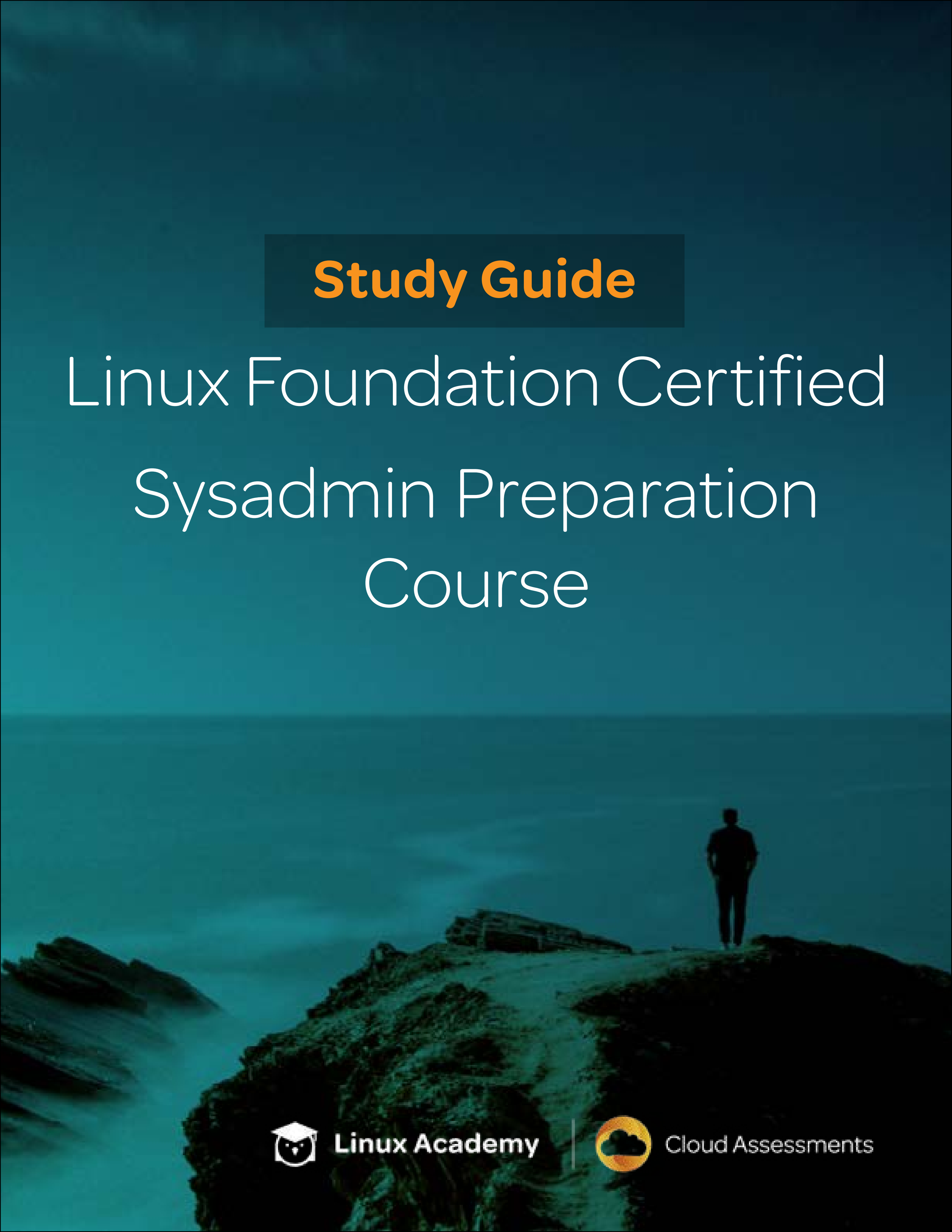

Study Guide

Linux Foundation Certified Sysadmin Preparation Course

Linux Academy

Cloud Assessments

Contents

The LFCS exam is centered around domains or competencies. The videos in the course explain and demonstrate these concepts. This guide contains notes about those videos, the commands used in them, brief summaries, and other information designed aid you in your studies. It is divided into sections, one for each video, and is intended to be a companion document that supplements the videos and labs in the course. But it is by no means a complete solution for studying for the exam. We strongly recommend that you view all the videos, review flashcards, take the quizzes, and perform the labs. Above all, practice your skills prior to taking the LFCS exam. Good luck@

Essential Commands

Essential commands account for 25% of the score on the exam.

Log in to Local and Remote Graphical and Text Mode Consoles

Both Ubuntu/Debian and CentOS/Redhat

VNC Viewer is a tool used to connect to a VNC Server which allows for remote operation of a Linux distribution in graphics mode. It is similar to other technologies such as Remote Desktop for Windows. VNC ordinarily runs over tcp/ip ports starting at 5800, but can be configured.

Secure Shell or ssh is used to connect to remote servers at a command line, ordinarily over port 22. The connection is encrypted, and while it usually relies on username/password authentication, it can be configured instead to make use of user-generated keys for authentication purposes.

How to master this skill: Practice! Fortunately, connecting via SSH is usually the first step to doing almost anything else. You will become adept at this naturally, throughout your studies in this course.

Search for Files

Remember: Linux is case sensitive! This means that “TEST”, “test”, “Test”, and “TesT” all register as completely different.

`find –help` for all of the `find` command options

Remember it also has a `man` entry:

```
man find
```

Examples:

Find all files in the entire system that are named test.txt:

```
find / -name “test.txt”
```

Find all files in the /etc directory not named test.txt:

```
find /etc/ -not -name “test.txt”
```

Find all character devices on the system:

```
find / -type c
```

Find all directories called log:

```
find / -type d -name “log”
```

Find all files in the /usr/bin directory or subdirecties that are larger than 27,000 bytes:

```
find /usr/bin -size +27000c
```

or

```
find /usr/bin -size +27k
```

Find all files in the /usr/bin directory or subdirectories that are larger than 1 megabyte:

```
find /usr/bin/ -size 1M
```

Find all files created more than a day ago:

```
find / -mtime 1
```

Find all files create less than a day ago:

```
find / -mtime -1
```

Find all files owned by the user named “chad”:

```
find / -user chad
```

Find all files in the /etc directory owned by the user root, and paginate the results:

```
find /etc/ -user root | more
```

Find all files in the /usr/bin directory with user permissions of 755:

```
find /usr/bin -perm 755
```

Find all files called test.txt and set their permissions to 700:

```
find / -name “test.txt” -exec chmod 700 {} \;
```

Philosophy: In Linux, the general rule is “No news is good news.” This means that when a command completes successfully, there will not likely be any command line confirmation or other feedback. If you don’t get a response from running a command, it probably completed exactly as you expected it to.

The more command will cause the output to pause after it fills a page and wait for you to hit the spacebar before continuing. There is also a less command that will allow you to scroll back and forth through the output using arrow keys.

The chmod command changes a file’s permissions and will be covered in a later lesson.

The which command allows you to find the location of a command that will be executed if invoked. This is useful for locating the executable file, especially if the behavior is not what you expected.

```
which
```

For instance, to find out which python executable would be invoked:

```
which python
```

The locate command can also help you find files but relies on a database rather than looking directly at the file system. The two can easily be out of sync. To synchronize locate’s database, run the command (as root) updatedb.

```
locate
```

How to Master this Skill: Practice! During your studies in this course, you will have many opportunities to practice hunting down files. Remember the find command and use it when possible. There is also a lab available that covers this topic.

Evaluate and Compare Basic File System Features and Options

This section is the only slide-based lecture in this class. The slides are available in the download section – where you found this guide – for your review.

Compare and Manipulate File Content and Use Input-Output Redirection

The cat tool will read file contents to standard out, which is usually the screen.

The more command pauses output.

The pipe character takes the standard output from one command and passes it to another command. For example, to send the output from cat shoppinglist through more:

```
cat shoppinglist | more
```

or

```
more shoppinglist
```

The less command will paginate, similar to more, but it will also allow you to use your up and down arrow keys to scroll backward and forward through the file.

```
cat shoppinglist | less
```

or

```
less shoppinglist
```

The sort command sorts content passed to it on standard in and places the output on standard out.

```
sort shoppinglist
```

To sort in reverse order:

```
sort -r shoppinglist
```

The cat tool's name is short for concatenate. You can combine the output of multiple files like this:

```
cat file1 file2
```

You can achieve complex results by combining commands:

```
cat file1 file2 | sort > file1and2
```

The > and < symbols allow you to send the standard output to a particular file on disk. The above example would have sorted the two files called file1 and file2 together and created a file in the current working directory called file1and2.

The fmt command (format) will clean up and format text.

The nl command counts the number of lines in a file, and also will number each line in a file for reference.

