

Navigating the File System

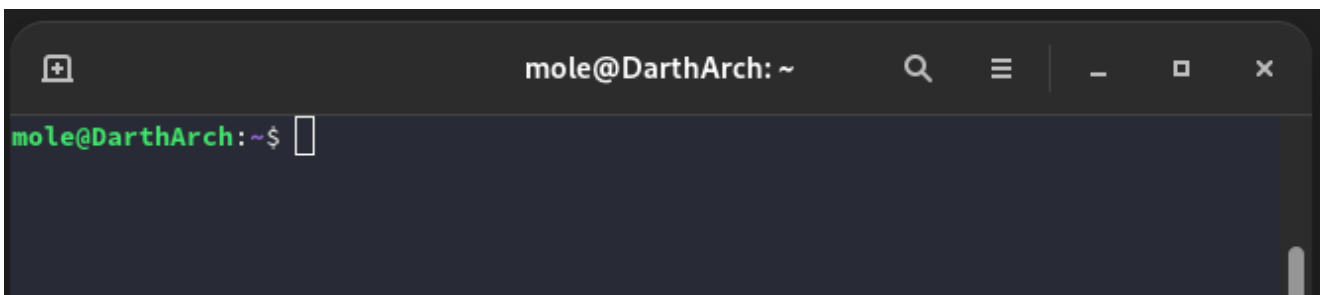
Category	Experience Level	Author
Linux	Complete Beginner	Nick Ruffles

Contents

- [Bash](#)
- [File system structure](#)
- [Listing files](#)
- [Displaying contents of files](#)
- [Creating and editing files](#)
- [Moving removing and copying files](#)
- [Changing user](#)
- [Locating files and text](#)

Bash

First of all, we need to know the 'shell' that we will be using. Bash is a binary (application) that comes pre-installed in most Linux operating systems. It is a way to directly interface with programs and files on the computer, and hence is very useful for controlling what the computer does.



This is what bash should look like on most installations. You can access bash via the GUI by searching for 'terminal' in the Linux application searcher (Press the windows key or the mac equivalent).

File system structure

```
mole@DarthArch: ~/SESH
mole@DarthArch:~/SESH$ ls /
bin  dev  home  lib64  mnt  proc  run  srv  sys  tmp  var
boot  etc  lib  lost+found  opt  root  sbin  swapfile  timeshift  usr
mole@DarthArch:~/SESH$
```

There are multiple directories within the 'root' (Top level) directory of the Linux file system, each with a specific purpose. In Linux and UNIX everything is a file, even folders. There is no such thing as a registry such as in Windows, it's just a file system which makes life easier for us.

Files and folders are neatly sorted into different folders in the root directory, which makes finding those files easier in the future for us and also other people.

`/bin`

- The bin directory stores binaries (applications) that the user is going to be able to use.

`/etc`

- The etc directory or 'etcetera' directory stores configuration files, this is where all the user changeable options are stored. For example `/etc/passwd` stores the home directories, uid, names and descriptions of the users on the machine.

`/home`

- The home directory simply stores all of the users' home directories.

`/root`

- The root directory stores the home directory for the root (admin/superuser) user.

`/opt`

- The opt directory stores 'optional' software. This will be applications such as Discord or Spotify that are not installed directly via a package manager.

`/usr`

- The usr directory stores user binaries and program data.

`/sbin`

- The sbin directory contains all of the binaries that the root user should have access to and run.

`/var`

- The var directory contains all variable information, this tends to be the files used while a process is running and needs to store data to the disk temporarily.

Listing files

First we want to be able to see what files we have access to, the `ls` command allows us to see the files in our current working directory. In the following example I will be in the `~/SESH` directory. I will explain this a bit later.

```
mole@DarthArch: ~/SESH
mole@DarthArch:~/SESH$ ls
File1.pdf File2.png File3.txt File4.txt
mole@DarthArch:~/SESH$
```

As you can see we can see some of the files in this directory, but this isn't all of the files. By default, `ls` doesn't show all hidden files. To show all files AND hidden files we can use the `-a` flag.

```
mole@DarthArch: ~/SESH
mole@DarthArch:~/SESH$ ls
File1.pdf File2.png File3.txt File4.txt
mole@DarthArch:~/SESH$ ls -a
. .. File1.pdf File2.png File3.txt File4.txt .hidden_passwords.txt
mole@DarthArch:~/SESH$
```

