**Основы линукс: Введение в bash**

**О чем статья**

Прочитав эту статью, вы узнаете что такое **bash** (стандартный командный интерпретатор линукс), научитесь обращаться со стандартными командами: **ls**, **cp**, **mv**… поймете назначение инодов, жестких и символических ссылок и многое другое.

Это пособие предназначено для новичков в линукс и для тех, кто хочет повторить или усовершенствовать понимание основных принципов линукс, таких как копирование и перемещение файлов, создание ссылок, использование стандартных команд линукс наряду с перенаправлениями и пайпами. В этой статье вы найдете множество примеров, поясняющих изложенный материал. Для начинающих большинство информации окажется новой, а для более продвинутых пользователей этот материал может стать прекрасным пособием для обобщения имеющихся знаний и навыков.

**Введение в bash**

**Shell**

Если вы используете линукс, то знаете что после логина вас приветствует приглашение командного интерпретатора. Например такое:

\$

Если после логина загружается графическая оболочка, то чтобы добраться до командного интерпретатора нужно запустить эмулятор терминала (gnome-terminal, xfce4-terminal, konsole, xterm, rxvt…) или переключиться на один из виртуальных терминалов нажав **Ctrl**–**Alt**–**F1** или **Ctrl**–**Alt**–**F2** и т.д.

Приглашение командного интерпретатора на вашем компьютере может отличаться от того что показано в примере. Оно может содержать имя пользователя, имя компьютера и название текущей рабочей директории. Но несмотря на все эти различия, программа которая печатает это приглашение называется «**shell**» (оболочка), и скорее всего в роли вашей командной оболочки выступает программа которая называется **bash**.

**У вас запущен bash?**

Проверить запущен ли bash можно следующей командой:

\$ **echo \$SHELL** /bin/bash

Если в результате выполнения этой команды вы получили ошибку или её вывод отличается от того что в примере, то возможно в вашей системе в качестве командной оболочки используется не bash. Несмотря на это, большая часть материала будет актуальна, но все же рекомендуем вам переключиться на bash. Сделать это можно (если bash установлен в системе) командой:

\$ **bash**

**Что такое bash**

Bash (акроним от «**B**ourne-**a**gain **SH**ell») это стандартный интерпретатор команд на большинстве линукс систем. В его обязанности входит обработка и исполнение команд с помощью которых пользователь управляет компьютером. После того как вы завершили работу, можно завершить процесс командного интерпретатора. После нажатия клавиш **Ctrl**–**D**, команд **exit** или **logout** процесс командного интерпретатора будет завершен и на экране снова появится приглашение ввести имя пользователя и пароль.

**Использование «cd»**

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

$ **cd /**

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

**Пути**

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

\$ **pwd** /

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

**Абсолютные пути**

Вот несколько примеров абсолютных путей

cd c:/Users/Дмитрий/Desktop/repo

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

Забыли написать «c:»

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

**Относительные пути**

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

\$ **cd /usr**

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

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

**Использование «..»**

Относительные пути могут содержать одну или несколько директорий **«..»**. «..» указывает на родительскую директорию по отношению к нашей рабочей директории. Пример:

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

Как вы видите, команда *cd ..* ‘поднимает нас на уровень выше’.

Можно добавить **..** к относительному пути. Это позволит переместиться в директорию, которая находится на одном уровне с той в которой мы находимся. Пример:

\$ **pwd** /usr/local \$ **cd ../share** \$ **pwd** /usr/share

**Примеры с использованием относительных путей**

Относительные пути могут быть довольно сложными. Вот несколько примеров. Результат выполнения команд не показан, попробуйте определить в какой директории вы окажетесь используя bash.

\$ **cd /bin** \$ **cd ../usr/share/zoneinfo** \$ **cd /usr/X11R6/bin** \$ **cd ../lib/X11** \$ **cd /usr/bin** \$ **cd ../bin/../bin**

**Рабочая директория «.»**

Перед тем как закончить разговор о команде **cd**, следует упомянуть еще несколько вещей. Во-первых, существует ещё одна специальная директория **«.»**, которая указывает на текущую директорию. Эта директория используется для запуска исполняемых файлов, находящихся в текущей директории.

\$ **./myprog**

В последнем примере myprog это исполняемый файл находящийся в текущей директории, который будет запущен на исполнение.

