1

GETTING STARTED WITH THE BASICS

By our very nature, hackers are doers. We want to touch and play with things. We also want to create and, sometimes, break things. Few of us want to read long tomes of information technology theory before we can do what we love most: hacking. With that in mind, this chapter is designed to give you some fundamental skills to get you up and running in Kali . . . now!

In this chapter, we won't go into any one concept in great detail—we'll cover just enough to let you play and explore in the operating system of hackers: Linux. We will save more in-depth discussions for later chapters.

Introductory Terms and Concepts

Before we begin our journey through the wonderful world of *Linux Basics for Hackers*, I want to introduce a few terms that should clarify some concepts discussed later in this chapter.

Binaries This term refers to files that can be executed, similar to executables in Windows. Binaries generally reside in the */usr/bin* or *usr/sbin* directory and include utilities such as `ps`, `cat`, `ls`, and `ifconfig` (we'll touch on all of four of these in this chapter) as well as applications such as the wireless hacking tool aircrack-ng and the intrusion detection system (IDS) Snort.

Case sensitivity Unlike Windows, the Linux filesystem is case sensitive. This means that *Desktop* is different from *desktop*, which is different from *DeskTop*. Each of these would represent a different file or directory name. Many people coming from a Windows environment can find this frustrating. If you get the error message "file or directory not found" and you are sure the file or directory exists, you probably need to check your case.

Directory This is the same as a folder in Windows. A directory provides a way of organizing files, usually in a hierarchical manner.

Home Each user has their own */home* directory, and this is generally where files you create will be saved by default.

Kali Kali Linux is a distribution of Linux specifically designed for penetration testing. It has hundreds of tools preinstalled, saving you the hours it would take to download and install them yourself. I will be using the latest version of Kali at the time of this writing: Kali 2018.2, first released in April 2018.

root Like nearly every operating system, Linux has an administrator or superuser account, designed for use by a trusted person who can do nearly anything on the system. This would include such things as reconfiguring the system, adding users, and changing passwords. In Linux, that account is called *root*. As a hacker or pentester, you will often use the root account to give yourself control over the system. In fact, many hacker tools require that you use the root account.

Script This is a series of commands run in an interpretive environment that converts each line to source code. Many hacking tools are simply scripts. Scripts can be run with the bash interpreter or any of the other scripting language interpreters, such as Python, Perl, or Ruby. Python is currently the most popular interpreter among hackers.

Shell This is an environment and interpreter for running commands in Linux. The most widely used shell is bash, which stands for *Bourne-again shell*, but other popular shells include the C shell and Z shell. I will be using the bash shell exclusively in this book.

Terminal This is a command line interface (CLI).

With those basics behind us, we will attempt to methodically develop the essential Linux skills you'll need to become a hacker or penetration tester. In this first chapter, I'll walk you through getting started with Kali Linux.

A Tour of Kali

Once you start Kali, you'll be greeted with a login screen, as shown in Figure 1-1. Log in using the root account username *root* and the default password *toor* (if you changed the password earlier, use your new password here).

Figure 1-1: Logging into Kali using the root account

You should now have access to your Kali desktop (see Figure 1-2). We'll quickly look at two of the most basic aspects of the desktop: the terminal interface and file structure.