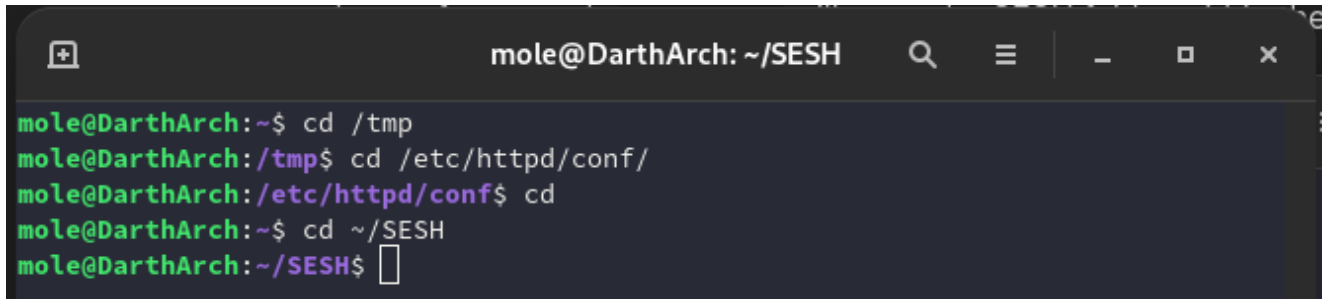
Now this shows us that we have access to the `.` and `..` directories, along with the `.hidden_passwords.txt` file. The `.` stands for the current working directory, while `..` stands for the directory above the current working directory (The parent directory).

We can also use `ls -la` to list more information about each file, including who owns that file, the size of the file in bytes, the time/day it was created and also the read/write/execute permissions about the file. More on the permissions later.

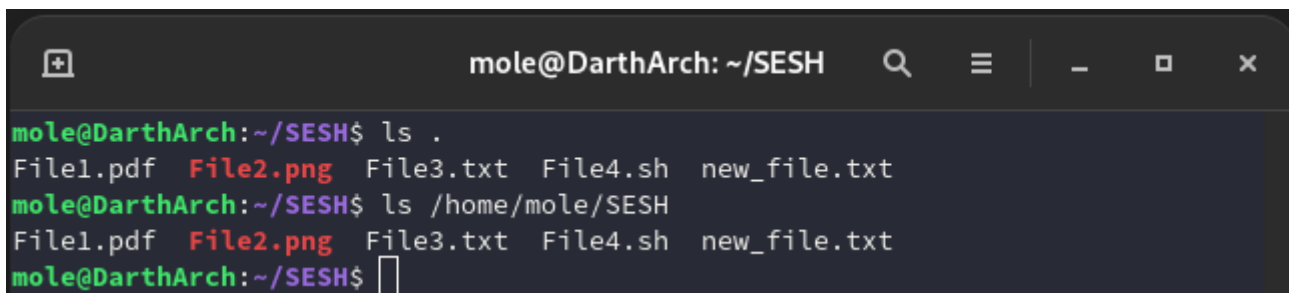
```
mole@DarthArch: ~/SESH
mole@DarthArch:~/SESH$ ls -la
total 8
drwxr-xr-x  2 mole mole 4096 Oct 19 19:25 .
drwx----- 35 mole mole 4096 Oct 19 19:22 ..
-rw-r--r--  1 mole mole   0 Oct 19 19:23 File1.pdf
-rw-r--r--  1 mole mole   0 Oct 19 19:23 File2.png
-rw-r--r--  1 mole mole   0 Oct 19 19:23 File3.txt
-rw-r--r--  1 mole mole   0 Oct 19 19:23 File4.txt
-rw-r--r--  1 mole mole   0 Oct 19 19:25 .hidden_passwords.txt
mole@DarthArch:~/SESH$
```

Changing directory

We can use the `cd` to change directory, fairly self explanatory. Typing just `cd` will take you to your home directory, similar to in Windows. You can also use the `~` prefix to denote your home, for example `cd ~/SESH` will go to the SESH folder within the home directory.