The cut command will remove a specified delimiter and write out a part to standard output.

```
One;Two

Three]Four
```

If we execute the following on the above file called numbers,

```
cat numbers | grep “;” | cut -d “;” -f1
```

The output would be One.

If we execute the following on the above file cat numbers | grep “;” | cut -d “;” -f2, then the output would be Two.

The command cat numbers | grep “]” | cut -d “]” -f1 yields Three.

And I leave it as an exercise to you to predict what happens with:

```
cat numbers | grep “]” | cut -d “]” -f2
```

The >> symbol appends to the end of a file instead of creating a new one. It will create a new file if none exists, but otherwise, it will simply add the content to the end of the file.

```
echo “One more line” >> bigfile.txt
```

In the above example, the line will be appended to the file called bigfile.txt, unless it does not exist, in which case it will create the file and add the line to it just like > would.

Philosophy: Linux is a series of small, simple commands which, when combined, result in powerful emergent tools.

Remember: The man pages are your friends! They’re the standard method of documentation, and we’ll talk more about them later.

Analyze text using basic regular expressions

Some basic regular expression help:

Token	**Meaning**
^	Start of line
$	End of line
\<	Start of word
\>	End of word
\d	Digit
\D	Not a digit
[Aa1]	Either matches on A, a, or 1
[A-Z] or [:upper:]	Matches on any capital letter
[a-z] or [:lower:]	Matches on any lower case letter
[^lmno]	Matches on any character but l, m, n. or o.
\$	Matches a literal dollar sign

How to Master this Skill: Lots of practice. Regular expressions almost require a different kind of thinking and are both loved and loathed by those who use them.

Archive, Backup, Compress, Unpack, and Uncompress Files

The cp command allows you to directly copy files from one location to another.

```
cp
```

Such as:

```
cp /home/user/test.txt /opt/test.txt
```

The scp command allows you to copy files securely over the network to another host running an ssh server.

```
scp
```

Such as sending a file to another host:

```
scp /home/user/test.txt user@otherhost:/home/user/test.txt
```

Or retrieving a file from another host:

```
scp user@otherhost:/home/user/test2.txt /home/user/test2.txt
```

The rm command immediately deletes files.

```
rm unneededfile.txt
```

The tar command is the de facto standard for creating backups. To create an uncompressed backup:

```
tar cvf backupfile.tar /path/to/backup
```

To create a compressed backup:

```
tar cfzv backupfile.tar.gz /path/to/backup
```

Mnemonics for commonly used tar flags:

Flag	**Means**
c	create
v	verbose
f	file (not implied!)
z	zip (compress or uncompress)
x	Extract
t	List

The grep tool allows you to search for a particular output. It accepts input on standard in and puts only lines containing the pattern you are looking for back out.

```
tar tvf databackup.tar | grep searchingfor.txt
```

The gzip tool is a compression utility that will compress and uncompress files. Text files compress very efficiently, so zipping them up will result in a far smaller file, with the disadvantage that it will have to be uncompressed before it can be read. Files of other types will also be smaller, though with varying degrees of efficiency. By default, it will compress files “in place,” meaning it will take a file, compress it into the same name with a “.gz” appended to the end, and remove the original file.

```
gzip somefile.tar
```

Creates in a file in the directory called somefile.tar.gz, but the original somefile.tar is gone.

To create a compressed backup file, use the compression options in tar:

```
tar cvfz databackup.tar.gz /path/to/backup
```

To extract the contents of a compressed tar file to the current directory:

```
tar xvfz databackup.tar.gz
```

How to Master this Skill: Read the man pages and practice. Practice backing up your home directory daily, and remove old backups as they accumulate. Later, when you learn how, set up a script and/or a cron job to do these tasks.

Create, Delete, Copy and Move Files and Directories

The touch command quickly creates a zero-byte file.

```
touch /path/to/file
```

The cp command allows you to copy files from one location to another.

```
cp  <destination
```

The nano command is a simple text editor. On some Ubuntu systems, there is an alias to this for users of the older, deprecated pico editor.

The mkdir command allows you to create directories.

```
mkdir newdirectory
```

You can create several at once using the -p flag:

```
mkdir -p newdirectory/subdir1/subdir2
```

The mv command is used for moving files from one directory to another. It's also used for renaming files. Think of renaming as just moving a file from one name to another:

```
mv file1 /newdirectory/subdir1/

mv file1 /newdirectory/file.txt

mv
```

The rm command allows you to delete or remove files (and sometimes directories with the right flags and wildcards).

```
rm unneededfile
```

or, **beware**, a command like this will remove all files and directories in /home/user that start with "unneeded" and, if a subdirectory, remove it completely:

```
rm -rf /home/user/unneeded*
```

The rmdir command is for specifically removing empty directories. It will fail if there is anything in the directory. For your own safety, get in the habit of using rmdir to remove directories, and use those error messages to your advantage so you don't accidentally wipe out half (or more) of your important files.

How to Master this Skill: This will come naturally as you study, as file manipulations are at the heart of learning Linux, but still… Practice!

Create and Manage Hard and Soft Links

The directory structure on your hard drive is basically a list of named pointers (directory file names) to locations on the disk. A hard link is a direct pointer to that location, and a soft link is a link to another file name in the directory structure. A file on disk can have multiple hard links pointing to it so that you can edit the file from multiple directory locations.

The ln tool allows you to create both hard and soft links to files.

To create a hard link:

```
ln /path/to/originalfile /path/to/newlink
```

To create a soft link:

```
ln -s /path/to/originalfile /path/to/newlink
```

How to Master this Skill: Practice making both soft and hard links to commonly viewed/edited files in your home directory.

List, Set, and Change Standard File Permissions

The chmod tool allows you to set permissions on files.

Directories listed with the -la flag show privileges in three sets of characters like this:

```
drwxrwxrwx 24 owner group size date filename
```

The first character tells you whether the entry represents a directory or not:

```
d– – –
```

The next three characters tell you which of the read, write, and execute rights that the file’s owner has:

```
-rwx– –
```

The next three characters tell which of the read, write, and execute rights that the file’s group has:

```
–-rwx–
```

The next three characters tell which of the read, write, and execute rights that everyone else has:

```
– –-rwx
```

To change file permissions, you can use chmod with the character method like below.

To add group read and write access to a file:

```
chmod g+rw file
```

To summarize the character method, a truncated usage statement is presented below:

```
chmod [ugoa]*([-+=][rwx]*[ugo]
```

Or you can use octal mode. Many Linux Administrators have a version of this table on a sticky note on their monitors:

Value	**Meaning**
4	Read
2	Write
1	Execute

So if you want to give the owner full rights, the group read and write rights, and everyone else only read rights to a file, you'd use:

```
chmod 764 file
```

Justification:

	Math	**Characters**	**Meaning**
7	= 4 + 2 + 1	rwx	Full rights.
6	= 4 + 2	rw-	Read/Write
4	= 4	r–	Read only

How to master this skill: In a word, practice! Learn to count to 7 in binary will actually really help you:

Decimal	**Binary**	**rwx**	**Rights**
0	000	—	None (Common)
1	001	–x	Execute only (?! Makes no sense)
2	010	-w-	Write only (Limited usefulness)
3	011	-wx	Write/Execute (Rarely used)
4	100	r–	Read only (Common)
5	101	r-x	Read/Execute (Common)
6	110	rw-	Read/Write (Common)
7	111	rwx	Read/Write/Execute (Common)

Read and Use System Documentation

Read the man pages.

```
man
```

How to master this skill: When troubleshooting a problem, don't run off to ask someone a question (online or otherwise) until you've AT LEAST read and tried to understand the associated man page(s). In almost all online forums, it is considered lazy and bad form not to read the documentation before asking questions online. No one wants to do your homework for you!

Manage Access to the root Account

Protect your server. Don't log in as root and don't allow anyone else to, either. Don't share a single password. Use the sudo command to escalate your privileges. Change to the root user (su or sudo su only when necessary, and log out as soon as you've completed your tasks.

The sudo command allows you to "do" things as the "super user" that you ordinarily wouldn't have appropriate permissions to do.

```
sudo
```

Such as the mount command:

```
sudo mount /dev/sdb3 /mnt/myfiles
```

You'll be asked for YOUR password to ensure that it's you.

Use the su command to become another user, usually the root user.

```
su root  (#root is implied if you only type "su")
```

You will be prompted for the password belonging to the user whose identity you are attempting to assume.

The whoami command will return the name of the Linux user you are currently logged in as.

```
root@somehost$ whoami

root
```

To provide a user with access to use the sudo command, use visudo to edit the sudoers file:

```
visudo
```

Add the user like others in the file.

Alternatively, you can set up a group that has sudo privileges in the sudoers file, and you can then assign users to the group to dynamically alter privileges.

How to master this skill: Live it. Your servers are important, and you need to protect them. Take security best practices to heart and accept nothing less, even on your home lab machines. We play like we practice, so practice good habits!

