Как получить ввод диалогового окна, направленного на переменную?

Я преподавал скрипты bash и столкнулся с проблемой. Я написал сценарий для ввода ввода от пользователя, используя команду «прочитать», и сделаю для этого ввод переменной, которая будет использоваться позже в скрипте. Скрипт работает, но ....

Я бы хотел, чтобы его можно было настроить с помощью «диалога». я узнал что

- BASH-копия в другой каталог, не работающий
- работа crontab не работает
- Подсистема Windows для Linux: / home / user (путь к корневому каталогу Linux) для использования в терминале ConEmu?
- Как автоматически отвечать на интерактивную программу cli (а не на скрипт) с помощью скрипта bash?
- Как настроить приглашение на использование двух цветов?

'dialog –inputbox' будет направлять вывод в 'stderr', и для того, чтобы получить этот вход в качестве переменной, вы должны направить его в файл и затем извлечь его. Код, который я нашел, чтобы объяснить это:

```
#!/bin/bash dialog --inputbox \ "What is your username?" 0 0 2> /tmp/inputbox.tmp.$$ retval=$? input=`cat /tmp/inputbox.tmp.$$` rm -f /tmp/inputbox.tmp.$$ case $retval in 0) echo "Your username is '$input'";; 1) echo "Cancel pressed.";; esac
```

Я вижу, что он отправляет sdterr в /tmp/inputbox.tmp.$$ с помощью 2>, но выходной файл выглядит как 'inputbox.tmp.21661'. Когда я пытаюсь загрузить файл, он дает мне ошибку. Поэтому я все еще не могу получить пользовательский ввод из -inputbox в качестве переменной.

Пример скрипта:

```
 echo " What app would you like to remove? " read dead_app sudo apt-get remove --purge $dead_app
```

Таким образом, вы можете видеть, что это простой скрипт. Возможно ли получить переменную как слово из `dialog --inputbox` ?

Related of "Как получить ввод диалогового окна, направленного на переменную?"

- Невозможно использовать файлы со специальными символами в оболочке
- Поместите все команды sed в один файл сценария оболочки
- Почему «if » выполняет оператор «then» в сценарии bash?
- regex search работает с `grep -E`, но не с bash-скриптом?

- Удалить ведущее или трейлинг-пространство (ы) в именах файлов или папок

: DI не может это объяснить !!! Если вы можете понять, что они говорят в Advanced Bash-Scripting Guide: Глава 20. Перенаправление ввода-вывода , напишите новый ответ, и я дам вам *50pp* :

```
 exec 3>&1; result=$(dialog --inputbox test 0 0 2>&1 1>&3); exitcode=$?; exec 3>&-;
echo $result $exitcode;
```

Ссылка: Диалог в bash не захватывает переменные правильно

^ ответ от @Sneetsher (4 июля 2014 г.)

В соответствии с запросом, я попытаюсь объяснить, что делает этот фрагмент, построчно.

Обратите внимание, что я упрощу это, опуская все `;` точки с запятой в конце строки, потому что они не нужны, если мы пишем одну команду на строку.

I / O – Потоки:

Во-первых, вам нужно понять потоки коммуникации. Есть 10 потоков, пронумерованных от 0 до 9:

- **Поток 0 («STDIN»):**
  «Стандартный вход» – входной поток по умолчанию для чтения данных с клавиатуры.
- **Поток 1 («STDOUT»):**
  «Стандартный вывод» – выходной поток по умолчанию, используемый для отображения обычного текста в терминале.
- **Stream 2 («STDERR»):** «Стандартная ошибка», выходной поток по умолчанию, используемый для отображения ошибок или другого текста для специальных целей в терминале.
- **Потоки 3-9:**
  Дополнительные, свободно используемые потоки. Они не используются по умолчанию и не существуют, пока что-то не попытается их использовать.

Обратите внимание, что все «потоки» внутренне представлены файловыми дескрипторами в `/dev/fd` (что является символической ссылкой на `/proc/self/fd` которая содержит еще одну символическую ссылку для каждого потока ... это немного сложная и не важная для их поведение, поэтому я останавливаюсь здесь.). Стандартные потоки также имеют `/dev/stdin` , `/dev/stdout` и `/dev/stderr` (которые снова символические ссылки и т. Д.).