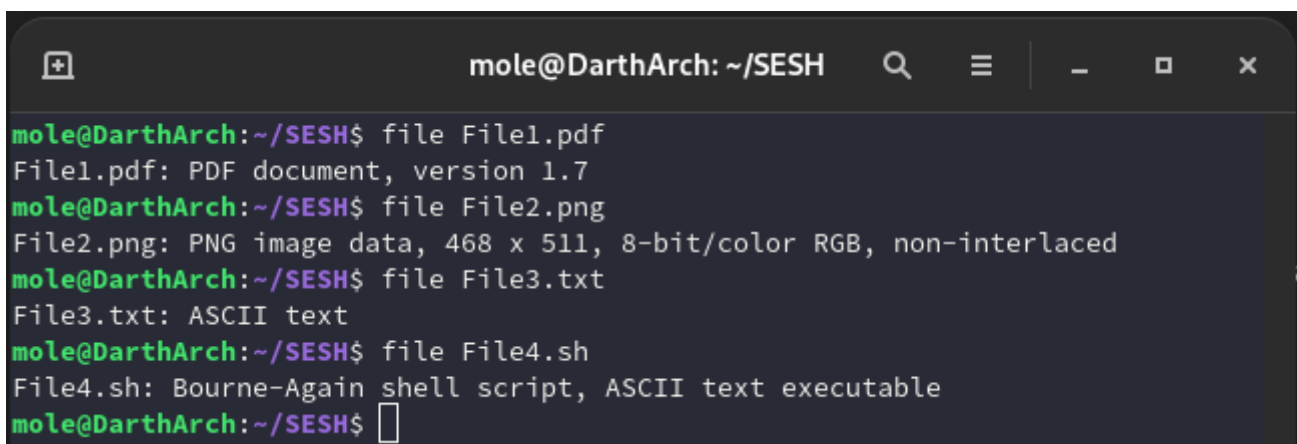
A terminal window titled 'mole@DarthArch: ~/SESH'. The prompt is 'mole@DarthArch:~\$'. The user enters 'cd /tmp', the prompt changes to 'mole@DarthArch:/tmp\$'. The user enters 'cd /etc/httpd/conf/', the prompt changes to 'mole@DarthArch:/etc/httpd/conf\$'. The user enters 'cd', the prompt changes to 'mole@DarthArch:~\$'. The user enters 'cd ~/SESH', the prompt changes to 'mole@DarthArch:~/SESH\$' with a cursor.

In Linux we can specify either relative paths or absolute paths. Relative paths use the current directory to calculate what folder/file to access, while absolute paths specify the whole path starting from the root directory. Below is an example of a relative path (top) and an absolute path (bottom).

A terminal window titled 'mole@DarthArch: ~/SESH'. The prompt is 'mole@DarthArch:~/SESH\$'. The user enters 'ls .' and the output is 'File1.pdf File2.png File3.txt File4.sh new_file.txt'. The user enters 'ls /home/mole/SESH' and the output is 'File1.pdf File2.png File3.txt File4.sh new_file.txt'. The prompt is 'mole@DarthArch:~/SESH\$' with a cursor.

Displaying contents of files

We can use the `file` command to see what the contents of a file is. It can tell us if the file is an executable binary, if it's an image, if it just contain text along with a load of other different file formats. We do this because we don't want to try and display a binary file to our terminal, as it will fill our screen with junk and it can often break our terminal.

A terminal window titled 'mole@DarthArch: ~/SESH'. The prompt is 'mole@DarthArch:~/SESH\$'. The user enters 'file File1.pdf' and the output is 'File1.pdf: PDF document, version 1.7'. The user enters 'file File2.png' and the output is 'File2.png: PNG image data, 468 x 511, 8-bit/color RGB, non-interlaced'. The user enters 'file File3.txt' and the output is 'File3.txt: ASCII text'. The user enters 'file File4.sh' and the output is 'File4.sh: Bourne-Again shell script, ASCII text executable'. The prompt is 'mole@DarthArch:~/SESH\$' with a cursor.