Operation of Running Systems

This topic accounts for 20% of your score on the exam.

Boot, Reboot, and Shut Down a System Safely

The shutdown command allows you to safely shut down the system while notifying users of what's going on.

To power off a system now with a message of "System Maintenance – New Hard drive! Back online in half an hour or so!":

```
shutdown -h now "New Hard drive! Back online in half an hour or
so!"
```

Common (but not complete) usage (Always see man pages for complete documentation)

```
shutdown -[h - halt | r - reboot ] <when?>
```

Shortcut commands reboot and poweroff do exactly what you'd expect them to do.

Note that changing the running status of the system requires root privileges.

How to master this skill: Working with Linux, you'll be shutting down and rebooting a lot of machines, so this will come naturally over time. But yes, practice.

Boot or Change System into Different Operating Modes

System V Common Runlevels:	**What it does**
0	Shutdown / Halt
1	Single User Mode
2	Multi user, no networking
3	Multi user mode, full networking
4	Undefined. No one uses this. Seriously.
5	Multi user graphics mode, full networking
6	Reboot

The runlevel command will let you know which runlevel the system is currently at.

To specify a runlevel to boot a system to, hold down the **shift** key during boot. This will bring up the Grub 2 bootloader. Find the kernel you wish to install and append the appropriate number to the command. See the video for a demonstration of this process.

How to master this skill: Runlevels are important when you need to rescue a dying system, but are rarely used otherwise. That makes it extra important to… Practice! Do this often enough that it is second nature, and you'll be the one users call when they need a hero.

Install, Configure, and Troubleshoot Bootloaders

Bootloaders take the system from power on to operating system initialization, and get their name from the expression: Pull yourself up by your boot straps. It means to more or less get started doing something all by yourself, without help or support from anyone. That's what the bootloader does. With no help from an operating system, it makes sure that the operating system can access the resources it needs to initialize.

Grub 2 is the de facto entrenched bootloader in use by most Linux distributions as of the writing of this guide.

/boot/grub/grub.cfg is grub's main configuration file. It is autogenerated and should not be manually edited. It is autogenerated by the update-grub tool.

/etc/grub.d is a configuration directory used by the update-grub tool to generate the grub.conf file and also specified menu entries.

The sample grub code I created in the video is:

```
bash
#!/bin/sh -e
echo "displayed when update-grub is run"
cat << EOF
menuentry "Other Linux Partition" {
set root=(hd0,3)
linux /boot/vmlinuz
initrd /boot/initrd.img
}
```

How to master this skill: Install a lot of Linux on hardware, and you'll be a pro at this in no time. Yes, in short, practice.

Diagnose and Manage Processes

The top command will show you a real-time, live updated console containing processes, their CPU utilization, and lots of other great information.

The htop command offers similar functionality, with a few more interactive options and major cosmetic differences. Which of these tools you prefer is a matter of personal preference.

The ps command will allow you to find processes running on the current machine to identify their owner or pid (process id).

Common ps invocation:

```
ps aux
```

The kill command is used to send termination signals to running applications.

Here is a brief list of kill signals commonly used (not complete).

Kill Signal	**Meaning**
SIGHUP	Shut down normally
SIGINT	Interrupt what you're doing if you can, and shut down
SIGQUIT	Like sending a control-C from the command line to a process.
SIGKILL	Terminate immediately, Hard kill. For when the process is ignoring your subtle SIGHUP hints.

How to master this skill: Be curious about what is going on on your system. If it's a test system or virtual machine, try stopping processes and see what happens. This method of teaching yourself troubleshooting can be extremely satisfying.

Locate and Analyze System Log Files

Usually in /var/log, most relevant logs are located there. For application files, consult their documentation.

Cat, less, more, grep, and input-output redirection covered in a previous lesson are your best friends with log file analysis. Practice them every chance you get.

How to master this skill: Open a log file and read it! Be comfortable with this, because you'll spend a lot of time doing it as a Linux Administrator. Different applications have different formats, but this is largely cosmetic, and with practice, you'll soon be able to glean pertinent information quickly. My favorite tip for reading log files is to always start from the bottom. It's where the most recent additions are, and recent errors are easy to find there.

Schedule Tasks to Run at a Set Date and Time

A user's crontab contains a list of commands a user wishes run at very particular times. These commands are run by a daemon called cron. To view a user's crontab, use this command (if you only want to see your own, you need not include the username):

```
crontab -u user -l
```

To delete a user’s entire crontab (all commands GONE!):

```
crontab -u saduser -r
```

To edit a user’s crontab:

```
crontab -u user -e
```

The format of the file is a line for each command and values for each row in the following table delimited by whitespace:

Position	**Meaning**
1	Minutes after the hour the scheduled command is to run.
2	Hour of the day
3	Day of the month (1 - 31)
4	Month (1 - 12)
5	Day of the week (0-6) (Sunday = 0)
6+	The command

Values are either numeric or a * to indicate “All.”

Here is an example of running a script called “/opt/do-the-things.sh” at 8:00 AM every weekday in March:

```
0 8 * 3 1,2,3,4,5 /opt/do-the-things.sh
```

How to master this skill: Automate your life! Cron jobs can run scripts and commands and can eliminate many regular tasks from your routine. Every time you do something simple at a terminal you should be thinking 1) can I script this? and 2) can I then create a cron job out of this script?

Verify Completion of Scheduled Jobs

The logs always have your answer. Again, practice reading them.

How to master this skill: Again, if you’ve automated your life, you’ll occasionally want to spy on your cronjob minions to be certain they’re keeping up day-to-day. That’s a long-winded way of saying practice, and it goes along with the previous section.

Update and Manage Software to Provide Required Functionality and Security Part 1: Ubuntu/Debian

On Debian/Ubuntu systems, the package management system is Aptitude and the two common tools used with it are apt and apt-get. Apt-get tends to receive preferential treatment because it doesn't do fancy stuff to the screen, thus making it ideal for inclusion in scripts.

Often, you'll need to update the repository cache by running an update:

```
apt update
```

or

```
apt-get update
```

And to update all out-of-date packages:

```
apt upgrade
```

or

```
apt-get upgrade
```

On most systems, you will need escalated privileges, so remember to sudo where necessary.

To install a new package, use:

```
apt install or apt-get install
```

To remove a package, you will usually want to remove everything associated with it, like so:

```
apt purge
```

or

```
apt-get purge
```

The dpkg tool is used when you already have a Debian package file (*.deb) that you want to query or install.

To install a package file, use the command:

```
dpkg -i somepackage.deb
```

Occasionally, bad things happen to the aptitude database, but you can generally clean it up with:

```
dpkg -a -configure
```

How to master this skill: If you play with a Linux system for longer than 3 minutes, you'll want to install some software. It becomes second nature quickly, and you'll rarely need to consult the man pages. But don't forget the man pages. In the beginning, you should be reading them for everything.

Update and Manage Software to Provide Required Functionality and Security Part 2: CentOS/Redhat

On Redhat and CentOS, the Redhat Package Management system is in place. The packages are .rpm files and the package management command, the equivalent to Debian's dpkg command is also called rpm. There is an apt type of system called yum.

To update the repository cache and packages on the system:

```
yum update.
```

This lets us know if a package is installed:

```
yum list
```

To find out if a packagename exists (wildcards are okay):

```
yum search
```

To install a package and any necessary dependencies:

```
yum install
```

The -y switch will answer yes to all questions so that we're not prompted during an install:

```
yum install -y
```

The yum-utils package includes some excellent additional tools, such as yumdownloader which fetches the package but does not install it.

The yum-list command will list all packages available in the database (grep is helpful here).

Display enabled yum repositories:

```
yum repolist
```

Display ALL yum repositories:

```
yum repolist all
```

The rpm tool is useful when you have a .rpm file that you'd like to install or query. To install a package from an rpm:

```
rpm -i somepackage.rpm
```

List all files installed by a particular package:

```
rpm -ql
```

Examine docs inside a package:

```
rpm -qdf
```

How to master this skill: Play with a Linux system enough and you'll want to install something. It becomes second nature quickly, and you'll rarely need to consult the man pages. But don't forget the man pages. In the beginning, you should be reading them for everything.

Verify the Integrity and Availability of Resources

Check your hard drives regularly with fsck, and your RAM with the memtest86+ on the grub menu. Fortunately, this isn't a skill most Linux administrators are called on to make use of often.

How to master this skill: Practice. Pretend things are wrong and check them.

Change kernel Runtime Parameters, Persistent and Non-Persistent

Since the kernel controls just about everything, sometimes you need to make changes to its parameters.

The sysctl tool is made for examining and changing kernel parameters at runtime. It examines the virtual process file system in /proc.

To change a system parameter until the next boot:

```
sysctl parameter.name=value
```

To change a system parameter permanently, you'll need to change the appropriate parameter by editing the appropriate file in /etc/sysctl.d/.

