Ubuntu Linux Fundamentals

Ubuntu Server - BASH Scripting - Controlling Flow - pipe, redirect, and tee

Linux Data Flow

When you're working at the command line, in Linux in a terminal window, there's a standard way for data to flow.

The three flows defined are standard input, standard output, and standard error.

Standard in is the keyboard.

Standard out is the terminal window or computer screen.

Standard error is also the terminal window or computer screen.

Here's how it works.

You type a command and hit Enter.

The command comes from Standard Input and is recognized and processed.

The results of normal processing are sent to Standard Output, or the screen.

The results of any errors encountered are also sent to Standard Output or the screen.

We'll call Standard Input stdin, Standard Output stdout, and Standard Error stderr for the rest of this lesson, and you'll often see them represented that way.

This behavior can be easily modified in Linux. You've done it already with the pipe command, which we'll go into further later in the lesson.

stdin, stdout, and stderr can be represented by the numbers 0, 1, and 2, respectively.

```
stdin  |  stdout  |  stderr
  0    |     1    |     2
```

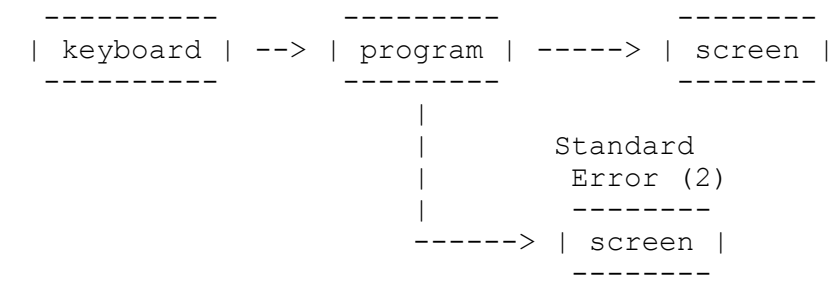
You'll see how this can be used later in the lesson.

Here's an "American Standard Code for Information Interchange (ascii) art" representation.

```
                Flow Control in Linux

Standard Input  = Keyboard
Standard Output = Screen
Standard Error  = Screen

   Standard                          Standard
    Input (0)                         Output (1)
```

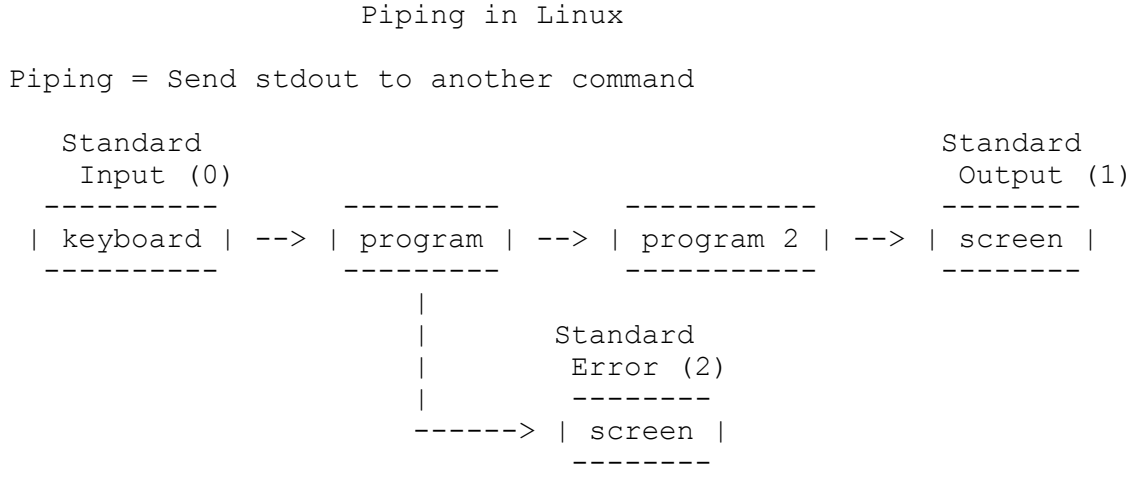


When representing data flow in Linux,

0 = Standard Input
1 = Standard Output
2 = Standard Error

Piping Output - |

When you pipe output, you take it from one command and feed it to the next.



stdin = Keyboard
stdout = Input for program 2
stderr = Print to screen

You can pipe multiple times.

Try this command:

```
cat /etc/group | grep <your username>
```

Substitute `<your username>` with the username you entered when installing.

Here are my results.

```
theo@ubuntu-server:~$ cat /etc/group | grep theo
adm:x:4:syslog,theo
cdrom:x:24:theo
sudo:x:27:theo,lskywalker
dip:x:30:theo
```

```
plugdev:x:46:theo
lxd:x:110:theo
theo:x:1000:
lpadmin:x:115:theo
sambashare:x:116:theo
```

Yours will likely be similar.

Now, pipe that to grep again, and grep for `cdrom`.

For me, that would be:

```
cat /etc/group | grep theo | grep cdrom
```

My output:

```
theo@ubuntu-server:~$ cat /etc/group | grep theo | grep cdrom
cdrom:x:24:theo
```

So, you can see the standard output was captured and redirected to grep, then the grep results were fed to grep again, this time looking for something different.