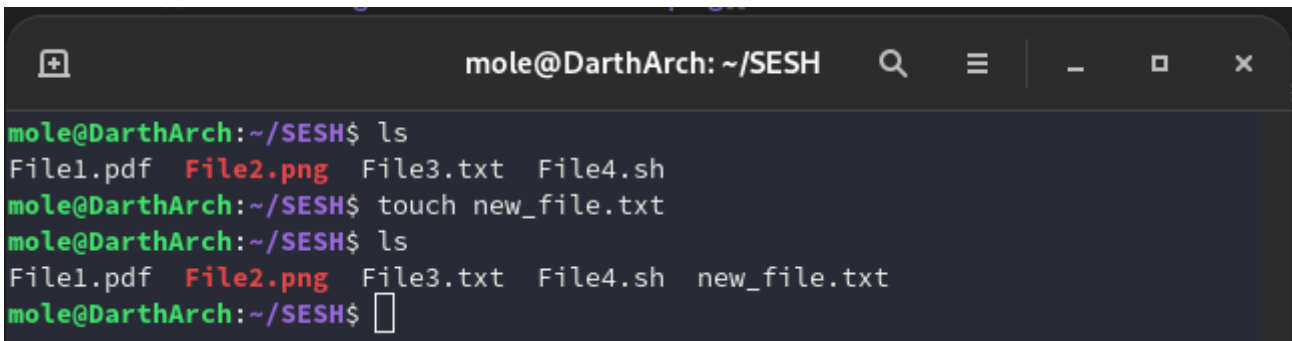
To display the contents of files, we use the `cat` command. It's fairly simple, and you'll find yourself using this command a lot.



```
mole@DarthArch: ~/SESH
mole@DarthArch:~/SESH$ cat File3.txt
Hello World!
mole@DarthArch:~/SESH$
```

Creating and editing files

We can create a new empty file by using the `touch` command.



```
mole@DarthArch: ~/SESH
mole@DarthArch:~/SESH$ ls
File1.pdf  File2.png  File3.txt  File4.sh
mole@DarthArch:~/SESH$ touch new_file.txt
mole@DarthArch:~/SESH$ ls
File1.pdf  File2.png  File3.txt  File4.sh  new_file.txt
mole@DarthArch:~/SESH$
```

To edit this file, we can use a terminal text editor called `nano`, it is possibly the easiest text editor to use in Linux. It allows us to insert, copy, paste, delete along with a load of other functions. To edit the file we just created, we will type `nano new_file.txt` (If the file hasn't already been made, then saving the file will create the file).

Now we can see the contents, we can type anything to be put inside this file. If we want to save the file we can use the `CTRL + O` keyboard shortcut to output the file, enter the name you want to save it as then press `ENTER` to save the file.

After we have saved the file, press `CTRL + X` to exit nano.

You can see the other keyboard shortcuts in the action bar at the bottom of the nano screen.

```
mole@DarthArch: ~/SESH
GNU nano 5.9 new_file.txt
[ Read 0 lines ]
^G Help      ^O Write Out  ^W Where Is  ^K Cut       ^T Execute
^X Exit      ^R Read File  ^\ Replace   ^U Paste     ^J Justify
```

Moving, removing and copying files

Now that we have a couple of files, we can try copying, moving and deleting them.

We can copy files by using the `cp` command, simply enter the file you want to copy and then its destination.

We can move a file by using the `mv` command, similar to `cp`, we enter the file we want to move then its destination. This command can also be used to rename files.

