Конфигурирование SHELL

Одним из ключевых компонентов оболочки bash являются **переменные** оболочки. Они хранят важную системную информацию и определяют поведение оболочки bash, а также многих команд. В данном лекции рассмотрено, что такое переменные оболочки и как их можно использовать для настройки оболочки.

Вы узнаете, как переменная PATH влияет на выполнение команд и как другие переменные влияют на вашу способность использовать историю ваших команд. Кроме того, вы также увидите, как использовать файлы инициализации, чтобы сделать переменные оболочки постоянными, поэтому они будут создаваться при каждом входе в систему.

Переменные

Переменная - это имя или идентификатор, которому можно присвоить значение. Оболочка (и другие команды) считывает значения этих переменных, что может привести к изменению поведения в зависимости от содержимого (значения) переменной.

Переменные обычно содержат одно значение, например 0 или bob. Некоторые могут содержать несколько значений, разделенных пробелами, такими как Джо Браун или другими символами, такими как двоеточия. /usr/bin:/usr/sbin:/bin:/usr/bin:/home/joe/bin

Вы присваиваете значение переменной, набирая имя переменной сразу после знака равенства = и затем значение. Например: `name="Bob Smith"`..

Имена переменных должны начинаться с буквы (буквенный символ) или подчеркивания и содержать только буквы, цифры и символ подчеркивания. Важно помнить, что имена переменных чувствительны к регистру; **A** и **a** - разные переменные. Как и в случае аргументов команд, одинарные или двойные кавычки должны использоваться, когда в значение, назначенное переменной, включены специальные символы, чтобы предотвратить расширение оболочки.

Допустимые назначения переменных	Недопустимые назначения переменных
`a=1`	`1=a`
`_1=a`	`a-1=3`
`LONG_VARIABLE='OK'`	`LONG-VARIABLE='WRONG'`
`Name='Jose Romero'`	`'user name'=anything`

Оболочка **bash** и команды широко используют переменные. Одним из основных применений переменных является предоставление средств для настройки различных предпочтений для каждого пользователя. В зависимости от значения переменных оболочка bash и команды могут вести себя по-разному. Что касается оболочки, переменные могут влиять на то, что отображает приглашение, каталоги, в которых оболочка будет искать команды для выполнения, и многое другое.

Локальные переменные и переменные среды

Локальная переменная доступна только для оболочки, в которой она была создана. Эти переменные определены только для текущей сессии. Они будут безвозвратно стерты после завершения сессии, будь то удаленный доступ или эмулятор терминала. Они не хранятся ни в каких файлах, а создаются и удаляются с помощью специальных команд.

Пользовательские переменные окружения оболочки Linux определяются для конкретного пользователя и загружаются каждый раз когда он входит в систему при помощи локального терминала, или же подключается удаленно. Такие переменные, как правило, хранятся в файлах конфигурации: .bashrc, .bash_profile, .bash_login, .profile или в других файлах, размещенных в директории пользователя.

Системные переменные окружения доступны во всей системе, для всех пользователей. Они загружаются при старте системы из системных файлов конфигурации: /etc/environment, /etc/profile, /etc/profile.d/ /etc/bash.bashrc.

КОНФИГУРАЦИОННЫЕ ФАЙЛЫ ПЕРЕМЕННЫХ ОКРУЖЕНИЯ LINUX

.BASHRC

Это файл переменных конкретного пользователя. Загружается каждый раз, когда пользователь создает терминальный сеанс, то есть проще говоря, открывает новый терминал. Все переменные окружения, созданные в этом файле вступают в силу каждый раз когда началась новая терминальная сессия.

.BASH_PROFILE

Эти переменные вступают в силу каждый раз когда пользователь подключается удаленно по SSH. Если этот файл отсутствует система будет искать .bash_login или .profile.

/ETC/ENVIRONMENT

Этот файл для создания, редактирования и удаления каких-либо переменных окружения на системном уровне. Переменные окружения, созданные в этом файле доступны для всей системы, для каждого пользователя и даже при удаленном подключении.

/ETC/BASH.BASHRC

Системный bashrc. Этот файл выполняется для каждого пользователя, каждый раз когда он создает новую терминальную сессию. Это работает только для локальных пользователей, при подключении через интернет, такие переменные не будут видны.

