

Поведение Баша

 неполный

Режимы запуска Bash

Оболочка входа

В качестве "оболочки входа" Bash считывает и устанавливает (выполняет) профиль пользователя из `/etc/profile` и одного из `~/.bash_profile` , `~/.bash_login` , или `~/.profile` (в таком порядке, используя первый доступный для чтения!).

Когда оболочка входа завершается, Bash считывает и выполняет команды из файла `~/.bash_logout` , если он существует.

Зачем нужен дополнительный режим оболочки входа? Существует множество действий и наборов переменных, которые имеют смысл только для первоначального входа пользователя в систему. Вот почему все оболочки UNIX® имеют (должны иметь) режим "входа".

Способы запуска Bash в качестве оболочки входа:

- первый символ `argv[0]` is - (дефис): традиционные оболочки UNIX® начинаются с `login` двоичного файла
- Bash запускается с `-l` опцией
- Bash запускается с `--login` опцией

Методы проверки режима оболочки входа в систему:

- параметр оболочки `login_shell` установлен

Связанные переключатели:

- `--noprofile` отключает чтение всех файлов профиля

Интерактивная оболочка

Когда Bash запускается как интерактивная оболочка без входа в систему, она считывает и выполняет команды из `~/.bashrc` . Этот файл должен содержать, например, псевдонимы, поскольку они должны быть определены в каждой оболочке, поскольку они не наследуются от родительской оболочки.

Возможность иметь общесистемный `/etc/bash.bashrc` или аналогичный общесистемный rc-файл специфична для поставщиков и дистрибьюторов, которые поставляют *свой собственный, исправленный вариант Bash*. Классический способ получить общесистемный rc-файл - `source /etc/bashrc` от каждого пользователя `~/.bashrc`.

Методы тестирования для режима интерактивной оболочки:

- специальный параметр `$-` содержит букву `i` (строчная буква l)

Связанные переключатели:

- `-i` запускает интерактивный режим
- `--norc` отключает чтение файлов запуска (например `/etc/bash.bashrc`, если поддерживается) и `~/.bashrc`
- `--rcfile` определяет другой загрузочный файл (вместо `/etc/bash.bashrc` and `~/.bashrc`)

Режим SH

Когда Bash запускается в режиме совместимости с SH, он пытается максимально точно имитировать поведение при запуске исторических версий `sh`, а также соответствует стандарту POSIX®. Считываемые файлы профиля - это `/etc/profile` и `~/.profile`, если это оболочка входа.

Если это не оболочка входа, вычисляется переменная среды `ENV`, и результирующее имя файла используется в качестве имени файла запуска.

После чтения файлов запуска Bash переходит в режим совместимости с POSIX (r) (для запуска, а не для запуска!).

Bash запускается в sh режиме совместимости, когда:

- базовое имя файла в `argv[0]` `sh` (⚠️ ПРИМЕЧАНИЕ: `/bin/sh` может быть связано с `/bin/bash`, но это не значит, что оно действует как `/bin/bash` ⚠️)

Режим POSIX

Когда Bash запускается в режиме POSIX®, он соответствует стандарту POSIX® для файлов запуска. В этом режиме **интерактивные оболочки** расширяют переменную `ENV`, а команды считываются и выполняются из файла, имя которого является расширенным значением.

Никакие другие файлы запуска не читаются. Следовательно, неинтерактивная оболочка не считывает никакие файлы запуска в режиме POSIX®.

Bash запускается в режиме POSIX®, когда:

- параметр командной `--posix` строки указан
- установлена переменная среды `POSIXLY_CORRECT`

Быстрая ссылка на файл запуска

- Возможные общесистемные rc-файлы обычно считываются, когда `~/ .bashrc` должны быть прочитаны (по крайней мере, Debian GNU / Linux ведет себя так)
- Независимо от того, какие общесистемные файлы `/etc` всегда считываются, Bash обычно считывает первый найденный файл, когда задается несколько вариантов (для пользовательских файлов в `~/`)

