Lists and Directories

Contents

Overview

"Everything in Linux is a file; If it's not a file, it's a process".

Linux stores programs and data in files which come in the form of:

- Text
- Scripts
- Directories
- Programs/Binaries
- and everything else is a process!

Linux file names can contain all ASCII characters except the `/` character.

Linux is *case sensitive*, meaning that all four files below are valid and separate

```
# ~/filename
# ~/Filename
# ~/FileName
# ~/FILENAME
```

The best practice is to stick to alphanumeric characters, dots, dashes and underscores.

There are no file extensions needed in Linux.
However, there are still naming conventions.

Linux adheres to the **F**ilesystem **H**ierarchy **S**tandard (`FHS`), all Linux distributions must follow this structure.
The reason `/` is not allowed is because it is reserved for the root user.

```
/ = root
```

Managing Files and Directories

Run the `pwd` command to find your present working directory.

```
# pwd
/home/ubuntu
```

The `pwd` example shows that the current working directory is the home directory of the user called *ubuntu*.

You can change your current directory with the `cd` command.
This will allow you to navigate the directory structure.
You can do this with either a relative path or the full path.

```
cd /home/user/example_dir #full path
cd ./example_dir #relative path
cd .. # This will take you to the previous directory on your current full path
cd # This will take the user to their home directory
```

Relative or Absolute Paths

Relative paths may be shorter commands and save time, but in a scripting environment, where your current location isn't always obvious, it is best to use the full paths when changing directory.

Wildcards (Globbing)

We can use wildcards to find files that have similar names. The following command would return all of the python files inside this directory.

```
ls *.py
```

There some other options you can use.

```
* # Zero or more characters
? # Exactly one character
[abc] # One character, either a, b or c
[A-Z] # One character in the range A-Z
[!abc] # One character, neither a, b or c
~ # Users home directory
```

Here are some examples.

```
rm test[0-9].txt # This will remove all files in your current directory that are
called test0.txt with any number between 0 and 9
mv ?.py ~ # This command will move every Python files with one character long
names into the users home directory.
```

Variables in Linux

We assign variables in Linux using `=` and reference them with `${}`. For example, the commands below would return `mylog1`

```
Log_file=mylog1
echo ${Log_file}
```

You can also assign a command to a variable. These commands would return the same output as the command `uname -r`

```
version=$(uname -r)
echo ${version}
```

You unassign a variable with `unset`

```
unset version
echo ${version} # This will return nothing
```

Making a File

We can make a file using the `touch` command.
This command would create a file called `example.txt` if the file doesn't exist.
If the file exists then the files modified time would be updated to the current time.

```
touch example.txt
```

We can also add content to a file using `echo` and `>`. This command would add the sentence "Hello" to our file and override any existing contents within it. Using `>>` would append to the end of file.

```
echo "Hello" > example.txt
```

The `cat` command allows us to read a file. The follwing command would return `Hello`.

```
cat example.txt
```

Copying Files

The `cp` command copies files to a target file or directory. This command would create an identical file to our `example.txt` but give it a new name.

```
# cp source_file target_file
cp example.txt identical.txt
```

You can also copy more than one file at a time.

```
cp test1.txt test2.txt new_dir
# This will copy both .txt files into the new_dir directory
```

Be aware that if the source section of the command refers to more than one file then the target must be a directory.

Moving Files

The `mv` command moves files from one place to another, it can also be used to rename a file.
It has a very similar layout to the `cp` command. This command would move our `example.txt` to the Documents directory.

```
mv example.txt /Documents
```

Removing Files

The command to remove a file is `rm`. You can delete more than one file at once. The set of commands below move the `example.txt` back to our working directory and then removes both the `.txt` files.

```
mv /Documents/example.txt .
rm example.txt identical.txt
```

Directories

You can make a directory with the `mkdir` command. The following command creates a `example_dir` directory.

```
mkdir exmaple_dir
```

You can delete an empty directory with the `rmdir` command, but you must use `rm -r` to remove a directory containing files. The following command will both remove our directory `example_dir`.

```
rm -r example_dir
rmdir example_dir
```

The `cp` and `mv` commands both work with directories.
These commands will move the directory to the /Documents directory and copy the directory and its contents to the /Documents directory.

```
mv example_dir /Documents
cp example_dir -r /Documents
```

Tutorial

There is no tutorial for this module.

Exercises

What does the `-r` flag do to modify the `rm` command?

▶ Answer

What does the `-f` flag do to modify the `rm` command?

▼ Answer

`-f` stands for `force`. This means it will not ask the user whether they would like to consider every file one by one when carrying out the command.

Now run commands to complete the following tasks.

▶ Tasks