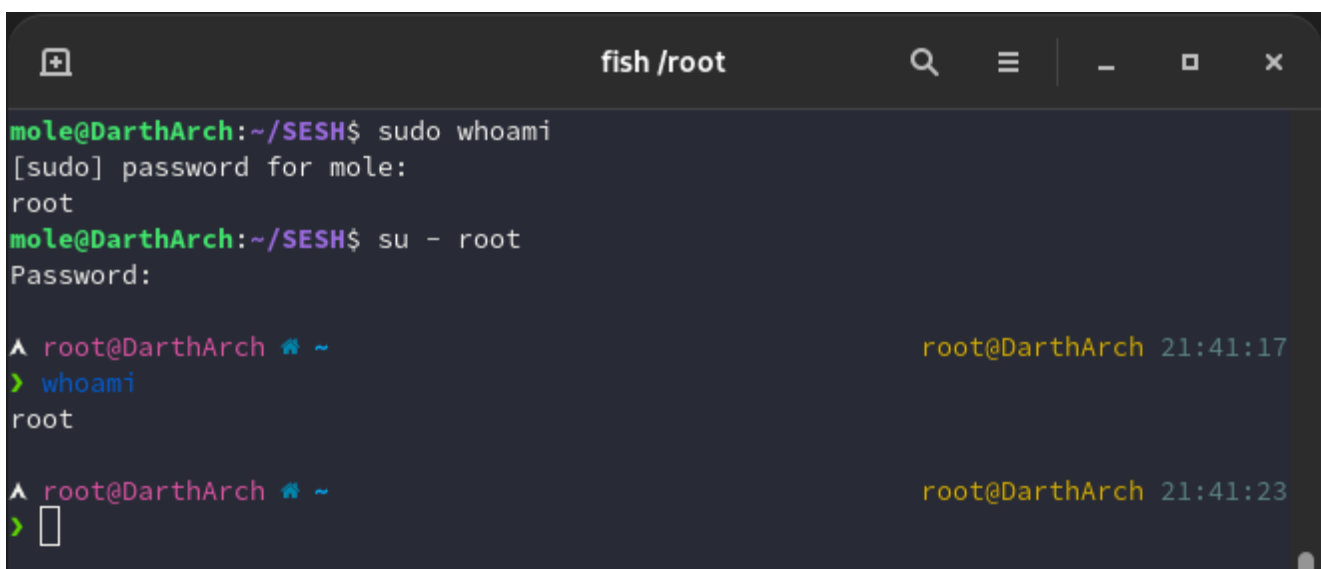
To remove a file we can use the `rm` command followed by the file or files to remove.

```
mole@DarthArch:~/SESH$ ls
File1.pdf File2.png File3.txt File4.sh new_file.txt
mole@DarthArch:~/SESH$ mv File1.pdf new_File1.pdf
mole@DarthArch:~/SESH$ cp File2.png File2.backup.png
mole@DarthArch:~/SESH$ rm new_file.txt
mole@DarthArch:~/SESH$ ls
File2.backup.png File2.png File3.txt File4.sh new_File1.pdf
mole@DarthArch:~/SESH$
```

Changing user

We are also able to change the user account that we are using, to do this we generally need to know the password to the account we are trying to access (Except if you are the root user). We can use the `su` command to switch user, simply entering `su - root` will try and switch you to be the root user account (although we could choose any account we know the password to). It will prompt you for the password. This would allow us to execute commands on behalf of that account, therefore potentially giving us access to more data and files.

We can also use the `sudo` command to run commands as the root user, you'll quite often see people use `sudo` when a higher level of privilege is needed to run a command. `sudo` works slightly different to `su` in that if you are part of a `wheel` or `sudo` group it means you can run `sudo` with your own password instead of the root password. The `wheel` and `sudo` groups are set by an administrator and should only be given to trusted individuals, as it effectively makes them an administrator on that computer.



```
fish /root
mole@DarthArch:~/SESH$ sudo whoami
[sudo] password for mole:
root
mole@DarthArch:~/SESH$ su - root
Password:

^ root@DarthArch ~ # root@DarthArch 21:41:17
> whoami
root

^ root@DarthArch ~ # root@DarthArch 21:41:23
> [
```

Locating files and text

The `find` command is incredibly useful for finding and filtering files on the Linux system, it allows you to filter by the name, creation date/time, permissions and owners among a load of other filters. This all allows you to easily find files on the system in a relatively quick amount of time. Use the `man find` command to see more info about the find command

For example, `find / -name "File4.sh"` would find all files where the name of the file contains `passwd`. This will, however, output some permission denied errors. To mute the permission denied errors, we can put `2>/dev/null` on the end of the command. This simply tells the errors (using file descriptor 2 which is standard error) to redirect to `/dev/null` (Unix equivalent to the void).

We can also use wildcards within the search if we don't know the full name of the file/directory to find: `find / -name "File4.*"`

```
mole@DarthArch: ~  
mole@DarthArch:~$ find / -name "File4.sh" 2>/dev/null  
/home/mole/SESH/File4.sh  
mole@DarthArch:~$ find / -name "File4.*" 2>/dev/null  
/home/mole/SESH/File4.sh  
mole@DarthArch:~$
```

Now it's all good and well trying to find files based on names and permissions, but what about if we want to find text within a load of files? For that we can use the `grep` command. Grep allows us to search through one or multiple files for text or regular expressions. To search through a single file for a string we can type `grep "Search term" /path/to/file`. To search through multiple files recursively we can use `grep -r "search term" /path/to/directory`. The grep command is increadibly useful, you will likely find yourself using it a lot in Linux.

```
fish /home/mole/SESH  
^ ~ /SESH mole@DarthArch 13:14:57  
> grep "Hello" File3.txt  
Hello World!  
^ ~ /SESH mole@DarthArch 13:15:08  
> grep -r "Hello" ./  
./File3.txt:Hello World!
```