Сценарий:

- `exec 3>&1`
  Встроенный `exec` Bash может использоваться для применения перенаправления потока к оболочке, что означает, что он влияет на все следующие команды. Для получения дополнительной информации запустите `help exec` в своем терминале.

  В этом специальном случае поток 3 перенаправляется в поток 1 (STDOUT), что означает, что все, что мы отправляем в поток 3 позже, появится в нашем терминале, как если бы оно обычно печаталось в STDOUT.

- `result=$(dialog --inputbox test 0 0 2>&1 1>&3)`
  Эта строка состоит из множества частей и синтаксических структур:

  - `result=$(...)`
    Эта структура выполняет команду в скобках и назначает вывод (STDOUT) на `result` переменной bash. Он читается через `$result` . Все это описывается как-то в облике `man bash` .
  - `dialog --inputbox TEXT HEIGHT WIDTH`
    Эта команда показывает окно TUI с заданным текстом, поле ввода текста и две кнопки OK и CANCEL. Если выбрано OK, команда выходит со статусом 0 и печатает введенный текст в STDERR, если CANCEL будет выбран, он выйдет с кодом 1 и ничего не напечатает. Для получения дополнительной информации прочитайте `man dialog` .
  - `2>&1 1>&3`
    Это две команды перенаправления. Они будут интерпретироваться справа налево:

    `1>&3` перенаправляет поток 1 команды (STDOUT) в пользовательский поток 3.

    `2>&1` перенаправляет поток команд 2 (STDERR) в поток 1 (STDOUT).

    Это означает, что все, что команда печатает в STDOUT, теперь появляется в потоке 3, а все, что предназначалось для отображения на STDERR, теперь перенаправляется на STDOUT.

  Таким образом, вся строка отображает текстовое приглашение (в STDOUT, которое было перенаправлено на поток 3, который оболочка снова перенаправляет обратно в STDOUT в конце – см. `exec 3>&1`) и назначает введенные данные (возвращается через STDERR, затем перенаправляется на STDOUT) на `result` переменной Bash.

- `exitcode=$?`
  Этот код извлекает код выхода ранее выполненной команды (здесь из `dialog`) через зарезервированную переменную Bash `$?` (всегда держит последний код выхода) и просто сохраняет его в нашем собственном `exitcode` переменном `exitcode` . Его можно снова прочитать

через `$exitcode` . Вы можете найти дополнительную информацию об этом в `man bash` , но это может занять некоторое время ...

- `exec 3>&-`
  Встроенный `exec` Bash может использоваться для применения перенаправления потока к оболочке, что означает, что он влияет на все следующие команды. Для получения дополнительной информации запустите `help exec` в своем терминале.

  В этом специальном случае поток 3 перенаправляется на «stream -», что означает, что он должен быть закрыт. Данные, отправленные в поток 3, больше не будут перенаправлены нигде с этого момента.

- `echo $result $exitcode`
  Эта простая команда `echo` (более подробная информация о `man echo`) просто печатает содержимое двух переменных Bash `result` и `exitcode` в STDOUT. Поскольку здесь нет явных или неявных переадресаций потоков, они действительно появятся на STDOUT и поэтому просто будут отображаться в терминале. Какое чудо! ☺

Резюме:

Во-первых, мы устанавливаем оболочку для перенаправления всего, что мы отправляем в пользовательский поток 3 обратно в STDOUT, чтобы он появился в нашем терминале.

Затем мы запускаем команду `dialog` , перенаправляем ее оригинальный STDOUT на наш пользовательский поток 3, потому что он должен быть отображен в конце, но нам временно нужно использовать поток STDOUT для чего-то другого.
Мы перенаправляем исходный STDERR команды, откуда возвращается пользовательский ввод диалогового окна, затем STDOUT.
Теперь мы можем захватить STDOUT (который содержит перенаправленные данные из STDERR) и сохранить его в нашем переменном `$result` . Он содержит желаемый пользовательский ввод сейчас!

Нам также нужен код выхода команды `dialog` , который показывает нам, было ли нажато OK или CANCEL. Это значение представлено в зарезервированной переменной Bash `$?` и мы просто скопируем его в нашу переменную `$exitcode` .

