различные пункты, но в основном там можно увидеть такие:

1. Заголовок с именем команды, за которым в скобках следует номер раздела;

2. Имя команды и релевантные команды;

3. Порядок перечисления опций и аргументов;

4. Описание команды;

5. Возможные опции;

6. Информация об авторах;

7. Ссылки на другие источники.

Обычно все man-страницы разбиты на 8 разделов. О некоторых командах есть информация в различных разделах, в этом случае необходимо указывать, информацию из какого раздела мы хотим получить. Например: man 7 signal и man signal ведут на абсолютно разные страницы.

1. Команды пользователя (ls, mkdir, env…);

2. Системные вызовы или функции ядра (poll, chown, chroot…);

3. Библиотечные функции (CGI, fclose, getpwent…);

4. Информация по оборудованию (tty, zero, vesa…);

5. Описание формата файлов (crontab, nfs, yum.conf…);

6. Игры;

7. Разное (boot, icmp, wireless…);

8. Системное администрирование (iptables, lsof, fdisk…).

При чтение объемных man-страниц может потребоваться функция поиска. Для этого находясь в man-странице нажмите клавишу "/" и введите слово, которое хотите найти (без пробела после /). Клавиша "/" осуществляет поиск вперед. Для обратного поиска (вверх по документу) используйте клавишу "?", например:

/concatenate – вниз по документу найти слово "concatenate"

?concatenate – вверх по документу найти слово "concatenate"

whatis

Команда whatis производит поиск по базе данных whatis для законченного слова.

```
[root@linuxbox ~]# whatis cat
cat             (1)  - concatenate files and print on the standard output
cat             (1p)  - concatenate and print files
[root@linuxbox ~]#
```

База данных создается командой makewhatis.

apropos

Команда apropos производит поиск по базе данных whatis и выводит совпадения.

```
[root@linuxbox ~]# apropos Hardware
hal                (rpm) - Hardware Abstraction Layer
hwdata             (rpm) - Hardware identification and configuration data
kudzu              (rpm) - The CentOS hardware probing tool.
[root@linuxbox ~]#
```

info

Команда info позволяет просматривать справочные страницы в гипертекстовом формате info. Файлы в этом формате удобны в навигации, в том числе есть возможность перехода в другие секции документа.

Например: info emacs

Справку о самой команде info можно получить с помощью info info:

```
[root@linuxbox ~]# info info
```

help

В командный интерпретатор bash встроена команда help, которая позволяет получить справку по builtin-командам bash.

```
[root@linuxbox ~]# help unset
unset: unset [-f] [-v] [name ...]
    For each NAME, remove the corresponding variable or function.  Given
    the `-v', unset will only act on variables.  Given the `-f' flag,
    unset will only act on functions.  With neither flag, unset first
    tries to unset a variable, and if that fails, then tries to unset a
    function.  Some variables cannot be unset; also see readonly.
[root@linuxbox ~]#
```

locate

Команда locate производит поиск файлов по одной или нескольким базам созданным с помощью updatedb и выводит совпадения.

```
[root@linuxbox ~]# locate crontab
/etc/anacrontab
/etc/crontab
```

```
/usr/bin/crontab
/usr/share/man/man1/crontab.1.gz
/usr/share/man/man1p/crontab.1p.gz
/usr/share/man/man5/anacrontab.5.gz
/usr/share/man/man5/crontab.5.gz
/usr/share/vim/vim70/syntax/crontab.vim
[root@linuxbox ~]#
```

Для обновления базы необходимо воспользоваться командой updatedb.

Статистику базы можно посмотреть с помощью опции -S

```
[root@linuxbox ~]# locate -S
Database /var/lib/mlocate/mlocate.db:
    9,998 directories
    119,567 files
    5,832,849 bytes in file names
    2,513,667 bytes used to store database
[root@linuxbox ~]#
```

which

Команда which показывает абсолютный путь к указанной команде.

```
[root@linuxbox ~]# which passwd
/usr/bin/passwd
[root@linuxbox ~]#
```

Быстрая справка

У многих команд есть встроенная справка, которую можно вызвать с

помощью опции -h или --help.

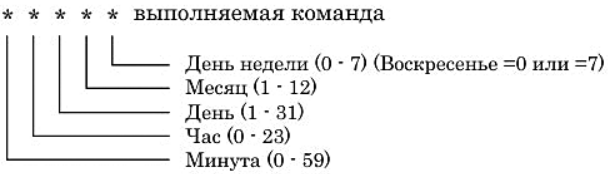
```
[root@linuxbox ~]# cat --help
Usage: cat [OPTION] [FILE]...
Concatenate FILE(s), or standard input, to standard output.

  -A, --show-all         equivalent to -vET
  -b, --number-nonblank    number nonblank output lines
  -e                     equivalent to -vE
...
[root@linuxbox ~]#
```

Выполнение заданий по расписанию

cron – это программа которая выполняет задания по расписанию. Если существует файл /etc/cron.allow то пользователи которым разрешено использовать cron должны присутствовать в этом файле. Если файл /etc/cron.allow отсутствует то происходит поиск файла /etc/cron.deny. Пользователи перечисленные в этом файле не могут использовать cron, всем остальным разрешено. Чтобы разрешить использовать cron всем пользователям в системе необходимо убедиться, что существует пустой файл /etc/cron.deny и отсутствует файл /etc/cron.allow. По умолчанию так и есть.

Формат crontab-файла



Каждое задание занимает одну строку и состоит из шести полей.

минута	час	день_месяца	месяц	день_недели	команда

Допустимые значения:

Минута: от 0 до 59

Час: от 0 до 23

день_месяца: от 1 до 31

месяц: от 1 до 12 или три буквы названия месяца (от jan до dec)

день_недели: от 0 до 6 (0 или 7 – это воскресенье) или три буквы дня недели (от sun до sat)

Каждое из полей даты и времени может содержать символ *, который означает любое возможное значение. Диапазон указывается с помощью дефиса. Список чисел указывается через запятую. Интервал можно указать через символ / (slash).

crontab-файлы

При старте, демон **cron** выполняет следующие файлы:

1. crontab-файлы в каталоге /var/spool/cron/

2. crontab-файлы в каталоге /etc/cron.d/

3. системный crontab-файл /etc/crontab

Сервис cron каждую минуту читает их и выполняет задания, которые необходимо выполнить в данный момент.

Системный crontab-файл обычно выглядит вот так:

```
[root@linuxbox ~]# cat /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/


# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
[root@linuxbox ~]#
```

Записи в файле указывают на следующее:

SHELL – какой командный интерпретатор использовать для выполнения команд. Если shell не указан то он будет взят из файла /etc/passwd для пользователя, который является владельцем файла.

PATH – добавить каталоги в переменную PATH.

MAILTO – на какое адрес отсылать результаты выполнения заданий.

HOME – корневой каталог для пользователя. При необходимости доступа к специальным свойствам интерпретатора, значения переменных SHELL и HOME можно изменить, не зависимо от того, что прописано в /etc/passwd. Параметр является не обязательным.

- Содержимое каталога /etc/cron.hourly будет запускаться каждый час на первой минуте часа
- Содержимое каталога /etc/cron.daily будет запускаться каждый день на второй минуте четвертого часа
- Содержимое каталога /etc/cron.weekly будет запускаться каждое воскресенье на 22-ой минуте 4-го часа
- Содержимое каталога /etc/cron.monthly будет запускаться каждый первый день месяца на 42-ой минуте 4-го часа

Размещение файлов в каталогах

/etc/cron.hourly

/etc/cron.daily

/etc/cron.weekly

/etc/cron.monthly

доступно только пользователю root и все файлы выполняются с его правами. Файлы должны обладать правом на выполнение.

Команда crontab

Для создания crontab файлов обычным пользователям необходимо использовать команду crontab. Команда служит для создания, изменения и добавления файла для демона cron.

Команда и опция	Описание
crontab -e	Редактировать свой crontab-файл
crontab -l	Просмотреть свой crontab-файл
crontab -r	Удалить свой crontab-файл
crontab -u	Позволяет задать конкретного пользователя, для которого будут применяться вышеописанные опции. Опция доступна только пользователю root crontab -u username -e crontab -u username -l crontab -u username -r

crontab-файл можно создать из существующего файла. Например создаем файл с именем file и содержимым как ниже

```
[user1@linuxbox ~]$ cat file
0-59 * * * * /bin/true
[user1@linuxbox ~]$ whoami
user1
[user1@linuxbox ~]$
```

Далее превращаем его в crontab-файл, передав имя файла в качестве аргумента команде crontab

```
[user1@linuxbox ~]$ crontab file
```

Все файлы пользователей размещаются в каталоге /var/spool/cron/ и их имя соответствует имени пользователя которому они принадлежат.

```
[root@linuxbox ~]# ls -l /var/spool/cron/
total 8
-rw------- 1 root  root   19  Sep  12  22:56  root
-rw------- 1 user1 root   23  Sep  13  18:44  user1
[root@linuxbox ~]# cat /var/spool/cron/user1
0-59 * * * * /bin/true
[root@linuxbox ~]#
```

crontab в примерах

Выполнять задание каждую минуту

```
0-59 * * * * /bin/true
```

Выполнять задание каждые 15 минут

```
*/15 * * * * /bin/true
```

Выполнять задание каждый день в полночь

```
0 0 * * * /bin/true
```

Выполнять задание в 0:30, 2:30, 4:30, 6:30 и т.д.

```
30 */2 * * * /bin/true
```

Выполнять задание в 23 часа 59 минут 31 декабря каждого года

```
59 23 31 dec * HappyNewYear.sh
```

Упаковщики и архиваторы

tar – это формат файлового архива и название популярной программы для упаковки множества файлов в один архив. tar не занимается сжатием архивов, для этого используются внешние программы, такие как gzip или bzip2.

