CHAPTER 5

Useful Unix Commands and Tools

In the last chapter, we learned a few more commands of intermediate difficulty. We are very comfortable now with the command prompt and can navigate the filesystem of Linux and other Unix-like OSs. We can use simple file and directory commands. We are also comfortable with various operators and piping.

In this chapter, we will learn advanced commands and tools in Unix. The following is the list of topics we will learn in this chapter:

- Shell and environment variables
- Useful Linux commands
- Useful Unix tools

After this chapter, we will be very comfortable with advanced tools in Unix. We will find these commands and concepts useful for learning shell scripting in the next chapter.

Shell and Environment Variables

Let us see how we can define variables in the shell. We can define numeric and string types of variables as follows:

```
a=2
str1='ASHWIN'
```

© Ashwin Pajankar 2021
A. Pajankar, *Practical Linux with Raspberry Pi OS*,
https://doi.org/10.1007/978-1-4842-6510-9_5

We do not have to declare them as we do in programming languages like C or Java. These variables are known as shell variables. We can access them by prefixing the variable names with a $ symbol as follows:

```
echo $a
echo $str1
```

The preceding statements will print the values stored in the variables. We can assign values belonging to any data type to a variable. Thus, the variables in the shell are not confined to storing values of any single data type.

An environment variable is a variable whose value is set with the functionality built into the operating system or shell. An environment variable is made up of a name and value pair. All system-related information is stored in the environment variable. We can see the list of environment variables by running the following command:

```
env
```

It will print a very long list of variables, and it will consume several pages. We have already seen that the variable `SHELL` stores the name of the executable file of the current shell. So we will have a look at the important environment variables. Run the following command to know the Bash shell version:

```
echo $BASH_VERSION
```

Run the following command to know the hostname of your RPi:

```
echo $HOSTNAME
```

Run the following command to know the location of the history file:

```
echo $HISTFILE
```

The following command returns the location of the **home** directory of the current logged user (in our case, the user **pi**):

```
echo $HOME
```

To know the directory location the shell searches for the executable files when we run any command, use the following command:

```
echo $PATH
```

Useful Linux Commands

Let us see a few useful commands in Linux. The command w shows who are logged in and what they are doing. Run the command and see the output.

The command uptime shows for how long the system is running:

```
pi@raspberrypi:~ $ uptime
 17:05:24 up 19:10,  3 users,  load average: 0.24, 0.22, 0.18
```

The command who shows who is logged in:

```
pi@raspberrypi:~ $ who
pi       tty1         2020-08-24 21:54
pi       pts/0        2020-08-25 16:42 (192.168.0.100)
pi       pts/1        2020-08-25 16:59 (192.168.0.100)
```

The command whoami prints the ID of the current user as follows:

```
pi@raspberrypi:~ $ whoami
pi
```

We can get information about the system with the following command:

```
pi@raspberrypi:~ $ uname -a
Linux raspberrypi 5.4.51-v7l+ #1333 SMP Mon Aug 10 16:51:40 BST
2020 armv7l GNU/Linux
```

We can get information about the current processes and utilization of resources using the commands htop and top. Run them to see the output.

We can see a snapshot of current processes with the command ps:

```
pi@raspberrypi:~ $ ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 Aug24 ?        00:00:04 /sbin/ini
root         2     0  0 Aug24 ?        00:00:00 [kthreadd
root         3     2  0 Aug24 ?        00:00:00 [rcu_gp]
```

This is the partial output of the execution of the command.

The command df reports the details of the filesystem:

```
pi@raspberrypi:~ $ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        15G  6.5G  7.3G  47% /
devtmpfs        1.8G     0  1.8G   0% /dev
tmpfs           1.9G     0  1.9G   0% /dev/shm
tmpfs           1.9G  8.7M  1.9G   1% /run
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           1.9G     0  1.9G   0% /sys/fs/cgroup
/dev/mmcblk0p1  253M   54M  199M  22% /boot
tmpfs           378M  4.0K  378M   1% /run/user/1000
```

We can see the list of connected USB devices as follows with the command lsusb:

```
pi@raspberrypi:~ $ lsusb
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 005: ID 046d:081b Logitech, Inc. Webcam C310
Bus 001 Device 004: ID 046d:c077 Logitech, Inc. M105 Optical Mouse
Bus 001 Device 003: ID 1c4f:0002 SiGma Micro Keyboard TRACER
Gamma Ivory
```

```
Bus 001 Device 002: ID 2109:3431 VIA Labs, Inc. Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

We can see processor information with the commands `lscpu` and `cat /proc/cpuinfo`. Run both the commands to see the output.

We can use the following commands to see information related to memory:

```
pi@raspberrypi:~ $ free -m
              total        used        free      shared  buff/cache   available
Mem:           3776         121        3320          36         334        3491
Swap:            99           0          99
pi@raspberrypi:~ $ cat /proc/meminfo
MemTotal:        3867184 kB
MemFree:         3399896 kB
MemAvailable:    3575016 kB
```

Unix commands are binary executable files. We can locate them with the commands `which` and `whereis`. The command which tells us the location of a binary executable:

```
pi@raspberrypi:~ $ which python3
/usr/bin/python3
```

We can retrieve information about the man page and documentation about the command with the command `whereis` as follows:

```
pi@raspberrypi:~ $ whereis python3
python3: /usr/bin/python3.7m /usr/bin/python3
/usr/bin/python3.7-config /usr/bin/python3.7
/usr/bin/python3.7m-config /usr/lib/python3 /usr/lib/python3.7
/etc/python3 /etc/python3.7 /usr/local/lib/python3.7
/usr/include/python3.7m /usr/include/python3.7
/usr/share/python3 /usr/share/man/man1/python3.1.gz
```

There is another Raspberry Pi OS–specific utility that can retrieve a lot of system information. It is vcgencmd. We can learn more about it at www.raspberrypi.org/documentation/raspbian/applications/vcgencmd.md.

The following are the examples of the execution:

```
pi@raspberrypi:~ $ vcgencmd measure_temp
temp=35.0'C
pi@raspberrypi:~ $ vcgencmd get_mem arm && vcgencmd get_mem gpu
arm=896M
gpu=128M
```

The first example shows the CPU temperature, and the second example shows the memory split (in megabytes) between CPU and GPU.

Useful Unix Tools

Let us study a few useful UNIX tools. These useful UNIX commands are found in all the distributions of Linux and other Unix-like operating systems. Let us create a simple CSV file for the demonstration. I have created a small CSV file for the demo, and its contents are as follows:

```
pi@raspberrypi:~ $ cat abc.csv
ASHWIN, 20k, INDIA
THOR, 10k, Asgard
JANE, 15k, UK
IRON MAN, 100k, USA
```

We can check the statistics of the file (number of words, lines, and characters including blank spaces) with the command wc as follows:

```
pi@raspberrypi:~ $ wc abc.csv
 4 13 71 abc.csv
pi@raspberrypi:~ $ wc -c abc.csv
71 abc.csv
```

```
pi@raspberrypi:~ $ wc -w abc.csv
13 abc.csv
pi@raspberrypi:~ $ wc -l abc.csv
4 abc.csv
```

The first example shows all the statistics of a file. The next three examples show the counts of characters, words, and lines, respectively. We can also use the command cut on this file for more practice. Have a look at the following examples:

```
pi@raspberrypi:~ $ cut -c 2-5 abc.csv
SHWI
HOR,
ANE,
RON
pi@raspberrypi:~ $ cut -d "," -f 2- abc.csv
 20k, INDIA
 10k, Asgard
 15k, UK
 100k, USA
```

We can use the command grep to find patterns of texts. For example, if I want to find my name in a text file, I can use the command grep as follows:

```
pi@raspberrypi:~ $ grep ASHWIN abc.csv
ASHWIN, 20k, INDIA
```

If I want the search to be case insensitive, then I can use it the following way:

```
pi@raspberrypi:~ $ grep -i asgard abc.csv
THOR, 10k, Asgard
```

We can sort data with the command `sort` as follows:

```
pi@raspberrypi:~ $ sort abc.csv
ASHWIN, 20k, INDIA
IRON MAN, 100k, USA
JANE, 15k, UK
THOR, 10k, Asgard
```

We can find out unique data items as follows:

```
pi@raspberrypi:~ $ sort abc.csv | uniq
ASHWIN, 20k, INDIA
IRON MAN, 100k, USA
JANE, 15k, UK
THOR, 10k, Asgard
```

To see the command in action, before running it, insert a duplicate line in the file `abc.csv`:

The following are the date and calendar commands in action:

```
pi@raspberrypi:~ $ date
Tue 25 Aug 2020 09:03:01 PM IST
pi@raspberrypi:~ $ cal
    August 2020
Su Mo Tu We Th Fr Sa
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
```

Finally, if we want to find a file, then we can use the command `find` as follows:

```
pi@raspberrypi:~ $ find . -name "*.conf"
./.config/lxterminal/lxterminal.conf
./.config/lxsession/LXDE/desktop.conf
./.config/pcmanfm/LXDE/desktop-items-0.conf
```

The command is followed by the path (in our case, it is the current directory, hence .) and criteria for the search. Here, we are searching for the configuration files in the current directory.

Summary

In this chapter, we learned many advanced Linux commands. We will use all the commands we learned in this and previous chapters to prepare shell scripts in the next chapter.

The next chapter will have detailed instructions on how to prepare and execute shell scripts on Linux.