How to master this skill: Few parameters need changing frequently, but you'll sometimes need to make changes to IP packet forwarding. Consult the documentation for your kernel and find interesting parameters and customize your system.

Use Scripting to Automate System Maintenance Tasks

Scripting is a huge topic, and the video and this study guide will not be able to do more than scratch the surface of the possibilities. Here are some key points to remember as you get started:

1. Always start with a shebang. That tells the system how to interpret the script. They look like this:

```
#!/bin/bash
```

And point to an appropriate shell or interpreter.

2. Scripts are executed sequentially. The order of the commands matters.

3. Avoid relative paths like ../../home/. Instead, use absolute paths like /home. This way, if you ever decide to move your script, you don't have to go back in and fix it immediately just because it lives in a new directory.

4. Log it! Get in the habit of "echoing" important information to the screen or writing results to either your own log file or appending it to the system log file. This can be very useful for debugging larger, more complex scripts later on.

How to master this skill: Automate all the things! At first, write some scripts that take care of remembering pesky flags on complex commands for you. Once you get hooked on that, you'll start finding new ways to script minor annoyances out of your life.

Scripting Conditionals and Loops

Sometimes, you'll want to take specific actions only if certain conditions are met. A condition like "Only rename the media file if the log file says it converted successfully. If that directory exists, delete all the *.txt files in it," or some such is a simple example.

The if statement (which ends with a matching fi statement) defines a condition under which to undertake specific actions:

```
bash
if (condition) then:
  do-something;
fi
```

The while command allows you to do the same block of code over and over again until some condition is true. This code will execute at least once:

```
bash
while (something=true){
    ls -la >> thisfilemightgetlargequick
}
```

How to master this skill: Once again, imagine ways to make your life at the keyboard more comfortable. Start small, and work your way up.

Manage the Startup Process and Services (In Services Configuration)

In SystemD systems such as Ubuntu 14+, Centos, and RedHat systems, starting a service at boot is simply a matter of using the systemctl tool. Starting one of these services at boot is referred to as "enabling" it:

```
systemctl enable servicename
```

To gain information about a particular service on a systemd system, you'd use:

```
systemctl status servicename
```

In older systems using upstart for services management, you'd create an appropriate configuration script in /etc/init.

To gain status information on upstart systems, you'd type:

```
status servicename
```

How to master this skill: Starting and stopping services is a common task you'll perform as an administrator, so you'll have plenty of opportunities to practice it.

List and Identify SELinux/AppArmor File and Process Contexts

SELinux (Redhat/Centos) and AppArmor (Ubuntu/Debian) prevent applications on their respective systems from overstepping due to defects or malicious behavior.

SELinux

SELinux can be enabled/enforced, logged, or disabled.

To view all contexts on your system:

```
semanage fcontext -l
```

To find a specific context:

```
semanage fcontext -l | grep httpd
```

To view security context on a file:

```
ls -Z filename
```

Used without a file name the command would show the security context of every file in a directory.

To show security context on processes:

```
ps auxZ
```

AppArmor:

The aa-status tool will show you similar security contexts on AppArmor systems.

Profiles are stored in the /etc/apparmor.d directory and you can view these profiles and see how they are constructed.

Regular expression experience is helpful, so you might want to review the basic regular expression lesson.

ls -Z is usually available on Ubuntu systems, but does not work for AppArmor profiles. ps auxZ, however, does work.

How to master this skill: Practice, and don't disable these critical systems. Many systems administrators would prefer to disable SELinux or AppArmor than to learn to use it properly. Don't be one of those! If you're just getting started, then start out with passive or complain mode on production systems. It won't prevent anything horrible from happening, but it also won't prevent you from reconfiguring your services as necessary. It will just log complaints. Read your log files and as you get used to how things function, throw yourself a graduation party and fully enable the systems.

Identify the Component of a Linux Distribution That a File Belongs To

You can do this with your package management tools that we've already discussed in a previous lesson. Specifically, you can find out which package provides a particular file on Redhat and Centos systems:

```
rpm -qf
```

or

```
yum whatprovides
```

On Debian/Ubuntu machines, we'll use dpkg:

```
dpkg -S
```

How to master this skill: Get curious about the packages installed on your machine. As you study Linux, whenever you encounter a new command, read the man page. Figure out what package it comes from and if there's a newer version.

User and Group Management

User and Group management skills account for 10% of the exam.

Create, Delete, and Modify Local User Accounts

The tools for user creation are pretty straightforward: useradd and adduser. If you've got a lot of manual scripts you need to run or other manual work, then:

1. You should put that on your list of things to automate.

2. You should use the useradd command, which on many systems will not create the home directory automatically. The adduser command will ask you a set of questions about the user to get the new person all set up.

```
useradd
```

```
adduser
```

How to master this skill: Invite all your nerdier friends to log in to your practice Linux machine. Creating accounts is a common task, and it won't require MUCH practice to get the hang of it, but it will require some.

Create, Delete, and Modify Local Groups and Group Memberships

To add groups to a system, you can use either the groupadd or addgroup command, or you can edit the /etc/group file to create your group and designate its membership.

```
groupadd newgroup
```

or

```
addgroup newgroup
```

To add a user to a group, either manually edit the /etc/group file or use the usermod tool, carefully.

```
usermod -a newgroup olduser
```

Note that if you hastily read the usage, you'll see that -G GROUPS is supposed to add users to groups, so if you do a:

```
usermod -G newgroup olduser
```

You will accidentally wipe out ***all of that user's other group memberships***. Be careful, and make sure you understand what the man pages are telling you before you run commands.

How to master this skill: Again, read the man pages and practice. Groups are a wonderfully powerful and, in my humble opinion, an underused feature of Linux.

Manage System-Wide Environment Profiles

Environment variables are easy to use and easier to set. I use them in live demonstrations all the time because it's easier to remember things like REMOTE_IP than 10.9.185.99 when you're talking in front of an audience answering questions. The export command is a way to explicitly define an environment variable in your current shell.

```
export VARIABLE="value"
```

To unset a variable, either:

```
unset VARIABLE
```

or

```
export VARIABLE=''
```

Set persistent environment variables in .bashrc for a particular user, or system-wide in a script file located in the /etc/profiles.d directory.

How to master this skill: You learn what you use, and since I make extensive use of environment variables, editing those profile scripts doesn't scare me a bit. Practice, practice, practice.

Manage Template User Environment

The /etc/skel directory is designed to be a template of a standard set up for your users. You can place files like a standard .bashrc script (see the previous section for an application of this) or standard directory tree, or whatever you like. It beats having to do it manually and is less work than writing a script.

How to master this skill: This one is super easy because it almost does itself. Go in, poke around, make some changes. As long as you know what happens, you know when you need to use it.

Configure User Resource Limits

The /etc/security/limits.conf file is used to set limits on user resource utilization.

The file is arranged in lines, where each line defines some limit. The format of each line is as follows:

Order	Value Expected
1	Domain: Like the username, group, or wildcard that is being limited
2	Type: Either "hard" or "soft". Either the limit is absolute, or the user may change the value (up to a hard limit, if one exists)
3	What is being limited, such as the core file size, maximum data, maximum memory locked, etc. *See the man page for limits.conf for a complete list
4	Value – The value of the limit being set, in appropriate units

How to master this skill: On a test or practice Linux system, set limits for yourself. Initially, set them low so they are annoying and you have to change the file a lot. Once you've gotten the hang of that, set yourself some soft limits and practicing using the prlimit tool to make changes.

Manage User Privileges

The /etc/access.conf file allows or disallows users from logging into a machine and starting up a shell. You can thereby create a user that cannot log in remotely, or can only log in from specific hosts or networks.

The user, group, and "everyone" execute "bit" (permissions) on executable files and scripts determine who can execute any given command.

From our previous experience with file permissions, that's these permissions:

```
–x-x-x
```

In fact, you should review the entire lesson on file permissions if you can't convert the above to a three digit number. I'll wait…

How to master this skill: Start restricting your scripts to only be owner-runnable. Other people shouldn't be running your scripts anyway, unless you consciously permit them, so get in the habit of setting permission on your scripts to something like 766 or 744. Yes, do that math on those, too, and as a quick exercise, write out the permissions as they look in a directory listing and list what they mean. If you killed a forest by printing this out, do that here:

Permission 3xOctet	bit	Owner Read	Owner Write	Owner	Group Read	Group Write	Group	World Read	World Write	World
755										
744										

Configure PAM

PAM stands for Pluggable Authentication Modules. They allow the admin to change the way applications authenticate users.

PAM is configured EITHER by the file /etc/pam.conf or by all the files in a directory called /etc/pam.d. If this directory exists, then the /etc/pam.conf file is completely ignored.