**cd и домашняя директория пользователя**

Для того чтобы перейти в домашнюю директорию, нужно набрать

\$ **cd**

Без аргумента cd переместит вас в домашнюю директорию. Для суперпользователя домашней обычно является директория /root, а для обычных пользователей — /home/username/. Но что если мы хотим указать конкретный файл, находящийся в домашней директории. Например, как аргумент к программе **‘myprog’**? Можно написать:

\$ **./myprog /home/user/myfile.txt**

Однако, использовать абсолютные пути к файлам не всегда удобно. Эту же операцию можно сделать при помощи **~**–тильды:

\$ **./myprog ~/myfile.txt**

**~** — специальное имя, указывающее в bash на домашнюю директорию пользователя.

**Домашние директории других пользователей**

Но что если нам нужно указать файл в домашней директории другого пользователя? Для этого после тильды нужно указать имя этого пользователя. Например, чтобы указать на файл fredsfile.txt находящийся в домашней директории пользователя fred:

\$ **./myprog ~fred/fredsfile.txt**

**Команды линукс**

**Введение в ls**

Вероятно вы уже знакомы с командой **ls**, которая, вызванная без аргументов, выводит на экран список файлов хранящихся в рабочей директории:

\$ **cd /usr** \$ **ls** X11R6 doc i686-pc-linux-gnu lib man sbin ssl bin gentoo-x86 include libexec portage share tmp distfiles i686-linux info local portage.old src

Если указать опцию **-a**, можно будет увидеть все файлы, включая скрытые (имена которых начинаются с точки).

\$ **ls -a** . bin gentoo-x86 include libexec portage share tmp .. distfiles i686-linux info local portage.old src X11R6 doc i686-pc-linux-gnu lib man sbin ssl

**Подробный список директорий**

После самой команды **ls** в качестве ее аргумента можно указать один или более файлов или директорий. Если указать имя файла, то команда **ls** выведет информацию только об этом файле. А если указать название директории, **ls** покажет все ее содержимое. Опция **‘-l’** команды **ls** бывает очень полезной если вы хотите кроме имен файлов узнать более подробную информацию о них (права на файл, имя владельца, время последнего изменения файла и его размер).

В следующем примере показано применение опции **‘-l’** для вывода информации о файлах хранящихся в директории /usr

\$ **ls -l /usr** drwxr-xr-x 7 root root 168 Nov 24 14:02 X11R6 drwxr-xr-x 2 root root 14576 Dec 27 08:56 bin drwxr-xr-x 2 root root 8856 Dec 26 12:47 distfiles lrwxrwxrwx 1 root root 9 Dec 22 20:57 doc -> share/doc drwxr-xr-x 62 root root 1856 Dec 27 15:54 gentoo-x86 drwxr-xr-x 4 root root 152 Dec 12 23:10 i686-linux drwxr-xr-x 4 root root 96 Nov 24 13:17 i686-pc-linux-gnu drwxr-xr-x 54 root root 5992 Dec 24 22:30 include lrwxrwxrwx 1 root root 10 Dec 22 20:57 info -> share/info drwxr-xr-x 28 root root 13552 Dec 26 00:31 lib drwxr-xr-x 3 root root 72 Nov 25 00:34 libexec drwxr-xr-x 8 root root 240 Dec 22 20:57 local lrwxrwxrwx 1 root root 9 Dec 22 20:57 man -> share/man lrwxrwxrwx 1 root root 11 Dec 8 07:59 portage -> gentoo-x86/ drwxr-xr-x 60 root root 1864 Dec 8 07:55 portage.old drwxr-xr-x 3 root root 3096 Dec 22 20:57 sbin drwxr-xr-x 46 root root 1144 Dec 24 15:32 share drwxr-xr-x 8 root root 328 Dec 26 00:07 src drwxr-xr-x 6 root root 176 Nov 24 14:25 ssl lrwxrwxrwx 1 root root 10 Dec 22 20:57 tmp -> ../var/tmp

В первой колонке показана информация о правах доступа к каждому файлу в списке. (Немного позже я объясню какая буква что обозначает) Следующая колонка показывает количество ссылок на каждый элемент списка. Третья и четвертая колонки — владелец и группа файла соответственно. Пятая колонка — размер. Шестая — время последнего изменения файла (‘last modified time’ или mtime). Последняя колонка — имя файла или директории (Если это ссылка, то после знака ‘**–>**‘ стоит имя объекта на который она ссылается).