Формат команды: tar [опции] [имя_файла-архива] [каталог/и]

Основные опции команды tar

Опция	Описание
-c	Создание нового архива
-j	Сжатие с помощью bzip2
-z	Сжатие с помощью gzip
-v	Выводить отладочную информацию
-f	Название файла архива
-x	Извлечение из архива

Создание tar-архива

```
[root@linuxbox ~]# du -sh /etc/

57M    /etc/

[root@linuxbox ~]# tar cvf etc.tar /etc/

[root@linuxbox ~]# ls -lh

total 48M

-rw-r--r-- 1 root root 48M Sep 13 23:30 etc.tar

[root@linuxbox ~]#
```

Как видно из листинга выше, размер tar-файла практически соответствует размеру оригинального каталога. Для создания сжатого архива необходимо добавить опцию z.

Создание tar-архива сжатого с помощью gzip

```
[root@linuxbox ~]# du -sh /etc/

57M    /etc/

[root@linuxbox ~]# tar czvf etc.tar.gz /etc/

[root@linuxbox ~]# ls -lh

total 4.8M

-rw-r--r-- 1 root root 4.8M Sep 14 13:29 etc.tar.gz

[root@linuxbox ~]#
```

Благодаря сжатию архива с помощью программы gzip, архив уменьшился по размеру в 10 раз.

Создание tar-архива сжатого с помощью bzip2

Для упаковки каталога с помощью tar и сжатия с помощью bzip2 вместо опции z используется опция j.

```
[root@linuxbox ~]# du -sh /etc/

57M    /etc/

[root@linuxbox ~]# tar cjvf etc.tar.bz2 /etc/

[root@linuxbox ~]# ls -lh

total 3.2M

-rw-r--r-- 1 root root 3.2M Sep 14 13:41 etc.tar.bz2

[root@linuxbox ~]#
```

С помощью bzip2 сжатие каталога /etc/ получилось лучше.

Разархивирование tar.gz-файла

```
[root@linuxbox ~]# tar xzvf etc.tar.gz
```

```
...
[root@linuxbox ~]# ls -l
total 4868
drwxr-xr-x 84  root  root     4096  Sep  14 13:24 etc
-rw-r--r--    1  root  root  4965172  Sep  14 14:59 etc.tar.gz
[root@linuxbox ~]#
```

Если необходимо получить только некоторые файлы из архива то перечисляем их через пробел. Символ / (slash) ставить перед именем каталога не нужно, команда tar убирает его, чтобы файл-архив при разархивации не перезаписал каталог в корне файловой системы.

```
[root@linuxbox ~]# tar xzvf etc.tar.gz etc/shells etc/profile etc/vimrc
etc/shells
etc/vimrc
etc/profile
[root@linuxbox ~]# ls -l etc/
total 12
-rw-r--r-- 1 root root  937   Jan   31  2006  profile
-rw-r--r-- 1 root root   69   Sep    6  23:21 shells
-rw-r--r-- 1 root root 1488  Nov 25  2008  vimrc
[root@linuxbox ~]#
```

Разархивирование bz2-файла

Для разархивирования bz2-файла может использоваться команда tar xjvf или bzip2 -d

```
[root@linuxbox ~]# tar xjvf etc.tar.bz2
[root@linuxbox ~]# ls -l
total 3268
```

```
drwxr-xr-x 84  root root      4096   Sep 14  13:24 etc
-rw-r--r--      1 root  root 3337026   Sep 14  15:16 etc.tar.bz2
[root@linuxbox ~]#
```

Если необходимо получить только некоторые файлы из архива то перечисляем их через пробел. Символ / (slash) ставить перед именем каталога не нужно, команда tar убирает его, чтобы файл-архив при разархивации не перезаписал каталог в корне файловой системы.

```
[root@linuxbox ~]# tar xjvf etc.tar.bz2 etc/shells etc/profile etc/vimrc
etc/shells
etc/vimrc
etc/profile
[root@linuxbox ~]# ls -l etc/
total 12
-rw-r--r-- 1  root  root   937  Jan   31   2006   profile
-rw-r--r-- 1  root  root    69  Sep    6   23:21  shells
-rw-r--r-- 1  root  root  1488  Nov  25    2008  vimrc
[root@linuxbox ~]#
```

Файловый менеджер mc

mc (Midnight Commander) – это файловый менеджер на базе библиотеки ncurses (псевдографика). mc поддерживает все основные операции с файлами: копирование, перемещение, архивация, изменение прав доступа, создание ссылок, каталогов и т.д.

F1 – справка

F2 – пользовательское меню

F3 – просмотреть файл

F4 – редактировать файл

F5 – копировать

F6 – переименовать

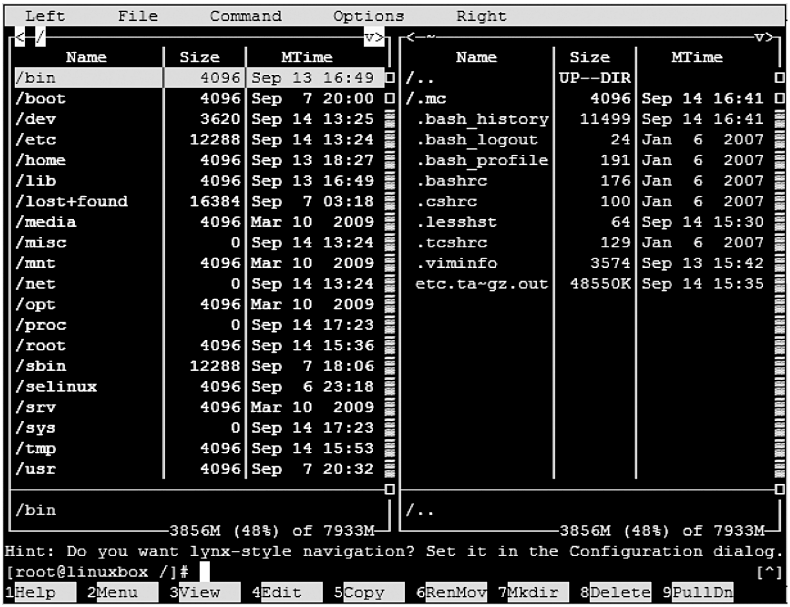
F7 – создать каталог

F8 – удалить

F9 – выпадающее меню

F10 – выход

Tab – перемещение между окнами



Работа с ssh: подключение к удаленной системе

ssh (Secure Shell) – протокол прикладного уровня позволяющий подключаться к удаленной системе Unix/Linux и выполнять в ней команды. В отличие от telnet, ssh шифрует весь трафик включая пароли. Для работы ssh необходим ssh-клиент, который используется для подключения к удаленной системе и ssh-сервер, который ожидает входящие соединения (обычно на порту 22) и проводит аутентификацию. Для соединения между собой, клиент и сервер должны создать пары ключей – открытых и закрытых и обменяться открытыми ключами. Но в большинстве случаев используются классические пароли.

Для подключения к удаленной системе используется команда ssh.

Формат команды: ssh [опции] [имя_пользователя]@[имя_хоста]

Подключимся под учетной записью user1 к хосту 192.168.79.129

```
[admin@system5 ~]# ssh user1@192.168.79.129
The authenticity of host '192.168.79.129 (192.168.79.129)' can't be established.
RSA key fingerprint is 39:4a:e8:5f:b7:33:14:e6:29:7f:12:08:da:06:77:38.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.79.129' (RSA) to the list of known hosts.
user1@192.168.79.129's password:
[user1@linuxbox ~]$
```

В первый раз будет выведено предупреждение о том, что подлинность хоста не может быть установлена и запрос подтверждения подключения. После ответа yes открытый ключ хоста к которому мы подключаемся будет помещен в наш файл ~/.ssh/known_host и в дальнейшем таких сообщений не будет.