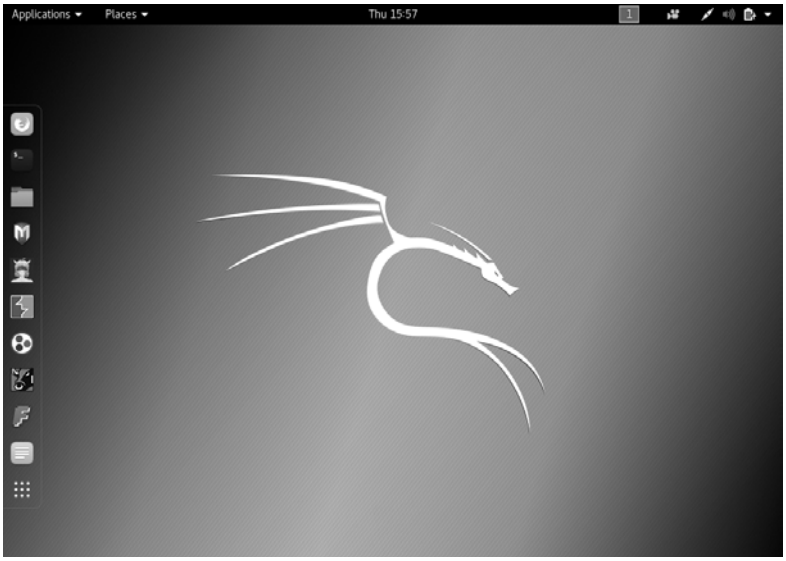

Figure 1-2: The Kali desktop

The Terminal

The first step in using Kali is to open the *terminal*, which is the command line interface we'll use in this book. In Kali Linux, you'll find the icon for the terminal along the left of the desktop. Click this icon to open the terminal. Your new terminal should look like the one shown in Figure 1-3.

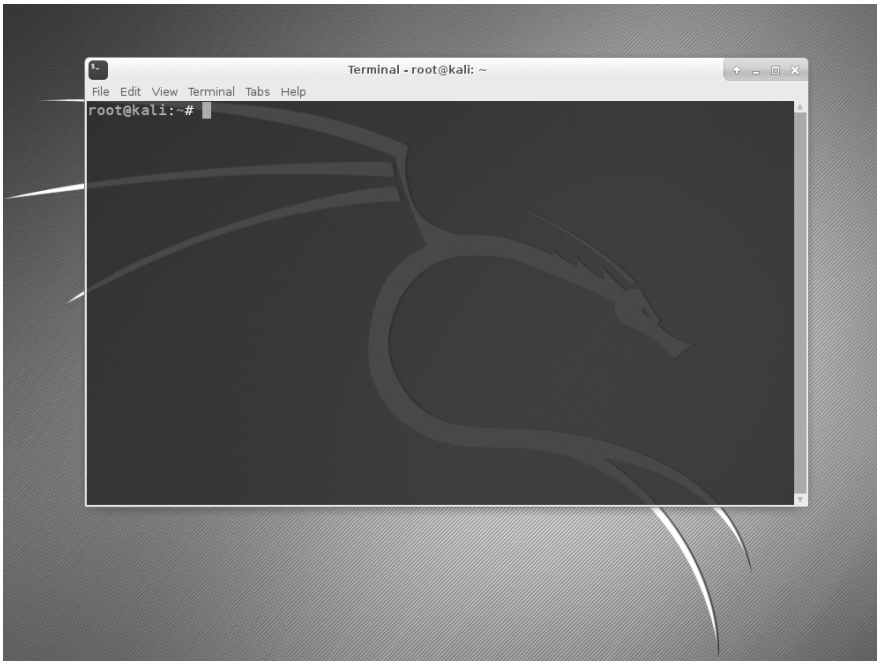

Figure 1-3: The Kali terminal

This terminal opens the command line environment, known as the *shell*, which enables you to run commands on the underlying operating systems and write scripts. Although Linux has many different shell environments, the most popular is the bash shell, which is also the default shell in Kali and many other Linux distributions.

To change your password, you can use the command `passwd`.

The Linux Filesystem

The Linux filesystem structure is somewhat different from that of Windows. Linux doesn't have a physical drive (such as the *C:* drive) at the base of the filesystem but uses a logical filesystem instead. At the very top of the file-system structure is */*, which is often referred to as the *root* of the filesystem, as if it were an upside-down tree (see Figure 1-4). Keep in mind that this is different from the root user. These terms may seem confusing at first, but they will become easier to differentiate once you get used to Linux.

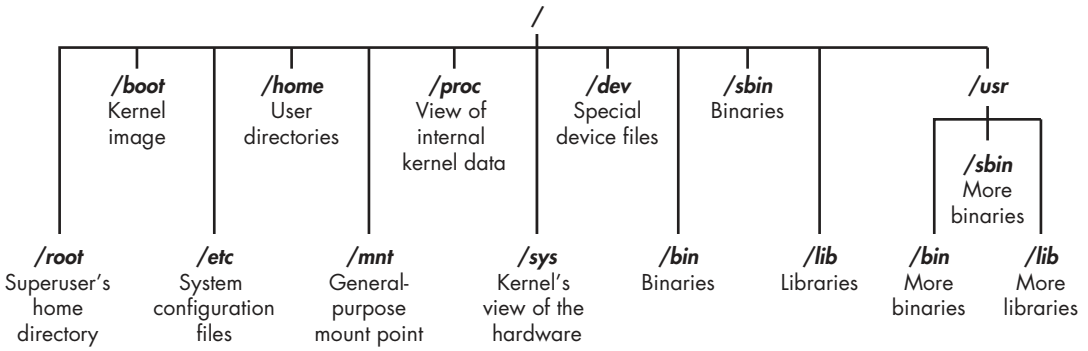

Figure 1-4: The Linux filesystem

The root (/) of the filesystem is at the top of the tree, and the following are the most important subdirectories to know:

/root The home directory of the all-powerful root user

/etc Generally contains the Linux configuration files—files that control when and how programs start up

/home The user's home directory

/mnt Where other filesystems are attached or mounted to the filesystem

/media Where CDs and USB devices are usually attached or mounted to the filesystem

/bin Where application *binaries* (the equivalent of executables in Microsoft Windows or applications in macOS) reside

/lib Where you'll find *libraries* (shared programs that are similar to Windows DLLs)

We'll spend more time with these key directories throughout this book. Understanding these first-level directories is important to navigating through the filesystem from the command line.

It's also important to know before you start that you should not log in as root when performing routine tasks, because anyone who hacks your system (yes, hackers sometimes get hacked) when you're logged in as root would immediately gain root privileges and thus "own" your system. Log in as a regular user when starting regular applications, browsing the web, running tools like Wireshark, and so on. For the practice you'll do in this book, staying logged in as root should be fine.

Basic Commands in Linux

To begin, let's look at some basic commands that will help you get up and running in Linux.

Finding Yourself with pwd

Unlike when you're working in a graphical user interface (GUI) environment like Windows or macOS, the command line in Linux does not always make it apparent which directory you're presently in. To navigate to a new directory, you usually need to know where you are currently. The *present working directory* (or *print working directory*) command, `pwd`, returns your location within the directory structure.

Enter `pwd` in your terminal to see where you are:

```
kali >pwd
/root
```

In this case, Linux returned `/root`, telling me I'm in the root user's directory. And because you logged in as root when you started Linux, you should be in the root user's directory, too, which is one level below the top of the filesystem structure (`/`).

If you're in another directory, `pwd` will return that directory name instead.

Checking Your Login with whoami

In Linux, the one "all-powerful" superuser or system administrator is named root, and it has all the system privileges needed to add users, change passwords, change privileges, and so on. Obviously, you don't want just anyone to have the ability to make such changes; you want someone who can be trusted and has proper knowledge of the operating system. As a hacker, you usually need to have all those privileges to run the programs and commands you need (many hacker tools won't work unless you have root privileges), so you'll want to log in as root.

If you've forgotten whether you're logged in as root or another user, you can use the `whoami` command to see which user you're logged in as:

```
kali >whoami
root
```

If I had been logged in as another user, such as my personal account, `whoami` would have returned my username instead, as shown here:

```
kali >whoami
OTW
```

Navigating the Linux Filesystem

Navigating the filesystem from the terminal is an essential Linux skill. To get anything done, you need to be able to move around to find applications, files, and directories located in other directories. In a GUI-based system, you can visually see the directories, but when you're using the command

line interface, the structure is entirely text based, and navigating the filesystem means using some commands.

Changing Directories with cd

To change directories from the terminal, use the *change directory* command, `cd`. For example, here's how to change to the */etc* directory used to store configuration files:

```
kali >cd /etc
kali:/etc >
```

The prompt changes to `root@kali:/etc`, indicating that we're in the */etc* directory. We can confirm this by entering `pwd`:

```
kali:/etc >pwd
/etc
```

To move up one level in the file structure (toward the root of the file structure, or `/`), we use `cd` followed by double dots (`..`), as shown here:

```
kali:/etc >cd ..
kali >pwd
/
kali >
```

This moves us up one level from */etc* to the `/` root directory, but you can move up as many levels as you need. Just use the same number of double-dot pairs as the number of levels you want to move:

- You would use `..` to move up one level.
- You would use `../..` to move up two levels.
- You would use `../../..` to move up three levels, and so on.

So, for example, to move up two levels, enter `cd` followed by two sets of double dots with a forward slash in between:

```
kali >cd ../..
```

You can also move up to the root level in the file structure from anywhere by entering `cd /`, where `/` represents the root of the filesystem.