/etc/profile

Системный файл profile. Все переменные из этого файла, доступны любому пользователю в системе, только если он вошел удаленно. Но они не будут доступны, при создании локальной терминальной сессии, то есть если вы просто откроете терминал.

Все переменные окружения Linux созданные с помощью этих файлов, могут быть удаленны всего лишь удалением их оттуда. Только после каждого изменения, нужно либо выйти и зайти в систему, либо выполнить эту команду:

```
source имя_файла
```

Переменная окружения доступна для оболочки, в которой она была создана, и передается во все другие команды/программы, запускаемые оболочкой.

По соглашению строчные буквы используются для создания имен локальных переменных, а заглавные буквы - при именовании переменных среды. Например, локальная переменная может называться test, а переменная среды может называться TEST. Хотя это соглашение, которому следует большинство людей, оно не является правилом.

По умолчанию, когда переменная назначается в оболочке bash, она изначально устанавливается как локальная переменная. Есть несколько способов сделать локальную переменную переменной среды. Во-первых, существующая локальная переменная может быть «экспортирована» с помощью команды export:

```
ENVIRONMENT=1
export ENVIRONMENT
```

Во-вторых, новую переменную можно «экспортировать» и присвоить значение одной командой. В следующем случае мы предполагаем, что переменная ранее не была установлена:

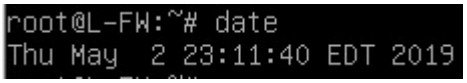
```
export ENVIRONMENT=2
```

В-третьих, можно использовать команду Declare или Typeset, чтобы «объявить» переменную как переменную окружения. Эти команды являются синонимами и работают одинаково:

```
declare -x ENVIRONMENT=3
typeset -x ENVIRONMENT=3
```

Команда **env** также может использоваться для временной установки переменной. Для серверов обычно устанавливается всемирное координированное время (UTC), что хорошо для поддержания согласованного времени на серверах по всей планете, но может разочаровать практическое использование, просто говоря время:

```
date
```



Чтобы временно установить переменную часового пояса, используйте команду **env**:

```
env TZ=EST date
```

Переменная **TZ** устанавливается только в среде текущей оболочки и только на время выполнения команды. Эта переменная не повлияет на остальную часть системы. Фактически, повторный запуск команды **date** проверит, что переменная TZ вернулась в UTC.

Отображение переменных

Есть несколько способов отображения значений переменных. Команда **set** сама по себе отобразит все переменные (локальные и окружающие):

```
set /head
```

Чтобы отобразить только переменные среды, вы можете использовать несколько команд, которые выдают почти одинаковые выходные данные: **env, Declare -x, typeset -x** или **export -p**:

```
env | sort
```

```
root@L-FW:~# env | sort
HOME=/root
HUSHLOGIN=FALSE
LANG=en_US.UTF-8
LOGNAME=root
MAIL=/var/mail/root
OLDPWD=/root
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Чтобы отобразить значение определенной переменной, используйте команду **echo** с именем переменной с префиксом **$** (знак доллара). Например, чтобы отобразить значение переменной **PATH**, вы должны выполнить

```
echo $ PATH
```

```
root@L-FW:~# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Переменные также могут быть заключены в **{}** (фигурные скобки), чтобы отделить их от окружающего текста. В то время как команда **echo ${PATH}** выдает тот же результат, что и команда **echo $PATH**, фигурные скобки визуально выделяют переменную, что облегчает ее просмотр в сценариях в некоторых контекстах.

Сброс переменных

Если вы создаете переменную и больше не хотите, чтобы эта переменная была определена, вы можете использовать команду **unset**, чтобы удалить ее:

```
example=12
echo $example
12
unset example
echo $example
```

Не сбрасывайте критические системные переменные, такие как переменная **PATH**, так как это может привести к неправильной работе среды.

Переменная PATH

Переменная PATH является одной из наиболее важных переменных среды для оболочки, поэтому важно понять, как она влияет на выполнение команд.

Переменная PATH содержит список каталогов, которые используются для поиска команд, введенных пользователем. Когда пользователь вводит команду, а затем нажимает клавишу Enter, в каталогах PATH выполняется поиск исполняемого файла, который соответствует имени команды. Обработка работает через список каталогов слева направо; первый исполняемый файл, который соответствует введенному, является командой, которую попытается выполнить оболочка.

