CUMULUS

White Paper by David Davis, ActualTech Media

Top Ten Basic Linux Administration Tips

In this Paper

Introduction

For anyone new to Linux, there are a few things that you must-know in order to get started. For example, you'll need to know how to gain access to Linux, how to login, and how to get around Linux, once logged in.

In this paper, you'll learn all of that, and more, as we run down the top 10 must-knows of basic Linux administration.

#1 – Where Do I Get Linux?

To get started with Linux, you need to download a Linux distribution, such as RedHat Enterprise Linux, Ubuntu, Debian, Fedora, openSUSE, CentOS, or Cumulus Linux. You want to make sure that you obtain a Linux distribution that is compatible with your hardware. For example, you might select a 32-bit i386 image or a 64-bit amd64 image.

For example, if you want to start with the Debian distribution, you can download an ISO-formatted image that you would use to install Debian Linux from https://www.debian.org/distrib/

While some people will want to run Linux directly on a physical server, desktop, or laptop, many people start learning Linux for the first time by running it inside of a virtual machine. With a VM option, you can run Linux inside your existing Microsoft Windows or Apple macOS operating system using virtualization tools such as VMware Workstation or VMware Fusion, both of which both offer a free, limited-time evaluation license. You can also go with a free product from Oracle called VirtualBox. Another option is to run Linux as a VM in the public cloud via a provider such as Amazon Web Services or Microsoft Azure.

CLI vs. GUI

Linux can be administered through a graphical user interface (GUI) or command line interface (CLI). Most direct uses of Linux by consumers/individuals are done with a GUI, as with Android phone users or Linux desktop users. Most Linux servers are administered through the CLI, as administrators typically find it to be more efficient.

#2 – How Do I Log In to Linux?

Because most Linux administration is done using a CLI, you log in to Linux at either the console of the Linux host machine or by remotely connecting to the Linux server over a network. For a new installation, you typically log in to the console to install system packages and then set up initial users with passwords and network access.

Most Linux servers are set up to allow users to connect via the network using the *Secure Shell* (SSH), an encrypted communications protocol. SSH is a secure alternative to the insecure *telnet* that was used in the past. With SSH, your SSH client connects to the SSH server running on the Linux host where you log in with a username and password.

Get an SSH client!

SSH clients are all over the place! If you're a Windows user, one of the most popular clients is PuTTY, which is available for download from www.putty.org. If you're a Mac user, you can just use the Terminal application that is already built into macOS. If you're on a mobile device, head to your device's application store and look around. You'll find a multitude of options.

Here's how establishing a connection might work from a system that already has an SSH client (such as macOS in this case) connecting to a Linux host (Cumulus Linux in this case) over the network:

```
macos1:~ david$ ssh cumulus@192.168.1.107
cumulus@192.168.1.107's password: ********
Welcome to Cumulus VX (TM)
Cumulus VX (TM) is a community supported virtual appliance designed for
experiencing, testing, and prototyping Cumulus Networks' latest technology.
For any questions or technical support, visit our community site at:
http://community.cumulusnetworks.com
The registered trademark Linux ® is used pursuant to a sublicense from LMI, the
exclusive licensee of
```

As you can see, with SSH, you connect using the command **`ssh`,** followed by the **Username,** an **@** symbol, and then the **Hostname** or **IP Address** of the Linux host to which you are trying to connect. You will be prompted for your password to log in. In the example above, the password is required, but is not echoed and therefore not shown.

#3 – How Do I Know What Type of Linux I Am Using?

Because there are so many different types of Linux, you want to be sure you know what distribution and version you are using (for the sake of searching the right documentation on the Internet, if nothing else). Keep in mind a couple different commands to identify your Linux version.

The **`uname`** command shows the basic type of operating system you are using, like this:

```
david@debian:~$ uname -a
Linux debian 3.16.0-4-686-pae #1 SMP Debian 3.16.43-2 (2017-04-30) i686 GNU/Linux
```

And the **`hostnamectl`** command shows you the hostname of the Linux server as well as other system information, like the machine ID, virtualization hypervisor (if used), operating system, and Linux kernel version. Here's an example:

```
david@debian:~$ hostnamectl
Static hostname: debian
Icon name: computer-vm
Chassis: vm
Machine ID: 0eb625ef6e084c9181b9db9c6381d8ff
Boot ID: 8a3ef6ecdfcf4218a6102b613e41f9ee
Virtualization: vmware
Operating System: Debian GNU/Linux 8 (jessie)
Kernel: Linux 3.16.0-4-686-pae
Architecture: x86
```

As shown above, this host is running Linux. More specifically, the host is running Debian GNU Linux version 8 (codename jessie) with a Linux 3.16 version kernel on an x86 CPU architecture. Among other things, you can also see that this Linux installation is running on a virtual machine with VMware as the hypervisor. Cool, huh?