**Как посмотреть только директории**

Иногда возникает потребность посмотреть информацию только о директориях, а не о всем их содержимом. С этой задачей поможет справиться опция **‘-d’**, которая указывает команде выводить информацию только о директориях. Пример:

\$ **ls -dl /usr /usr/bin /usr/X11R6/bin ../share** drwxr-xr-x 4 root root 96 Dec 18 18:17 ../share drwxr-xr-x 17 root root 576 Dec 24 09:03 /usr drwxr-xr-x 2 root root 3192 Dec 26 12:52 /usr/X11R6/bin drwxr-xr-x 2 root root 14576 Dec 27 08:56 /usr/bin

**Рекурсивный список и информация о инодах**

Действие опции **‘-R’** противоположно действию **‘-d’**. Она позволяет выводить информацию о файлах находящихся в директории рекурсивно. Сначала показывается содержимое директории верхнего уровня, потом по очереди содержимое всех поддиректорий и так далее. Вывод этой команды может быть достаточно объемным, поэтому мы не приводим ее пример, но вы можете попробовать сделать это самостоятельно, набрав в командной строке ‘**ls -R**‘ или ‘**ls -Rl**‘.

И, наконец, опция **‘-i’** используется для вывода инодов каждого объекта файловой системы.

\$ **ls -i /usr** 1409 X11R6 314258 i686-linux 43090 libexec 13394 sbin 1417 bin 1513 i686-pc-linux-gnu 5120 local 13408 share 8316 distfiles 1517 include 776 man 23779 src 43 doc 1386 info 93892 portage 36737 ssl 70744 gentoo-x86 1585 lib 5132 portage.old 784 tmp

**Что такое иноды?**

Каждый объект файловой системы (файл, директория…) имеет свой уникальный номер, называемый *инодом* (inode number). Эта информация может показаться незначительной, но понимание функции инодов поможет вам разобраться во многих операциях над файловой системой. Например, посмотрим на **«.»** и **«..»** как на ссылки, присутствующие в каждой директории. Чтобы понять что из себя представляет директория **«..»**, узнаем инод директории /use/local

\$ **ls -id /usr/local** 5120 /usr/local

Как можем видеть, инод директории /usr/local — 5120. Теперь посмотрим какой инод у директории /usr/local/bin/..:

\$ **ls -id /usr/local/bin/..** 5120 /usr/local/bin/..

Получается, что иноды директорий /usr/local и /usr/local/bin/.. совпадают! Это значит, что на инод 5120 ссылаются два имени: /usr/local и /usr/local/bin/.. То есть это два разных имени одной директории. Каждый инод указывает на определенное место на диске.

С каждым инодом может быть связано несколько имен объектов файловой системы. Количество ‘синонимов’ файла (объектов файловой системы, ссылающихся на один инод) показывает число во втором столбце вывода команды ‘**ls -l**‘.

\$ **ls -dl /usr/local** drwxr-xr-x 8 root root 240 Dec 22 20:57 /usr/local

На этом примере видно (второй столбец), что на директорию /usr/local ссылаются 8 разных объектов файловой системы. Вот их имена:

/usr/local /usr/local/. /usr/local/bin/.. /usr/local/games/.. /usr/local/lib/.. /usr/local/sbin/.. /usr/local/share/.. /usr/local/src/..

**mkdir**

Давайте рассмотрим команду **mkdir**. Она служит для создания новых директорий. В следующем примере демонстрируется создание трех новых директорий (tic, tac, toe) в директории /tmp

\$ **cd /tmp** $ **mkdir tic tac toe**

По умолчанию команда **mkdir** не может создать вложенной структуры директорий. Поэтому, если вам нужно создать несколько вложенных одна в другую директорий (**won/der/ful**), то вам придется три раза поочередно вызывать эту команду:

\$ **mkdir won/der/ful** mkdir: cannot create directory 'won/der/ful': No such file or directory \$ **mkdir won** \$ **mkdir won/der** \$ **mkdir won/der/ful**

Упростить эту операцию можно добавив опцию **‘-p’** к команде mkdir. Эта опция позволяет создавать вложенную структуру директорий:

\$ **mkdir -p easy/as/pie**

Чтобы узнать о возможностях этой утилиты подробнее, прочитайте справку, которая вызывается командой **man mkdir**. Справки есть практически ко всем командам из этого руководства (например **man ls**), кроме **cd**, т.к. она встроена в bash (для таких команд справка вызывается так: **help cd**)

**touch**

Перейдем к изучению команд **cp** и **mv**, служащих для копирования, переименования и перемещения файлов и директорий. Но перед этим создадим пустой файл в директории /tmp при помощи команды **touch**:

\$ **cd /tmp** \$ **touch copyme**

Команда **touch** обновляет время последнего доступа к файлу (шестая колонка вывода команды **ls -l**) если он уже существует или создает новый пустой файл, если его ещё нету. После этой операции у нас должен появиться пустой файл **/tmp/copyme**.

