

**Save 10%
on Exam
Voucher**

See Inside

PEARSON IT
CERTIFICATION



Practice
Tests



Flash
Cards



Review
Exercises



Study
Planner

Cert Guide

Advance your IT career with hands-on learning

CompTIA®

Linux+

XK0-005



ROSS BRUNSON

CompTIA® Linux+ XK0-005 Cert Guide

Companion Website and Pearson Test Prep Access Code

Access interactive study tools on this book's companion website, including practice test software, review exercises, Key Term flash card application, a study planner, and more!

To access the companion website, simply follow these steps:

1. Go to **www.pearsonitcertification.com/register**.
2. Enter the **print book ISBN: 9780137866885**.
3. Answer the security question to validate your purchase.
4. Go to your account page.
5. Click on the **Registered Products** tab.
6. Under the book listing, click on the **Access Bonus Content** link.

When you register your book, your Pearson Test Prep practice test access code will automatically be populated with the book listing under the Registered Products tab. You will need this code to access the practice test that comes with this book. You can redeem the code at **PearsonTestPrep.com**. Simply choose Pearson IT Certification as your product group and log in to the site with the same credentials you used to register your book. Click the **Activate New Product** button and enter the access code. More detailed instructions on how to redeem your access code for both the online and desktop versions can be found on the companion website.

If you have any issues accessing the companion website or obtaining your Pearson Test Prep practice test access code, you can contact our support team by going to **pearsonitp.ehelp.org**.

- As a secure replacement for file transfer methods, such as FTP or RCP. On the client side, the **sftp** command is used to replace **ftp**, and the **scp** command is used to replace **rcp**.
- As a secure replacement for remote execution methods, like **rsh**. On the client side, the **ssh** command is used for this feature.

NOTE SSH is covered in detail in Chapter 11, “Using Remote Connectivity for System Management.”

Key Topic

Using curl and wget

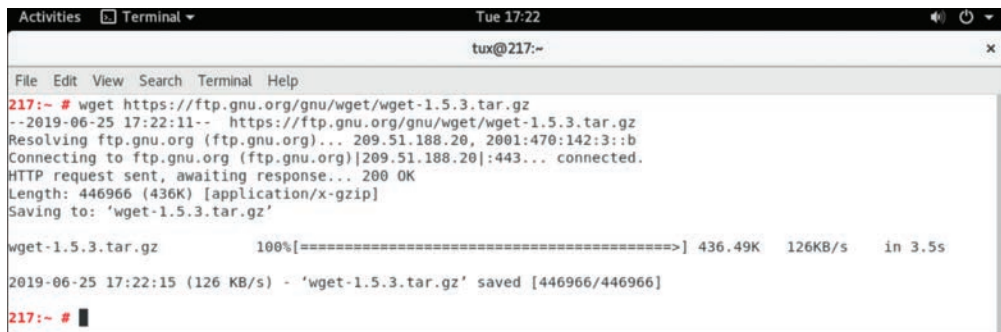
When you want to script the download of a package or file from a remote source as part of an update or script that will use that file, you can use the **wget** and **curl** commands.

For example, to get a file from a remote location without a need for further interaction, you use the following command:

```
# wget http://ftp.gnu.org/gnu/wget/wget-1.5.3.tar.gz
```

This command downloads the latest (at the time of publication) version of the source code for the **wget** command, as demonstrated in Figure 5-8.

Key Topic



```

Activities  Terminal  Tue 17:22
tux@217:~
File Edit View Search Terminal Help
217:~ # wget https://ftp.gnu.org/gnu/wget/wget-1.5.3.tar.gz
--2019-06-25 17:22:11-- https://ftp.gnu.org/gnu/wget/wget-1.5.3.tar.gz
Resolving ftp.gnu.org (ftp.gnu.org)... 209.51.188.20, 2001:470:142:3::b
Connecting to ftp.gnu.org (ftp.gnu.org)|209.51.188.20|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 446966 (436K) [application/x-gzip]
Saving to: 'wget-1.5.3.tar.gz'

wget-1.5.3.tar.gz      100%[=====] 436.49K  126KB/s  in 3.5s

2019-06-25 17:22:15 (126 KB/s) - 'wget-1.5.3.tar.gz' saved [446966/446966]

217:~ #
  
```

Figure 5-8 Downloading software with the **wget** command

There is a lot of confusion about when to use **curl** instead of **wget** to download a file. While these two commands are very similar in function, **curl** is typically used to both download and execute a script file, such as in situations of automated installation of virtual machine or container environments.

Simply enough, the **wget** source code file previously downloaded can also be downloaded with the **curl** command, as demonstrated in Figure 5-9.

```

Activities  Terminal  Tue 17:41
tux@217:~
File Edit View Search Terminal Help
217:~ # curl -O https://ftp.gnu.org/gnu/wget/wget-1.5.3.tar.gz
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           %             Dload  Upload    Total   Spent    Left   Speed
100 436k 100 436k  0     0 60902      0  0:00:07  0:00:07 --:--:-- 114k
217:~ # ls wget-1.5.3.tar.gz
wget-1.5.3.tar.gz
217:~ #

```

Figure 5-9 Downloading software with the **curl** command

The **curl** command needs to have either the **-O** option or the **--remote-name** option specified in order to download a file like **wget** does; otherwise, the file is sent to the standard output/console and is not created as a file on the disk.

The nc Command

The man page of the **nc** command (also referred to as the **netcat** command) provides an excellent summary of the **nc** command:

The **nc** (or **netcat**) utility is used for just about anything under the sun involving TCP or UDP. It can open TCP connections, send UDP packets, listen on arbitrary TCP and UDP ports, do port scanning, and deal with both IPv4 and IPv6. Unlike **telnet(1)**, **nc** scripts nicely and separates error messages onto standard error instead of sending them to standard output, as **telnet(1)** does with some.