#4 – Where Do I Find Things?

An operating system has a file system that, similar to a filing cabinet, allows you to store and retrieve data. Most file systems use the concept of *directories*—also called *folders*—and files that are stored inside the directories. Everything in Linux—even hardware—is represented in this folder and file structure.

Where do usernames and passwords come from?

You may be wondering where these usernames and passwords come from. The "superuser" username in Linux is called "root" because it is the only user that can modify the root directory. During installation, the root user is created, and you are able to select a password. Post-installation, administrators can use the root user account and account management commands to create new users (with associated passwords) for normal user activities.

If you're new to Linux, you might be wondering how the Linux file system compares to something familiar like the Microsoft Windows file system. In Windows, you may be used to drive letters (like the C: drive) being used as the highest point of a storage volume. Linux represents the highest level of the volume differently. The Linux file system can span multiple physical drives, which are all a part of the same tree. The highest point of the Linux file system is the "/," or "root," with all other directories branching down the tree from there, as shown in Figure 1.

Interaction with and navigation of the Linux file system is done up and down the tree with commands such as:

- **`pwd`**. Display the directory you're currently in (short for print working directory)
- **`ls`**. List out files that are present in the folder
- **`cd`**. Change directory
- **`rm`**. Remove files
- **`mkdir`** and **`rmdir`**. Make and remove folders or directories, respectively

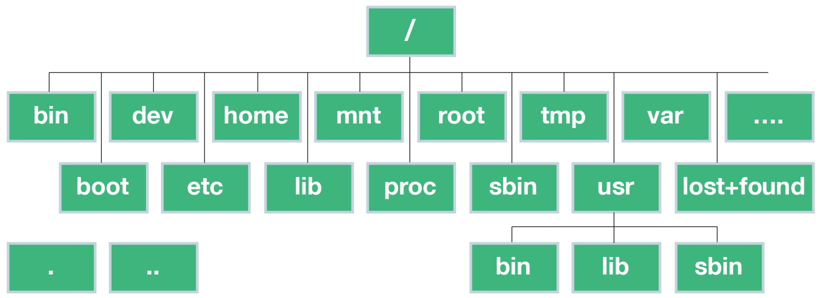

Figure 1. The typical Linux file system

Let's do a quick exercise. First, by using the **pwd** command, you can see what directory I'm currently in.

```
david@debian:~$ pwd
/home/david
```

Next, to change to the root directory, you can use the **cd** command.

```
david@debian:~$ cd /
```

To get a simple list of files, you can use the **ls** command. This will display a very concise list of the files and folders that exist in the current directory.

```
david@debian:/$ ls
bin  boot  dev  etc  home  initrd.img  lib  lost+found  media  mnt  opt  proc  root
run  sbin  srv  sys  tmp  usr  var  vmlinuz
```

But, in most cases, you probably want more information than just a simple list of files. Linux uses command line *flags* or *switches* to extend what a command can do. For example, to list out all the files and folders in the current directory, along with full details about each one, you would type **ls -la**. This long listing format then shows you each file and directory, as well as the permissions and access rights for each object, the name of the user that owns the object (root), the name of the group that owns the object (again, root), the file size, and the data and time that the object was last modified. Here's what this output looks like for the root folder on my test system:

```
david@debian:/$ ls -la
total 88
drwxr-xr-x  21 root root  4096 May 15 11:50 .
drwxr-xr-x  21 root root  4096 May 15 11:50 ..
drwxr-xr-x   2 root root  4096 May 15 12:11 bin
drwxr-xr-x   3 root root  4096 May 15 15:53 boot
drwxr-xr-x  18 root root  3200 Jul 14 01:52 dev
drwxr-xr-x 134 root root 12288 Jul 14 01:55 etc
drwxr-xr-x   3 root root  4096 May 15 15:53 home
lrwxrwxrwx   1 root root    33 May 15 11:50 initrd.img -> /boot/initrd.img-3.16.0-
4-686-pae
drwxr-xr-x  19 root root  4096 May 17 00:41 lib
drwx------   2 root root 16384 May 15 11:49 lost+found
drwxr-xr-x   3 root root  4096 May 15 11:49 media
drwxr-xr-x   2 root root  4096 May 15 11:49 mnt
drwxr-xr-x   2 root root  4096 May 15 11:49 opt
dr-xr-xr-x 150 root root     0 Jul 14 01:52 proc
drwx------   2 root root  4096 May 16 14:29 root
drwxr-xr-x  23 root root   880 Jul 14 01:57 run
drwxr-xr-x   2 root root  4096 May 17 00:41 sbin
drwxr-xr-x   2 root root  4096 May 15 11:49 srv
dr-xr-xr-x  13 root root     0 Jul 14 01:52 sys
drwxrwxrwt  13 root root  4096 Jul 14 02:02 tmp
```

```
drwxr-xr-x  10 root root  4096 May 15 11:49 usr
drwxr-xr-x  12 root root  4096 May 15 12:12 var
lrwxrwxrwx   1 root root    29 May 15 11:50 vmlinuz -> boot/vmlinuz-3.16.0-4-686-
pae
```

