

[Перейти к основному контенту](#)   Поиск в r / bash[Log In](#) **r/bash** • 3 года назад
kevors

Инструмент для обнаружения непреднамеренного затенения переменных в вашем коде bash

[отправка](#)

Привет, ребята. Я писал сложный скрипт и столкнулся с некоторыми проблемами при передаче переменных по имени в качестве аргументов функций. Проблема заключалась в непреднамеренном затенении переменных.

Вот пример. Давайте создадим функцию с тремя аргументами. Она должна суммировать `$2+$3` и присвоить результат переменной с именем `$1`. Я знаю, что приведенный ниже код не является оптимальным: это такой способ продемонстрировать проблему.

```
sum2 () {
```

[Подробнее](#) 

3



10

[Поделиться](#)[Добавить комментарий](#)Сортировать по: [Лучшие](#)   [Комментарии к поиску](#)**QliXeD** • 3 года назад

Краткое описание: не используйте `* sh` для сложных скриптов, используйте `python` / `perl` / и др.

Для сложных скриптов и скриптов из сотен строк с большим количеством функций не рекомендуется использовать `bash` / `sh` / etc. Вам следует использовать `python` / `perl` / `other-real-script-lang`. Я не хочу унижать `bash` / `sh`, но возможности сценариев более ограничены из-за подобных вещей. Также проблемой может быть использование производительности / памяти и стабильность оболочки после выполнения.

Рад, что вы нашли взлом. Но все же это взлом, который при других обстоятельствах может дать сбой.

Также не уверен, что вы пишете, но проверьте такие вещи, как `ansible` / `puppet`, на наличие альтернатив, которые намного упростят проблему, которую вы хотите решить.

Возможно, это не тот ответ, который вы хотите услышать, но я думаю, что, вероятно, это тот, который вам нужно услышать.

[Перейти к основному контенту](#)[Log In](#)

Краткое описание: не используйте `* sh` для сложных скриптов, используйте `python / perl /` и др.

Вы можете смеяться, но я переношу его с некоторых из них на `bash`. Я хочу, чтобы он работал без каких-либо ограничений.

Также не уверен, что вы пишете, но проверьте такие вещи, как `ansible / puppet`, на наличие альтернатив, которые намного упростят проблему, которую вы хотите решить.

Это чистая оболочка, никакие внешние двоичные файлы не используются в более чем 1k строках кода.

↑ 3 ↓ [Reply](#) [Поделиться](#) ...

1 еще один ответ



bigfig · 3 года назад

Ограничьте область действия переменных функциями, где это возможно, и объявляйте константы как доступные только для чтения. Вы даже можете подделать область действия блока следующим образом:

```
объявите foo='a'
echo "$ foo"

_(){
  unset -f _
  объявите foo='v'
  echo "$ foo"
};_

echo "$ foo"
```

⊖ ↑ 0 ↓ [Reply](#) [Поделиться](#) ...



kevors OP · 3 года назад

Ваш пример кода противоположен тому, что я пытаюсь выполнить. Если я хочу передать `var` по его имени функции, важно НЕ затенять его локальным `var`, объявленным в функции. `VARR` обнаруживает затенение.

↑ 1 ↓ [Reply](#) [Поделиться](#) ...

2 еще ответы



whetu · 3 года назад

[Перейти к основному контенту](#)[Log In](#)

переменных: это может быть очень неприятная ошибка, которую нужно выяснить.

За свою карьеру мне приходилось писать сценарии, которые можно было использовать в последней версии `bash`, или ее можно было использовать в оболочке, которая не поддерживает `local`. Простое решение, которое я придумал для этого сценария, - просто добавить переменные, которые я хочу сделать "локальными" переменными.

То есть: я не использую следующий подход постоянно, только когда мне это нужно. Это выглядит так:

```
$ SUM # Это переменная среды / глобальная переменная
$ sum # Это переменная уровня скрипта
$ _sum # Это локальная переменная
```

Когда я использую этот подход, я также стараюсь `unset` добавлять любые "локальные" переменные в конце функции.