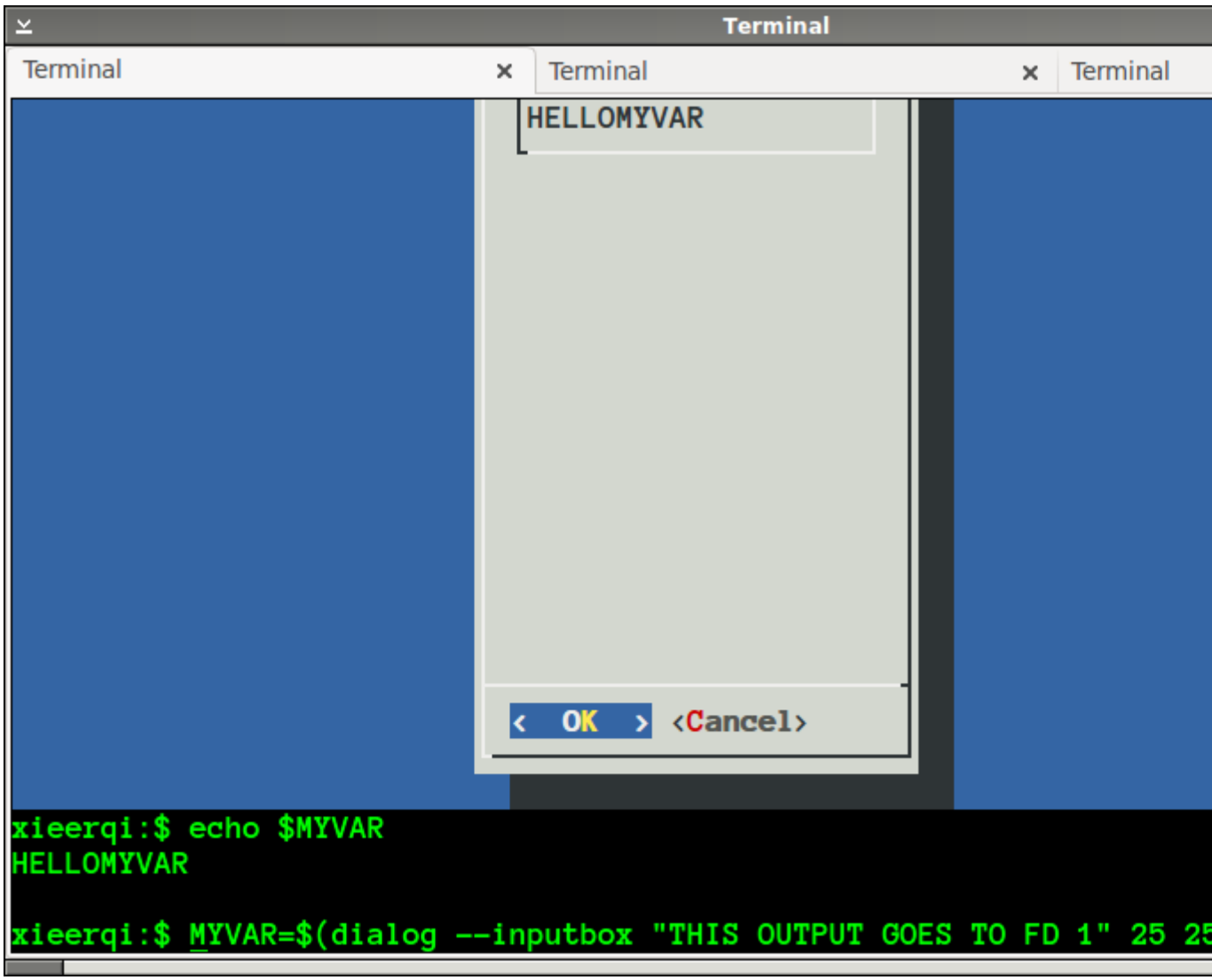
После этого мы снова закрываем поток 3, поскольку нам это больше не нужно, чтобы остановить дальнейшие перенаправления.

Наконец, мы обычно `$exitcode` на терминал содержимое обеих переменных `$result`(пользовательский ввод диалогового окна) и `$exitcode` (0 для OK, 1 для CANCEL).

Если вы читаете справочную страницу для диалога, есть опция `--output-fd` , которая позволяет явно указывать, где идет выход (STDOUT 1, STDERR 2), а не по умолчанию – STDERR.