**echo**

Теперь, когда у нас есть пустой файл, запишем в него текстовую строку при помощи команды **echo**, которая выводит переданный ей аргумент на стандартное устройство вывода (текстовый терминал в нашем случае).

\$ **echo "firstfile"** firstfile

Чтобы записать строку в наш файл, перенаправим в него вывод команды **echo**:

\$ **echo "firstfile" > copyme**

Знак **>** (больше) указывает командной оболочке что нужно перенаправить вывод команды стоящей слева в файл, имя которого находится справа. Если файла с таким именем не существует, он будет создан автоматически. А если такой файл уже есть, то он будет перезаписан (все его содержимое будет стерто перед записью нашей строки). Команда **‘ls -l’** покажет что размер нашего файла теперь равен 10 байтам — девять байт занимает слово ‘firstfile’ и один байт символ перевода строки.

\$ **ls -l copyme** -rw-r--r-- 1 root root 10 Dec 28 14:13 copyme

**cat и cp**

Для вывода содержимого файла на терминал используется команда **cat**:

\$ **cat copyme** firstfile

Теперь мы можем приступить к разбору базовой функциональности команды **cp**. Эта команда принимает два аргумента. Первый — имя уже существующего файла (‘copyme’), второй — название новой копии, которую мы хотим сделать (‘copiedme’).

\$ **cp copyme copiedme**

Можем убедиться, что новая копия файла имеет другой номер инода (это значит что мы получили действительно новый отдельный файл, а не просто ссылку на старый)

\$ **ls -i copyme copiedme** 648284 copiedme 650704 copyme

**mv**

Теперь применим команду **mv** чтобы переименовать файл («copiedme» –> «movedme»). Номер инода после этой операции не меняется, а изменяется только название файла.

\$ **mv copiedme movedme** \$ **ls -i movedme** 648284 movedme

Номер инода не изменяется только при условии, что переименованный файл остается в пределах той файловой системы где находился исходный файл. Мы рассмотрим подробнее устройство файловых систем в одной из следующих частей этого пособия.

Команда **mv** позволяет не только переименовывать файлы, но и перемещать их. Например, чтобы переместить файл **/var/tmp/myfile.txt** в директорию **/home/user** нужно дать команду:

\$ **mv /var/tmp/myfile.txt /home/user**

Файл будет перемещен в домашнюю директорию пользователя **user** даже если она находится в другой файловой системе (в этом случае файл будет скопирован в новое место после чего оригинал будет удален). Как вы могли уже догадаться, перемещение файла в другую файловую систему приводит к изменению его инода. Это происходит потому, что каждая файловая система имеет свой отдельный набор инодов.

Нужно заметить, существует вероятность, что новый присвоенный номер инода может совпасть со старым, но она чрезвычайно мала.

Чтобы переместить одновременно несколько файлов в одну директорию нужно написать:

\$ **mv /var/tmp/myfile1.txt /var/tmp/myfile2.txt /home/user**

или

\$ **mv -t /home/user /var/tmp/myfile1.txt /var/tmp/myfile2.txt**

Если добавить опцию **‘-v’**, на экран будет выведен отчет о проделанной операции:

\$ **mv -vt /home/user /var/tmp/myfile1.txt /var/tmp/myfile2.txt** '/var/tmp/myfile1.txt' -> '/home/user/myfile1.txt' '/var/tmp/myfile2.txt' -> '/home/user/myfile2.txt'

**ПОПУЛЯРНЫЕ СТАТЬИ**

* [Firefox & samba](https://linuxgeeks.ru/smb.htm)
* [Ручная сборка ядра linux](https://linuxgeeks.ru/kernel.htm)
* [Создание Live-usb или Как сделать загрузочную флэшку с линукс](https://linuxgeeks.ru/live-usb.htm)
* [Использование текстового редактора Vim](https://linuxgeeks.ru/vi.htm)
* [Упражнения по работе с командной строкой linux](https://linuxgeeks.ru/test-cmd.htm)
* [Один из вариантов набора программного обеспечения](https://linuxgeeks.ru/my-software-choice.htm)
* [Сравнение бесплатных веб-браузеров под линукс](https://linuxgeeks.ru/web-browser.htm)
* [Bash в примерах. Часть вторая.](https://linuxgeeks.ru/bash-2.htm)

© 2009-2014 [LinuxGeeks.ru](https://linuxgeeks.ru/) -- [Карта сайта](https://linuxgeeks.ru/sitemap)