Listing the Contents of a Directory with ls

To see the contents of a directory (the files and subdirectories), we can use the `ls` (list) command. This is very similar to the `dir` command in Windows.

```
kali >ls
bin    initrd.img        media        run        var
```

```
boot   initrd.img.old    mnt          sbin       vmlinuz
dev    lib               opt          srv        vmlinuz.old
etc    lib64             proc         tmp
home   lost+found        root         usr
```

This command lists both the files and directories contained in the directory. You can also use this command on any particular directory, not just the one you are currently in, by listing the directory name after the command; for example, `ls /etc` shows what's in the */etc* directory.

To get more information about the files and directories, such as their permissions, owner, size, and when they were last modified, you can add the `-l` switch after `ls` (the `l` stands for *long*). This is often referred to as *long listing*. Let's try it here:

```
kali >ls -l
total 84
drw-r--r--    1    root     root    4096    Dec    5  11:15    bin
drw-r--r--    2    root     root    4096    Dec    5  11:15    boot
drw-r--r--    3    root     root    4096    Dec    9  13:10    dev
drw-r--r--    18   root     root    4096    Dec    9  13:43    etc
--snip--
drw-r--r--    1    root     root    4096    Dec    5  11:15    var
```

As you can see, `ls -l` provides us with significantly more information, such as whether an object is a file or directory, the number of links, the owner, the group, its size, when it was created or modified, and its name.

I typically add the `-l` switch whenever doing a listing in Linux, but to each their own. We'll talk more about `ls -l` in Chapter 5.

Some files in Linux are hidden and won't be revealed by a simple `ls` or `ls -l` command. To show hidden files, add a lowercase `-a` switch, like so:

```
kali >ls -la
```

If you aren't seeing a file you expect to see, it's worth trying `ls` with the `a` flag When using multiple flags, you can combine them into one, as we've done here with `-la` instead of `-l -a`.

Getting Help

Nearly every command, application, or utility has a dedicated help file in Linux that provides guidance for its use. For instance, if I needed help using the best wireless cracking tool, aircrack-ng, I could simply type the `aircrack-ng` command followed by the `--help` command:

```
kali >aircrack-ng --help
```

Note the double dash here. The convention in Linux is to use a double dash (`--`) before word options, such as `help`, and a single dash (`-`) before single-letter options, such as `-h`.

When you enter this command, you should see a short description of the tool and guidance on how to use it. In some cases, you can use either `-h` or `-?` to get to the help file. For instance, if I needed help using the hacker's best port-scanning tool, nmap, I would enter the following:

```
kali >nmap -h
```

Unfortunately, although many applications support all three options (`--help`, `-h`, and `-?`), there's no guarantee the application you're using will. So if one option doesn't work, try another.

Referencing Manual Pages with man

In addition to the help switch, most commands and applications have a manual (man) page with more information, such as a description and synopsis of the command or application. You can view a man page by simply typing `man` before the command, utility, or application. To see the man page for aircrack-ng, for example, you would enter the following:

```
kali >man aircrack-ng
NAME
       aircrack-ng - a 802.11 WEP / WPA-PSK key cracker
SYNOPSIS
       aircrack-ng [options] <.cap / .ivs file(s)>
DESCRIPTION
       aircrack-ng is an 802.11 WEP and WPA/WPA2-PSK key cracking program.
       It can recover the WEP key once enough encrypted packets have been
       captured with airodump-ng. This part of the aircrack-ng suite deter-
       mines the WEP key using two fundamental methods. The first method is
       via the PTW approach (Pyshkin, Tews, Weinmann). The main advantage
       of the PTW approach is that very few data packets are required to
       crack the WEP key. The second method is the FMS/KoreK method. The
       FMS/KoreK method incorporates various statistical attacks to dis-
       cover the WEP key and uses these in combination with brute forcing.
       Additionally, the program offers a dictionary method for determining
       the WEP key. For cracking WPA/WPA2 pre-shared keys, a wordlist (file
       or stdin) or an airolib-ng has to be used.
```

This opens the manual for aircrack-ng, providing you with more detailed information than the `help` screen. You can scroll through this manual file using the ENTER key, or you can page up and down using the PG DN and PG UP keys, respectively; you can also use the arrow keys. To exit, simply enter **`q`** (for quit), and you'll return to the command prompt.

Finding Stuff

Until you become familiar with Linux, it can be frustrating to find your way around, but knowledge of a few basic commands and techniques will go a long way toward making the command line much friendlier. The following commands help you locate things from the terminal.

