Read and write data from anywhere with redirection in the Linux terminal

Redirection is an efficient way to get data from one place to another without a lot of mouse moving and key pressing.

30 Jun 2020 | Seth Kenlon (Red Hat) (/users/seth) | 42 | 5 comments

Image by : *WOCinTech Chat. Modified by Opensource.com. CC BY-SA 4.0*

Redirection of input and output is a natural function of any programming or scripting language. Technically, it happens inherently whenever you interact with a computer. Input gets read from **`stdin`** (standard input, usually your keyboard or mouse), output goes to **`stdout`** (standard output, a text or data stream), and errors get sent to **`stderr`**. Understanding that these data streams exist enables you to control where information goes when you're using a shell, such as Bash (/resources/what-bash) or Zsh (/article/19/9/getting-started-zsh).

Standard in, standard out, and standard error exist as filesystem locations on Linux. You can see them in **`/dev`**:

```
$ ls /dev/std*
/dev/stderr@  /dev/stdin@  /dev/stdout@
```

The basics of redirection are simple: use some number of `>` characters to redirect output, and some number of `<` characters to redirect input.

Redirecting output

To write the output of the ls (https://opensource.com/article/19/7/master-ls-command) command to a file:

```
$ ls > list.txt
```

You don't see the output of `ls` as you normally would, because the output is written to the `list.txt` file instead of your screen. This is so versatile, in fact, that you can even use it to copy the contents of one file to another. It doesn't have to be a text file, either. You can use redirection for binary data:

```
$ cat image.png > picture.png
```

(In case you're wondering why you'd ever want to do that, it's for a sometimes-useful repercussion on file permissions (https://opensource.com/article/19/8/linux-permissions-101).)

Redirecting input

You can redirect input "into" a command, too. This is arguably less useful than redirecting output because many commands are already hard-coded to take input from an argument you provide. It can be useful, however, when a command expects a list of arguments, and you have those arguments in a file and want to quickly "copy and paste" them from the file into your

```
$ sudo dnf install $(<package.list)
```

Common uses of input redirection are the **here-document** (or just **here-doc** for short) and **here-string** techniques. This input method redirects a block of text into the standard input stream, up to a special end-of-file marker (most people use `EOF`, but it can be any string you expect to be unique). Try typing this (up to the second instance of `EOF`) into a terminal:

```
$ echo << EOF
> foo
> bar
> baz
> EOF
```

The expected result:

```
foo
bar
baz
```

A **here-doc** is a common trick used by Bash (https://opensource.com/resources/what-bash) scripters to dump several lines of text into a file or onto the screen. As long as you don't forget to end the clause with your end-of-file marker, it's an effective way to avoid unwieldy lists of `echo` or `printf` statements.

A **here-string** is similar to a **here-doc**, but it consists of just one string (or several strings disguised as a single string with quotation marks):

Redirecting error messages

Error messages go to a stream called **stderr**, designated as **2>** for the purposes of redirection. This command directs error messages to a file called **output.log**:

```
$ ls /nope 2> output.log
```

Sending data to /dev/null

Just as there are locations for standard in, standard out, and error, there's also a location for *nowhere* on the Linux filesystem. It's called **null**, and it's located in **/dev**, so it's often pronounced "devnull" by people who use it too frequently to say "slash dev slash null."

You can send data to **/dev/null** using redirection. For instance, the **find**

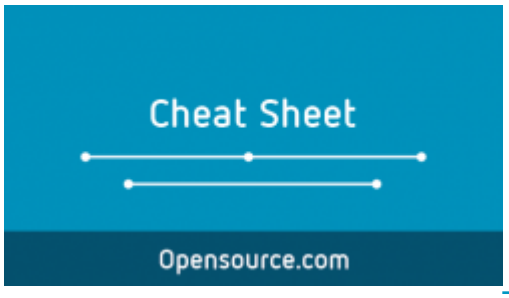
```
$ find ~ -type f
/home/seth/actual.file
find: `/home/seth/foggy': Permission denied
find: `/home/seth/groggy': Permission denied
find: `/home/seth/soggy': Permission denied
/home/seth/zzz.file
```

The `find` command processes that as an error, so you can redirect just the error messages to `/dev/null`:

```
$ find ~ -type f 2> /dev/null
/home/seth/actual.file
/home/seth/zzz.file
```

Using redirection

Redirection is an efficient way to get data from one place to another in Bash. You may not use redirection all the time, but learning to use it when you need it can save you a lot of needless opening files and copying and pasting data, all of which generally require mouse movement and lots of key presses. Don't resort to such extremes. Live the good life and use redirection.