The single configuration file method, while still valid, is rarely used. It's much easier to organize services by storing them in separate files within /etc/pam.d/.

The *@include* directive in a configuration file includes another configuration file in the current file, allowing system administrators to organize and develop their own shorthand through careful planning.

PAM Deals with 4 management tasks:

- AuthenticationManagement
- Account Management
- Session Management
- Password Management

How to master this skill: Install pam-aware applications as part of your studies, and then change how they authenticate. Practice, practice!

Networking

Networking skills account for 12% of the exam.

Configure Networking and Hostname Resolution Statically or Dynamically

The ifconfig command will give you information about your networking interfaces.

Debian/Ubuntu Systems

On older Debian/Ubuntu systems check /etc/network/interfaces file or the /etc/network/interfaces.d for a group of the configurations

Here is an example of a simple DHCP client configuration for an ethernet device:

```
bash
auto eth0
iface eth0 inet dhcp
```

Here's an example for a simple static host for an ethernet device:

```
bash
auto eth0
iface eth0 inet static
address 10.9.8.7
netmask 255.255.255.0
gateway 10.9.8.1
dns-search mydomain.com
dns-nameservers 8.8.8.8 8.8.4.4
```

To restart your network interface from the command line.

```
sudo ifdown eth0 && sudo ifup eth0
```

Tip: Be careful, you might lose your connection, and if you've configured anything wrong, things could be bad. Make sure you have a backup of the good configuration files someplace safe and on the machine and are accessible without networking. If you're feeling clever, set up a cron job to run at some point in the near future that reverts back to the old configuration files and reboots. If everything works, you can delete your cron job and pat yourself on the back. If everything fails though, then just wait until the ordained time, reconnect and try again, and congratulate yourself on how that could have gone **MUCH** worse.

Redhat/CentOS Systems

On Redhat and CentOS systems, the network is configured by scripts in a directory called /sysconfig/network-scripts.

Ordinarily, each interface has its own configuration file along the lines of ifcfg-ethX.cfg which is very similar in many respects to the /etc/network/interfaces.d files on an Ubuntu or Debian system.

Here is an example of a DHCP configured client:

```
bash
BOOTPROTO=dhcp
DEVICE=eth0
HWADDR=0a:67:42:8d:24:9e
ONBOOT=yes
TYPE=Ethernet
USERCTL=no
```

Here is an example of a static client:

```
bash
BOOTPROTO=none
DEVICE=eht0
HWADDR=0a:67:42:8d:24:9e
ONBOOT=yes
TYPE=Ethernet
IPADDR=10.9.8.7
PREFIX=24
GATEWAY=10.9.8.1
DNS1=10.9.8.53
DNS2=8.8.8.8
DNS3=8.8.4.4
```

The same warnings apply here as in the Debian/Ubuntu section; go back and read them if you are skipping around.

Restart the network with this:

```
systemctl restart network
```

How to master this skill: Put Linux on your laptop, and then go places to practice. Usually, you'll use DHCP, but you might talk a few friends into letting you use a static IP on their network and get used to changing files around.

Configure Network Services to Start Automatically at Boot

A lot of this material is the same as what's in a previous lesson on starting services at boot; we're just being specific about networking services this time. You should review that lesson if these concepts are unclear.

A telnet server was the progenitor of modern SSH. It allowed shell access across plain text connections, making it inherently extremely insecure, so its use is actively discouraged today. But it's an easy way to test network connectivity. If you choose to use this tool, you should forcibly disable it when you're not actively using it. And you should not use it at all on a machine connected directly to the internet, unless you like being a spam relay. (Hint: You don't.)

How to master this skill: Install Linux a lot of times. A lot - like 30 times at least. You should be a pro at installing your distribution of choice, but you should also look at others just for the troubleshooting practice. Configuring network services to start automatically at boot is something you usually only do once on a server – when you're first setting it up. Besides, you might find a distribution you like a lot. Some suggestions of ones to try (The test is on one of the first two, then they are in no particular order): Ubuntu, CentOS, Redhat, Debian, openSUSE, Alpine, Gentoo, Mint, Arch, Puppy, Elementary, Knoppix, Mandriva, and my very first distro: Slackware.

Implement Packet Filtering

Sometimes you want to stop anything on your system from processing packets that show up at your doorstep. They look like cute little orphan puppies but are really monsters ready to take over the whole system. The iptables tool puts you in control of which packets should be "let in," which should be "turned away," and which ones should just be ignored. That last is called the If the monster doesn't know anyone's home, maybe it will go away approach.

Just like commands in previous lessons, a misconfiguration with the iptables utility can lock you out of a system. Remember that you're probably connecting via a network, and if you do something like block port 22 (the ssh port), then you're going to have a hard time telling the server to do **ANYTHING**. Proceed with caution.

The ping tool allows you to send special packets to another host to see if that host can respond. The -c flag is used to limit the number of requests that the tool sends. You may use a host IP address or host name.

Send three ping requests to host 10.9.8.7:

```
ping -c 3 10.9.8.7
```

Continue sending ping requests to host www.linuxacademy.com until manually canceled:

```
ping www.linuxacademy.com
```

List of current policies:

```
iptables -L
```

Remove current rule set: (Danger Will Robinson…)

```
iptables -flush
```

Change the current “forward” policy to “Accept” everything:

```
iptables -P FORWARD ACCEPT
```

Reject all ping requests sent to device eth0 with an error message:

```
iptables -A INPUT -protocol icmp -in-interface eth0 -j REJECT
```

To drop all ping requests (stealth mode):

```
iptables -A INPUT -protocol icmp -in-interface eth0 -j DROP
```

How to master this skill: Practice. Practice carefully. This skill is probably better obtained by using a Linux virtual machine on your local laptop than by using one of the Linux Academy Cloud Servers. Mistakes on a local VM are easily rectified, but if you mess up one of your cloud servers, you might have to delete it and reinstall to regain access.

Start, Stop, and Check the Status of Network Services

If you’ve been going in order, you know how to start and stop services already, on both Ubuntu/Debian and Redhat/CentOS machines. Review those lessons before continuing here if you need to.

The `netstat` tool is a favorite of Linux administrators as it provides the process and port-level details about what is listening and communicating on your system, using what protocols.

Running `netstat` as root will give you additional information than it would if you’re just a user.

`netstat -a`: Shows the current status of everything port on your system

`netstat -at`: Shows the current status of only TCP ports and connections

`netstat -au`: Does the same for UDP

netstat -l: Lists TCP listeners

netstat -lu: Lists UDP listeners

netstat -s: Will show you lots of cool statistics

netstat -tp: Will include PID numbers with active Internet connections

Show the kernel routing table:

netstat -r: (similar to ip route list)

netstat -ie: Will list all the network interfaces, similar to ipconfig

netstat -c: Will continuously monitor and update every few moments - (Get out with Control-C)

How to master this skill: Whenever you install a new process that communicates over the network, use netstat to figure out which ports it is listening on, and then check your answer against what the documentation says it should be.

Statically Route IP Traffic

Network routing is hard. When it's down, don't panic. Just remember that networking is merely signals on a wire. A device sends a signal onto that wire. Then the signal moves along the wire to another device. That other device does something with the information carried by the signal, and then it usually decides which other wire to put that signal (packet) back out on. Take it slow and don't get frustrated.

IP routing has to do with the logic of deciding what to do with a given network packet in order to get it closer to its destination. Which interface do I use for sending the packet? It's a little like knowing whether to send a text message to your brother and call your mother on her land-line.

To view the kernel route table, you can make an ip route list, or you can use the older method of netstat -r (see the previous lesson).

This will show you which networks will be sent which traffic based on the destination host's IP address. A route table might look like this:

```
bash
10.9.185.0/24 dev eth0 proto kernel scope link src 10.9.185.148
10.9.185.1 dev eth0 proto dhcp scope link src 10.9.185.148 metric
100
```

This basically says if a packet is on the network, 10.9.185.0/24 (that slash 24 at the end means it's got a netmask of 255.255.255.0) then that's the local network, and it doesn't need a gateway – everything is right here.

The second line says "Okay, for everything else, send it to 10.9.185.1, because it needs routing. All I know about those hosts is that they are NOT "right here.""That gateway is often referred to as my "next hop" because it's the next place packets get sent on their way to their destination.

I could set up another network inside my home network at 10.9.185.0/24 and provide manual, static routing to it. Let's say I had another network of 192.168.1.0/24 that was just a few Raspberry Pis behind a router plugged into my home network at 10.9.185.31. I could add a route for those IP addresses on my host by typing:

```
ip route add 192.168.1.0/24 via 10.9.8.31 dev eth0
```

The ip command is dense. There's a lot to it. Take a look at the man page, and see that the routing portion is broken out into its own man page of ip-route. Take a look at THAT one. Don't let the syntax frighten you – just work through the usage one value at a time.