Аутентификация по ключам

Ssh имеет несколько способов аутентификации. Самые распространенные это по паролю и метод Identity/Pubkey, когда вы генерируете два ключа, личный и публичный. Публичный ключ записываете в файл ~/.ssh/authorized_keys пользователю в удаленной системе. В этом случае при подключение у нас не будут запрашивать пароль, но необходимо убедиться, что доступ в помещение с вашим компьютером ограничено. Так как для подключения к удаленной системе будет достаточно набрать команду:

ssh [пользователь]@[хост]

Генерируем пару ключей с помощью команды ssh-keygen.

```
[user@system5 ~]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user1/.ssh/id_rsa):
Created directory '/home/user1/.ssh'.
Enter passphrase (empty for no passphrase):
```

```
Enter same passphrase again:
Your identification has been saved in /home/user1/.ssh/id_rsa.
Your public key has been saved in /home/user1/.ssh/id_rsa.pub.
The key fingerprint is:
96:3f:3e:df:bd:4b:db:0d:97:3f:bb:b7:b0:c1:72:04 user1@linuxbox.company.ru
[user@system5 ~]$
```

С помощью команды ssh-copy-id наш публичный ключ переносим пользователю в удаленную систему.

```
[user@system5 ~]$ ssh-copy-id -i ~/.ssh/id_rsa.pub user2@192.168.79.129
27
The authenticity of host '192.168.79.129 (192.168.79.129)' can't be established.
RSA key fingerprint is 39:4a:e8:5f:b7:33:14:e6:29:7f:12:08:da:06:77:38.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.79.129' (RSA) to the list of known hosts.
user2@192.168.79.129's password:
Now try logging into the machine, with "ssh 'user2@192.168.79.129'", and check in:

  .ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.

[user@system5 ~]$
```

Подключаемся к системе без пароля.

```
[user@system5 ~]$ ssh user2@192.168.79.129
[user2@linuxbox ~]$
```

ssh-клиент для ОС Windows

Для операционной системы Windows самым распространенным ssh-клиентом является бесплатная программа putty, скачать которую можно по адресу:

http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

Модуль 14. Мониторинг системы

Журнальные файлы

Журнальные файлы или log-файлы служат для сбора сообщений по работе операционной системы и приложений. Это позволяет отслеживать сбои в работе и является хорошим средством диагностики.

Для работы с системой журналирования необходимо сделать следующее:

1. Создать политику журналирования;

2. Производить ротацию log-файлов;

3. Удалять не актуальные log-файлы.

Каждое приложение может обладать собственным файлом журнала или записывать сообщения в стандартный системный журнал /var/log/messages

Файлы журналов располагаются в каталоге /var/log/. Основные файлы перечислены в таблице.

Файл	Описание
messages	Основной системный журнальный файл
audit/audit.log	Сообщения авторизации
cron	Сообщения от cron
dmesg	Сообщения процесса загрузки системы
wtmp	Учет времени соединения. Бинарный файл
yum.log	Сообщения от yum
secure	Сообщения связанные с безопасностью
maillog	Сообщения от почтовой службы
lastlog	Регистрация последних подключений пользователей

Файлы журналов внешних приложений скорее всего будут носить созвучные названия, например – mysqld.log, httpd.log, proftpd.log.

Syslog

Для регистрации и управления сообщениями используется специальное программное обеспечение – Syslog.

Syslog – это система регистрации сообщений. До ее появления различные программы использовали собственные методы журналирования что

было не очень удобно для конечных пользователей. Syslog позволяет сортировать сообщения по источникам и степени важности и направлять их в соответствующие файлы. Также есть возможность отправки сообщений на терминал пользователя или на другой сервер.

Syslog автоматически запускается при старте системы.

```
[root@linuxbox ~]# ps aux | grep syslogd
root     2474  0.0  0.2   1716   564 ?        Ss   16:17   0:00 syslogd -m 0
root     6976  0.0  0.2   3912   648 pts/0    R+   19:38   0:00 grep syslogd
[root@linuxbox log]#
```

Для работы Syslog используется демон syslogd, конфигурационный файл которого /etc/syslog.conf

Базовый формат файла: селектор действие

Например строка ниже обеспечивает сохранение всех сообщений от cron в файл /var/log/cron

```
cron.*                                  /var/log/cron
```

Селектор означает программу ("средство" в терминологии Syslog) которая посылает регистрационные сообщения. Уровень обозначает важность сообщения, в файле конфигурации Syslog указывается, как минимум какой уровень важности должны иметь сообщения, чтобы быть зарегистрированными. Селекторы могут содержать * и none, что означают все и нечего соответственно. Селектор может включать группу средств разделенных запятыми или точкой с запятой.

Формат селектора: средство.уровень

Средства и уровни необходимо выбирать из стандартного списка.

Возможные средства (facilities) Syslog

Средства	Описание
auth	Сообщения связанные с безопасностью
authpriv	Сообщения авторизации пользователей
cron	Сообщения демона cron
daemon	Сообщения системных демонов
kern	Сообщения ядра

lpr	Система буферизации печати
mail	Почтовые сообщения
mark	Метки времени генерируемые через определенный интервал
news	Система телеконференций Usenet
syslog	Внутренние сообщения Syslog
user	Пользовательские процессы
uucp	Сообщения UUCP
local0 – local7	Для любого использования. Red Hat использует local7 для сообщений загрузки ОС
*	Используется для всего

Возможные приоритеты (priorities) Syslog

Уровень серьезности	Описание
emerg	Экстренные ситуации
alert	Срочные ситуации
crit	Критические состояния
err	Другие ошибочные состояния
warning	Предупреждающие сообщения
notice	События которые заслуживают внимания
info	Информационные сообщения
debug	Отладочные сообщения

Возможные действия Syslog

Действие	Описание
Имя_файла	Записать сообщение в файл. Файл уже должен существовать
@Имя_компьютера	Переслать сообщение демону syslogd на компьютер с указанным именем
@IP_адрес	Переслать сообщение на компьютер с заданным IP-адресом
Пользователь1, пользователь2,…	Вывести сообщение на терминалы указанных пользователей
*	Вывести сообщение на терминалы всех зарегистрированных пользователей

Квалификаторы

В последних версиях syslogd появилась возможность использования квалификаторов, которые позволяют использовать условия.

Квалификатор	Описание
=	Только этот приоритет
!	За исключением этого и более высоких приоритетов

Примеры использования квалификаторов

Селектор	Описание
mail.*	Регистрировать все почтовые сообщения
mail.=info	Выбор почтовых сообщений только уровня info
mail.info;mail.!err	Выбор почтовых сообщений уровней info, notice и warning
mail.debug;mail.!=notice	Выбор почтовых сообщений любого уровня, кроме notice

logrotate

Для бережного использования дискового пространства необходимо производить ротацию журнальный файлов, то есть архивировать файлы по прошествии какого-то времени или при достижение журнальным файлом определенного размера. И хранить определенное количество таких архивов. Если это не отслеживать то журнальный файлы могут очень быстро заполнить файловую систему, что может привести к проблемам. Для автоматизации ротации в Linux используется logrotate. Он ежедневно запускается с помощью cron и работает в соответствие со своим конфигурационным файлом /etc/logrotate.conf

Файл конфигурации /etc/logrotate.conf

Опция	Описание
weekly	Игнорировать размер файла, производит ротацию логов раз в неделю
rotate 4	Хранить журнальные файлы за 4 недели
create	Создавать новый файл после ротации старого
compress	Сжимать файлы в архиве
include	Подключить к конфигурации файл или каталог

В каталоге /etc/logrotate.d/ расположены настройки logrotate для различных программ. На примере рассмотрим файл конфигурации syslog.

/etc/logrotate.d/syslogd

```
/var/log/messages /var/log/secure /var/log/maillog /var/log/spooler /var/log/
boot.log /var/log/cron {
    sharedscripts
    postrotate
        /bin/kill -HUP `cat /var/run/syslogd.pid 2> /dev/null` 2> /dev/null || true
        /bin/kill -HUP `cat /var/run/rsyslogd.pid 2> /dev/null` 2> /dev/null || true
    endscript
}
```

До фигурных скобок через пробел перечислены файлы, для которых будет выполняться ряд действий (перечисленные в фигурных скобках).

Опция	Описание
sharedscripts	Скрипты будут запускаться только один раз, не важно сколько файлов задано
postrotate … endscript	Между postrotate и endscript заданы команды bash, исполняемые после ротации. В нашем случае демонам syslogd и rsyslogd будет отправлен сигнал HUP, что заставить их перечитать файлы конфигурации

Команда logger

Команда logger позволяет обрабатывать сообщения поступающие от shell-сценариев.