(/article/19/11/bash-cheat-sheet)

Bash cheat sheet: Key combos and special syntax (/article

your computer.

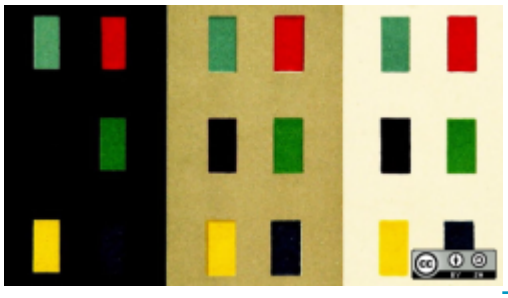
Seth Kenlon (Red Hat) (/users/seth)

(/article/20/4/bash-programming-guide)

Get started with Bash programming (/article/20/4/bash-programming-guide)

Learn how to write custom programs in Bash to automate your repetitive tasks. Download our new eBook to get started.

Seth Kenlon (Red Hat) (/users/seth)



(/article/19/12/ratpoison-linux-desktop)

Go mouseless with the Linux Ratpoison window manager (/article/19/12/ratpoison-linux-desktop)

This article is part of a special series of 24 days of Linux desktops. If you'd rather live in a terminal all day and avoid mousing around, the Ratpoison window manager is the solution for you.

Seth Kenlon (Red Hat) (/users/seth)

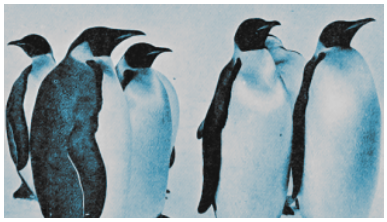
(/users/seth)

About the author

Seth Kenlon - Seth Kenlon is an independent multimedia artist, free culture advocate, and UNIX geek. He has worked in the film (http://www.imdb.com/name/nm1244992) and computing (http://people.redhat.com/skenlon) industry, often at the same time. He is one of the maintainers of the Slackware-based multimedia production project, http://slackermedia.info (http://slackermedia.info)

• More about me (/users/seth)

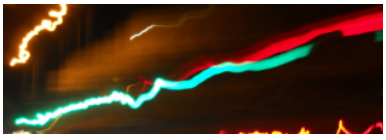
Recommended reading



Manage your SSH connections with this open source tool (/article/20/9/ssh-connection-manager?utm_campaign=intrel)

A beginner's guide to SSH for remote connection on Linux (/article/20/9/ssh?utm_campaign=intrel)

Program hardware from the Linux command line (/article/20/9/hardware-command-line?utm_campaign=intr



learning awk (/article/20/9/awk-ebook?utm_campaign=intr...)

traffic through your firewall (/article/20/9/firewall?utm_campaign=int...)

using tcpdump at the Linux command line (/article/18/10/introduction-tcpdump?utm_campaig...)

5 Comments

Andrejs on 02 Jul 2020 1

FYI: simple redirection like this will not work in all the cases. In particular, if you do output redirection of a script that spawns another shell process(-es). That's why I wonder why you haven't mentioned a tool called `script`: script -c "./my-script" log.txt

Seth Kenlon (/users/seth) on 02 Jul 2020 1

Keep checking back. There's a script article coming up as its own topic.

Abhishek Chaudhary (/users/theabbie) on 08 Jul 2020 2

Amazing post, loved it

Rajan Bhardwaj (/users/rajabhar) on 26 Jul 2020 0

Good one

DanielDaugs on 29 Jul 2020 0