

 незавершенная работа...

Синтаксический анализ и выполнение

Почти все в грамматике Bash можно свести к "простой команде". Единственное, что Bash должен развернуть, оценить и выполнить, - это простая команда.

Простое расширение команды

- <http://lists.gnu.org/archive/html/bug-bash/2013-01/msg00040.html>
(<http://lists.gnu.org/archive/html/bug-bash/2013-01/msg00040.html>)
- <http://lists.research.att.com/pipermail/ast-developers/2013q2/002456.html>
(<http://lists.research.att.com/pipermail/ast-developers/2013q2/002456.html>)

Этот шаг выполняется после первоначального разделения командной строки.

Расширение простой команды выполняется в четыре этапа (интерпретация простой команды **слева направо**):.

1. Слова, помеченные синтаксическим анализатором как **присваивания переменных** и **перенаправления**, сохраняются для последующей обработки.
 - присвоения переменных предшествуют имени команды и имеют вид `WORD=WORD`
 - перенаправления могут появляться в любом месте простой команды
2. Остальные слова расширены. Если после расширения остаются какие-либо слова, первое слово принимается за **название команды**, а остальные слова являются **аргументами**.
3. Выполняются перенаправления.
4. Текст после `=` присвоения переменной `in` каждой переменной подвергается расширению тильды, расширению параметра, замене команд, арифметическому расширению и удалению кавычек, прежде чем быть присвоенным переменной.

Если после расширения **имя команды не** отображается:

- Присвоения переменных влияют на **текущую среду оболочки**.
 - Это то, что происходит, когда вы вводите только назначение переменной в командной строке.

- Присвоение переменным, доступным только для чтения, вызывает ошибку, и команда завершается с ненулевым значением.
- Перенаправления выполняются, но не влияют на текущую среду оболочки.
 - это означает, что `a > FILE` без какой-либо команды **будет** выполнено: `FILE` будет создано!
- Команда завершает работу
 - с кодом выхода, указывающим на ошибку перенаправления, если таковая имеется
 - с кодом выхода последней разобранной команды-подстановки, если таковой имеется
 - с кодом выхода 0 (ноль), если ошибка перенаправления не произошла и замена команды не была выполнена

В противном случае, если имя команды приводит:

- Сохраненные и проанализированные переменные добавляются в среду выполняемой команды (и, таким образом, не влияют на текущую среду).
 - Присвоение переменным, доступным только для чтения, вызывает ошибку, и команда завершается с ненулевым кодом ошибки.
 - **Ошибки присваивания** в режимах, отличных от *POSIX*, приводят к полному завершению входящих команд (например, циклов)
 - **Ошибки назначения** в (неинтерактивном) режиме *POSIX* приводят к завершению всего сценария

Поведение в отношении ошибок присвоения переменных может быть протестировано:

<http://lists.gnu.org/archive/html/bug-bash/2013-01/msg00054.html>
(<http://lists.gnu.org/archive/html/bug-bash/2013-01/msg00054.html>)

Этот полностью завершает сценарий

```
#!/bin/sh
# Эта оболочка работает в режиме POSIX!

эхо ПРЕ

# Ниже приведено присвоение ошибки, так как нет цифры "9"
# для базового восьмеричного числа!
foo=$((8#9))

эхо - СООБЩЕНИЕ
```

Этот завершает только заключающую составную команду (the `{ _;_ }`).):

```
#!/bin/bash
# Эта оболочка работает в собственном режиме Bash!

эхо ПРЕ

# Следующее - ошибка назначения!
# "Эхо-ТЕСТ" не будет выполнен, так как {...; } завершается
{ фу=$((8#9)); эхо-ТЕСТ; }


эхо-СООБЩЕНИЕ
```

Простое выполнение команды

Если проанализированная простая команда не содержит косых черт, оболочка пытается найти и выполнить ее:

- функции оболочки
- встроенные команды оболочки
- проверьте собственную хэш-таблицу
- ищите вместе PATH

Начиная с версии Bash 4, при сбое поиска команды оболочка выполняет функцию оболочки с именем `command_not_found_handle()`, использующим сбойную команду в качестве аргументов. Это может быть использовано для предоставления удобных для пользователя сообщений, установки пакетов программного обеспечения и т.д. Поскольку эта функция выполняется в отдельной среде выполнения, вы не можете реально повлиять на основную оболочку с ее помощью (изменение каталога, настройка переменных).

 продолжение следует

См . также

- Внутренний: Перенаправление
- Внутренний: Введение в расширения и замены

Обсуждение

Rodolfo (<http://www.facebook.com/profile.php?id=100003418804641>), 2013/03/21 08:39
()

Какая радость найти такую ясную мысль. Спасибо за сообщение!

Oleg Dunauskas, 2015/02/08 16:43 ()

Вы написали: → Текст после = в каждом присваивании переменной подвергается расширению тильды, расширению параметров, замене команд, арифметическому расширению и удалению кавычек, прежде чем быть присвоенным переменной. Что такое расширение пути?

Ян Шампера, [2015/02/08 17:37 \(\)](#)


Привет,

при назначении нет расширения имени пути. Пример:

```
> foo=*  
> echo "$foo" # обратите внимание на кавычки, в противном случае '*' был бы расширен здесь (но все же, не выше при назначении)  
*
```

Oleg Dunauskas, [2015/02/09 15:24 \(\)](#)

Спасибо за объяснение.

 [syntax/grammar/parser_exec.txt](#)  Последнее изменение: 2019/10/31 16:18 автор : эрсен

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

Bash Хакеры Вики



Except where otherwise noted, content on this wiki is licensed under the following license:
GNU Free Documentation License 1.3