Добавим в файл /etc/syslog.conf строку

```
local0.notice /tmp/myscript.log
```

Перезагрузим syslog

```
[root@linuxbox ~]# service syslog restart
```

Создаем пустой файл

```
[root@linuxbox ~]# touch /tmp/myscript.log
```

С помощью команды logger на local0 с приоритетом notice отправим строку “my own notice message”. Согласно конфигурации Syslog, сообщения отправляемые на local0 с приоритетом notice будут записываться в /tmp/myscript.log

```
[root@linuxbox ~]# logger -p local0.notice «my own notice message»
```

Проверим наш журнальный файл

```
[root@linuxbox ~]# cat /tmp/myscript.log

Sep  4 20:35:10 linuxbox root: my own notice message

[root@linuxbox ~]#
```

Мониторинг различных параметров системы

Мониторинг свободного места

Выведем список 5 пользователей занимающих наибольшее количество дискового пространства в своем домашнем каталоге.

```
[root@linuxbox ~]# du -cms /home/* | sort -rn | head -6

8200 total

6100 /home/james

900 /home/ronaldo

570 /home/neo

420 /home/tester

210 /home/mybestuser
```

Как видим пользователь james занимает 6.1 Гб места и т.д.

Защита системы от пользовательских процессов

Пользовательские процессы без проблем могут занять все дисковое пространство и тем самым вывести сервер из строя. Чтобы этого избежать

воспользуемся командой ulimit и ограничим возможный размер файла до 10Мб.

```
[root@linuxbox ~]# ulimit -f 10000

[root@linuxbox ~]# yes 'some shit' > shit.txt

File size limit exceeded

[root@linuxbox ~]# ls -l

total 10016

-rw-r--r-- 1 root root 10240000 Mar 14 21:36 shit.txt
```

Точно также можно задавать другие опции ограничения, такие как максимальное количество файлов, максимальное количество открытых дескрипторов, процессорное время, объем виртуальной памяти доступный в shell и т.д.

Просмотреть все возможные параметры

```
[root@linuxbox ~]# ulimit -a
```

Учитывайте специфику вашей системы, например на серверах баз данных размеры файлов могут быть очень большими.

Мониторинг S.M.A.R.T.-параметров жесткого диска

```
[root@linuxbox ~]# smartctl -a /dev/sda
```

S.M.A.R.T. – это Self-Monitoring, Analysis and Reporting Technology. Технология самоконтроля и анализа, которой снабжены все современные диски. Она позволяет отслеживать основные параметры жестких дисков и прогнозировать потенциальный отказ.

Один из способов выяснить как ваш HDD называется в системе это выполнить команду fdisk -l.

Мониторинг сетевых портов в Linux

Посмотреть конкретный сетевой порт

```
[root@linuxbox ~]# netstat -nlp | grep :8080
```

```
tcp        0      0      :::8080 :::*          LISTEN      2697/httpd
```

Как видим, на порту 8080 работает веб-сервер Apache.

Весь список активных портов

```
[root@linuxbox ~]# netstat -tlnp
```

Мониторинг открытых файлов и сокетов

Для этих целей подойдет lsof, которая выводит список открытых файлов.

Вывод всех открытых файлов

```
[root@linuxbox ~]# lsof
```

Смотрим открытые файлы работающие с сетью по IPv4

```
[root@linuxbox ~]# lsof -i 4
COMMAND    PID     USER   FD   TYPE    DEVICE  SIZE  NODE    NAME
portmap    2341     rpc    3u   IPv4     6692         UDP    *:sunrpc
rpc.statd  2366    root    7u   IPv4     6748         TCP    *:852 (LISTEN)
cupsd      2657    root    3u   IPv4     7392         TCP    localhost.localdomain:ipp
cupsd      2657    root    5u   IPv4     7395         UDP    *:ipp
php-cgi    2662   nginx    0u   IPv4     7415         TCP    localhost.localdomain:...
nginx      3915   nginx    5u   IPv4    10864         TCP    *:https (LISTEN)
nginx      3915   nginx    6u   IPv4    10865         TCP    *:http (LISTEN)
```

Список открытых файлов по NFS

```
[root@linuxbox ~]# lsof -N
```

Список открытых файлов в /var/log/

Опция +d покажет открытые файлы только в этом каталоге а +D и во вложенных.

```
[root@linuxbox ~]# lsof +d /var/log
```

```
COMMAND   PID   USER  FD  TYPE  DEVICE  SIZE    NODE      NAME
syslogd   2289  root  1w  REG   253,0   382311  3047763   /var/log/messages
syslogd   2289  root  2w  REG   253,0   25939   3047764   /var/log/secure
syslogd   2289  root  3w  REG   253,0   6285    3047765   /var/log/maillog
syslogd   2289  root  4w  REG   253,0   7348    3047768   /var/log/cron
syslogd   2289  root  5w  REG   253,0   0       3047766   /var/log/spooler
syslogd   2289  root  6w  REG   253,0   1532    3047767   /var/log/boot.log
acpid     2590  root  2w  REG   253,0   3374    3047640   /var/log/acpid
nginx     3914  root  2u  REG   253,0   3398    3047715   /var/log/nginx.er...
```

Список открытых файлов для конкретного пользователя

```
[root@linuxbox ~]# lsof -u james
```

Мониторинг запущенных процессов

Все процессы

```
[root@linuxbox ~]# ps -eF
```

Вывести дерево процессов

```
[root@linuxbox ~]# ps -ejH
```

Мониторинг системных ресурсов в реальном времени

Команда top предоставляет информацию о запущенных процессах, включая данные об использовании процессом CPU и памяти, пользователях, запустивших процесс и их PID (Process ID), а также о времени, прошедшем с момента запуска процесса.

```
[root@linuxbox ~]# top
```

Команда htop выводит эту статистику в более красивом виде. Для установки htop необходимо подключить репозиторий EPEL, откуда она и будет установлена.

```
[root@linuxbox ~]# rpm -Uvh http://download.fedora.redhat.com/pub/epel/5/
i386/epel-release-5-3.noarch.rpm
[root@linuxbox ~]# yum -y install htop
[root@linuxbox ~]# htop
```

Мониторинг свободного места в разделах

```
[root@linuxbox ~]# df -h
Filesystem      Size  Used  Avail  Use%  Mounted on
/dev/sda1       7.8G  1.6G  5.9G   21%   /
tmpfs           25M   0     125M   0%    /dev/shm
[root@linuxbox ~]#
```

Мониторинг сетевой подсистемы в реальном времени

В этом нам отлично поможет пакет ntop.

Убедитесь что репозиторий EPEL подключен.

```
[root@linuxbox ~]# rpm -Uvh http://download.fedora.redhat.com/pub/epel/5/
i386/epel-release-5-3.noarch.rpm
[root@linuxbox ~]# yum -y install ntop
```

Инициализируем ntop, задаем пароль администратора.

```
[root@linuxbox ~]# ntop
```

Останавливаем работу ntop нажатием CTRL+C

В файл /etc/ntop.conf добавляем IP-адрес машины на которой установлен ntop в опциях --http-server и --https-server

Запускаем ntop

```
[root@linuxbox ~]# service ntop start
```

Теперь через браузер можно получить доступ к статистики ntop. По

умолчанию ntop работает на порту 3000.

К своему интерфейсу ntop я получил набрав в браузере

http://192.168.146.132:3000/

Есть и консольная программа для просмотра статистики по сетевому интерфейсу – iftop. Для ее установки подключите репозиторий EPEL.

```
[root@linuxbox ~]# rpm -Uvh http://download.fedora.redhat.com/pub/epel/5/
i386/epel-release-5-3.noarch.rpm
[root@linuxbox ~]# yum -y install iftop
[root@linuxbox ~]# iftop
```

Мониторинг работы DNS-сервера в реальном времени

Ставим пакет dnstop на том же сервере где и dns-сервер.

```
[root@linuxbox ~]# rpm –Uvh ftp://fr.rpmfind.net/linux/dag/redhat/el5/en/i386/
dag/RPMS/dnstop-0.0.20070510-1.el5.rf.i386.rpm
```

Далее запускаем dnstop и смотрим за статистикой.

```
[root@linuxbox ~]# dnstop eth0
```

В момент когда dnstop запущен можно нажимать на специальные клавиши и смотреть статистику различного рода.

Основные кнопки:

s – показать таблицу с адресами источников запросов;

d – показать таблицу с пунктами назначения запросов, куда они передаются;

t – показывать статистику по типам запросов.

Другие опции в документации – man dnstop

Мониторинг соединений proftpd в реальном времени

```
[root@linuxbox ~]# ftptop
```

Статистика по виртуальной памяти

```
[root@linuxbox ~]# vmstat
```

Статистика по процессору и устройствам ввода-вывода

Ставим пакет sysstat (содержит в себе программы sa, mpstat, iostat, sar)

```
[root@linuxbox ~]# yum -y install sysstat
```

Информация по вводу-выводу

```
[root@linuxbox ~]# iostat
```

Информация по процессору

```
[root@linuxbox ~]# mpstat
```

Мониторинг подключенных пользователей

Для того чтобы посмотреть кто сейчас подключен к системе используем команду who.

```
[root@linuxbox ~]# who
root    pts/0       2009-09-05 12:57 (192.168.79.1)
[root@linuxbox ~]#
```

История подключений в систему

Для того чтобы посмотреть историю подключений в системе, используем команду last.