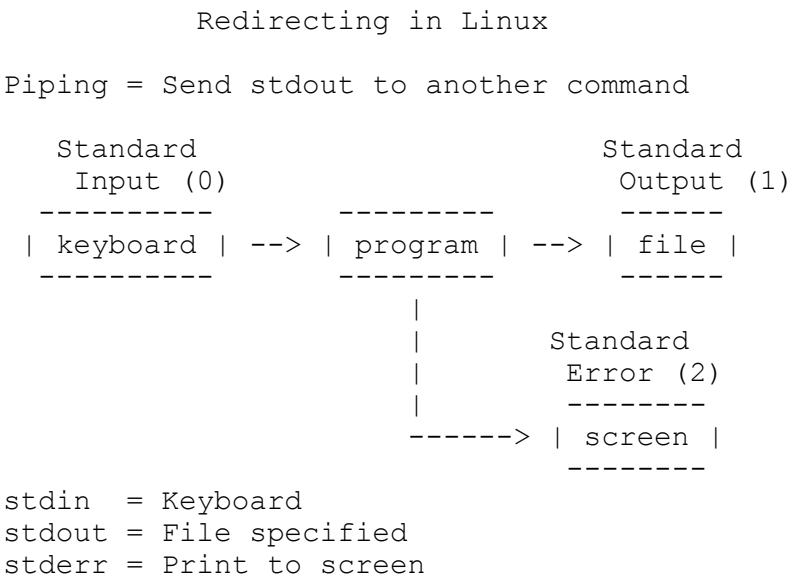
Redirecting Output - >, and >>

If, instead of sending output to another command, you want to send it to a file, you use the redirect symbol.

> will overwrite the contents of a file with the output from stdout.

>> will append stdout to the end of a file.

In either case, if the file doesn't exist, it will be created.



Try this:

```
ls /etc > file1
```

Have a look at the contents of file1.

```
cat file1
```

You can see the output of listing the contents of the `/etc` directory.

Now try:

```
ls /etc | grep ^p > file1
```

This lists the /etc directory, but uses grep to filter for only things starting with p, then overwrites the content of file1.

`cat file1` to check.

Now try:

```
ls /etc >> file1
```

Now, the normal contents of the ls command are written below the entries captured that begin with p in the last command.

Redirecting Errors

What if you know you'll get some errors, but you don't want to see them on the screen?

You can redirect `stderr` to a place where you won't see it.

A special device, `dev null`, is like a trash chute. Anything sent to `dev null` is quietly discarded.

Try the following:

```
find /var -user <your username>
```

You should get several errors.

Now, try this:

```
find /var -user <your username> 2> dev null
```

Nothing prints to screen because all of the errors were sent to `dev null` and discarded.

Reading A File Into stdin - <

You can have stdin come from a file instead of from the keyboard.

If you still have `file1` from the Redirecting Output portion of the lesson, please do the next steps. If not, please follow the steps above to re-create it then do the following:

```
sort -r < file1
```

The sort command will sort what it is fed from lowest to highest numerically and alphabetically. The `-r` option sorts in reverse order, or highest to lowest.

That's not super useful, and the same thing can be accomplished by just typing `sort -r file1`.

We'll then take that output and redirect it to a new file.

```
sort -r < file1 > file1-reverse-sort
```

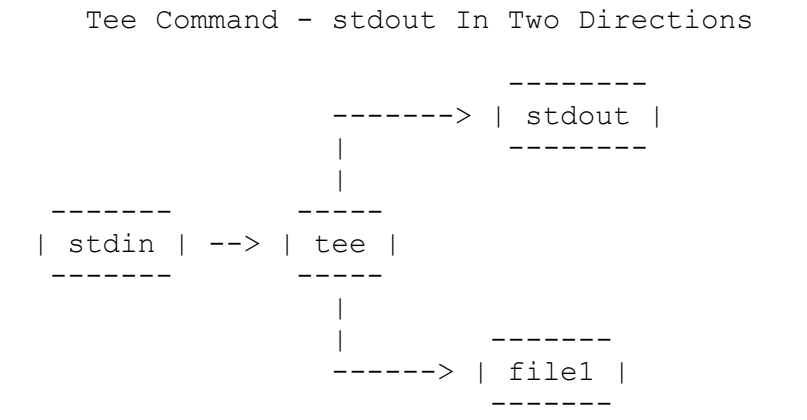
If you `cat file1-reverse-sort`, you'll see that it's reverse sorted.

tee

What if you want to send output to two places at once?

That's what the `tee` command does.

Like a t-fitting for a pipe, it takes what comes in and sends it in two directions, kind of down two different pipes.




```
stdout and file1 are just examples.
The output can be directed any way you choose.
```

Type:

```
ls /etc | tee file2
```

You'll see the output of the command, but if you `cat file2`, it will also be in the file.

Please practice with these commands and try some things yourself.

Great work getting through it!

More Information

Ryans Tutorials, Piping and Redirection
https://ryanstutorials.net/linuxtutorial/piping.php

LearnLinux.org tutorial on stdin, stdout, and stderr
http://www.learnlinux.org.za/courses/build/shell-scripting/ch01s04.html

Linux 101 Hacks, Tee Command Usage Examples
http://linux.101hacks.com/unix/tee-command-examples/