Remember, routing on any given machine comes down to which interface do I use to send this packet? We're only worrying about the next hop. After that, it's the next device's job to move our data on down the network.

How to master this skill: Play with applications that need routes when you're experimenting with them, like Kubernetes. If you're not ready to take THAT dive (Kubernetes is fun but pretty in-depth), then practice changing your own next hop, checking the results, and changing it back. Again, BE CAREFUL. If the machine you're connected to doesn't have a route to you and has no way to get back its old route, then your packets are just going off into the ether.

Synchronize Time Using Other Network Peers

Let's end this section on an easy note, huh? NTP, or network time protocol, is a way for you to synchronize the time on your servers with the time on other servers. Setups like that are usually synchronized to the atomic clock or some other super-accurate clock.

Most of what you need to do to get your time synchronized is:

1. Install ntpd
2. Configure ntpd
3. Drink 64 fluid ounces of water per day

(Okay, that has nothing to do with time synchronization; we just need those first two things. #3 isn't a bad idea, though.)

Install ntpd by using the package manager on your distribution on the package "ntp" – it's probably already installed.

Edit /etc/ntp.conf to contain pointers to servers of your choosing by adding pool lines to the configuration file.

How to master this skill: You should already have it mastered just by reading this section and watching the video. Install and configure it a few times, and you'll see that this is probably one of the easiest competencies on the LFCS exam.

Service Configuration

Service configuration skills account for 20% of the exam.

Note: In this section, we'll be presenting you with the steps and scripts we used to create the tutorial videos in a recipe type format. For discussion on these commands and configurations, see the associated video in the course.

Configure a Caching DNS Server

In this lesson, we install and configure BIND9 on CentOS 7.

```
bash
yum install bind bind-utils
nano /etc/named.conf
```

Alter the file such that the options section with the line allow-query looks like this:

```
allow-query { localhost; any };
```

And add this line immediately following the line above:

```
allow-query-cache { localhost; any; }
```

Check the permissions on the file /etc/named.conf. It should be owned by root and be assigned to the named group. The permissions on it should be 640. In short, an ls -la /etc/named.conf should return something that looks mostly like this (different dates and precise file sizes okay!):

```
-rw-r—-. 1 root named 1757 Apr 1 00:48 named.conf
```

ls -lZ should return:

```
-rw-r—-. 1 root named system_u:object_r:named_conf_t:s0 named.conf
```

If not, then we need to correct its SELinux context by:

```
semanage fcontext -a -t named_conf_t:s0 /etc/named.conf

semanage fcontext -a -t named_conf_t:s0 /etc/named.rfc1912.zones
```

Continue the process:

```
bash
named-checkconf /etc/named.conf
systemctl restart named
systemctl enable named
systemctl status named
```

Maintain a DNS Zone

Writing a zone file is truly a test of your patience. For the exam, you should do it a couple of times. If you aren't here for that, you might also check out zonefile.org, and they will do the heavy lifting for you.

Sample zone file and cheatsheet:

```
$ORIGIN=linuxacademy.com
$TTL=600

@  IN  SOA    dns1.linuxacademy.com     mail.linuxacademy.com (
        318   ;   Serial number -- update this every time
       21600  ;   Time to live in seconds - Refresh after 6 hours
        3600  ;   Retry after one hour
      604800  ;   Expires after a week
      86400 ) ;   Minimum time to live of a day

host-records IN    A    1.1.1.1
are-all-of   IN A   1.1.1.2
type-a       IN A   1.1.1.3

host10       IN A   1.1.1.10
host11       IN A   1.1.1.11

web-server     IN A    1.1.1.80
mail         IN A   1.1.1.99
dns1         IN A   1.1.1.53
dns2         IN A   1.1.1.54

cname-records IN CNAME type-a
point-to      IN CNAME are-all-of
other-records IN CNAME host-records

www            IN CNAME     web-server
            IN MX    10    mail
            IN NS    dns1.linuxacademy.com
            IN NS    dns2.linuxacademy.com
```

Connect to Network Shares

On the server:

```
bash
yum install nfs-utils
mkdir /share
chmod -R 755 /share
chown nfsnobody:nfsgroup /share
systemctl enable rpcbind
systemctl enable nfs-server
systemctl enable nfs-idmap
systemctl start rpcbind
systemctl start nfs-server
systemctl start nfs-idmap
```

On the server, configure the share and the client (you will need IP addresses for all clients you wish to grant access to):

Edit the /etc/exports file to look like this:

```
bash
/share 172.31.96.178 (ro,sync,no_root_squash,no_all_squash)
```

Restart the server to pick up the changes to the configuration files.

```
systemctl restart nfs-server
```

On the client:

```
bash
yum install nfs-utils
mkdir -p /mnt/remote
mount -t nfs 172.31.124.130:/share /mnt/remote
```

Test and verify, on client:

```
echo “Test File” > /mnt/remote/testfile.txt
```

```
cat /mnt/remote/testfile.txt
```

Test and verify, on server:

```
echo /share/testfile.txt
```

Configure Email Aliases

Using Postfix as the email server, we create a new file in /etc/postfix called aliases.

Send webmaster's email to the user named chad:

```
webmaster: chad
```

Send terry's email to both terry and boss:

```
terry: terry, boss
```

And so the whole file would read, simply:

```
webmaster: chad
terry: terry, boss
```

You must tell Postfix about the aliases changes:

```
postalias /etc/postfix/aliases
```

Configure SSH Servers and Clients

```
apt-get install openssh-server
```

Make desired configuration changes to the config file /etc/ssh/sshd_config.

To generate a key-pair:

```
ssh-keygen
```

This will create two files. A private key file which should be kept to yourself as you would a password. This file is called, by default, /.ssh/id_rsa. There is also a matching public file, which is similar to a username in that servers you will be logging in to will all have a copy, and it will be associated with your account. This file is called, by default /.ssh/id_rsa.pub.

Copy your public key file to the server you wish to log in to without typing a password:

```
ssh-copy-id you@otherserver
```

Sometimes, albeit rarely, this might not work. In those cases, you'll have to transfer it manually. Here's a handy bit of script to help with that:

```
cat ~/.ssh/id_rsa.pub | ssh you@otherserver "mkdir -p ~/.ssh &&
cat >> ~/.ssh/authorized_keys"
```

Restrict Access to HTTP Proxy Servers

Using Squid as the proxy server, we edit the configuration file /etc/squid/squid.conf.

For a single host, add a line like this to the file:

```
acl hosttodeny src 10.9.8.200
```

For a whole network, add a line like this to the file:

```
acl nettodeny src 10.10.10.0/24
```

Then alter the line that says:

```
http_access allow localnet
```

To deny the appropriate acls:

```
http_access allow localnet !hosttodeny !nettodeny
```

Configure an IMAP and IMAPS Service (Including POP3 and POP3S)

We're going to use Dovecot as our IMAP, IMAPS, POP3 and POP3s servers.

```
bash
sudo apt install dovecot
cd /etc/dovecot
cat 10-mail.conf | grep mail_location # make a note of this value
for later.
cat 10-mail.conf | grep mail_privileged_group # make a note of
this value for later.
```

Verify that the location of the mail_location exists.

Edit the file /etc/dovecot/dovecot.conf.

Find the line protocols = and add the protocol names so it looks like:

```
protocols = imap pop3 imaps pop3s
```

Verify that certificates have been created and placed in /etc/dovecot/conf.d directory or in the /etc/pki/dovecot directory, depending on your distribution.

Verify that the location of the certificates is identical to the location cited in /etc/dovecot/10-ssl.conf.

```
systemctl dovecot restart
```

Verify:

```
ps aux | grep dove
netstat -nptlu | grep dove
```

We should be listening on ports 110, 143, 993, and 995.

Configure an HTTP Server (RHEL/CentOS)

As root:

```
bash
yum install httpd
systemctl enable httpd
systemctl start httpd
systemctl status httpd
curl http://localhost # should get the Apache2 test page
```

To enable Ubuntu/Debian-like virtual hosts in separate files, which is extremely convenient, edit the /etc/httpd/httpd.conf file. At the very bottom of the file, add this line:

```
IncludeOptional vhost.d/*.conf
```

Create the new directory:

```
mkdir -p /etc/httpd/vhost.d
```

Restart the service:

```
systemctl restart httpd
```

Configure an HTTP Server (Ubuntu/Debian)

As root:

```
bash
apt-get install apache2
lynx localhost
```

Configuration files are located in /etc/apache2 instead of where they are on CentOS/Redhat machines. The main configuration file is /etc/apache2/apache2.conf, though it's been split up into multiple files for ease of management.