Все подключения

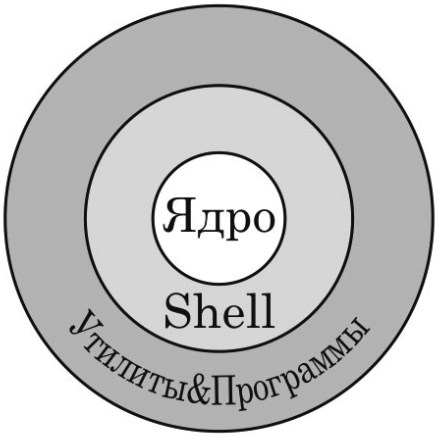
```
[root@linuxbox ~]# last
```

Последние 10 подключений в систему

```
[root@linuxbox ~]# last | head -n 10
```

Модуль 15. Ядро операционной системы

Ядро – это центральная часть операционной системы. Оно непосредственно взаимодействует с аппаратной частью, изолируя прикладные программы от особенностей архитектуры. С помощью системных вызовов, ядро предоставляет пользовательским процессам ряд услуг. Таким образом в системе можно выделить два уровня привилегий: уровня системы (пользователь root) и уровня пользователей (обычные пользователи).



UNIX/Linux-систему можно разбить на три основных уровня:

1. Аппаратные средства;

2. Ядро операционной системы;

3. Пользовательские программы.

Монолитное ядро

В своей работе Linux использует монолитное ядро, это когда компоненты операционной системы являются не самостоятельными модулями, а составными частями одной большой программы. Монолитное ядро работает в едином адресном пространстве. Преимущества такого подхода заключаются в высокой скорости работы и легкости разработки новых модулей. Основной недостаток как раз заключается в едином адресном пространстве в случае которого сбой в одном из компонентов может нарушить работоспособность всей системы.

Когда необходимо обновлять ядро?

Можно выделить два основных момента:

1. Адаптировать ядро под конкретную систему;

2. Добавить поддержку новых устройств.

Если что-то конкретное от ядра не требуется то обновлять его не обязательно. Кроме случаев, когда новое ядро содержит важные исправления и это может сказаться на стабильности системы или ее безопасности.

Установка нового ядра

Есть два способа установить новое ядро.

1. Установка из репозитория. Самый простой и рекомендуемый для новичков способ. Но обновление ядра в репозиториях появляется достаточно медленно и такой способ устроит не каждого;

2. Ручная установка. Необходимо скачать исходные коды ядра из Интернета, сконфигурировать, скомпилировать, установить в систему и настроить загрузчик GRUB для нового ядра. Этот способ сложнее первого.

Установка ядра из репозитория

Для того чтобы посмотреть текущею версию ядра воспользуйтесь командой uname -r:

```
[root@linuxbox ~]# uname -r
2.6.18-92.el5
[root@linuxbox ~]#
```

Теперь посмотрим, есть ли более новая версия в репозитории CentOS.

```
[root@linuxbox ~]# yum list | grep kernel
kernel.i686                 2.6.18-92.el5          installed
kernel.i686                 2.6.18-128.7.1.el5     updates
kernel-PAE.i686             2.6.18-128.7.1.el5     updates
kernel-PAE-devel.i686       2.6.18-128.7.1.el5     updates
kernel-devel.i686           2.6.18-128.7.1.el5     updates
```

```
kernel-doc.noarch              2.6.18-128.7.1.el5   updates
kernel-headers.i386            2.6.18-128.7.1.el5   updates
kernel-xen.i686                2.6.18-128.7.1.el5   updates
kernel-xen-devel.i686          2.6.18-128.7.1.el5   updates
...
[root@linuxbox ~]#
```

Самые интересные строки в нашем случае это 1 и 2.

```
kernel.i686                    2.6.18-92.el5        installed
kernel.i686                    2.6.18-128.7.1.el5   updates
```

Первая строка показывает какое ядро сейчас установлено (installed). Вторая строка показывает последнею доступную версию ядра.

Для установки новой версии ядра выполняем команду yum install

```
[root@linuxbox ~]# yum -y install kernel
```

После установки в конфигурационный файл загрузчика GRUB будут помещены следующие строки:

```
title  CentOS (2.6.18-128.7.1.el5)
      root (hd0,0)
      kernel /boot/vmlinuz-2.6.18-128.7.1.el5 ro root=LABEL=/
      initrd /boot/initrd-2.6.18-128.7.1.el5.img
```

Новая запись будет находиться перед записью старого ядра, что сделает новое ядро загружаемым по умолчанию. Перезагрузим систему.

```
[root@linuxbox ~]# shutdown -r now
```

После перезагрузки проверяем версию ядра.

```
[root@linuxbox ~]# uname -r
2.6.18-128.7.1.el5
[root@linuxbox ~]#
```

Ядра из репозитория уже протестированы и гарантированно работают сразу же после установки.

Описание возможных пакетов установки

Пакет	Описание
kernel	Содержит ядро. Для систем x86 используется только первые 4 Гб оперативной памяти. Для систем с более чем 4 Гб необходимо использовать пакет kernel-PAE
kernel-devel	Содержит заголовочные и make-файлы для пакета kernel
kernel-PAE	Ядро с поддержкой **Physical Address Extension,** которое позволяет использовать больше чем 4 Гб оперативной памяти
kernel-PAE-devel	Содержит заголовочные и make-файлы для пакета kernel-PAE
kernel-doc	Файлы документации ядра
kernel-headers	Заголовочные файлы которые описывают интерфейс между ядром Linux и пространством пользователя
kernel-xen	Ядро с поддержкой виртуализации Xen
kernel-xen-devel	Содержит заголовочные и make-файлы для пакета kernel-xen

Установка ядра вручную

Перед установкой ядра вручную убедитесь, что в системе есть следующие пакеты:

1. gcc (если нет то устанавливаем – yum -y install gcc);

2. ncurses-devel (если нет то устанавливаем – yum -y install ncurses-devel).

Сборку ядра принято делать в каталоге /usr/src/ поэтому переходим в него.

```
[root@linuxbox ~]# cd /usr/src/
```

Скачиваем с сайта kernel.org последнею стабильную версия ядра. Сейчас активно поддерживается две ветки ядер – это 2.4.x и 2.6.x мы будем использовать 2.6.x как более современную и функциональную.

```
[root@linuxbox src]# wget http://www.kernel.org/pub/linux/kernel/v2.6/linux-
2.6.30.5.tar.bz2
```

Разархивировать архив в формате .bz2 можно с помощью команды tar xjvf

```
[root@linuxbox ~]# tar xjvf linux-2.6.30.5.tar.bz2
```

Переходим в каталог с исходными кодами ядра.

```
[root@linuxbox ~]# cd linux-2.6.30.5
```

Можно почитать файл README, чтобы прояснить для себя многие моменты связанные с компиляцией ядра.

```
[root@linuxbox linux-2.6.30.5]# less README
```

Этап 1. Конфигурирование ядра

Для конфигурирования ядра может применяться несколько видов интерфейсов.

make menuconfig – конфигурирование будет происходит в интерфейсе псевдографики основанном на библиотеке ncurses.

make xconfig – конфигурирование будет происходит в графическом режиме X Window.

make config – пошаговый консольный сценарий.

Если вы используете X Window то xconfig будет наиболее удобным. Для консольного режима конфигурирования лучше всего подойдет menuconfig.

```
[root@linuxbox linux-2.6.30.5]# make menuconfig
```

Конфигурируя ядро, то есть добавляя и удаляя из него функционал, есть следующие варианты выбора:

M – как модуль. Поддержка какого либо устройства или функции будет добавлена в виде загружаемого модуля.

* – это built-in то есть встроенная в ядро поддержка устройства или какой либо функции.

После завершения конфигурирования ядра выходим из интерфейса нажав <Exit>. Следующим этапом будет очистка окружения от не нужных файлов.

Этап 2. Очистка окружения от временных и не нужных файлов

Выполняем команду make clean

```
[root@linuxbox linux-2.6.30.5]# make clean
```

Этап 3. Создание сжатого образа ядра

```
[root@linuxbox linux-2.6.30.5]# make bzImage
```

Процесс создания ядра может занять продолжительное время.

Этап 4. Компиляция модулей

```
[root@linuxbox linux-2.6.30.5]# make modules
```

Процесс компиляции модулей может занять продолжительное время.

Этап 5. Установка модулей

```
[root@linuxbox linux-2.6.30.5]# make modules_install
```

Этап 6. Установка ядра

```
[root@linuxbox linux-2.6.30.5]# make install
```

На этом этапе в каталог /boot копируются файлы:

System.map-<версия ядра> – символьная таблица используемая ядром;

initrd-<версия ядра>.img – RAM disk содержащий временную файловую систему и загружаемый в память раньше ядра;

vmlinuz-<версия ядра> – файл ядра Linux.

