

11.8.1 Copying Files

Like so many Linux features, you have a variety of options from which to choose when you want to manipulate files and directories. You can also use wildcards when you are copying, moving, or deleting files and directories.

To copy a file, type the following command:

```
cp <source> <destination>
```

So, to copy the file `sneakers.txt` to the directory `tigger` in your login directory, move to your login directory and type:

```
cp sneakers.txt tigger
```

Notice that you also used relative pathnames to copy the file. You can use both relative and absolute pathnames with `cp`. Our login directory is the parent of the directory `tigger`; `tigger` is one directory down from our login directory.

Read the `cp` man page (`man cp`) for a full list of the options available with `cp`. Among the options you can use with `cp` are the following:

- `-i` — interactive. Prompts you to confirm if the file is going to overwrite a file in your destination. This is a handy option because it can help prevent you from making mistakes.
- `-r` — recursive. Rather than just copying all the files and directories, this will copy the whole directory tree, subdirectories and all.
- `-v` — verbose. shows the progress of the files being copied.

If you use `cp` with no options, you will not see much when the command is executed. Using an option, such as `-i`, can make the process a little more useful. If you want to copy a file to a location that already has a file with the same name, you will be asked first if you really want to overwrite (or replace) the file that is already there.

Now that you have the file `sneakers.txt` in the `tigger` directory, use `cp -i` to copy the file again to the same location.

```
[newuser@localhost newuser]$  
cp -i sneakers.txt tigger  
cp: overwrite 'tigger/sneakers.txt'?
```

To overwrite the file that is already there, press [Y] and then [Enter]. If you do not want to overwrite the file, press [N] and [Enter].

11.8.2 Moving Files

To move files, use the **mv** command. It is similar to the **cp** command, except that with **mv** the file is physically moved from one place to another, instead of being duplicated, as with **cp**. For more about **mv**, see the **mv** man page (type **man mv**).

Common options for **mv** include the following:

- **-i** — interactive. This will prompt you if the file you have selected will overwrite an existing file in the destination directory. This is a good option, because like the **-i** option for **cp**, you will be given the chance to make sure you want to replace an existing file.
- **-f** — force. Overrides the interactive mode and moves without prompting. Unless you know what you are doing, this option is dangerous; be very careful about using it until you become more comfortable with your system.
- **-v** — verbose. Shows a list of the files being moved.

If you want to move a file out of your home directory and into another directory, type the following (you will need to be in your home directory):

```
mv sneakers.txt tigger
```

Alternatively, the same command using absolute pathnames looks like **mv sneakers.txt /home/newuser /home/newuser/tigger**.

11.8.3 Renaming Files

Actually, we have already covered half of renaming, because when you copy or move files, you can also rename.

To copy the file `sneakers.txt` from your login directory to the `tigger` subdirectory, just type:

```
cp sneakers.txt tigger
```

To copy and rename that file from `sneakers.txt` to `piglet.txt`, type:

```
cp sneakers.txt tigger/piglet.txt
```

To *move* and rename the file, just substitute **mv** for **cp** in the above example.

If you **cd** to `tigger` and then type **ls**, you will see the file `piglet.txt`.

If you just want to rename the file and keep its location, just **mv** in your current directory:

```
mv sneakers.txt piglet.txt
```

11.8.4 Deleting Files and Directories

You learned about creating files with the `touch` command and by using redirection in Chapter 10, *Shell Prompt Basics*. And you created the directory `tigger` using `mkdir`.

Now you need to learn how to delete files and directories. Deleting files and directories with the `rm` command is a straightforward process. See the `rm` man page for more information. Options for removing files and directories include:

- `-i` — interactive. Prompts you to confirm the deletion. This option can stop you from deleting a file by mistake.
- `-f` — force. Overrides interactive mode and removes the file(s) without prompting. This might not be a good idea, unless you know exactly what you are doing.
- `-v` — verbose. Shows a list of files as they are being removed.
- `-r` — recursive. Will delete a directory and all (if any) files and the subdirectories it contains.

To delete the file `piglet.txt` from the `tigger` directory with the `rm` command:

```
rm piglet.txt
```

What happens if you did not really want to get rid of it? Too late! That is where the `-i` (interactive) option is helpful, because it gives you a second chance to think about whether or not you really want to delete the file.

```
[newuser@localhost newuser]$  
rm -i piglet.txt  
rm: remove 'piglet.txt'?
```

You can also delete files using the wildcard `*`, but be careful, because you can easily delete files you did not intend to throw away.

To remove a file using a wildcard, you would type:

```
rm pig*
```

The above command will remove all files in the directory which start with the letters "pig."

You can also remove more than one file using one command:

```
rm piglet.txt sneakers.txt
```

Options for removing files and directories include the following:

- `-i` — interactive. Prompts you to confirm the deletion. This option can stop you from deleting a file by mistake. .

- `-f` — force. Overrides interactive mode and removes the file(s) without prompting. This might not be a good idea, unless you know exactly what you are doing.
- `-v` — verbose. Shows a list of files as they are being removed.
- `-r` — recursive. Will delete a directory and all (if any) files and the subdirectories it contains.

You can use `rmdir` to remove a directory (`rmdir foo`, for example), but only if the directory is empty. To remove directories with `rm`, you must specify the `-r` option.

For example, if you want to recursively remove the directory `tigger` you would type:

```
rm -r tigger
```

If you want to combine options, such as forcing a recursive deletion, you can type:

```
rm -rf tigger
```



The `rm` command can delete your entire filesystem! If you are logged in as root and you type the simple command `rm -rf /`, you are in trouble; this command will recursively remove everything on your system.

A safer alternative to using `rm` for removing directories is the `rmdir` command. With this command, you will not be allowed to use recursive deletions, so a directory which has files in it will not be deleted.

Read the `rmdir` man page (`man rmdir`) to find out more about this command.