Searching with locate

Probably the easiest command to use is `locate`. Followed by a keyword denoting what it is you want to find, this command will go through your entire filesystem and locate every occurrence of that word.

To look for aircrack-ng, for example, enter the following:

```
kali >locate aircrack-ng
/usr/bin/aircrack-ng
/usr/share/applications/kali-aircrack-ng.desktop
/usr/share/desktop-directories/05-1-01-aircrack-ng.directory
--snip--
/var/lib/dpkg/info/aircrack-ng.md5sums
```

The `locate` command is not perfect, however. Sometimes the results of `locate` can be overwhelming, giving you too much information. Also, `locate` uses a database that is usually only updated once a day, so if you just created a file a few minutes or a few hours ago, it might not appear in this list until the next day. It's worth knowing the disadvantages of these basic commands so you can better decide when best to use each one.

Finding Binaries with whereis

If you're looking for a binary file, you can use the `whereis` command to locate it. This command returns not only the location of the binary but also its source and man page if they are available. Here's an example:

```
kali >whereis aircrack-ng
aircarck-ng: /usr/bin/aircarck-ng /usr/share/man/man1/aircarck-ng.1.gz
```

In this case, `whereis` returned just the aircrack-ng binaries and man page, rather than every occurrence of the word *aircrack-ng*. Much more efficient and illuminating, don't you think?

Finding Binaries in the PATH Variable with which

The `which` command is even more specific: it only returns the location of the binaries in the `PATH` variable in Linux. We'll look more closely at the `PATH` variable in Chapter 7, but for now it's sufficient to know that `PATH` holds the directories in which the operating system looks for the commands you execute at the command line. For example, when I enter `aircrack-ng` on the command line, the operating system looks to the `PATH` variable to see in which directories it should look for aircrack-ng:

```
kali >which aircrack-ng
/usr/bin/aircrack-ng
```

Here, `which` was able to find a single binary file in the directories listed in the `PATH` variable. At minimum, these directories usually include */usr/bin*, but may include */usr/sbin* and maybe a few others.

Performing More Powerful Searches with find

The `find` command is the most powerful and flexible of the searching utilities. It is capable of beginning your search in any designated directory and looking for a number of different parameters, including, of course, the filename but also the date of creation or modification, the owner, the group, permissions, and the size.

Here's the basic syntax for `find`:

```
find directory options expression
```

So, if I wanted to search for a file with the name *apache2* (an open source web server) starting in the root directory, I would enter the following:

```
kali >find /❶ -type f❷ -name apache2❸
```

First I state the directory in which to start the search, in this case `/` ❶. Then I specify which type of file to search for, in this case `f` for an ordinary file ❷. Last, I give the name of the file I'm searching for, in this case `apache2` ❸.

My results for this search are shown here:

```
kali >find  / -type f -name apache2
/usr/lib/apache2/mpm-itk/apache2
/usr/lib/apache2/mpm-event/apache2
/usr/lib/apache2/mpm-worker/apache2
/usr/lib/apache2/mpm-prefork/apache2
/etc/cron.daily/apache2
/etc/logrotate.d/apache2
/etc/init.d/apache2
/etc/default/apache2
```

The `find` command started at the top of the filesystem (`/`), went through every directory looking for *apache2* in the filename, and then listed all instances found.

As you might imagine, a search that looks in every directory can be slow. One way to speed it up is to look only in the directory where you would expect to find the file(s) you need. In this case, we are looking for a configuration file, so we could start the search in the */etc* directory, and Linux would only search as far as its subdirectories. Let's try it:

```
kali >find /etc -type f -name apache2
/etc/init.d/apache2
/etc/logrotate.d/apache2
/etc/cron.daily/apache2
/etc/default/apache2
```

This much quicker search only found occurrences of *apache2* in the */etc* directory and its subdirectories. It's also important to note that unlike some other search commands, `find` displays only *exact* name matches. If the

file *apache2* has an extension, such as *apache2.conf*, the search will *not* find a match. We can remedy this limitation by using *wildcards*, which enable us to match multiple characters. Wildcards come in a few different forms: `*` `.` `,` `?` and `[]`.

Let's look in the */etc* directory for all files that begin with *apache2* and have any extension. For this, we could write a `find` command using the following wildcard:

```
kali >find /etc -type f -name apache2.\*
/etc/apache2/apache2.conf
```

When we run this command, we find that there is one file in the */etc* directory that fits the `apache2.*` pattern. When we use a period followed by the * wildcard, the terminal looks for any extension after the filename *apache2*. This can be a very useful technique for finding files where you don't know the file extension.

