# **Redirection in Linux/Unix - Demystified!**

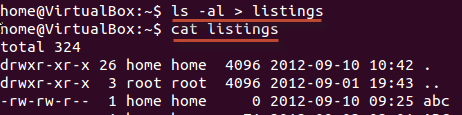
Most of the commands , we have learned so far, take an input and give an output.

* The standard input (stdin) device is the keyboard.
* The standard output (stdout) device is the screen.

Linux , is a very flexible system and you can change the standard input / output devices. Let's learn how this re-direction works

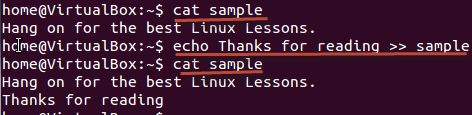
**Output Redirection**

|  |  |
| --- | --- |
| Redirection in Linux/Unix - Demystified! | The **'>**' symbol is used for output (STDOUT) redirection.  Example:  ***ls -al > listings***  Here the output of command ls -al is re-directed to file "listings" instead of your screen. |



**Note**: Use the right file name while redirection. If there is an existing file with the same name, it will be overwritten.

If you do not want a file to be overwritten but want to add more content to an existing file then you should use **'>>**' operator.



You can re-direct standard output, to not just files, but also devices!

***$ cat music.mp3 > /dev/audio***

The cat command reads the file music.mp3 and sends the output to /dev/audio which is the audio device. If the sound configurations in your PC are correct, this command will play the file music.mp3

## **Input redirection**

|  |  |
| --- | --- |
| Redirection in Linux/Unix - Demystified! | The **'<**' symbol is used for input(STDIN) redirection  Example: The mail program in Linux can help you send e-mails from the Terminal.  You can type the contents of the email using the standard device keyboard. But if you want to attach a File to email you can use the input re-direction operator in following format -  ***Mail -s "Subject" to-address < Filename*** |



This would attach the file with the mail and it would be sent to the recipient.

The above examples were simple. Let's look at some advance re-direction techniques which makes use of File Descriptors

## **File Descriptors**

In Linux/Unix everything is a file. Regular file, Directories and even Devices are files. Every File has an associated number called File Descriptor (FD).

Your screen also has a File Descriptor. When a program is executed the output is sent to File Descriptor of the screen and you see program output on your monitor. If the output is sent to File Descriptor of the printer, the program output would have been printed.

## **Error Redirection**

Whenever you execute a program/command at the terminal, 3 files are always open, viz., standard input, standard output, standard error**.**



These files are always present whenever a program is run. As explained before a file descriptor, is associated with each of these files

|  |  |
| --- | --- |
| **File** | **File Descriptor** |
| **Standard Input STDIN** | **FD0** |
| **Standard Output STDOUT** | **FD1** |
| **Standard Error STDERR** | **FD2** |

By default, error stream is displayed on the screen. Error redirection is routing the errors to a file other than the screen.

### **Why Error Redirection?**

Error re-direction is one of the very popular features of Unix/Linux.

Frequent UNIX users will reckon that many commands give you massive amounts of errors.

* For instance, while searching for files, one typically gets permission denied errors. These errors usually do not help the person searching for a particular file.
* While executing shell scripts, you often do NOT want error messages cluttering up the normal program output.

The solution is to re-direct the error messages to a file.

**Example 1**

**$ myprogram 2>errorsfile**

Redirection in Linux/Unix - Demystified!

Above we are executing a program names myprogram.

The file descriptor for standard error is 2.

Using "2>" we re-direct the error output to a file named "errorfile"

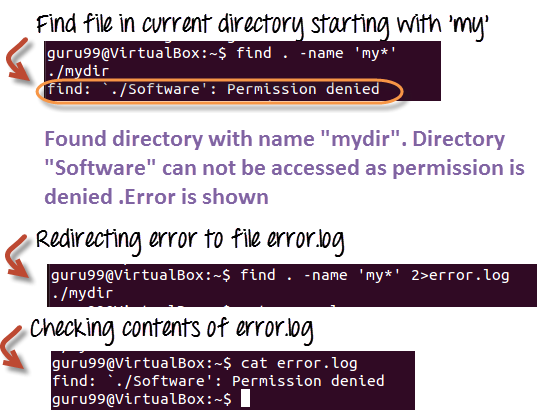
Thus, program output is not cluttered with errors.

**Example 2**

Here is another example which uses find statement -

***find . -name 'my\*' 2>error.log***

Using the "find" command we are searching the "." current directory for a file with "name" starting with "my"



**Example 3**

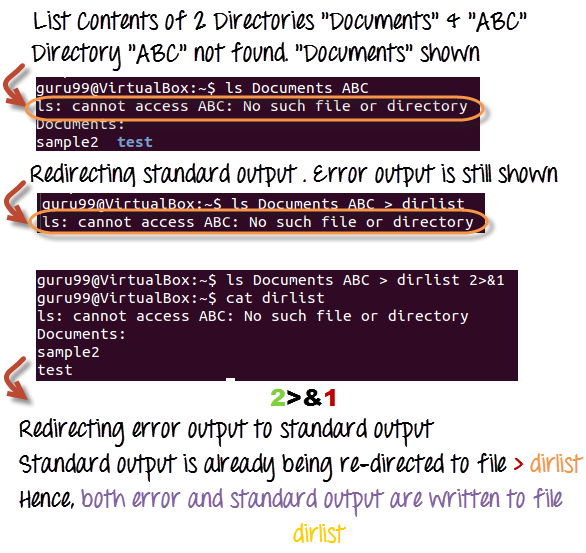
Let's see a more complex example,

Server Administrators frequently, list directories and store both error and standard output into a file, which can be processed later. Here is the command.

***ls Documents ABC> dirlist 2>&1***

Here,

* ">&" which writes the output from one file to the input of another file.
* We are redirecting error output to standard output which in turn is being re-directed to file dirlist. Hence , both the output is written to file dirlist



Summary

* Each file in Linux has a corresponding File Descriptor associated with it
* The keyboard is the standard input device while your screen is the standard output device
* ">" is the output redirection operator. ">>" appends output to an existing file
* "<" is the input redirection operator
* ">&"re-directs output of one file to another.
* You can re-direct error using its corresponding File Descriptor 2.