NDG Linux Unhatched - NDG Linux Unhatched

Redirection

Adding content to files in Linux can be done in a variety of ways. Linux has a few text editors that can be used to add content to a file. However, this method requires some familiarity with Linux text editor commands.

Note

Linux text editors are covered in the next section of this course.

There is a way in Linux to quickly add content to a file using a command line feature called *input/output (I/O) redirection*. I/O redirection allows for information in the command line to be sent to files, devices, and other commands. The input or output of a command is redirected from its default destination to a different location. I/O redirection is like a series of train tracks, where a switch can be enabled to direct the output of a command on a different track so it goes somewhere else in the shell. In this section, we are writing to files by redirecting the output of a command to a file.

When it comes to command input and output there are three paths, or “tracks”. These paths are called *file descriptors*. The first file descriptor is *standard input*, abbreviated as STDIN. Standard input is the information the command receives and processes when it is executed, essentially what a user types on the keyboard. The second file descriptor is *standard output*, abbreviated as STDOUT. Standard output is the information that the command displays, the output of the command. The last file descriptor is *standard error*, abbreviated as STDERR. STDERR, are the error messages generated by commands that are not correctly executed. The following are examples of how file descriptors will appear in the terminal:

Standard Input (STDIN)

```
sysadmin@localhost:~$ ls ~/Documents
```

Standard Output (STDOUT)

```
sysadmin@localhost:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
```

Standard Error (STDERR)

```
sysadmin@localhost:~$ ls fakefile
ls: cannot access fakefile: No such file or directory
```

This section will cover one of the three file descriptors, STDOUT, and how to redirect STDOUT from where you normally see it, in the terminal, to a file in the filesystem. To use redirection, simply use a greater-than symbol `>` along with a filename:

```
[COMMAND] > [FILE]
```

To demonstrate redirection, we will use the output of the `cat` command. Without redirection, the output of the `cat` command will be displayed in the terminal:

Follow Along

Use the following command to switch to the `Documents` directory:

```
sysadmin@localhost:~$ cd ~/Documents
```

```
sysadmin@localhost:~/Documents$ cat food.txt
Food is good.
```

Now use the `>` character to redirect the STDOUT of the `cat food.txt` command above to a new file called `newfile1.txt`:

```
sysadmin@localhost:~/Documents$ cat food.txt > newfile1.txt
sysadmin@localhost:~/Documents$
```

As you can see, there is no output displayed since the STDOUT has been redirected to the `newfile1.txt` file. Verify that the STDOUT of the `cat food.txt` command is in `newfile1.txt`:

```
sysadmin@localhost:~/Documents$ cat newfile1.txt
Food is good.
```

This is useful if you need to copy content from an important file to another file in order to edit the contents without modifying the original file. However, what if you want to add a comment or note to a file? To do this, you can use the `echo` command. The `echo` command is used to print output in the terminal:

```
sysadmin@localhost:~/Documents$ echo "Hello"
Hello
sysadmin@localhost:~/Documents$
```

Printing comments to the screen is a fun feature but the `echo` command can be made more useful by using redirection. Using the `echo` command, content can be added to the `newfile1.txt` file:

```
sysadmin@localhost:~/Documents$ cat newfile1.txt
Food is good.
sysadmin@localhost:~/Documents$ echo "I like food." > newfile1.txt
sysadmin@localhost:~/Documents$ cat newfile1.txt
I like food.
sysadmin@localhost:~/Documents$
```

Notice that the STDOUT of the `echo` command has replaced the original content of the file. This is because the single `>` character will overwrite any contents in an existing file. To append rather than overwrite content to a file, use a double greater-than symbol `>>`:

```
sysadmin@localhost:~/Documents$ echo "This food is good." >> newfile1.txt
sysadmin@localhost:~/Documents$ cat newfile1.txt
I like food.
This food is good.
sysadmin@localhost:~/Documents$
```

Important

To redirect information to an existing file, the user must have write permissions on that file.