В конфигурационный файл /boot/grub/grub.conf добавляется запись для нового ядра. Но по умолчанию будет загружаться старое ядро. После того как вы протестируете новое ядро его можно установить ядром по умолчанию.

Перезагружаем систему чтобы протестировать работоспособность нового ядра. Его необходимо будет выбрать в меню выбора во время загрузки.

```
[root@linuxbox linux-2.6.30.5]# shutdown -r now
```

Проверяем текущею версия ядра если система удачно загрузилась.

```
[root@linuxbox ~]# uname -r
2.6.30.5
[root@linuxbox ~]#
```

Мои поздравления! :)

Команда sysctl: изменение параметров ядра в режиме реального времени

Команда sysctl используется для изменения параметров ядра на лету. Список доступных параметров находится в /proc/sys/.

Для того чтобы посмотреть список доступных параметров с использованием sysctl выполним следующею команду:

```
[root@linuxbox ~]# sysctl -a
```

Пример изменения hostname в режиме реального времени.

```
[root@linuxbox ~]# hostname
linuxbox.company.ru
[root@linuxbox ~]# sysctl -a | grep kernel.hostname
kernel.hostname = linuxbox.company.ru
[root@linuxbox ~]# sysctl -w kernel.hostname="newbox.company.ru"
kernel.hostname = newbox.company.ru
[root@linuxbox ~]# hostname
newbox.company.ru
[root@linuxbox ~]# sysctl -a | grep kernel.hostname
kernel.hostname = newbox.company.ru
[root@linuxbox ~]#
```

Включение передачи пакетов между интерфейсами.

```
[root@linuxbox ~]# sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
[root@linuxbox ~]#
```

Для того чтобы настройки не сбросились после перезагрузки системы их необходимо добавить в файл /etc/sysctl.conf в формате параметр=значение.

Конфигурационные файлы в /etc/

Файл	Описание
/etc/aliases	База псевдонимов для MTA (Mail Transfer Agent) sendmail. После редактирования файла необходимо выполнить команду newaliases
/etc/anacrontab	Конфигурационный файл anacron. anacron позволяет выполнять задания по расписанию. Но в отличие от cron, компьютер не обязательно должен быть включенным 24 часа в сутки
/etc/at.deny	Файл команды at. at позволяет выполнить команду в указанное время. Если существует файл /etc/at.allow то команду at разрешается выполнять только тем, кто присутствует в этом файле. Если файла /etc/at.allow не существует то проверяется наличие файла /etc/at.deny и всем кто в нем перечислен запрещается использовать команду at
/etc/bashrc	Глобальный bashrc, который выполняется каждый раз когда вы открываете новый терминал. Содержит в себе различные настройки командного интерпретатора bash
/etc/blkid/blkid.tab	Кэш-файл утилиты blkid, которая выводит информацию о блочных устройствах
/etc/crontab	Файл с командами и временем их выполнения для демона cron. cron позволяет выполнять задания с определенным интервалом
/etc/cron.deny	Файл демона cron. Если существует файл /etc/cron.allow то команду crontab разрешается использовать только тем, кто присутствует в этом файле. Если файла /etc/cron.allow не существует то проверяется наличие файла /etc/cron.deny и всем кто в нем перечислен запрещается использовать команду crontab
/etc/cron.hourly/	Задания для cron которые должны выполняться каждый час
/etc/cron.daily/	Задания для cron которые должны выполняться ежедневно

/etc/cron.weekly/	Задания для cron которые должны выполняться каждую неделю
/etc/cron.monthly/	Задания для cron которые должны выполняться каждый месяц
/etc/cups/	Конфигурационные файлы принт-сервера CUPS
/etc/exports	Конфигурационный файл NFS, в котором перечисляются файловые системы которые будут экспортированы
/etc/filesystems	Конфигурационный файл для команды mount. В случае если при монтирование файловой системы ее тип не задан явно, будет произведен ее поиск по этому файлу
/etc/fstab	Файл содержит файловые системы которые автоматически монтируются при старте системы
/etc/group	Файл содержит список групп и пользователей, которые в эти группы входят. Формат файла: <имя группы>:<пароль группы>:<GID>:<список пользователей>
/etc/grub.conf	Символическая ссылка на /boot/grub/grub.conf. Это конфигурационный файл загрузчика GRUB
/etc/gshadow	Файл групповых паролей
/etc/host.conf	Файл настроек преобразования имен. В этом файле задается порядок разрешения имени. По умолчанию поиск соответствия имени происходит в файле /etc/hosts затем происходит обращение к DNS-серверу
/etc/hosts	Статическая таблица в формате <IP-адрес> <имя хоста> Используется для разрешения имени хоста
/etc/hosts.allow	Конфигурационный файл tcpd. В файле перечисляются хосты которым разрешено выполнять INET сервисы в локальной системе
/etc/hosts.deny	Конфигурационный файл tcpd. В файле перечисляются хосты которым запрещено выполнять INET сервисы в локальной системе
/etc/inittab	Файл описывает, что процесс INIT должен делать на каждом runlevel
/etc/inputrc	Файл используется в планировании клавиатуры для специфических ситуаций
/etc/issue	Файл используется mingetty при отображение “welcome” строки при подключение пользователя с помощью терминала

/etc/issue.net	Файл используется mingetty при отображение “welcome” строки во время подключения пользователя по telnet
/etc/login.defs	Файл содержит настройки подсистемы теневых (shadow) паролей
/etc/logrotate.conf	Конфигурационный файл logrotate, который занимается ротацией, упаковкой и отправкой по сети журнальный файлов
/etc/logrotate.d/	Настройки logrotate для различных программ
/etc/logwatch/	Конфигурационные файлы logwatch. logwatch – это системный анализатор логов, он сканирует логи и формирует отчет в удобном для пользователя виде
/etc/mail/	Конфигурационные файлы sendmail
/etc/man.config	Конфигурационный файл для команды man
/etc/mime.types	Файл с MIME-типами (Multipurpose Internet Mail Extension, многоцелевые расширения электронной почты для Internet) для принт-сервера CUPS
/etc/modprob.conf	Конфигурационный файл для modprobe, программы которая позволяет добавить/удалить модуль из ядра
/etc.modprobe.d	Каталог с конфигурационными файлами для modprobe
/etc/motd	Файл с сообщением дня. Это сообщение выводится всем кто подключается в систему
/etc/mtab	Файл содержит список файловых систем которые примонтированы в данный момент
/etc/nsswitch.conf	Конфигурационный файл для целой группы библиотечных программ, которые связаны с получением имен хостов и использованию этих имен локально
/etc/ntp.conf	Конфигурационный файл ntpd (Network Time Protocol Daemon)
/etc/pam.conf **/etc/pam.d**	Конфигурационные файлы pam - Pluggable Authentication Modules for Linux
/etc/passwd	База данных пользователей системы
/etc/printcap	Конфигурационный файл для принтера. Используется принт-сервером CUPS
/etc/profile	Файл запускается во время логина в систему. Позволяет задать глобальные настройки по умолчанию для всех пользователей системы

/etc/protocols	Таблица соответствий имени протокола и его номера
/etc/rc.d/	Наборы скриптов которые выполняются при старте системы на различных уровнях выполнения
/etc/redhat-release	В это файле можно посмотреть версию ОС Linux
/etc/resolv.conf	В файле задаются DNS-сервера которые отвечают за разрешение имен
/etc/securetty	Файл содержит список терминалов с которых пользователь root может подключаться к системе
/etc/services	Файл переводит имя сервиса в номер порта и протокол
/etc/shadow	Файл с паролями пользователей в зашифрованном виде
/etc/shells	Список доступных командных интерпретаторов (shells)
/etc/skel/	Набор файлов и каталогов которые будут скопированы новому пользователю в домашний каталог
/etc/ssh/	Ключи для работы SSH и конфигурационные файлы
/etc/sudoers	Список пользователей с описанием их возможностей. Файл позволяет делегировать обычному пользователю возможность исполнять некоторые привилегированные команды. Редактирование файла происходит с помощью команды visudo
/etc/syslog/	Каталог содержит различные конфигурационные файлы
/etc/sysctl.conf	Файл с опциями ядра
/etc/syslogd.conf	Конфигурационный файл syslogd
/etc/termcap	База данных всех возможных типов терминалов и их возможностей
/etc/vimrc	Глобальный конфигурационный файл для текстового редактора vim
/etc/wgetrc	Конфигурационный файл для программы wget
/etc/X11/	Конфигурационные файлы для X Window
/etc/xinetd.d/	Конфигурационные файлы программ для супердемона xinetd
/etc/yum.conf	Конфигурационный файл yum
/etc/yum.repos.d/	Каталог с репозиториями для программы yum. Файл должен иметь расширение .repo

