KIT304 Server Administration and Security Assurance

Tutorial 5

Goal

This tutorial will give you some exposure to file systems, and the formatting and mounting of hard drives, before you get some more practice at sharing files between users on a system.

Introduction

System administrators provide a wide range of services to the users of a computer. Perhaps the most basic of these is to provide a place to store content, and a way to be able to share it with other users in a controlled way. In a previous tutorial you shared a directory between two users, and today you will build on that.

Servers have a finite amount of storage space. That amount is the sum of the space on all of hard drives that are currently connected. Unix has, as you will see today, a unified file structure that may span multiple hard disk drives. Unlike what you may be used to seeing with Windows-based operating systems, they don't appear as separate drives. Instead, they appear as subdirectories (or folders) in the overall directory structure. You interact with the various files and directories in the file system, and don't generally need to consider where the drive boundaries exist.

Before looking at how to increase the storage space on the CentOS server by adding a new drive, we will have a brief aside about disks and file systems. In KIT104 and KIT213 these were discussed in relation to how data is stored on the disk. Those units described how a drive can have a range of different index structures (tables, indexes or trees commonly) to keep track of where the files and directories are stored. While it is common for consumer drives to come preformatted for use on Windows and macOS, this is not common for enterprise-grade drives. Your first step today will be to format a new (virtual) disk so that it can then be used to store data.

Formatting a disk includes deciding on which file system format to use. There are many different formats that are used today, including FAT (12, 16, 32), NTFS, ExFAT, HFS, HFS+, APFS, UFS, ext2, ext3, ext4, XFS, btrfs and ZFS. The first three (FAT, NTFS and ExFAT) are most closely associated with Windows, the next three are mostly associated with macOS, and the remainder are most often used as Unix or Linux file systems. This was not an exhaustive list. While you may be familiar with FAT32 or NTFS, today you will see and use **ext4** which is a commonly used Unix file system.

Up until this point in the tutorials you have been interacting with CentOS as it was installed, with a virtual hard drive of 20 gigabytes, and you've learned about some of the more important directories within its file structure. Today you'll learn about another – the **`/dev/`** directory, which is short for **devices**. This is where Unix operating systems store entities that represent all of the different types of devices that are connected to the system, including hard drives.

Activity 1 – Storage

1. Launch **CentOS**, log in with the student account, open a terminal window, and become the root user. Then change to the **`/dev/`** directory and do a directory listing so that you can see all of the files that it contains.

   You can probably figure out what some of those files represent based on their names. For example:

   - **`tty`** – terminals
   - **`lp`** – line printers
   - **`sd`** – storage device
   - **`mem`** – system memory

   What is the one named **`null`** used for?

   _____

2. More than one hard drive entry is present in the **`/dev/`** directory. One of them represents the hard drive you have been using for the past few weeks, and it is clearly set up and functioning as it should be. Another entry exists, however, for a hard drive that you are not aware of.

   The directory listing also shows entries for the multiple *partitions* that exist on the connected drive. Disk partitioning is the process of dividing a single physical hard disk into multiple logical storage units. These partitions can be used for different purposes, and each can even have a different file system type.

   You're going to use three commands to learn about the disks connected to the system – these are **`lsblk`**, **`df`**, and **`fdisk`**. **`lsblk`** is the first to use – it shows a listing of block devices (another name for hard drives) that are connected to the system. Using **`lsblk`**, what is the **device name** of the hard drive that is already set up, and what are the **device names** of its partitions?

   _____

   What is the **device name** of the other hard drive that connected but not otherwise set up? (This is the one you will get to set up during this tutorial.)

   _____

   Can you see where these are listed in **`/dev/`** now that you know what to look for?

Now run the command **df** in the terminal – this reports file system disk space usage. Included in its output are **tmpfs** filesystems, which are temporary filesystems that are actually held in RAM, and not on physical disks. **lsblk** and **df** will not show the exact same devices, but based on their output, and ignoring the **tmpfs** filesystems, what are the three purposes being fulfilled by the main hard drive, and what are their sizes?