Here is a snippet of the file containing the hierarchy outline:

bash It is split into several files forming the configuration hierarchy outlined below, all located in the /etc/apache2/ directory:

```
/etc/apache2/
|-- apache2.conf
|       \--  ports.conf
|-- mods-enabled
|       |-- *.load
|       \-- *.conf
|-- conf-enabled
|       \-- *.conf
\-- sites-enabled
        \-- *.conf
```

Configure HTTP Server Log Files

Log format types are defined in httpd.conf on CentOS/RedHat systems and apache2.conf on Debian/Ubuntu systems in the ifmodule log_config_module section.

For any vhost, there are two primary log files for Apache: error and access. Both are in /var/log by default, but this can be customized.

The error log's format is static and cannot be customized.

Access logs can be customized almost however you'd like. Two are already defined for us: customlog is logs/access_log combined meaning it's in this format:

	Meaning
h	hostname/ip address of remote requester
l	remote login name
u	remote user
t	date/time
r	first line of request to server
s	final status
b	size of response
	Legacy
	(browser)

Examine the logs as they are:

```
cat /etc/httpd/logs/access_log
```

Creating a simplified log format called mycustom:

```
“host: %h - Date and Time: %t - Requested: %r” mycustom
```

Change the access_log definition to use the new mycustom format and save the file. Then restart Apache:

```
systemctl restart httpd
```

If you made any mistakes in the configuration file, it will be revealed at this point. Generate some data with your new log:

```
lynx http://localhost
```

Verify the changes:

```
cat /etc/httpd/logs/access_log
```

Restrict Access to a Web Page

In either the apache2.conf or httpd.conf, depending on which distribution you're running, find the section that contains <Directory entries.

Create one for a site you wish to hide, except from certain networks or IP addresses:

```
bash
<Directory /var/www/html/test/>
     Order allow,deny
     Allow from 52.206.180.246
     Allow from 172.31.34.52
     Allow from 127
</Directory>
```

Save the file and restart the web server:

```
systemctl restart httpd
```
(RH/CentOS)

or

```
systemctl restart apache2
```
(Debian/Ubuntu)

In this example, the site at /var/www/html/test will be accessible from only the local machine, and from the IP addresses 52.206.180.246 and 172.31.34.52.

Configure a Database Server

To install MariaDB, a MySQL drop-in replacement:

```
bash
apt-get install mariadb-server mariadb-client
mysql_secure_installation # CHANGE THE ROOT PASSWORD!
```

To execute SQL statements interactively:

```
mysql -u root -p
```

Some simple commands for reference:

- show databases;
- create database SOMEDATABASE;
- show databases;
- use SOMEDATABASE;

Manage and Configure Containers

Containers are fabulous, and I could go on all day about them. In fact, I **do** in my two other classes, LXD/LXC Deep Dive and Certified Kubernetes Administrator Exam Preparation here at Linux Academy. Also, my colleague Terry Cox has a great one called Docker Certified Associate Prep Course that you should check out if you think you might enjoy containers. Okay, end of the shameless plugs.

A great place to find docker images is the Docker Hub.

To install Docker, install **Docker.io** using either apt or yum.

List of running containers:

```
docker ps
```

Create a containerized web server that will serve content out of /home/user/webstuff:

```
docker run -dit -name la-test-web -p 80:80 -v /home/user/
webstuff/:/usr/local/apache2/htdocs/ httpd:2.4
```

Pull up the host's ip address in a web browser, and you should see your content.

Add additional content. The directory is shared between the host and the container.

Stop and start the container (Verify in a browser between commands).

```
docker stop la-test-web
```

```
docker start la-test-web
```

List local docker images:

```
docker image ls
```

Remove an image:

```
docker image rm :
```

Manage and Configure Virtual Machines

On a Centos 7 machine:

```
bash
yum install qemu-kvm libvirt libvirt-client virt-install virt-
viewer
cat /proc/cpuinfo |grep vmx # grep for svm if you're on an AMD
system
virt-install --name=virtly --vcpu=1 --memory=1024 --cdrom=/tmp/
image.iso --disk size=5
virsh list --all # will list all running virtual machines on cli
virsh edit virtly # will open the XML defining the running VM to
alter in vi
virsh autostart virtly #will autostart this vm on boot
virsh autostart --disable virtly # will disable the above option
vmmanager #will launch the graphic version of Virtual machine
Manager
virt-clone --original=virtly --name=virtly2 --file=/path/to/
mynewdisk/virtly2.qcow2
```

Storage Management

Storage manage skills account for 13% of the exam.

List, Create, Delete, and Modify Physical Storage Partitions

The lsblk tool lists all block devices and partitions on the system, along with their size, current mount status, and mount point.

The fdisk tool allows you to format block storage devices.

```
fdisk /dev/sdx # or whatever block device you want to partition
```

Fdisk Menu Options:

Command	Action	Notes
a	Toggle bootable flag	Tells older non-UEFI systems that the partition contains a bootable operating system
b	Edit bsd label	
c	Toggle the dos compatibility flag	
d	Delete a partition	Danger! If you accidentally delete a partition, hit **q** and start over. It doesn't get written to disk immediately, so if you get out now, then you avoid nuking your system.
g	Create a new empty GPT partition table	
G	Create an IRIX (SGI) partition table	
l	List known partition types	
m	Print this menu	
n	Add a new partition	
o	Create a new empty DOS partition table	
p	Print the partition table	
q	Quit without saving changes	
s	Create a new empty Sun disklabel	

Command	Action	Notes
t	Change a partition's system id	
u	Change display/entry units	
v	Verify the partition table	
w	Write table and exit	
x	Extra functionality (experts only)	I bet the man page has details about this…

How to master this skill: How many USB thumb-drives do you have lying around that are too small to put anything interesting on? Use them to practice partitioning and see what kinds of interesting configurations you can come up with.

Manage and Configure LVM Storage

The lvm2 package allows you to take several physical block devices and bind them together and present them to the system as a single device mountable at a single mount point while using all the drive space.

When you use fdisk to partition your devices, make sure you change the partition type to **8e**, which is the Linux LVM file type.

The hierarchy of pieces you'll be using to assemble something you can mount is:

Partition > Physical Volume > Logical Volume > Volume Group > Filesystem

Object	Create Tool	List Tool	Detail Tool
Partition	fdisk	fdisk	fdisk
Physical Volume	pvcreate	pvs	pvdisplay
Logical Volume	lvcreate	lvs	lvdisplay
Volume Group	vgcreate	vgs	vgdisplay
Filesystem	mkfs	lsblk	df

To add a device to an existing LVM set:

1. Back up your existing volume group and unmount.
2. Partition your new device. Make sure the partition type is 8e.
3. Create a new physical volume in the new partition with pvcreate.
4. Extend the volume group with the new partition using vgextend.

5. Extend the logical volume with the new with lvextend.

6. Check the existing file system with e2fsck.

7. Resize the file system using resize2fs.

8. Remount the volume group.

How to master this skill: Practice creating them. See how much usable space you can get by making a volume group out of old USB thumb drives. If you have old USB hubs lying around, don't be shy. Plug as many in as physically possible. It's great practice and a fun project to show off to your nerd friends.

Create and Configure Encrypted Storage

To install and set up your first encrypted partition:

```
bash
grep -i config_dm_crypt /boot/config-$(uname -r)
lsmod | grep dm_crypt
 # We need to be sure our system supports encrypted file systems!
yum install cryptsetup
cryptsetup -y luksFormat /dev/xvdf1
 # enter a passphrase and verify
cryptsetup luckOpen /dev/xvdf1 mySecret
mkfs -t ext4 /dev/mapper/mySecret
mount /dev/mapper/mySecret /mnt/encrypted
df -h
```

To lock up the partition when not in use:

```
cryptsetup luksClose mySecret
```

This protects it from being mounted without the passphrase.

To unlock it:

```
bash
cryptsetup luksOpen /dev/xvdf1 mySecret
mount /dev/mapper/mySecret /mnt/encrypted
 # Next, write a file to test that everything works:
echo "encryption" > /mnt/encrypted/encryptedcontent
```

Lock it back up:

```
umount /mnt/encrypted
```

```
cryptsetup luksClose mySecret
```

Test mounting the drive without unlocking first:

```
mount /dev/xvdf1 /mnt/encrypted
```

We can't mount it unless we use the encryption system. Our passphrase protects it.

How to master this skill: Encrypt a thumb drive to keep your contact list or other contents on and practice using it daily until locking and unlocking the drive is a habit. Yes, I have a lot of old thumb drives sitting around. I may have an addiction.

Configure Systems to Mount File Systems at or During Boot

This is all controlled from the /etc/fstab file. Like many other Linux configuration files, this one is arranged such that each line in the file refers to a single device.

```
device /mnt/point/ fs-type options 0 2
```

The last number is a 0, 1, or 2 depending on what you want the fsck at boot behavior to be: 0 is don't do it, 1 is this is the root drive, so this gets the primary check, and 2 checks it after the primary drives.