Ниже вы можете увидеть, как я запускаю команду образца `dialog` , явно указывая, что вывод должен перейти к файловому дескриптору 1, что позволяет мне сохранить его в MYVAR.

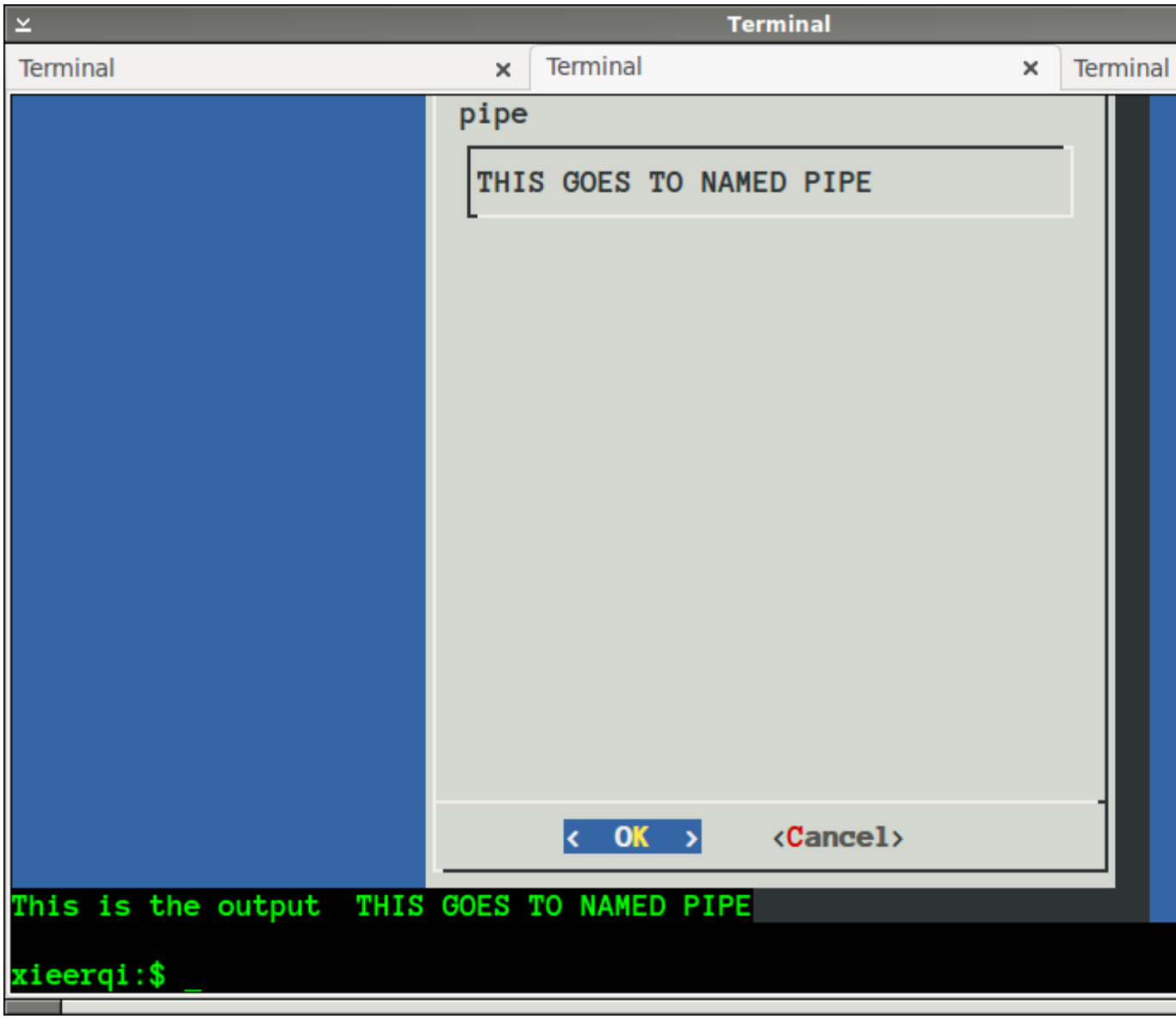
```
MYVAR=$(dialog --inputbox "THIS OUTPUT GOES TO FD 1" 25 25 --output-fd 1)
```



Альтернативный подход, который имеет много скрытого потенциала, заключается в использовании чего-то известного как именованный канал .