#5 – Where Are the Applications, and How Do I Run Them?

One of the most common questions from new Linux users who are at a command line is, “What are the applications available to me, and how do I run them?” As mentioned previously, most user tools are found in the directories /bin, /usr/bin and system tools are typically located in /sbin and /usr/sbin. For example, tools like **`cp`** (to copy a file), **`ps`** (for process status), and **`cat`** (to display the contents of a file) are all found in /bin. The great thing is you don’t need to go into any of these directories to run these types of tools because these directories are included in your $PATH variable by default.

The $PATH variable includes all the locations that are searched when you run a command in the CLI. Because the /bin directories are in your path, when you execute the name of any of these sample tools, they will be found.

Fun with the file system

The image shown in Figure 1 also showed you some of the most important directories in the Linux file system, including:

- **/bin, /sbin, /usr/bin, and /usr/sbin**. Where executable programs are stored.
- **/dev.** Where files representing hardware devices are stored. For example, if your Linux system had a floppy drive device, there would be a file named fd0 in the dev folder (/dev/fd0).
- **/etc**. Where configuration files are stored.
- **/home.** Where user home directories are stored, one for each user.
- **/var.** Where variable-length files, like log files, are stored.

Of course, not all applications play nice, and not all Linux administrators are consistent. This is just where stuff is *supposed* to go, but things occasionally end up where they don’t belong. While there may be some differences between Linux distributions when it comes to where things are located, in general, the baseline directory structure and usage of it should be the same because this is defined by the *file system Hierarchy Standard* (FHS). For more information on the FHS see: https://en.wikipedia.org/wiki/Filesystem_Hierarchy_Standard.

Here's what your $PATH variable might look like (shown by using the **`echo`** command to show the $PATH variable):

```
david@debian:~$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
```

You can execute applications or commands simply by typing the name of the command if application's location is in your $PATH. If that application is not in one of the folders listed in your $PATH, you have to do one of the following:

- Navigate to the folder where the application is found and tell Linux that you want to execute the application in that folder, like this:

  ```
  david@debian:~$ cd /opt/app/bin
  david@debian:~$ ./myapp
  ```

- (the "dot slash" refers to the current folder, with the full command saying "in the current directory, execute 'my app'")

- Specify the full path of the application when you execute it, like this:

  ```
  david@debian:~$ /opt/app/bin/myapp
  ```

A useful command in determining which command will be run and from what directory it will be run is the **`which`** command. Use **`which`** with the executable of a command afterward to get a list of the location of the command that will be executed.

Besides the standard types of Linux tools, there are tens of thousands of applications you can install into Linux in just a few commands. Linux distributions offer *package managers* that help you search online package or application repositories and then download and install just about any application you might want. Package managers also make it easy to update your packages to get the latest version. Examples of package managers are **`apt`**, **`dpkg`**, **`rpm`**, and **`yum`**. The package manager that is available to you will be determined by the Linux distribution that you have installed. Linux running on Android mobile devices also has its own package manager (similar to the Apple "App Store").

On Debian and Ubuntu systems, you can run **`apt list --installed`** and get a list of the packages that are already installed, like this:

```
david@debian:~$ apt list --installed
accountsservice/stable,now 0.6.37-3+b1 i386 [installed,automatic]
acl/stable,now 2.2.52-2 i386 [installed]
acpi/stable,now 1.7-1 i386 [installed]
acpi-support-base/stable,now 0.142-6 all [installed]
```

(**Output truncated**)

Any **`apt list`** command will result in very long output, so you may consider *piping* it to the "less" pager tool, like this: **`apt list | less`**. This will show you the output page by page and allow you to press the space bar after each page to see the next page.

#6 – How Do I Install Applications?

