Input Output Redirection in Linux/Unix Examples

What is Redirection?

Redirection is a feature in Linux such that when executing a command, you can change the standard input/output devices. The basic workflow of any Linux command is that it takes an input and give an output.

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

With redirection, the above standard input/output can be changed.

Output Redirection

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 correct file name while redirecting command output to a file. If there is an existing file with the same name, the redirected command will delete the contents of that file and then it may 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 redirect 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

The **'<**' symbol is used for input(STDIN) redirection

Example: The mail program in Linux can help you send emails 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 the following format.

```
Mail -s "Subject" to-address < Filename
```

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

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

File Descriptors (FD)

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	**0**
Standard Output STDOUT	**1**
Standard Error STDERR	**2**

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
```

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.

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