Индивидуальные конфигурационные файлы

Команда	Описание
~/.bash_logout	Сценарий выполняется при выходе из системы
~/.bash_history	Файл с историей команд, которые были введены из под этой учетной записи
~/.bash_profile	Сценарий выполняется при вашем подключение к системе
~/.bashrc	Этот сценарий выполняется каждый раз когда открывается новый shell
~/.emacs	Читается редактором emacs при запуске
~/.forward	Если файл содержит e-mail адрес то вся почта к этому пользователю будет перенаправляться на него
~/.pinerc	Конфигурационный файл для почтового клиента pine
~/.muttrc	Конфигурационный файл для почтового клиента mutt
~/.vimrc	Конфигурационный файл для текстового редактора vim
~/.netrc	Логины и пароли для программы ftp
~/.rhosts	Используется набором команд r: rlogin, rsh и т.д.
~/mbox	Файл где сохраняется старая почта
~/.ssh/	Файлы ssh

Команды для управления файловой системой

Команда	Описание
badblocks	Поиск плохих блоков на диске или разделе
debugfs	Позволяет получить прямой доступ к структуре файловой системы
df	Статистика по свободному месту для одной или нескольких файловых системах
du	Сколько места занимает каталог и его содержимое
dump	Используется для резервного копирования ext2/ext3 файловых систем или файлов. Используется в связке с командой restore
dumpe2fs	Выводит информацию о файловой системе
e2fsck	Проверка ext2/ext3 файловой системы
e2label	Поменять метку для ext2/ext3 файловой системы
exportfs	Создание списка экспортируемых файловых систем
fdisk	Управление разделами на жестком диске

fsck	Проверка и восстановление файловой системы. Не должна запускаться для примонтированой файловой системы
hdparm	Получить/установить параметры для жесткого диска
mkfs	Инициализация файловой системы
mkfs.ext3	Инициализация файловой системы ext3
mkswap	Создание swap-файла
mount	Монтирование файловой системы
restore	Восстановление файлов/файловых систем из резервных копий сделанных программой dump
swapon	Включение swap-раздела
swapoff	Отключение swap-раздела
sync	Запись информации из буферов системы на диск
tune2fs	Настройка ext2/ext3 файловой системы
umount	Отмонтирование файловой системы

Команды для управления файлами и каталогами

Команда	Описание
cd	Сменить текущий каталог
chmod	Сменить права доступа на файл или каталог
chown	Сменить владельца файла
chgrp	Сменить группу для файла или каталога
cksum	Посчитать контрольную сумму для файла и количество байт
md5sum	Посчитать md5 контрольную сумму для файла
cp	Скопировать файл или каталог
dd	Преобразование и копирование файла
dir	Вывести содержимое каталога
dircolors	Настройка цветов в выводе команды ls
file	Определение типа файла
find	Поиск файлов
install	Копирование файлов и установка атрибутов
ln	Создание линков между файлами
locate	Поиск программы по базе mlocate
ls	Вывести содержимое каталога
mkdir	Создание каталога
mknod	Создание блочного или символьного файла
mktemp	Создание временного файла
mv	Перемещение файла или каталога
pwd	Вывод текущего каталога

rm	Удаление файла или каталога
rmdir	Удаление каталога
stat	Вывод информации по файлу
test	Проверка статуса файла и сравнение значений
touch	Изменить штамп времени на файле. Создать пустой файл если его не существует
vdir	Вывод содержимого каталога
whatis	Поиск слова по базе whatis
whereis	Поиск бинарного файла, исходных кодов и man-страниц для указанной команды
which	Показать полный путь к команде

Просмотр и редактирование файлов

Команда	Описание
ed или red	Текстовый редактор
emacs	Текстовый редактор
head	Просмотреть первые N строк файла. По умолчанию 10
joe	Текстовый редактор
less	Просмотр файла
more	Просмотр файла
nano	Текстовый редактор
tail	Просмотреть последние N строк файла. По умолчанию 10
vi	Текстовый редактор
vim	Усовершенствованный текстовый редактор vi

Сжатие файлов, создание архивов и извлечение из них

Команда	Описание
ar	Создание, модификация и извлечение из архивов
bunzip2	Распаковка архива в формате bz
bzcat	Распаковка архива на стандартный вывод
bzip2	Упаковка файлов в архив в формате bz
bzip2recover	Восстановление файлов из поврежденных bz-архивов
cpio	Копирование файлов в архив/из архива
dump	Используется для резервного копирования ext2/ext3 файловых систем или файлов. Используется в связке с командой restore
gunzip	Разархивирование архива в формате gz

gzexe	Архивирование исполняемых файлов. Они автоматически распаковываются когда вы их запускаете. Используется в системах с маленьким дисковым пространством
gzip	Создание архива файлов в формат gz
tar	Утилита архивирования в формат tar
unzip	Разархивирование архива в формате zip
zcat	Разархивирование архива на стандартный вывод
zcmp	Сравнение архивов между собой
zdiff	Сравнение архивов между собой
zgrep	Ищет выражение в сжатых файлах
zmore	Разархивирование архива на стандартный вывод
znew	Конвертирование Z архивов в формат gz
zip	Создание zip-архивов

Манипуляции с текстом

Команда	Описание
basename	Убирает имена каталогов и суффикс из имени файла. Например результатом выполнения basename /usr/bin/find будет find
cat	Объединение файлов и печать их на стандартный вывод
cmp	Сравнение двух файлов
colrm	Удаление колонок из файла
column	Разбивает вывод по колонкам
comm	Сравнение двух файлов строка за строкой
csplit	Разбиение файла на части по контексту
cut	Удаление секций с каждой строки файла
diff	Показывает различия между двумя файлами
diff3	Показывает различия между тремя файлами
dirname	Преобразование абсолютного или относительного пути к файлу или каталогу в имя родительского каталога
echo	Отображение строки текста
egrep	Соответствует grep -E
expand	Преобразовывает отступы табулятора в пробелы
expr	Универсальный обработчик арифметических, логических или строковых выражений
false	Нечего не делает, выход с отрицательным статусом
fgrep	Соответствует grep -F
fold	Перенос строк по заданной длине

join	Объединяет строки двух файлов в общее поле
grep	Используется для поиска конкретных текстовых строк в файле
logname	Выводит имя текущего пользователя
look	Команда для поиска по отсортированным файлам
mkfifo	Создание именованного канала
nl	Печать файла на стандартный вывод с указанием нумерации строк
od	Просмотр двоичного файла
patch	Применить различия в файле к оригиналу
paste	Объединить строки файлов
printf	Печать и форматирование данных
rev	Печать на стандартный вывод файла с символами на строках в обратном порядке
script	Создание машинописного текста сессии терминала
sdiff	Поиск различий между двумя файлами и слияние в интерактивном режиме
sed	Потоковый редактор. Используется для трансформации входного потока
sleep	Создание искусственной задержки во времени
sort	Сортировать строки текстового файла
split	Команда копирует файл, разбивая его на отдельные файлы заданной длины
strings	Выводит на экран содержимое файла, игнорируя все нечитаемые символы
tac	Объединяет и печатает файлы в обратном порядке
tee	Команда читает со стандартного ввода и печатает в стандартный вывод или файл
tr	Преобразование символов
true	Нечего не делает. Выход с положительным статусом
tsort	Команда выполняет топологическую сортировку
ul	Сделать подчеркивание
unexpand	Преобразовывает отступы табулятора в пробелы
uniq	Удаление повторяющихся строк из отсортированного файла
uudecode	Кодирует/декодирует бинарный файл
uuencode	Кодирует/декодирует бинарный файл
wc	Подсчет строк, слов и символов в файле
xargs	Формирование списка аргументов и выполнение команды
yes	Команда выводит на экран строку “y” пока не будет остановлена

Вывод справки

Команда	Описание
apropos	Поиск по базе whatis указанной строки
help	Справка по builtin-командам bash
info	Команда info позволяет просматривать справочные страницы в гипертекстовом формате info
locate	Команда locate производит поиск по одной или нескольким базам созданным с помощью updatedb и выводит соответствия
man	Получить справку по команде
manpath	Определение путей по которым происходит поиск справки для команды
which	Показывает полный путь к указанной команде

Управление заданиями

Команда	Описание
at	Команда похожа на cron но запускается только один раз
atq	Просмотр списка заданий команды at ожидающих выполнения
atrm	Удаление заданий at
atrun	Выполнение следующего по списку задания
batch	Команда представляет системе задание которое ставится в очередь и запускается как только у системы появляется возможность его выполнить
crontab	Команда используется для планирования регулярных заданий
nice / renice	Изменение приоритета выполнения процесса
nohup	Запуск программы с иммунитетом к hangups
watch	Выполняет указанную команду через определенные интервалы времени