Here is a sample line that will mount a usb drive to the /media/usb mount point.

```
UUID=xxxxxxx /media/usbdisk ext4 noatime,noexec,nobootwait,nofail
0 2
```

fstab **option**	**Meaning**
default	Use rw, suid, dev, exec, auto, nouser, and async as the options
relatime	Access times are only updated if they are earlier than the modification time
noatime	Turns off tracking of access times
user	Permit users to mount the file system (imples noexec, nosuid, nodev unless you override these)
noauto	Do not mount when mount -a is given (like at boot time, this useful to temporarily turn off a file system from boot time without removing the entry)
exec / noexec	Permit or block the execution of binaries from the file system
suid / nosuid	Permit of blocking the operation of suid and sgid bits

fstab **option**	**Meaning**
nobootwait	Mounts the device, but the rest of the boot process does not wait for it to complete.
nofail	Do not report errors for this device if it doesn't exist. Particularly useful for usb drives which may or may not be plugged in.
credentials=/path/to/file	Used to pass a username and password to cifs network partitions
usrquota	Enable quota on this device (quota package must be installed and configured)

To create a new entry in this file, you'll need its UUID. You can get this with the blkid tool.

How to master this skill: Have I mentioned USB thumb drives at all? Practice mounting them and making them available at boot.

Configure and Manage Swap Space

Swap is used when a system needs more RAM but is using all its RAM. It swaps out pages to swap space – either a partition or a file. Partitions should be type **82**, Linux swap.

Turn all your swap options off with swapoff -a and back on again with swapon -a.

Swap is defined for the system in the fstab like this:

```
/dev/sd3 swap swap sw 0 0
```

You can also create a 512M swap file using the dd tool like this:

```
dd if=/dev/zero of=/path/to/extraswap bs=1024 count=523288
```

The new file should be owned by root and have permissions of 600. Start using it by using the swapon command.

```
swapon /path/to/extraswap
```

You can add this file to be mounted as boot time, as well, just by specifying the file as the device.

```
/path/to/extraswap swap swap sw 0 0
```

How to master this skill: Swap is important, but honestly I only set it up when I'm installing new systems or recording tutorials. Like anything else in this guide, practice doing it a few times. This one's pretty easy by comparison.

Create and Manage RAID Devices

The mdadm tool, or Multi Disk Admin tool, allows you to manage software RAID on Linux.

Partitions that will be used as part of a RAID array should have a type of **fd**.

RAID Type	Meaning
0	Many disks, no parity. For assembling disks into a single unit, though LVM might be a better choice if this is your goal
1	Also called Mirroring. All files on at least two drives
5	Drives with parity, for file and backup servers with high availability requirements
6	Drives with double parity
Others	There are more RAID types than we discussed in the video. See the man pages for details!

To create a RAID0 array from two partitions:

```
bash
mdadm --create --verbose /dev/md0 --level=stripe --raid-devices=2
/dev/xvdf1 /dev/xvdf2
cat /proc/mdstat
mdadm --detail /dev/md0
mkfs -t ext4 /dev/md0
mdadm --detail --scan
 #(add this line to /etc/mdadm/mdadm.conf on Ubuntu/Debian
 # or /etc/mdadm.conf on CentOS/RedHat)
mdadm --assemble --scan
update-rc.d mdadm defaults
nano /etc/default/mdadm
 # set    AUTOSTART=true
 # On CentOS/Redhat:
systemctl enable mdmonitor
systemctl start mdmonitor
```

To check on your RAID array:

```
mdadm -detail /dev/md0
```

If a device fails, then add a new device to have the raid array go into self-repair mode (if you've chosen other than RAID0):

```
mdadm /dev/md0 -add /dev/NEWDEVX
```

You can also add a spare device when everything is fine when you create the array, which will immediately be swapped in should any of the other devices fail.

How to master this skill: Are you running out of thumb drives yet? I'm nerdy in that I love building RAID6 arrays out of several thumb drives. There's not very much space on the resulting array, but it's redundant!

Configure Systems to Mount File Systems on Demand

By this point, you've mounted and unmounted file systems right and left throughout this whole course. One video outlines connecting to a samba share.

Install utilities:

```
bash
yum install samba-client samba-common cifs-utils
 # on Debian/Ubuntu systems, use apt-get
 # get available shares (omit -I and the IP address if you're on a
private network)
smbclient -I x.x.x.x -U <username> -L share
mkdir -p /mnt/newmountpoint
 # create a credentials file with YOUR Samba credentials
echo "username=chad" > /mnt/.smbcredentials
echo "password=i<3kittens" >> /mnt/.smbcredentials
chmod 600 /mnt/.smbcredentials
```

Edit the /etc/fstab file – use the IP address, not the hostname!

```
//x.x.x.x/share /mnt/newmountpoint cifs credentials=/mnt/.
smbcredentials,defaults 0 0
```

Then, to mount the share,

```
mount -a
```

or

```
mount -t cifs -o credentials=/mnt/.smbcredentials,defaults
//x.x.x.x/share /mnt/newmountpoint
```

How to master this skill: Network shares are incredibly useful, so practice making use of them – even when something like an scp would do the trick.

Create, Manage, and Diagnose Advanced File System Permissions

How to master this skill:

Setup User and Group Disk Quotas for Filesystems

Disk quotas are managed with the quota tool. Install the quota package with your system's package manager.

Edit the fstab to enable quotas. Add the keyword usrquota to the options section.

For example:

```
bash
/dev/sda1  /opt ext4  defaults 0 2
 # to enable quotas, change the line above to the line below
/dev/sda1  /opt ext4  defaults,usrquota 0 2
```

Once that's done, you'll need to remount the device (or reboot the system):

```
mount -o remount /dev/sda1
```

Create the database (note this will remove any old databases along with any information they contained):

```
quotacheck -cu /opt # Note the mount point is the same one for the
one we're using
```

Optional m and g flags also (see the video for details).

Check the database with the a (all) and v (verbose) and u (user) options:

```
quotacheck -avu
```

Edit quotas with the command:

```
edquota
```

Quotas are in blocks, not bytes. You can get the block size of your partition's filesystem with the command:

```
blockdev -getbsz /dev/sda1
```

Then, you'll just do the math. For example, if I wanted to give the user "chad" a 5GB quota on the file system above, I'd multiply 5 x 1,000,000,000 (or the closest binary even number, 1,073,741,824 if you're into high precision math) roughly and divide the result by my block size, 1024. This yields either the less accurate 4,882.812.5 (you'd need to eliminate that decimal, so if you're feeling generous that's 4,882,813) or the extremely accurate 5,242,880.

Yeah, the math can get weird. In short, depending on who you ask, a 5GB quota translates into something between 4,880,000 1k blocks and 5,370,000 1k blocks. When dealing with huge numbers and something like quotas, I am very comfortable with ballpark figures, so I'll probably assign the quota count value for this at 5000000.

So, back to editing the file. You'll see something like this:

```
bash
Disk quotas for user chad (uid 401):
Filesystem       blocks       soft         hard        inodes   soft
hard
/dev/sda1        999999          0            0            99      0
0
```

And for our discussion, you can change it to something like this:

```
bash
Disk quotas for user chad (uid 401):
Filesystem       blocks       soft         hard        inodes   soft
hard
/dev/sda1        999999          0     50000000            99      0
0
```

Save the file and then run `quota` to verify that things worked out like you'd hoped.

You can run a quota report to see how things are going with:

```
repquota -a
```

How to master this skill: Read all the man pages, and then practice! Quotas are a broad topic, and there's a lot you can do with them.

Create and Configure File Systems

The mkfs command will create filesystems. In other operating systems, this is referred to as formatting.

To make a filesystem, you'll usually need a partition on a block device, so use fdisk to set that up. Some newer file systems don't strictly need a partition table, so consult your file system of choice's man pages to find out precisely what you need.

To use it:

```
mkfs -t
```

like:

```
mkfs -t ext4 /dev/sd3
```

Once you've created the file system, you can mount it and add it to your system's /etc/fstab so it gets mounted at boot time.

How to master this skill: Hey, I didn't mention thumb drives in the previous section! Guess what? Yes. More thumb drives, but this is fun. You can experiment with all the different file systems and find out which ones have the features you need and like.

Conclusion

I hope this study guide was useful to you! This course wouldn't have been possible without the support of my wife, Colleen, and my mentor and friend, Terry Cox. Thanks also to my peer review partner Kevin Thompson who has many excellent courses here at Linux Academy that you should check out. He does Python coding and stuff! I'm a one-man herd of cats that Megan Thompson somehow managed to organize, and if you found any spelling or grammatical mistakes, it's totally my fault because there were SO MANY that Rebecca Gibbs and Craig Parker likely went blind trying to decipher what I meant. Drop by the community and let me know if you find ways I can improve the course or this guide! Good luck on your exam!

Sincerely, Chad Miller