Режим	<code>/etc/profile</code>	<code>~/ .bash_profile</code>	<code>~/ .bash_login</code>	<code>~/ .profile</code>
Оболочка входа	X	X	X	X
Интерактивная оболочка	-	-	-	-
Логин, совместимый с SH	X	-	-	X
Совместимость с SH	-	-	-	-
Совместимость с POSIX®	-	-	-	-



Режимы запуска Bash

Обычный Bash

Режим запуска POSIX

В режиме POSIX® Bash следует стандарту POSIX® в отношении поведения и синтаксического анализа (выдержка из документа сопровождающего Bash):

Запуск Bash с параметром командной строки `--posix` или выполнение `set`

`-o posix` во время выполнения Bash приведет к тому, что Bash будет более точно соответствовать

стандарту POSIX, изменяя поведение в соответствии с поведением, указанным

POSIX в областях, где значение Bash по умолчанию отличается.

При вызове как `sh`, Bash переходит в режим POSIX после чтения файлов запуска

.

Ниже приведен список того, что изменилось, когда Bash находится в режиме POSIX":

1. Когда команда в хэш-таблице больше не существует, Bash повторно выполнит поиск `$PATH`, чтобы найти новое местоположение. Это также доступно с помощью `shopt -s checkhash`.

2. Сообщение, печатаемое кодом управления заданием и встроенными, когда задание завершается с ненулевым статусом, является "Выполнено (статус)".

3. Сообщение, печатаемое управляющим кодом задания и встроенными функциями при остановке задания, называется 'Stopped (SIGNAME)', где SIGNAME - это, например, `SIGTSTP`.

4. Встроенный файл `bg` использует требуемый формат для описания каждого задания, размещенного в фоновом режиме, который не включает указание на то, является ли задание текущим или предыдущим заданием.

5. Зарезервированные слова, появляющиеся в контексте, где распознаются зарезервированные слова, не подвергаются расширению псевдонима.

6. Расширения POSIX `PS1` и `PS2` от `!` до номера истории и `!!` до `!` включены, и расширение параметров выполняется для значений `PS1` и `PS2` независимо от настройки опции `promptvars`.

7. Запускаемые файлы POSIX выполняются (`$ ENV`), а не обычные файлы Bash.

8. Расширение тильды выполняется только для назначений, предшествующих имени команды, а не для всех операторов присваивания в строке.

9. Файл истории по умолчанию - `~/.sh_history` (это значение по умолчанию `$ HISTFILE`).

10. Выходные данные `kill -l` печатают все названия сигналов в одной строке, разделенные пробелами, без префикса `SIG`.

11. Встроенная функция ``kill`` не принимает имена сигналов с ``SIG`` префикс.
12. Неинтерактивные оболочки завершают работу, если ИМЯ ФАЙЛА в файле ``.`` не найдено.
13. Неинтерактивные оболочки завершают работу, если синтаксическая ошибка в арифметическом расширении приводит к недопустимому выражению.
14. Операторы перенаправления не выполняют расширение имени файла для слова в перенаправлении, если оболочка не является интерактивной.
15. Операторы перенаправления не выполняют разбиение слова на слова в перенаправлении.
16. Имена функций должны быть допустимыми именами оболочки. То есть они не могут содержать других символов, кроме букв, цифр и символов подчеркивания, и не могут начинаться с цифры. Объявление функции с недопустимым именем приводит к фатальной синтаксической ошибке в неинтерактивных оболочках.
17. Специальные встроенные функции POSIX обнаруживаются перед функциями оболочки во время поиска команд.
18. Если специальная встроенная функция POSIX возвращает статус ошибки, неинтерактивная оболочка завершает работу. Неустраняемые ошибки - это те, которые перечислены в стандарте POSIX, и включают в себя такие вещи, как передача неправильных параметров, ошибки перенаправления, ошибки присвоения переменных для назначений, предшествующих имени команды, и т.д.
19. Если задан ``CDPATH``, встроенный ``cd`` не будет неявно добавлять к нему текущий каталог. Это означает, что ``cd`` завершится ошибкой, если ни одно допустимое имя каталога не может быть создано из любой записи в ``$CDPATH``, даже если каталог `a` с тем же именем, что и имя, указанное в качестве аргумента для ``cd``, существует в текущем каталоге.
20. Неинтерактивная оболочка завершает работу со статусом ошибки, если переменная ошибка присваивания возникает, когда имя команды не следует за инструкциями присваивания. Ошибка присвоения переменной возникает, например, при попытке присвоить значение переменной, доступной только для чтения.