_____

_____

_____

3. Now that you know what the new drive is called you can set about making it useful. First you need to partition it using the **fdisk** command using this format:

   fdisk /dev/*devicename*

   When **fdisk** starts, it will display a welcome message, and wait for a command. You can list all of the available commands by pressing **m** for help. Find the command to **add a new partition**, and then enter this command and follow the prompts to add the partition, accepting the default values whenever they are offered. This creates a *primary* partition that fills the entire drive.

   The command you've run hasn't actually written anything to the disk, but it has prepared a new partition table for the disk that is ready to be written to it. You can preview this new partition table with the **p** command.

   When you're happy to write the changes to the disk, use the **w** command. This step can't be undone, so always check that the new table is what you expect before writing it back to the disk.

   At this point if you run **lsblk** again, you should see a change in relation to the new drive. What is the name of the newly created partition?

   _____

4. You now have a drive with a partition, but it still doesn't have a file system. As you know, there are a number of different file system formats you could choose for the new drive. Use the **man** page for the **df** command to find out which option you use with **df** to display each filesystem's type, and then use that version of the **df** command to find the filesystem type of the **/boot** and **/** filesystems. What is the command you used, and what is the filesystem type:

   _____

5. To create a filesystem on a partition, you use the **mkfs** command. Read its **man** page to determine how to format the primary partition of the new drive using the **ext4** filesystem format, and write the full command to do this below. Ask your tutor to check your work before proceeding:

   _____

   Once you've been given the go-ahead, format the drive.

6. You now have a partitioned drive that contains a file system, but it's still not accessible – you can verify this with the **df** command, which still won't show the newly formatted volume. That's because, unlike macOS and Windows, Linux systems don't usually automatically mount new volumes – instead, they have to be manually *mounted*, or you must configure the system to mount them at boot time.

   Before you can mount the new file system, you need to have, or create, a location in the existing filesystem on which to mount it. Create a new directory called **projects** in the root of the filesystem (that is, at **/**). You'll use this as the mount point for the new filesystem. Mounting drives is done with the **mount** command. Use the **man** page to work out what **mount** command options you need to mount your new partition onto the **/projects/** directory, and write down the full mount command here:

   _____

   Once you run this command, the new drive gets attached to the **/projects/** directory. Files you store in this directory are actually stored on the new drive (**/dev/sdb1**) and not on the original drive (**/dev/sda**). If you do a directory listing of **/projects/**, you'll find an additional directory that you did not create called **lost+found**. This is used to store any lost parts of files that are discovered though the usage of disk-checking tools like **fsck**.

   Now run **lsblk** and **df -k** again – this time, you should see the new partition reflected properly in the output of both commands.

7. Click the On/Off "power" icon in the upper-right corner of the CentOS VM, click the same icon lower down in the menu that pops up, and select the **Restart** option. Once again, log in with the student account, open a terminal window, and become the root user. Use the **lsblk** command, and the **df -k** command. You should find that while the new partition is visible, it is no longer mounted – that's because the system hasn't been configured to mount it at boot time.

   To configure **CentOS** to mount a device at boot up you need to edit the file **/etc/fstab** (as root), and add a new line at the bottom that specifies the volume to mount and the mount point. The line you need to add has the following syntax:

> ➢ Note that if you get this line wrong, your system will not boot correctly, and you'll need to ask for help from your tutor.

```
devicePath mountPointPath ext4 auto 0 0
```

After making this change, reboot again, and then verify that the new partition is now mounted correctly.

Activity Part 2 – Sharing

In a previous tutorial, you started exploring how to change group ownership permissions on one user's home directory so that another user could have shared access to their files. In a real-world system, it would be rare to set up shared access to documents in this way. Instead, a more appropriate approach would be to:

- establish an overall shared space for project work (most likely not in **`/home`**)
- create Unix groups that represent each of the projects that users will be collaborating on
- make users members of the appropriate Unix groups that they need to be in
- create folders inside the shared space for each of the projects
- use group ownership settings on each of those folders to limit access to just the members of the relevant group

In this part of this tutorial, you're going to use the new **`/projects/`** space that you've set up as a storage space for your various projects, and you'll create some new groups so that you can enforce permissions at the group level, rather than at the user level.

To carry out the following steps, you'll need to be the **`root`** user in the terminal.

8. You're the server administrator at a company named **Impossible Technologies**, and you need to create a place for the company researchers to collaborate and share their work. Most of the researchers have signed non-disclosure agreements with their project's financiers, and so you must ensure that they can't see the research work of projects they are not a part of.