Перед поиском команды для переменной PATH оболочка сначала определит, является ли команда псевдонимом или функцией, что может привести к тому, что переменная PATH не будет использоваться при выполнении этой конкретной команды.

Кроме того, если команда встроена в оболочку, переменная PATH не будет использоваться.

На следующем рисунке показана типичная переменная PATH с именами каталогов, отделенными друг от друга символом «:»

```
echo $PATH
```

```
root@L-FW:~# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

В следующей таблице показано назначение некоторых каталогов, отображаемых в выходных данных предыдущей команды:

Каталог	Содержание
/home/sysadmin/bin	Каталог для текущего пользователя sysadmin для размещения

Каталог	Содержание
	программ. Обычно используется пользователями, которые создают свои собственные сценарии.
/usr/local/sbin	Обычно пустой, но может иметь административные команды, которые были скомпилированы из локальных источников.
/usr/local/bin	Обычно пустой, но может иметь команды, которые были скомпилированы из локальных источников.
/usr/sbin	Содержит большинство административных командных файлов.
/usr/bin	Содержит большинство команд, доступных для выполнения обычными пользователями.
/sbin	Содержит основные административные команды.
/bin	Содержит самые основные команды, которые необходимы для функционирования операционной системы.

Для выполнения команд, которые не содержатся в каталогах, перечисленных в переменной PATH, существует несколько параметров:

- Команда может быть выполнена путем ввода абсолютного пути к команде.
- Команда может быть выполнена с относительным путем к команде.
- Переменная PATH может быть включена в каталог, в котором находится команда.
- Команду можно скопировать в каталог, указанный в переменной PATH.

Абсолютный путь указывает местоположение файла или каталога от каталога верхнего уровня через все подкаталоги к файлу или каталогу. Абсолютные пути всегда начинаются с символа /, представляющего корневой каталог. Например, **/usr/bin/ls** - это абсолютный путь.

Относительный путь указывает местоположение файла или каталога относительно текущего каталога. Например, в каталоге /home/sysadmin относительный путь test/newfile будет фактически ссылаться на файл /home/sysadmin/test/newfile. Относительные пути **никогда не начинаются** с символа /.

Использование относительного пути для выполнения файла в текущем каталоге требует использования символа «.», который символизирует текущий каталог:

```
sysadmin @ localhost: ~ $ ./my.sh
```

Иногда пользователь хочет, чтобы его домашний каталог был добавлен в переменную PATH, чтобы запускать сценарии и программы, не используя ./ перед именем файла. У них может возникнуть желание изменить переменную пути следующим образом:

```
pwd
/root
PATH=/root
```

К сожалению, все, что ранее содержалось в переменной PATH, будет потеряно.

```
echo $PATH
/root
```

Программы, перечисленные за пределами каталога **/root**, теперь будут доступны только по их полному пути.

Можно добавить к переменной PATH, не перезаписывая ее предыдущее содержимое. Импортируйте текущее значение переменной $PATH во вновь определенную переменную PATH, используя его с обеих сторон оператора присваивания.

```
PATH=$PATH
```

Завершите его значением дополнительного пути к домашнему каталогу.

```
PATH=$PATH:/home/sysadmin
```

```
root@L-FW:~# PATH=$PATH:/root
root@L-FW:~# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/root
```

Теперь сценарии, расположенные в каталоге **/root**, могут выполняться без указания пути:

```
nano my.ch
```

```
#!/bin/bash
echo 'Hello!'

