Содержание

Карманный справочник по оболочке Bash

В этом карманном справочнике вкратце описывается оболочка Bash и, в частности, ее версия 4.4 для GNU/Linux и Mac OS X. Оболочка Bash имеется также для Solaris и различных систем BSD. Ее можно скомпилировать практически для любой другой системы Unix и даже для OpenVMS! В этом справочнике рассматриваются следующие вопросы.

- История развития оболочки.
- Краткий обзор функциональных средств.
- Вызов оболочки.
- Код завершения команды.
- Синтаксис.
- Функции.
- Переменные.
- Арифметические выражения.
- Предыстория выполнения команд.
- Автозавершение вводимых команд.
- Управление заданиями.
- Параметры оболочки.
- Выполнение команд.
- Сопроцессы.
- Ограниченные оболочки.
- Встроенные команды.
- Дополнительные источники информации.

Условные обозначения

Имена файлов, команд, параметров и приводимые примеры выделены в тексте данного справочника моноширинным шрифтом. Все, что должно вводиться пользователем в командной строке, выделено **полужирным моноширинным** шрифтом, а текст, который должен быть заменен настоящими данными в примерах и описаниях синтаксиса, — *наклонным моноширинным* шрифтом. Новые термины, особо подчеркиваемые слова и выражения выделяются *курсивом*. И, наконец, ссылки в форме *имя*(*N*) обозначают номер оперативной страницы руководства, где *имя* — наименование оболочки, *N* — раздел руководства, оперативно доступного по команде `man`. Имена переменных оболочки, в том числе и переменных окружения, обозначаются как `$VAR`, где `VAR` — имя переменной.

История развития оболочки

Первоначально оболочка Bourne shell (т.е. оболочка Борна) получила распространение в 1979 году вместе с системой V7 Unix и впоследствии стала стандартной для написания сценариев оболочки. Оболочку Bourne shell можно по-прежнему обнаружить в каталоге `/bin/sh` многих коммерческих систем Unix. Она не претерпела особых изменений с момента своего выпуска, хотя с годами подвергалась незначительным улучшениям. К числу наиболее примечательных новых средств, внедренных в этой оболочке, относятся переменная `CDPATH` и встроенная команда `test` (появились в Unix System III в 1980 году), хеширование команд и функции оболочки для версии System V Release 2 в 1984 году, а также средства управления заданиями, внедренные в версии System V Release 4 в 1989 году.

В оболочке Berkeley C shell (`csh`) предоставлялись средства, которые оказались более удобными для применения в диалоговом режиме (например, предыстория выполнения команд и управление заданиями), и поэтому долгое время стандартной

для программирования в Unix считалась оболочка Bourne shell, а для повседневной работы — оболочка C shell. Дэвид Корн из компании Bell Labs стал первым разработчиком, усовершенствовавшим оболочку Bourne shell, дополнив ее такими `csh`-подобными средствами, как предыстория команд, управление заданиями и дополнительные удобства программирования. В конечном итоге оболочка Korn shell (т.е. оболочка Корна) превзошла своим набором функциональных средств оболочки Bourne shell и C shell, сохранив в то же время совместимость с предыдущими средствами программирования на языке оболочки. А ныне язык и режим работы "стандартной оболочки" определены в стандарте POSIX на основании оболочки Bourne shell для системы Unix System V вместе с избранным поднабором функциональных средств из оболочки Korn shell.

Преследуя цель произвести систему, полностью отвечающую Unix по своим рабочим характеристикам, Фонд свободного программного обеспечения (Free Software Foundation) разработал копию оболочки Bourne shell, написанную заново и получившую название *Bash* (Bourne-Again Shell — "возрожденная" оболочка Борна). Со временем Bash стала совместимой со стандартом POSIX версией оболочки со многими дополнительными средствами, перекрывающими аналогичные средства оболочки Korn shell, хотя Bash не является точной копией оболочки Korn shell. В настоящее время оболочка Bash считается едва ли не самой широко распространенной версией, производной от оболочки Bourne shell.

Краткий обзор функциональных средств

В оболочке Bash предоставляются следующие функциональные средства и возможности.

- Переадресация ввода-вывода.
- Применение метасимволов подстановки для сокращения имен файлов.