   Create a subdirectory in **`/projects/`** for each of the three research groups, named:

   a. **`levitation`**
   b. **`perpetualmotion`**
   c. **`teleportation`**

   What command did you use to create each directory (only write one)?

   _____

9. Create Unix groups for each of those projects, using the same names. Again, what commands did you use (to keep this simple, only write one):

   _____

10. Set the *group permissions* on each of the research folders you created in step 8 to **`rwx`** so that group members have full access to the contents of their project's folder. World (*other*) permissions should be set to `---`. The project folders should already be owned by the root user, and so their permissions don't matter. What commands did you use (only write one)?

    _____

11. Set the *group owner* of each research folder to the correct group. What command did you use?

    _____

12. Now create 5 user accounts for your research scientists. Their names are Dave, Erin, Frida, George, and Helen. Use lower-case versions of these as their Unix usernames, and give them passwords (to keep this simple, each password can be the same). What commands did you use for the first user?

    _____

13. The research scientists are conducting research as follows:

    Dave: perpetual motion
    Erin: levitation and teleportation
    Frida: levitation
    George: perpetual motion and levitation
    Helen: teleportation

    Make each user a member of the groups as shown above, and write one of the commands as an example of how you did that:

    _____

14. To verify that this has worked as planned, in the top right-hand corner of the CentOS screen click on the "power" icon, and in the menu that pops down, click **student** and then select **Log Out**. Log in as **Dave**, and verify that you can access the **perpetualmotion** project folder, but not the **levitation** or **teleportation** folders. Check that you can create a document and a subdirectory in your group's project folder.

15. Now log out as Dave, and log in as **Erin**. Now, you should be able to access the **levitation** and **teleportation** folders, and create documents in them, but you should not be able to access the **perpetualmotion** project folder. Create a file in the **levitation** folder that contains notes for your colleague Frida.

16. Now, log out as Erin, and log in as **Frida**. As Frida, can you read the **levitation** project document created by **Erin**? Can you modify this file? You should find that you can't, meaning that currently, researchers can *read* documents they create to share with each other, but they can't *modify* them.

    You can overcome this by setting the *set-group-id* bit on a directory that is shared (like your three project group folders). When the *set-group-id* bit is set on a directory, all new files created inside that directory, by any user, inherit the *group owner* of the parent directory, not the *primary group* of the user who created the file(s) inside it. This allows group members who share the folder to automatically have write permission on the files that other group members create.

    To set the *set-group-id* bit you need to use the **`chmod`** command. What is the full **`chmod`** command you would need to use, as the root user, to set the *set-group-id* bit on the levitation project subdirectory?

    _____

    Use the appropriate **`chmod`** command (as root) on each of your three project folders.

17. Still logged in as **Frida**, can you now write (make changes) to the file that Erin created earlier? Why not?

    _____

18. What command would you need to use (as root) to enable Frida to make changes to the file Erin created earlier?

    _____

19. Still logged in as **Frida**, create a new file in the **levitation** project folder with notes for Erin. Then log out as Frida, and log in as **Erin**. Check that you can both read and modify the new file.

> ➢ Demonstrate to your tutor that you can modify the file created by Frida as Erin.

Conclusion

That concludes our coverage of Unix system administration features. If you have not finished any of the previous tutorials, or you have any problems or issues with any of the content, please ask for help now while your tutor is readily available. The next tutorial will be your first practical exam. You will be asked to carry out tasks that have been covered in the previous tutorials in a set time limit, and so you need to be comfortable enough with the material that even if you need to look something up in a system man page, you know how to find the relevant information quickly.

Skills to Master

To successfully complete this tutorial, you must have mastered the following skills:

- use the **`lsblk`** and **`df`** commands to learn about drive partitions, volume formats, partitions, and which partitions are mounted
- use the **`fdisk`** command to prepare a drive for use
- modify the **`/etc/fstab`** file to ensure a partition is mounted at boot time
- use **group permissions** and **group membership** to create shared areas for group members that are inaccessible to non-group members
- use the **set-group-id** bit to ensure shared group files are writable by all group members

Anything identified in this skills section could be part of an assessment item in this module.