chmod u+x my.sh
```

Файлы инициализации

Когда пользователь открывает новую оболочку, либо во время входа в систему, либо когда он запускает терминал, который запускает оболочку, оболочка настраивается с помощью файлов, называемых файлами инициализации (или конфигурации). Эти файлы инициализации устанавливают значение переменных, создают псевдонимы и функции и выполняют другие команды, которые полезны при запуске оболочки.

Существует два типа файлов инициализации: «глобальные» файлы инициализации, которые влияют на всех пользователей в системе, и «локальные» файлы инициализации, характерные для отдельного пользователя.

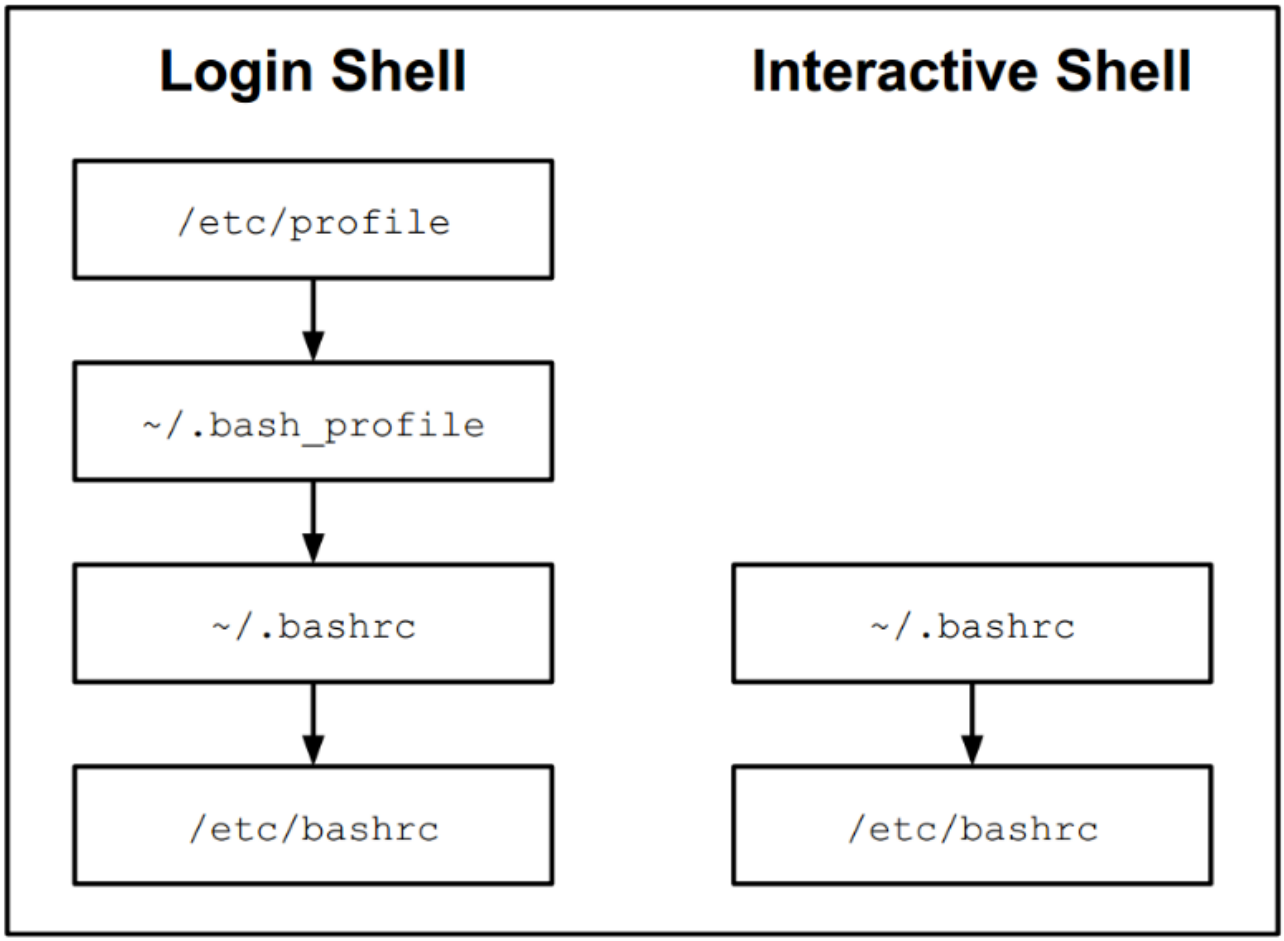
Глобальные файлы конфигурации находятся в каталоге **/etc**. Локальные файлы конфигурации хранятся в домашнем каталоге пользователя (/home/user, /root – для суперпользователя).

Файлы инициализации BASH

Каждая оболочка использует разные файлы инициализации. Кроме того, большинство оболочек выполняют разные файлы инициализации, когда оболочка запускается через процесс входа в систему (называемая оболочкой входа в систему), в отличие от того, когда оболочка запускается терминалом (называемой оболочкой без регистрации или интерактивной оболочкой).

Следующая диаграмма иллюстрирует различные файлы, которые запускаются с типичной оболочкой входа в систему по сравнению с интерактивной оболочкой:

Примечание. Символ ~ представляет домашний каталог пользователя и имена файлов, перед которыми стоит символ. указать скрытые файлы.



Когда bash запускается как оболочка входа в систему, сначала выполняется файл **/etc/profile**. Этот файл обычно выполняет все файлы, заканчивающиеся на **.sh**, которые находятся в каталоге **/etc/profile.d**. Следующий исполняемый файл - обычно **~ /.bash_profile**, но некоторые пользователи могут использовать файл **~ /.bash_login** или **~/.profile**. Файл **~/.bash_profile** обычно также выполняет файл **~ /.bashrc**, который, в свою очередь, выполняет файл **/etc/bashrc**.

Поскольку файл **~/.bash_profile** находится под контролем пользователя, выполнение файлов **~ /.bashrc** и **/etc/bashrc** также контролируется пользователем.