21. Неинтерактивная оболочка завершает работу со статусом ошибки, если переменная итерации в операторе ``for'` или переменная выбора в операторе ``select'` является переменной только для чтения.
22. Замена процесса недоступна.
23. Операторы присваивания, предшествующие специальным встроенным функциям POSIX, сохраняются в среде оболочки после завершения сборки.
24. Операторы присваивания, предшествующие вызовам функций оболочки, сохраняются в среде оболочки после возврата функции, как если бы была выполнена специальная встроенная команда POSIX.
25. Встроенные команды ``export'` и ``readonly'` отображают свои выходные данные в формате, требуемом POSIX.
26. Встроенная ``ловушка'` отображает названия сигналов без начального ``SIG'`.
27. Встроенная ``ловушка'` не проверяет первый аргумент на предмет возможности спецификации сигнала и не возвращает обработку сигнала к исходному расположению, если это так, если только этот аргумент не состоит исключительно из цифры и является действительным номером сигнала. Если пользователи хотят сбросить обработчик для данного сигнала в исходное расположение, они должны использовать ``-'` в качестве первого аргумента.
28. Встроенные функции ``.`` и ``source'` не выполняют поиск аргумента `filename` в текущем каталоге, если он не найден с помощью поиска ``PATH'`.
29. Подоболочки, созданные для выполнения подстановок команд, наследуют значение параметра ``-e'` из родительской оболочки. Когда он не находится в режиме POSIX, Bash очищает параметр ``-e'` в таких подоболочках.
30. Расширение псевдонимов всегда включено, даже в неинтерактивных оболочках.
31. Когда встроенный "псевдоним" отображает определения псевдонимов, он не отображает их с ведущим "псевдонимом", если не указана опция `"-p"`.
32. Когда встроенный компонент `set` вызывается без параметров, он не отображает имена и определения функций оболочки.
33. Когда встроенная функция `set` вызывается без параметров, она отображает

значения переменных без кавычек, если только они не содержат метасимволы оболочки, даже если результат содержит непечатаемые символы.

34. Когда встроенный ``cd'` вызывается в ЛОГИЧЕСКОМ режиме, и имя пути созданный на основе ``$ PWD'`, а имя каталога, указанное в качестве аргумента, не ссылается на существующий каталог, ``cd'` завершится ошибкой вместо возврата в ФИЗИЧЕСКИЙ режим.

35. Когда встроенному ``pwd'` предоставляется опция ``-P'`, он сбрасывает ``$PWD'` к имени пути, не содержащему символических ссылок.

36. Встроенный модуль ``pwd'` проверяет, что значение, которое он выводит, совпадает с текущим каталогом, даже если его не просят проверить файловую систему с помощью опции ``-P'`.

37. При перечислении истории встроенный `"fc"` не включает указание на то, была ли изменена запись истории.

38. Редактор по умолчанию, используемый ``fc'`, - это ``ed'`.

39. Встроенные команды `'type'` и `'command'` не сообщают о найденном не исполняемом файле, хотя оболочка попытается выполнить такой файл, если это единственный файл с таким именем, найденный в ``$PATH'`.

40. Режим редактирования ``vi'` будет вызывать редактор ``vi'` непосредственно при выполнении команды ``v'`, вместо проверки ``$ FCEDIT'` и ``$EDITOR'`.


41. Когда включена опция ``xpg_echo'`, Bash не пытается интерпретировать какие-либо аргументы для ``echo'` как опции. Каждый аргумент отображается после преобразования управляющих символов.

Существует другое поведение POSIX, которое Bash не реализует по умолчанию, даже когда находится в режиме POSIX. В частности:

1. Встроенный ``fc'` проверяет ``$EDITOR'` как программу для редактирования записей истории, если ``FCEDIT'` не установлен, вместо того, чтобы по умолчанию переходить непосредственно к ``ed'`. ``fc'` использует ``ed'`, если значение ``EDITOR'` не задано.