- Переменные и параметры для специальной настройки рабочей среды.
- Встроенный набор команд для написания программ оболочки.
- Функции и оболочки для модульной организации задач, выполняемых в программах оболочки.
- Управление заданиями.
- Редактирование с использованием синтаксиса команд редактора `vi` или Emacs, имитирующее режим командной строки.
- Доступ к предыдущим командам (из предыстории их выполнения) и возможность редактировать эти команды.
- Целочисленные арифметические операции.
- Массивы и арифметические выражения.
- Применение псевдонимов для сокращения имен команд.
- Прямая совместимость со стандартом POSIX.
- Средства интернационализации.
- Цикл `for` для повторного выполнения арифметических операций.

Вызов оболочки

Интерпретатор команд оболочки Bash (`bash`) можно вызвать следующим образом:

`bash [`*`параметры`*`] [`*`аргументы`*`]`

Команды оболочки Bash можно выполнять с терминала, из файла (когда в качестве первого *аргумента* указан сценарий) или из стандартного ввода (если аргументы отсутствуют или указан параметр **`-s`**). Оболочка автоматически выводит приглашение на ввод команд, если стандартный ввод производится с терминала или в командной строке указан параметр **`-i`**.

Во многих системах путь к оболочке Bash осуществляется по ссылке /bin/sh. Если же оболочка Bash вызывается как sh, она действует в большей степени как традиционная оболочка Bourne shell. В частности, исходные оболочки читают содержимое файлов /etc/profile и ~/.profile, а обычные оболочки — содержимое переменной окружения $ENV, если она установлена. Подробнее об этом можно узнать на оперативной странице руководства *bash*(1).

Параметры командной строки

Во встроенной команде set (подробное ее описание приведено ниже, в разделе "Встроенные команды") могут быть указаны однобуквенные параметры командной строки, приведенные ниже.

-c *строка*. Читать команды из указанной *строки*.

-D, **-dump-strings**. Вывести из программы все символьные строки, представленные в форме $"…".

-i. Создать интерактивную оболочку (с приглашением на ввод команд). Этот параметр может и не применяться в команде set.

-l, **--login**. Обозначает такой же режим, как и для исходной оболочки.

-O *параметр*. Активизирует заданный *параметр* команды shopt. Для установки заданного *параметра* в исходное состояние служит параметр **+O**.

-p. Начать работу в качестве привилегированного пользователя. Не читать содержимое переменной окружения $ENV или $BASH_ENV; не пытаться импортировать функции из рабочей среды, а также игнорировать значения переменных BASHOPTS, CDPATH, GLOBIGNORE и SHELLOPTS. В этом случае читается содержимое обычных файлов запуска (например, ~/.bash_profile).

-r, **--restricted**. Создать ограниченную оболочку (см. далее раздел "Ограниченные оболочки").

-s. Читать команды из стандартного ввода. Результаты, выводимые из встроенных команд, направляются в файл с дескриптором **1**, а все остальные результаты, выводимые из оболочки, — в файл с дескриптором **2**.

-v, **--verbose**. Выводить строки по мере их чтения оболочкой.

--debugger. Если при запуске доступен отладочный профиль, прочитать его и активизировать параметр `extdebug` команды `shopt`. Служит для применения в отладчике оболочки Bash (подробнее об этом см. по адресу `http://bashdb.sourceforge.net`).

--dump-po-strings. То же, что и параметр **-D**, но на этот раз строки выводятся в формате `gettext` по общей лицензии GNU.

--help. Вывести сообщение об использовании команды и успешно завершить.

--init-file *файл*, **--rcfile** *файл*. Использовать указанный *файл* вместо файла `~/.bashrc` для запуска интерактивных оболочек.

--noediting. Не пользоваться библиотекой GNU Readline для ввода, даже в интерактивной оболочке.

--noprofile. Не читать файл `/etc/profile` или любые другие личные файлы запуска.

--norc. Не читать файл `~/.bashrc`. Активизируется автоматически, когда оболочка вызывается как `sh`.

--posix. Активизировать режим работы по стандарту POSIX.

--version. Вывести сообщение об используемой версии оболочки и успешно завершить.

-, **--**. Завершить обработку параметров.

Пояснение остальных параметров приведено далее, в разделе "Встроенные команды" при описании команды `set`.

Аргументы

Аргументы присваиваются позиционным параметрам `$1`, `$2` и т.д. Если в качестве первого аргумента указывается сценарий, из него читаются команды, а остальные аргументы присваиваются позиционным параметрам `$1`, `$2` и т.д. Имя сценария доступно в качестве позиционного параметра `$0`. Файл самого сценария не обязательно должен быть исполняемым, но доступным для чтения.