xieerqi: $ cat dialog_with_npipe.sh

```
 #!/bin/bash mkfifo /tmp/namedPipe1 # this creates named pipe, aka fifo dialog --
inputbox "This is an input box with named pipe" 40 40 2> /tmp/namedPipe1 & # to make
sure the shell doesn't hang, we run redirection in background, because fifo waits for
output to come out OUTPUT="$( cat /tmp/namedPipe1 )" # release contents of pipe echo
"This is the output " $OUTPUT
```



~~: DI не может это объяснить !!! Если вы можете понять, что они говорят в ссылке: Advanced Bash-Scripting Guide: Chapter 20. Перенаправление ввода-вывода , напишите новый ответ, и я дам вам *50pp*~~

Была дана Баунти, для объяснения см . Ответ ByteCommander . □ Это часть истории.

```
 exec 3>&1; result=$(dialog --inputbox test 0 0 2>&1 1>&3); exitcode=$?; exec 3>&-;
echo $result $exitcode;
```

Источник: Диалог в bash не правильно захватывает переменные
Ссылки: Расширенное руководство по созданию Bash-скриптов: Глава 20. Перенаправление ввода-вывода

Если кто-то еще приземлился здесь из Google, и хотя этот вопрос задается специально для bash, вот еще одна альтернатива:

Вы можете использовать зенит . Zenity – *графическая* утилита, которая **может**использоваться внутри сценариев bash. Но, конечно, для этого потребуется X-сервер, как указано в user877329.

```
sudo apt-get install zenity
```
Затем в вашем скрипте:

```
RETVAL=`zenity --entry --title="Hi" --text="What is your username"`
```
Полезная ссылка .

Ответ, предоставленный Sneetsher, несколько более изящный, но я могу объяснить, что не так: значение `$$` отличается внутри backticks (потому что оно запускает новую оболочку, а `$$` – это PID текущей оболочки). Вы захотите поместить имя файла в переменную, а затем вместо этого ссылаться на эту переменную.

```
#!/bin/bash t=$(mktemp -t inputbox.XXXXXXXXX) || exit trap 'rm -f "$t"' EXIT # remove temp file when done trap 'exit 127' HUP STOP TERM # remove if interrupted, too dialog --inputbox \ "What is your username?" 0 0 2>"$t" retval=$? input=$(cat "$t") # Prefer $(...) over `...` case $retval in 0) echo "Your username is '$input'";; 1) echo "Cancel pressed.";; esac
```
В этом случае избежать временного файла было бы лучшим решением, но будет много ситуаций, когда вы не сможете избежать временного файла.

Это работает для меня:

```
#!/bin/bash input=$(dialog --stdout --inputbox "What is your username?" 0 0) retval=$? case $retval in ${DIALOG_OK-0}) echo "Your username is '$input'.";; ${DIALOG_CANCEL-1}) echo "Cancel pressed.";; ${DIALOG_ESC-255}) echo "Esc pressed.";; ${DIALOG_ERROR-255}) echo "Dialog error";; *) echo "Unknown error $retval" esac
```
На странице руководства `dialog` говорится о –stdout:

Прямой вывод на стандартный вывод. Этот параметр предоставляется для совместимости с Xdialog, однако его использование в переносных сценариях не рекомендуется, так как curses обычно записывает свои обновления экрана в стандартный вывод. Если вы используете эту опцию, диалог пытается открыть терминал, чтобы он мог писать на дисплей. В зависимости от платформы и вашей среды это может закончиться неудачей.

Может ли кто-нибудь сказать, в какой платформе или среде он не работает? Вызывает ли перенаправление вывода `dialog` на `2>&1 >/dev/tty` лучше?

- Как создать совпадение для регулярного выражения?
- Как увидеть команду, связанную с псевдонимом bash?
- / usr / bin / env: 'python3 \ r': Нет такого файла или каталога
- ping несколько IP с помощью bash?
- Выполнение команды в новом процессе оболочки bash
- Wget как утилита для загрузки всех изображений в каталог mysite.com/img/
- для массива символов с чередованием дает неожиданный "("
- Как сохранить длину массива в переменной на языке сценариев bash
- Bash one-liner для проверки версии> =
- Как изменить цвета в подсказке bash
- bash: [: слишком много аргументов?
- Как перемещать сразу несколько файлов в определенный целевой каталог?