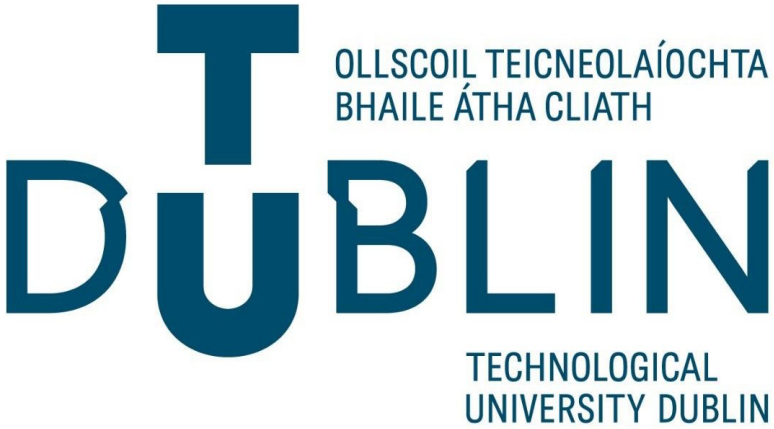

Goals

In this lab we're going to

- Get familiar with pipes and redirects
- Grep and Sort
- Introduce ps to view processes running on a machine

Pipes and Redirects

Normally, we get the output of any command into the terminal. Sometimes, we need to send the output of the command as the input of another command. This task can be done by executing the commands multiple times. But this task can be done easily using a pipe command that executes two or more commands at a time in Linux where the output of the one command is the input of the next command. The '|' operator is used for piping. The uses of the pipe command are described in this tutorial.

Tasks

Open up a terminal window in your codespace.

Task 1: Run the following command to write a string value into the test.md file. Here, the output of the “echo” command is passed as the input of the “cat” command:

```
echo "Learn bash programming" | cat > test.md
```

Run the following command to check the content of the **test.md** file:

```
cat test.md
```

Task 2: Assign the output of a command to a variable

The method of counting the total number of text files that start with the word “test” is shown in this example. This task can be done using multiple commands or a single command with the pipe (|).

Run the following command to create a number of empty md (markdown) files.

```
touch sample1.md test2.md sample3.md test4.md
sample5.md test6.md sample7.md
```

Run the following command to print the list of markdown files of the current location:

```
ls -la *.md
```

Run the following command to store the list of text files in the $list variable:

```
list=`ls *.md`
```

Run the following command to show the contents of the $list variable

```
echo $list
```

Task 3: Introduce grep

The `grep` command supports many functionalities. Users can select a functionality of their choice by setting the right flags or arguments. Check out `man grep` for details of all the options you can pass to grep.

Run the following command to find out the list of text files that start with the word “test” and store them in the filter.md file:

```
grep test*.md $list > filter.md
```

Display the contents of filter.md – what is the command to do this? What do we think happened here?

Task 4: Count lines

Run the following command to count the total number of lines of the filter.md file:

```
wc -l filter.md
```

The task of the previous commands can be done easily using the following single command where the output of the “ls” command is sent as the input of the “grep” command. The output of the “grep” command is as the input of the “wc” command using pipe (|)

```
ls *.md | grep test | wc -l
```

This is how piping works. The output of one command feeds into the input of the next command.

Task 5: Sort the contents of a file

The “sort” command can be used to sort the content of the file in different ways. The method of sorting the content of a text file using the “cat” and “sort” commands is shown in this part of the tutorial.

Create a markdown file named **products.md** with the following content:

```
Mouse A4Tech 100
Monitor DELL 120
Keyboard Defender 200
Scanner Epson 230
Headphone Apple 111
```

Run the following command to check the content of the products.md file:
`cat products.md`

Run the following command to sort the content of the file in ascending order based on the first column of the file:

```
cat products.md | sort
```

Run the following command to sort the content of the file in descending order based on the first column of the file:

```
cat products.md | sort -r
```

Question: If I wanted to get more information of the `sort` command on the command line, what command would I type?

Run the following command to sort the content of the file in ascending order based on the third column of the file:

```
cat products.md | sort -k3
```

Task 6 – Check the contents of a file

Run the following commands to check the content of the products.txt file and print the last line from the first three lines of the products.md file:

```
cat products.md
cat products.md | head -3 | tail -1
```

The pipe (|) operator is a very useful operator of Bash which is used for multiple purposes. Some common uses of this operator are shown in this tutorial using various examples. We hope that the use of the pipe (|) operator is cleared for the Bash users after reading this tutorial.

Task 7 – Introducing PS

Linux provides us a utility called **ps** for viewing information related with the processes on a system which stands as abbreviation for **"Process Status".** ps command is used to list the currently running processes and their PIDs along with some other information depends on different options. It reads the process information from the virtual files in **/proc** file-system. /proc contains virtual files, this is the reason it's referred as a virtual file system.

Simple process selection. Run the following command to show the processes for the current shell

```
ps
```

The result contains four columns of information.
Where,
PID – the unique process ID
TTY – terminal type that the user is logged into
TIME – amount of CPU in minutes and seconds that the process has been running
CMD – name of the command that launched the process.

Note **–** Sometimes when we execute **ps** command, it shows TIME as 00:00:00. It is nothing but the total accumulated CPU utilization time for any process and 00:00:00 indicates no CPU time has been given by the kernel till now. In above example we found that, for bash no CPU time has been given. This is because bash is just a parent process for different processes which needs bash for their execution and bash itself is not utilizing any CPU time till now.

Lots more examples here to try if you finish these exercises early.

Run through all the commands in this document.
https://www.geeksforgeeks.org/ps-command-in-linux-with-examples/