Такой простой подход с привычными программными областями, как этот, не сработает для вашего сценария? Я неправильно понимаю проблему?

1 [Reply](#) [Поделиться](#) ...



kevors OP • 3 года назад • Отредактировано 3 года назад

[Перейти к основному контенту](#)[Log In](#)

Но функции могут вызывать функции. Ваш `_sum` var в некоторой функции не отличается от локального var с тем же именем в вызываемой им функции.

Я неправильно понимаю проблему?

Позвольте мне привести другой пример. `count_zero` это функция для подсчета количества нулевых элементов в массиве с именем `$2` и сохранения результата в переменной с именем `$1`.

```
count_zero () {
  [[ $ 1 == результат ]] || {
    результат local -n
    результат = $ 1
  }

  [[ $ 2 == список ]] || {
    local -n list
    list= $ 2
  }

  local el n # <=== 'n' local to 'count_zero'
  n= 0

  для el в "$ {list[@]}"; выполните
  ! ((el == 0)) || ((++ n))
  Выполнено

  результат = $ n
}

main () {
  local -список=(1 0 2 3 0 4 5 6)
  local n # <=== 'n' local to 'main'

  count_zero n список
  объявить -p n
}

Главная
```


Вывод:


объявить -- n

Если я использую `m` вместо `n` in `main()`, вывод становится

[Перейти к основному контенту](#)[Log In](#)

Проблема в том, `count_zero` что он делает не только (устанавливает значение переменной с именем \$ 1 в число нулевых значений в массиве с именем \$ 2), но также в том, какие локальные имена переменных он объявляет внутренне, что не должно иметь значения (поскольку попытка еще один ответровку с помощью `setopt setopt` срабатывает один раз, а затем зависит после это всего лишь значение переменной области видимости, объявленное в этой функции), но это имеет значение! В данном конкретном случае произойдет сбой, если его первый (или второй) аргумент будет либо `n`, либо `el`. `0 result` и `list` локальных переменных не может быть и речи, поскольку они объявлены с учетом затенения.

 [r/learnjavascript](#)
Представьте, что вы написали такую функцию, которая работает с переменными, переданными по Рефакторинг стороннего Javascript для упрощения отладки имени давным-давно. В ее документе говорится, что установите переменную с именем \$ 1 на количество нулевых элементов в массиве с именем \$ 2. Итак, вы думаете "количество элементов? Позвольте мне назвать переменную `n`". Запустите его, и он не работает.

 [r / swift](#)
С помощью VARR вы можете начать `count_zero` с `varr "$1" "$2"` запрета локальных переменных с именами `$1`, `$2` и это решит проблему за вас:

7 положительные отзывы · 5 Комментарии

 [r/P](#) `varr на 18: 'n' может быть затенен; цепочка вызовов: main > count_zero`

У кого-нибудь есть советы по написанию анализатора рекурсивного спуска для AST? [JS]
В примере я следовал правилам VARR для защищенных функций (локальные переменные объявляются с помощью `local` инструкции, никаких присваиваний в объявлениях, только статические имена), так что единственная строка (помимо поиска `varr.sh`) - это единственное изменение, позволяющее обнаружить проблему.

Is this solution understandable ? (Day 4 - Javascript)

3 upvotes · 8 comments

[r/PowerBI](#)

How to loop call a web API - Need Help

10 upvotes · 11 comments

[r/golang](#)

Custom error capable of wrapping another error?

4 upvotes · 3 comments

[r/neovim](#)

[Help needed] SegFault Error because of lua async? And how to detect if a buffer is a file.

3 upvotes · 4 comments

TOP POSTS



reReddit: Top posts of July 3, 2021



Log In



Reddit

reReddit: Top posts of 2021