There are quite a few uses for the **nc** command. For example, suppose you want to know whether a specific port is being blocked by your company firewall before you bring online a service that makes use of this port. On the internal server, you can have the **nc** command listen for connections on that port:

```
# nc -l 3333
```

You should end up with a blank line below the **nc** command. Next, on a remote system outside your network, you could run the following **nc** command to connect (replacing *server* with the resolvable hostname or IP address of the local system):

```
# nc server 3333
```

If the connection is established, you see a blank line under the **nc** command line. If you type something on this blank line and press the **Enter** key, then what you typed appears below the **nc** command on the server. Actually, the communications work both ways: What you type on the server below the **nc** command appears on the client as well.

The following are some useful options to the **nc** command:

- **-w**: This option is used on the client side to close a connection automatically after a timeout value is reached. For example, **nc -w 30 server 333** closes the connection 30 seconds after it is established.
- **-6**: Use this option to enable IPv6 connections.
- **-k**: Use this option to keep the server process active, even after the client disconnects. The default behavior is to stop the server process when the client disconnects.
- **-u**: Use this option to use UDP connections rather than TCP connections (the default). This is important for correctly testing firewall configurations, as a TCP port might be blocked while the UDP port might not be blocked.

You can also use the **nc** command to display open ports, similar to the way you use the **netstat** command:

```
# nc -z localhost 1000-4000
Connection to localhost 3260 port [tcp/iscsi-target] succeeded!
Connection to localhost 3333 port [tcp/dec-notes] succeeded!
```

The **-z** option can also be used for port scanning on a remote host.

NOTE There is one feature of the **nc** command that I don't expect you will see on the exam; however, it is a useful technique for transferring all sorts of data. Assuming that the transfer is from the client to the server, on the server, you use the following format:

```
nc -l 3333 | cmd
```

And on the client, you use this format:

```
cmd | nc server 3333
```

For example, you can transfer an entire **/home** directory structure from the client to the server, using the **tar** command, by first executing the following on the server:

```
nc -l 333 | tar xvf -
```

Then on the client, you execute the following command:

```
tar cvf - /home | nc server 333
```

The client merges the contents of the **/home** directory structure into a tarball. The **-** tells the **tar** command to send this output to standard output. The data is sent to the server via the client's **nc** command, and then the server's **nc** command sends this data to the **tar** command. As a result, the **/home** directory from the client is copied into the current directory of the server.

This is just one technique of many for using this powerful feature of the **nc** command.

Using rsync

This was covered in the *Everything and the Kitchen rsync* section of Chapter 2, “Managing Files and Directories.”

Using scp

This was covered in the *Using the scp Command* section of Chapter 2.

Summary

In this chapter you learned a lot about how to configure and manage your network interfaces, how to configure and manage name resolution, including for local and remote hosts and domains, how to monitor and troubleshoot your network traffic, and how to use remote networking tools to transfer data from system to system, securely and with proper data integrity.

Exam Preparation Tasks

As mentioned in the section “Goals and Methods” in the Introduction, you have a couple of choices for exam preparation: the exercises here, Chapter 23, “Final Preparation,” and the exam simulation questions in the Pearson Test Prep Software Online.

Review All Key Topics

Review the most important topics in this chapter, noted with the Key Topic icon in the outer margin of the page. Table 5-3 lists these key topics and the page number on which each is found.



Table 5-3 Key Topics for Chapter 5

Key Topic Element	Description	Page Number
Figure 5-1	Utilities obsoleted by the <code>iproute2</code> suite of tools	189
Example 5-1	Example ip addr show Command Output	189
Table 5-2	ss Command Options	192
Paragraph	NetworkManager nmcli command	192
Paragraph	The ifconfig Command	195
Paragraph	Using the arp command to view or change the ARP table	198
List	Local system name configuration files	200
List	hostnamectl command hostname levels	204
Example 5-4	Example dig Command Output	205
Paragraph	Determining if a host is functioning	208
Paragraph	Using the netstat command to view the routing table	212
Section	The tcpdump Command	216
Section	Using <code>curl</code> and wget	218
Figure 5-8	Downloading software with the wget command	218

Define Key Terms

Define the following key terms from this chapter and check your answers in the glossary:

interface, iproute2, **ip**, **ss**, **nmcli**, **ifconfig**, **ifcfg**, **hostname**, **arp**, **route**, **network-scripts**, **resolv.conf**, **nsswitch.conf**, **resolvectl**, **hostnamed**, **hostnamectl**, **dig**, **nslookup**, **whois**, **traceroute**, **ping**, **mtr**, **netstat**, Wireshark, **tcpdump**, Secure Shell (SSH), **wget**, **curl**, **nc**, **rsync**, **scp**

Review Questions

The answers to these review questions are in Appendix A.

1. You wish to retrieve an RPM package file from a remote package repository, and you have the full URL of the required file. Which command do you use? (Choose two.)
 - a. **wget**
 - b. **dload**
 - c. **getfile**
 - d. **curl**
2. Which SSH command is designed to replace the **telnet** command?
 - a. **rlogin**
 - b. **sftp**
 - c. **scp**
 - d. **ssh**
3. Which SSH command is designed to function as a secure equivalent to the **rcp** command?
 - a. **rlogin**
 - b. **sftp**
 - c. **scp**
 - d. **ssh**
4. Which of the following commands display all of the routers that a packet travels through to get to the destination? (Choose all that apply.)
 - a. **traceroute**
 - b. **tracehop**
 - c. **mtr**