Когда bash запускается как интерактивная оболочка, он запускает файл **~/.bashrc**, который также может выполнять файл **/etc/bashrc**, если он существует. Опять же, поскольку файл **~/.bashrc** принадлежит пользователю, вошедшему в систему, он может предотвратить выполнение файла **/etc/bashrc**.

При таком количестве файлов инициализации общий вопрос на данный момент: «какой файл я должен использовать?» Следующая таблица иллюстрирует назначение каждого из этих файлов, предоставляя примеры команд, которые вы можете поместить в каждый файл:

File	Purpose
/etc/profile	Этот файл может быть изменен только администратором и будет выполняться каждым пользователем, который входит в систему. Администраторы используют этот файл для создания ключевых переменных среды, отображения сообщений пользователям при их входе в систему и установки ключевых системных значений.
~/.bash_profile	Каждый пользователь имеет свой собственный файл .bash_profile в своем домашнем каталоге. Назначение этого файла такое же, как и в файле /etc/profile, но наличие этого файла позволяет пользователю настраивать оболочку по своему вкусу. Обычно используется для создания пользовательских переменных среды.
~/.bashrc	Каждый пользователь имеет свой собственный файл .bashrc в своем домашнем каталоге. Цель этого файла - генерировать вещи, которые необходимо создать для каждой оболочки, такие как локальные переменные и псевдонимы.
/etc/bashrc	Этот файл может повлиять на каждого пользователя в системе. Только администратор может изменить этот файл. Как и в файле .bashrc, целью этого файла является создание объектов, которые необходимо создать для каждой оболочки, таких как локальные переменные и псевдонимы.

Изменение файлов инициализации

Способ работы пользовательской оболочки может быть изменен путем изменения файлов инициализации этого пользователя. Изменение глобальной конфигурации требует административного доступа к системе, поскольку файлы в каталоге **/etc** могут быть изменены только администратором. Пользователь может изменять только файлы инициализации в своем домашнем каталоге.

Рекомендуется создать резервную копию перед изменением файлов конфигурации в качестве страховки от ошибок; резервную копию всегда можно восстановить, если что-то пойдет не так.

Файл **~/.bash_profile** по умолчанию, поставляемый с CentOS 7, содержит следующие две строки, которые настраивают переменную среды PATH:

```
PATH=$PATH:$HOME/bin
export PATH
```

Первая строка устанавливает переменную PATH в существующее значение переменной PATH с добавлением подкаталога bin домашнего каталога пользователя. Переменная $ HOME ссылается на домашний каталог пользователя. Например, если пользователь вошел в систему как «joe», то **$HOME/bin** - это **/home/joe/bin**.

Вторая строка преобразует локальную переменную PATH в переменную окружения.

Файл ~/.bashrc по умолчанию выполняет /etc/bashrc, используя следующую инструкцию:

```
./etc/bashrc
```

Символ «.», используемый для создания файла с целью его исполнения. Символ «.» также имеет команду синонима, исходную команду, но использование «.» более распространено, чем использование исходной команды.

Sourcing может быть эффективным способом проверки изменений, внесенных в файлы инициализации, позволяя исправлять любые ошибки без необходимости выхода из системы и повторного входа. Если вы обновите файл **~/.bash_profile**, чтобы изменить переменную **PATH**, вы можете проверить правильность внесенных изменений, выполнив следующее:

```
.~/.bash_profile
echo $PATH
```

Сценарии выхода BASH

Так же, как bash выполняет один или несколько файлов при запуске, он также может выполнять один или несколько файлов при выходе. Когда bash завершает работу, он выполняет файлы **~/.bash_logout** и **/etc/bash_logout**, если они существуют. Обычно эти файлы используются для «очистки» тактики при выходе пользователя из оболочки. Например, по умолчанию **~/.bash_logout** выполняет команду **clear**, чтобы удалить любой текст, присутствующий на экране терминала.

Когда создается новый пользователь, файлы из каталога **/etc/skel** автоматически копируются в домашний каталог нового пользователя. Как администратор, вы можете изменять файлы в каталоге **/etc/skel** для предоставления пользовательских функций новым пользователям.

История команд

В некотором смысле файл **~/.bash_history** также можно считать файлом инициализации, поскольку **bash** также читает этот файл при запуске. По умолчанию этот файл содержит историю команд, которые пользователь выполнил в оболочке **bash**. Когда пользователь выходит из оболочки **bash**, он записывает в этот файл недавнюю историю.

Есть несколько способов, которыми эта история команд выгодна для пользователя:

- Вы можете использовать стрелки вверх и вниз, чтобы просмотреть историю и выбрать предыдущую команду для повторного выполнения.
- Вы можете выбрать предыдущую команду и изменить ее перед выполнением.
- Вы можете выполнить обратный поиск по истории, чтобы найти предыдущую команду, чтобы выбрать, отредактировать и выполнить ее. Чтобы начать поиск, нажмите CTRL + R, а затем начните вводить часть предыдущей команды.
- Вы снова выполняете команду, основываясь на числе, связанном с командой.

Выполнение предыдущих команд

Оболочка bash позволит пользователю использовать клавишу со стрелкой вверх для просмотра предыдущих команд. При каждом нажатии стрелки вверх оболочка отображает еще одну команду в списке истории.

Если пользователь заходит слишком далеко, можно использовать клавишу со стрелкой вниз, чтобы вернуться к более новой команде. После отображения правильной команды можно нажать клавишу Enter, чтобы выполнить ее.

Клавиши «Стрелка влево» и «Стрелка вправо» также можно использовать для позиционирования курсора в команде, клавиши «Backspace» и «Delete» для удаления текста, а дополнительные символы можно вводить в командной строке, чтобы изменить его, прежде чем нажимать клавишу «Enter» для его выполнения.

Есть еще ключи, которые можно использовать для редактирования команды в командной строке. В следующей таблице приведены некоторые полезные ключи редактирования:

Action	Key	Alternate Key Combination
Previous history item	↑	**Ctrl+P**
Next history item	↓	**Ctrl+N**
Reverse history search		**Ctrl+R**
Beginning of line	**Home**	**Ctrl+A**
End of line	**End**	**Ctrl+E**
Delete current character	**Delete**	**Ctrl+D**
Delete to left of cursor	**Backspace**	**Ctrl+X**
Move cursor left	←	**Ctrl+B**
Move cursor right	→	**Ctrl+F**

Изменение ключей редактирования

Ключи, доступные для редактирования команды, определяются настройками библиотеки Readline. Ключи обычно устанавливаются так, чтобы они соответствовали назначениям клавиш, найденным в текстовом редакторе emacs (популярный редактор Linux) по умолчанию.

Чтобы связать ключи в соответствии с назначениями клавиш, найденными в другом популярном текстовом редакторе, редакторе vi, оболочку можно настроить с помощью команды **set -o vi**. Чтобы вернуть привязки клавиш обратно в текстовый редактор emacs, используйте команду **set -o emacs**.

Чтобы автоматически настроить параметры истории редактирования, отредактируйте файл **~/.inputrc**. Если этот файл не существует, вместо него используется файл **/etc/inputrc**. Привязки клавиш задаются в конфигурационных файлах иначе, чем в командной строке; например, чтобы включить режим привязки ключа vi для отдельного пользователя, добавьте следующие строки в файл **~/.inputrc**:

```
set editing-mode vi
set keymap vi
```

Использование команды History

Команда **history** может использоваться для «повторного выполнения» ранее выполненных команд. При использовании без аргументов команда history предоставляет список ранее выполненных команд:

```
history
```

Обратите внимание, что каждой команде присваивается номер, который пользователь может использовать для повторного выполнения команды.

Команда **history** имеет множество параметров, наиболее распространенные из которых перечислены ниже:

Option	Purpose
-c	Очистить историю
-r	Считать историю из файла и заменить ее
-w	Записать историю в файл

Поскольку список истории обычно содержит пятьсот или более команд, часто полезно фильтровать список. Команда **history** принимает число в качестве аргумента, чтобы указать, сколько команд перечислить. Например, выполнение команды **history 10** покажет только последние десять команд из вашей истории.

Настройка истории Команда

Когда вы закрываете программу оболочки, она берет команды из списка истории и сохраняет их в файле истории **(~/.bash_history**). По умолчанию пятьсот команд будут сохранены в файле истории. Переменная HISTFILESIZE будет определять, сколько команд записать в этот файл.

Если пользователь хочет сохранить команды истории в файле, отличном от **~/.bash_history**, пользователь может указать абсолютный путь к другому файлу в качестве значения для локальной переменной HISTFILE:

```
HISTFILE = /path/to/file
```

Переменная HISTSIZE будет определять, сколько команд хранить в памяти для каждой оболочки bash. Если размер HISTSIZE больше, чем размер HISTFILESIZE, то при выходе из оболочки bash в файл истории будет записано только самое последнее число команд, указанное в HISTFILESIZE.

Хотя по умолчанию для него обычно ничего не установлено, вы можете воспользоваться установкой значения для переменной HISTCONTROL в файле инициализации, таком как файл **~/.bash_profile**. Переменная HISTCONTROL может быть установлена для любой из следующих функций:

HISTCONTROL = ignoredups предотвратит дублирование команд, которые выполняются последовательно.

HISTCONTROL = ignorespace не будет хранить команды, начинающиеся с пробела. Это предоставляет пользователю простой способ выполнить команду, которая не войдет в список истории.

HISTCONTROL = ignoreboth не будет хранить последовательные дубликаты или любые команды, начинающиеся с пробела.

HISTCONTROL = erasedups не будет хранить команду, идентичную другой команде в вашей истории (фактически предыдущая запись команды будет удалена из списка истории).

HISTCONTROL = ignorespace: erasedups будет включать в себя преимущество «erasedups» с преимуществом «ignorespace».

Другой переменной, которая будет влиять на то, что хранится в истории команд, является переменная **HISTIGNORE**. Большинство пользователей не хотят, чтобы список истории был забит базовыми командами, такими как ls, cd, exit и history. Переменная **HISTIGNORE** может использоваться для указания **bash** не хранить определенные команды в списке истории.

Чтобы команды не были включены в список истории, включите в файл **~/.bash_profile** команду, подобную следующей:

```
HISTIGNORE = 'ls *: cd *: history *: exit'
```

Выполнение предыдущих команд

! (восклицательный знак) - это специальный символ оболочки bash, указывающий на выполнение команды в списке истории. Есть много способов использовать! символ для повторного выполнения команд; В следующей таблице приведены некоторые примеры использования символа !:

History Command	Meaning
`!!`	Повторите последнюю команду
`!-4`	Выполнить команду, которая была запущена четыре команды назад
`!555`	Выполнить команду № 555
`!ec`	Выполните последнюю команду, которая началась с ec
`!?joe`	Выполните последнюю команду, которая содержала joe

Еще один способ использования списка истории - воспользоваться тем фактом, что последний аргумент каждой команды «запоминается». Часто пользователь вводит команду для ссылки на файл, возможно, для отображения некоторой информации об этом файле. Впоследствии пользователь может захотеть ввести другую команду, чтобы сделать что-то еще с тем же файлом. Вместо того, чтобы снова вводить путь, пользователь может использовать несколько сочетаний клавиш, чтобы вызвать этот путь к файлу из предыдущей командной строки.

Нажав либо Esc + . (Escape Period) или Alt + . (Alt Period), оболочка «вернет» последний аргумент предыдущей команды. Повторное нажатие одной из этих комбинаций клавиш вызывает в обратном порядке последний аргумент каждой предыдущей команды.