2. Как отмечалось выше, Bash требует, чтобы опция ``xpg_echo'` была включена, чтобы встроенная функция ``echo'` была полностью совместимой.

Bash можно настроить на соответствие POSIX по умолчанию, указав `--enable-strict-posix-default` на `configure` при сборке.

 **Fix Me!** помогите мне выяснить, что происходит в режиме POSIX®!

Режим POSIX® может быть включен с помощью:

- Bash, начинающийся как `sh` (базовое `argv[0]` имя `is sh`)
- запуск Bash с помощью опции командной строки `--posix`
- при запуске устанавливается переменная среды `POSIXLY_CORRECT`
- команда `set -o posix`

Тесты для режима POSIX®:

- переменная `SHELLOPTS` содержит `posix` в ее списке

Ограниченная оболочка

В ограниченном режиме Bash настраивает (и запускает) среду оболочки, которая намного более контролируема и ограничена, чем стандартный режим оболочки. Он действует как обычный Bash со следующими ограничениями:

- `cd` команда не может использоваться для изменения каталогов
- переменные `SHELL`, `PATH`, `ENV` и `BASH_ENV` не могут быть установлены или отменены
- имена команд, содержащие `/` (косую черту), не могут быть вызваны (следовательно, вы ограничены `PATH`)
- имена файлов, содержащие `/` (косую черту), не могут быть указаны в качестве аргумента `source` . встроенной команды `or`
- имена файлов, содержащие `/` (косую черту), не могут быть указаны в качестве аргумента `-p` опции `hash` встроенной команды
- определения функций не наследуются из среды при запуске оболочки
- переменная среды `SHELLOPTS` игнорируется при запуске
- перенаправление вывода с использованием операторов `>`, `>|`, `<>`, `>&`, `&>`, и `>>` перенаправления запрещено
- `exes` встроенная команда не может заменить оболочку другим процессом
- добавление или удаление встроенных команд с параметрами `-f` и `-d` в команду `enable builtin` запрещено
- использование `enable` встроенной команды для включения отключенных встроенных оболочек не работает
- `-p` опция `command` встроенной команды не работает
- отключение ограниченного режима с `set +r` помощью или `set +o restricted` (конечно) запрещено

Ограничения "-r" включаются **после** того, как Bash прочитал свои файлы запуска.

Когда запускаемая команда является сценарием оболочки, ограничения **отключаются** для (вспомогательной) оболочки, которая запускает этот сценарий оболочки.

Оболочка с ограниченным доступом может быть включена:

- вызов Bash как `rbash` (базовое `argv[0]` имя `is rbash`)
- вызов Bash с `-r` опцией
- вызов Bash с `--restricted` опцией

Тесты для ограниченного режима:

- специальный параметр `$-` содержит букву `r` (строчная буква `R`)
- параметр оболочки `restricted_shell` установлен и может быть проверен `shopt` встроенной командой

Обсуждение

Ed, 2012/08/23 09:30 ()

[цитата] Методы проверки режима оболочки входа в систему:

```
установлен параметр оболочки login_shell[/quote]
```

Ну, это говорит mw о ТОМ, ЧТО установлено, но не О том, КАК это проверить.

Это то, что я на самом деле пришел сюда, чтобы выяснить!

Ян Шампера, 15.02.2012 17:18 ()

```
-q shopt если login_shell; тогда  
...
```

`shopt` устанавливает статус выхода 0/1 в соответствии с `on / off`

Леонардо Д'Иаз, 2012/09/20 14:51 ()

Файл `/etc/profile` Debian также является источником `/etc/bash.bashrc` и `/etc/profile.d/*.sh`

`/etc/bash.bashrc` выполняет некоторую магию `PS1`, устанавливает `bash_completion(ы)` и `command-not-found` (предлагать пакеты)

Этот сайт поддерживается Performing Databases - вашими экспертами по администрированию баз данных

Bash Hackers Wiki



Если не указано иное, контент на этой вики лицензируется по следующей лицензии:
Лицензия на бесплатную документацию GNU 1.3