When I run this command, I find two files that start with *apache2* in the */etc* directory, including the *apache2.conf* file.

A QUICK LOOK AT WILDCARDS

Let's say we're doing a search on a directory that has the files *cat, hat, what,* and *bat*. The `?` wildcard is used to represent a single character, so a search for `?at` would find *hat, cat,* and *bat* but not *what,* because *at* in this filename is preceded by two letters. The `[]` wildcard is used to match the characters that appear inside the square brackets. For example, a search for `[c,b]at` would match *cat* and *bat* but not *hat* or *what*. Among the most widely used wildcards is the asterisk (*), which matches any character(s) of any length, from none to an unlimited number of characters. A search for `*at`, for example, would find *cat, hat, what,* and *bat*.

Filtering with grep

Very often when using the command line, you'll want to search for a particular keyword. For this, you can use the `grep` command as a filter to search for keywords.

The `grep` command is often used when output is piped from one command to another. I cover piping in Chapter 2, but for now, suffice it to say that Linux (and Windows for that matter) allows us to take the *output* of one command and send it as *input* to another command. This is called *piping*, and we use the | command to do it (the | key is usually above the ENTER key on your keyboard).

The `ps` command is used to display information about processes running on the machine. We cover this in more detail in Chapter 6, but for this

example, suppose I want to see all the processes running on my Linux system. In this case, I can use the ps (processes) command followed by the aux switches to specify which process information to display, like so:

```
kali >ps aux
```

This provides me with a listing of *all* the processes running in this system—but what if I just want to find one process to see if it is running?

I can do this by piping the output from ps to grep and searching for a keyword. For instance, to find out whether the apache2 service is running, I would enter the following.

```
kali >ps aux | grep apache2
root  4851 0.2 0.7 37548  7668 ?  Ss  10:14  0:00   /usr/sbin/apache2 -k start
root  4906 0.0 0.4 37572  4228 ?  S   10:14  0:00   /usr/sbin/apache2 -k start
root  4910 0.0 0.4 37572  4228 ?  Ss  10:14  0:00   /usr/sbin/apache2 -k start
--snip--
```

This command tells Linux to display all my services and then send that output to grep, which will look through the output for the keyword *apache2* and then display only the relevant output, thus saving me considerable time and my eyesight.

Modifying Files and Directories

Once you've found your files and directories, you'll want to be able to perform actions on them. In this section, we look at how to create files and directories, copy files, rename files, and delete files and directories.

Creating Files

There are many ways to create files in Linux, but for now we'll just look at two simple methods. The first is cat, which is short for *concatenate*, meaning to combine pieces together (not a reference to your favorite domesticated feline). The cat command is generally used for displaying the contents of a file, but it can also be used to create small files. For creating bigger files, it's better to enter the code in a text editor such as vim, emacs, leafpad, gedit, or kate and then save it as a file.

Concatenation with cat

The cat command followed by a filename will display the contents of that file, but to create a file, we follow the cat command with a *redirect*, denoted with the > symbol, and a name for the file we want to create. Here's an example:

```
kali >cat > hackingskills
Hacking is the most valuable skill set of the 21st century!
```

When you press ENTER, Linux will go into *interactive mode* and wait for you to start entering content for the file. This can be puzzling because the prompt disappears, but if you simply begin typing, whatever you enter will go into the file (in this case, *hackingskills*). Here, I entered `Hacking is the most valuable skill set of the 21st century!`. To exit and return to the prompt, I press CTRL-D. Then, when I want to see what's in the file *hackingskills*, I enter the following:

```
kali >cat hackingskills
Hacking is the most valuable skill set of the 21st century!
```

If you don't use the redirect symbol, Linux will spit back the contents of your file.