Управление процессами

Команда	Описание
bg	Позволяет перевести задание в фоновый режим
fg	Позволяет перевести задание в активный режим
jobs	Список выполняющихся заданий

kill	Уничтожение процесса или передача определенного сигнала
killall	Уничтожение процесса по имени или передача определенного сигнала
pidof	По имени программы узнать её ID
ps	Вывести список выполняющихся процессов
pstree	Вывести дерево процессов
top	Просмотр текущих процессов в реальном времени
CTRL+C	Завершение текущего задания
&	Запуск команды в фоновом режиме

Управление сетевой подсистемой

Команда	Описание
arp	Позволяет посмотреть/модифицировать arp cache
dig	Позволяет отправлять запросы к DNS-серверам
finger	Выводит информацию о системных пользователях
ftp	Программа для передачи файлов по протоколу FTP
ifconfig	Конфигурирование сетевого интерфейса
ifdown	Отключение сетевого интерфейса
ifup	Включение сетевого интерфейса
netstat	Вывод информации по портам, маршрутизации, сетевым интерфейсам
nslookup	Позволяет делать запросы к DNS-серверу
ping	Отправка ICMP ECHO_REQUEST к хостам в сети
portmap	Связывает DARPA порт с номером RPC программы. Необходимо для RPC calls. Используется в работе NFS
rcp	Копирование файлов между компьютерами
rlogin	Подключение к удаленной системе
route	Просмотр и модификация таблицы маршрутизации
rsh	Удаленное выполнение команд
showmount	Информация по статусу NFS-сервера на удаленном компьютере
tcpd	Демон может обслуживать входящие запросы для telnet, finger, ftp, exec, rsh, rlogin, tftp, talk и т.д.
tcpdump	Популярный анализатор трафика
telnet	Подключение к удаленному компьютеру по telnet
traceroute	Получение маршрута до удаленного компьютера

Переменные окружения

Команда	Описание
env	Показать все переменные окружения
export	Экспортировать переменную в окружение
printenv	Показать все переменные окружения
reset	Сброс runtime параметров сессии на значения по умолчанию
set	Показать все переменные окружения включая пользовательские
unset	Уничтожение значения переменной

Управление библиотеками

Команда	Описание
ldconfig	Настройка динамического связывания во время выполнения
ldd	Выдаёт список зависимостей от динамических библиотек
ltrace	Перехватывает и выводит все выполняемые процессом вызовы подпрограмм из динамических библиотек, все системные вызовы и все получаемые сигналы

Управление модулями и ядром

Команда	Описание
depmod	Создание файла зависимостей, который команда modprobe использует для загрузки набора модулей
dmesg	Используется для просмотра и настройки уровня выходных сообщений, которые появляются в процессе начальной загрузки ядра Linux и хранятся в кольцевом буфере
insmod	Установка модуля в работающее ядро
lsmod	Вывод списка всех модулей, установленных в ядре
modprobe	Загрузка набора модулей, определенного командой depmod
modinfo	Вывод информации о модуле
rmmod	Удаление модуля из ядра
sysctl	Конфигурирование опций ядра в реальном времени

Управление уровнями выполнения

Команда	Описание
exit	Выход из командного интепретатора
halt	Остановить систему
init	Контроль процесса инициализации
logout	Выход из системы
poweroff	Выключение системы
reboot	Перезагрузка системы
runlevel	Вывод текущего и предыдущего уровня выполнения
setsid	Запуск программы в новой сессии
shutdown	Выключение системы с возможностью оповещения пользователей
telinit	Изменение уровня выполнения

Управление системой

Команда	Описание
mesg	Управление возможностью записи на терминал другими пользователями
quota	Показать использование диска и лимиты
quotacheck	Сканирование файловой системы и сбор статистики по использованию дискового пространства
quotaoff	Отключение квот
quotaon	Включение квот
setpci	Конфигурирование PCI устройств
setserial	Конфигурирование последовательного порта
setterm	Настройка атрибутов терминала
setup	Конфигурирование базовых системных настроек
stty	Конфигурирование параметров терминала
swapon	Включение возможности свопинга для файлов и устройств
swapoff	Выключение возможности свопинга для файлов и устройств
tset	Инициализация терминала

Информация о системе

Команда	Описание
arch	Вывод информации об архитектуре компьютера
df	Информация о свободном дисковом пространстве

du	Информация о том, сколько дискового пространства занимает конкретный каталог
free	Информация о свободной и занятой памяти в системе
ipcrm	Удаление идентификаторов средств межпроцессной связи
ipcs	Выдает информацию о состоянии средств межпроцессной связи
lsof	Вывод списка открытых файлов
lspci	Вывод списка PCI устройств
procinfo	Системный статус взятый из /proc
pstree	Вывод дерева процессов
runlevel	Вывод текущего и предыдущего уровня выполнения
strace	Трассировка вызовов и сигналов для бинарных программ
tload	Вывод информации о средней нагрузке на систему
tty	Вывод имени файла терминала подключенного к стандартному вводу
uname	Вывод информации о системе (версия ядра, архитектура, тип ОС,…)
vmstat	Статистика по виртуальной памяти

Системное время

Команда	Описание
cal	Календарь
date	Выводит текущею дату и время
hwclock	Выводит аппаратное время (CMOS)
tzselect	Выбор временной зоны (time zone)
uptime	Как долго система работает (её uptime)

Управление пользователями

Команда	Описание
ac	Информация о продолжительности подключения пользователей к системе
accton	Включение системы учета процессов (запись всех команд, выполняемых в системе и подведение суммарных итогов)
adduser	Добавление нового пользователя
chage	Изменение информации по устареванию пароля пользователя

chfn	Изменение информация для finger. Она хранится в /etc/passwd и выводится при выполнение команды finger
chgrp	Изменение группы владельца для файла
chown	Изменение владельца файла
chpasswd	Групповое изменение паролей для пользователей
chroot	Изменение корневого каталога
chsh	Изменение командного интерпретатора во время подключения к системе
edquota	Редактирование пользовательских или групповых квот
faillog	Статистика по неудачным подключениям в систему и изменение лимита
finger	Информация по подключенным пользователям
gpasswd	Администрирование /etc/group
groupadd	Добавление новой группы
grpck	Проверка целостности групповых файлов
grpconv	Создание файла /etc/gshadow из /etc/group с конвертацией открытых паролей в shadow
grpunconv	Создаёт файл group из файлов group и gshadow, а затем удаляет файл gshadow
groupdel	Удаление группы
groupmod	Модификация группы
groups	Выводит список групп в которых находится пользователь
id	Выводит подлинные и действующие UID и GID
last	Список пользователей подключающихся к системе в последнее время
lastb	Тоже самое что и last, только по умолчанию выводит статистику из файла /var/log/btmp который содержит неудачные попытки подключения в систему
lastcomm	Показывает последние выполненные команды в обратном порядке. Работает если включен process accounting
lastlog	Информация по последнему подключению к системе каждого пользователя
logname	Вывод имени под которым подключен пользователь
newgrp	Переход в новую группу
newusers	Создание группы пользователей
passwd	Изменение пароля пользователя
pwck	Проверка целостности файлов паролей

pwconv	Команда создаёт файл shadow из файла passwd и необязательно существующего файла shadow
quota	Информация по занятому месту и лимите пользователя
quotaoff	Отключение квот
quotaon	Включение квот
quotacheck	Сканирование файловой системы и сбор статистики по использованию дискового пространства
repquota	Итоговая информация по использованию квот в файловой системе
su	Переключиться на другого пользователя
sulogin	Команда вызывается процессом init, когда система переходит в однопользовательский режим
ulimit	Управление доступными ресурсами для командного интерпретатора
useradd	Добавление нового пользователя
userdel	Удаление пользователя
usermod	Модификация аккаунта пользователя
users	Выводит список пользователей подключенных в систему
utmpdump	Используется для отладки
vigr	Редактировать файл групп
vipw	Редактировать файл паролей
w	Список подключенных пользователей и что они делают
wall	Отправить сообщение на все терминалы
write	Отправить сообщение другому пользователю
who	Список подключенных пользователей
whoami	Выводит эффективный ID пользователя

Печать файлов

Команда	Описание
lpr	Печать файлов
lpoptions	Просмотр и настройка опций принтера
lpc	Ограниченное управление принтером
lpq	Просмотр очереди печати
lprm	Удаление задания из очереди
pr	Подготавливает текст для печати
tunelp	Настройка опций для lp устройства

Запись дисков

Команда	Описание
cdrecord	Запись CD/DVD дисков из командной строки
cdrwtool	Разные манипуляции с CD-R, CD-RW и DVD-R дисками
mkisofs	Создание .iso файла из файлов и каталогов

Разное

Команда	Описание
alias	Позволяет за определенной последовательностью символов закрепить конкретную команду
history	Просмотр истории команд
mc	Файловый менеджер основанный на псевдографике
nc	Команда может открыть порт и выводить информацию из него на экран
sleep	Создать задержку на экране. Например sleep 2 создаст паузу в выполнение на 2 секунды
screen	Оконный менеджер с эмуляцией терминала VT100/ANSI
unalias	Удалить alias созданный командой alias