Код завершения команды

По завершении любой команды предоставляется числовой *код* ее *завершения* или *возвращаемое значение*. Внешние команды вроде `ls` предоставляют такое значение операционной системе. А внутренние команды вроде `cd` предоставляют его непосредственно оболочке.

Оболочка автоматически извлекает возвращаемое значение по завершении команды. По общепринятому соглашению нулевой код завершения означает *истинный* или *удачный* исход выполнения команды, а любой другой код завершения — *ложный* или *неудачный* исход ее выполнения. Именно таким образом в оболочке организуется управление потоком команд в таких операторах, как `if`, `while` и `until`.

Кроме того, оболочка делает значение, возвращаемое последней выполнявшейся командой, доступным для своего сценария в переменной `$?`. Но, как правило, это значение следует сохранять в другой переменной, поскольку последующие команды из сценария перезапишут его в переменной `$?`.

Значения кодов завершения могут находиться в пределах от **`0`** до **`255`**. Для обозначения отдельных условий завершения команд в оболочке применяются конкретные значения, перечисленные ниже.

Числовое значение кода завершения	Назначение
`0`	Обозначает удачный исход выполнения команды
`2`	Возвращается встроенными командами для указания на ошибки в их использовании
`126`	Обозначает, что команда была обнаружена, но не выполнена
`127`	Обозначает, что команда не была обнаружена
`128+`***N***	Обозначает, что выполнение команды было прервано, поскольку был получен сигнал ***N***

Синтаксис

В этом разделе описываются многие обозначения, присущие рассматриваемой здесь оболочке. В нем поочередно рассматриваются следующие вопросы:

- Специальные файлы.
- Метасимволы подстановки имен файлов.
- Раскрытие скобок.
- Управляющие последовательности символов.
- Заключение в кавычки.
- Формы команд.
- Формы переадресации ввода-вывода.

Специальные файлы

Оболочка читает содержимое одного или нескольких файлов запуска. Содержимое некоторых из этих файлов читается только в том случае, если оболочка является исходной, т.е. служит для входа в систему. Оболочка Bash читает содержимое этих файлов в следующем порядке.

- Файл `/etc/profile`. Выполняется автоматически при входе в систему.

- Первый файл, обнаруживаемый в следующем списке: `~/.bash_profile`, `~/.bash_login` или `~/.profile`. Выполняется автоматически при входе в систему.
- Файл `~/.bashrc`. Читается всякой неисходной оболочкой. Но если оболочка Bash вызывается как `sh` или с параметром **`--posix`**, то она читает содержимое переменной окружения `$ENV` ради обеспечения совместимости со стандартом POSIX.

Функции `getpwnam()` и `getpwuid()` из библиотеки C служат для выдачи информации о начальных каталогах по сокращениям *~имя*. (В персональных системах база данных пользователя хранится в файле `/etc/passwd`. Но в сетевых системах эта информация может поступать из служб NIS, NIS+, LDAP и других источников, а не из файла паролей на рабочей станции пользователя.)

Когда происходит выход из интерактивной исходной оболочки или же в неинтерактивной исходной оболочке выполняется встроенная команда `exit`, оболочка Bash читает и выполняет содержимое файла `~/.bash_logout`, при условии, что этот файл существует. (Исходной является та оболочка, где установлен параметр **`-l`**.)

Метасимволы подстановки имен файлов

Ниже перечислены метасимволы, применяемые для подстановки имен файлов.

`*`	Совпадение с любой строкой, содержащей от нуля и больше символов
`?`	Совпадение с любым символом
`[`*`abc`*`...]`	Совпадение с любым символом в квадратных скобках, где дефис может обозначать заданный диапазон символов (например, **`a-z`**, **`A-Z`**, **`0-9`**)
`[!`*`abc`*`...]`	Совпадение с любым символом, кроме тех, которые указаны в квадратных скобках
`~`	Начальный каталог текущего пользователя
`~`*`имя`*	Начальный каталог пользователя, имеющего указанное ***`имя`***
`~+`	Текущий рабочий каталог (содержимое переменной **`$PWD`**)
`~-`	Предыдущий рабочий каталог (содержимое переменной **`$OLDPWD`**)