To add, or *append*, more content to a file, you can use the `cat` command with a double redirect (>>), followed by whatever you want to add to the end of the file. Here's an example:

```
kali >cat >> hackingskills
Everyone should learn hacking
```

Linux once again goes into interactive mode, waiting for content to append to the file. When I enter `Everyone should learn hacking` and press CTRL-D, I am returned to the prompt. Now, when I display the contents of that file with `cat`, I can see that the file has been appended with `Everyone should learn hacking`, as shown here:

```
kali >cat hackingskills
Hacking is the most valuable skill set of the 21st century! Everyone should
learn hacking
```

If I want to *overwrite* the file with new information, I can simply use the `cat` command with a single redirect again, as follows:

```
kali >cat > hackingskills
Everyone in IT security without hacking skills is in the dark
kali >cat hackingskills
Everyone in IT security without hacking skills is in the dark
```

As you can see here, Linux goes into interactive mode, and I enter the new text and then exit back to the prompt. When I once again use `cat` to see the content of the file, I see that my previous words have been overwritten with the latest text.

File Creation with touch

The second command for file creation is `touch`. This command was originally developed so a user could simply *touch* a file to change some of its details, such as the date it was created or modified. However, if the file doesn't already exist, this command creates that file by default.

Let's create *newfile* with `touch`:

```
kali >touch newfile
```

Now when I then use `ls -l` to see the long list of the directory, I see that a new file has been created named *newfile*. Note that its size is `0` because there is no content in *newfile*.

Creating a Directory

The command for creating a directory in Linux is `mkdir`, a contraction of *make directory*. To create a directory named *newdirectory*, enter the following command:

```
kali >mkdir newdirectory
```

To navigate to this newly created directory, simply enter this:

```
kali >cd newdirectory
```

Copying a File

To copy files, we use the `cp` command. This creates a duplicate of the file in the new location and leaves the old one in place.

Here, we'll create the file *oldfile* in the root directory with `touch` and copy it to */root/newdirectory*, renaming it in the process and leaving the original *oldfile* in place:

```
kali >touch oldfile
kali >cp oldfile  /root/newdirectory/newfile
```

Renaming the file is optional and is done simply by adding the name you want to give it to the end of the directory path. If you don't rename the file when you copy it, the file will retain the original name by default.

When we then navigate to *newdirectory*, we see that there is an exact copy of *oldfile* called *newfile*:

```
kali >cd newdirectory
kali >ls

newfile   oldfile
```

Renaming a File

Unfortunately, Linux doesn't have a command intended solely for renaming a file, as Windows and some other operating systems do, but it does have the `mv` (move) command.

The `mv` command can be used to move a file or directory to a new location or simply to give an existing file a new name. To rename *newfile* to *newfile2*, you would enter the following:

```
kali >mv newfile newfile2
kali >ls
oldfile newfile2
```

Now when you list (`ls`) that directory, you see *newfile2* but not *newfile*, because it has been renamed. You can do the same with directories.

Removing a File

To remove a file, you can simply use the `rm` command, like so:

```
kali >rm newfile2
```

If you now do a long listing on the directory, you can confirm that the file has been removed.

Removing a Directory

The command for removing a directory is similar to the `rm` command for removing files but with `dir` (for directory) appended, like so:

```
kali >rmdir newdirectory
rmdir:failed to remove 'newdirectory': Directory not empty
```

It's important to note that `rmdir` will not remove a directory that is not empty, but will give you a warning message that the "directory is not empty," as you can see in this example. You must first remove all the contents of the directory before removing it. This is to stop you from accidentally deleting objects you didn't intend to delete.

If you do want to remove a directory and its content all in one go, you can use the `-r` switch after `rm`, like so:

```
kali >rm -r newdirectory
```

Just a word of caution, though: be wary of using the `-r` option with `rm`, at least at first, because it's very easy to remove valuable files and directories by mistake. Using `rm -r` in your home directory, for instance, would delete every file and directory there—probably not what you were intending.

Go Play Now!

Now that you have some basic skills for navigating around the filesystem, you can play with your Linux system a bit before progressing. The best way to become comfortable with using the terminal is to try out your newfound skills right now. In subsequent chapters, we will explore farther and deeper into our hacker playground.

EXERCISES

Before you move on to Chapter 2, try out the skills you learned from this chapter by completing the following exercises:

1. Use the `ls` command from the root (/) directory to explore the directory structure of Linux. Move to each of the directories with the `cd` command and run `pwd` to verify where you are in the directory structure.
2. Use the `whoami` command to verify which user you are logged in as.
3. Use the `locate` command to find wordlists that can be used for password cracking.
4. Use the `cat` command to create a new file and then append to that file. Keep in mind that > redirects input to a file and >> appends to a file.
5. Create a new directory called *hackerdirectory* and create a new file in that directory named *hackedfile*. Now copy that file to your */root* directory and rename it *secretfile*.