Before you start installing new services, you should typically ensure that you have the list of the most recent versions of available packages from the update repository. This command doesn't actually update any software, but it does make sure you're looking at a list of currently available package versions. You can update the list of packages that are available to you with **`apt update`**, like this:

```
david@debian:/opt$ sudo apt update
Ign http://ftp.us.debian.org jessie InRelease
Get:1 http://ftp.us.debian.org jessie-updates InRelease [145 kB]
Get:2 http://security.debian.org jessie/updates InRelease [63.1 kB]
Get:3 http://ftp.us.debian.org jessie Release.gpg [2,373 B]
```

(**output truncated**)

Then, install a package with **`apt install`**, like this:

```
david@debian:~$ sudo apt install apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  apache2-doc apache2-suexec-pristine apache2-suexec-custom
The following NEW packages will be installed:
  apache2
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 0 B/208 kB of archives.

After this operation, 361 kB of additional disk space will be used.
Selecting previously unselected package apache2.
(Reading database ... 137657 files and directories currently installed.)
Preparing to unpack .../apache2_2.4.10-10+deb8u8_i386.deb ...
Unpacking apache2 (2.4.10-10+deb8u8) ...
Processing triggers for man-db (2.7.0.2-5) ...
Processing triggers for systemd (215-17+deb8u7) ...
Setting up apache2 (2.4.10-10+deb8u8) ...
```

Important!

For commands requiring elevated privileges, we'll be prepending those commands with the **`sudo`** command. The **sudo** command allows you to run the command as an administrator.

In the above example, we used **`apt install`** to install the Apache web server. To verify that a package is installed correctly (and that you installed what you think you installed), you can use **`apt show`**.

```
david@debian:~$ apt show apache2
Package: apache2
Version: 2.4.10-10+deb8u8
Installed-Size: 361 kB
Maintainer: Debian Apache Maintainers <debian-apache@lists.debian.org>
Replaces: apache2.2-common, libapache2-mod-macro (<< 1:2.4.6-1~)
Provides: httpd, httpd-cgi
Depends: lsb-base, procps, perl, mime-support, apache2-bin (= 2.4.10-10+deb8u8),
apache2-utils (>= 2.4), apache2-data (= 2.4.10-10+deb8u8)
Pre-Depends: dpkg (>= 1.17.14)
Recommends: ssl-cert
Suggests: www-browser, apache2-doc, apache2-suexec-pristine | apache2-suexec-custom
Conflicts: apache2.2-common (<< 2.3~)
Breaks: libapache2-mod-macro (<< 1:2.4.6-1~)
Homepage: http://httpd.apache.org/
Tag: role::metapackage, suite::apache
Section: httpd
Priority: optional
Download-Size: 208 kB
APT-Manual-Installed: yes
APT-Sources: http://ftp.us.debian.org/debian/ jessie/main i386 Packages
Description: Apache HTTP Server
 The Apache HTTP Server Project's goal is to build a secure, efficient and
 extensible HTTP server as standards-compliant open-source software. The
 result has long been the number one web server on the Internet.
 Installing this package results in a full installation, including the
 configuration files, init scripts and support scripts.
```

You can see that the Apache 2.4.10 web server was installed, and it says that this package results in a full installation; however, it also suggests that we install the apache-doc (for documentation) and www-browser (to act as our HTTP client/ web browser) packages.

What is piping?

You can direct the output of a command to another command. Say you want to get a directory listing that doesn't scroll off the bottom of the screen. You can use the **`less`** paging tool by piping the output of **`ls -al`** to **`less`**. In this case, type **`ls -al | less`** at the command prompt and, when the screen fills up, you are prompted to hit a key to view the second page of the directory. Understanding the pipe character "|" and its usage is important as you begin your Linux journey. In fact, as you get deeper into Linux territory, you will find that the ability to pipe command output to other commands is invaluable when it comes to creating scripts to automate certain functionality.

#7 – Linux Processes, Programs, and Services

When you start a program, in Linux, it will run interactively by default, which means you can interact with it via your terminal session with all input and output presented to you, the user. However, you can also run programs in the background (often called "services") so that you don't see their output and can still use your command prompt to continue your work (and continue running the service even when you are logged out). This can also be useful if you have a program that will take some time to process; you can just put it in the background and be alerted when it is completed.

Getting help

Linux commands can, at times, be confusing and can become complex. In Linux, help is always available!

Use the **man** command (shorthand for "manual") to provide detailed documentation for just about every Linux command. For example:

```
david@Debian$ man ls
NAME
       ls - list directory contents
SYNOPSIS
       ls [OPTION]... [FILE]...
DESCRIPTION
       List  information  about the FILEs (the
current directory by default).  Sort entries
alphabetically if none of
       -cftuvSUX nor --sort is specified.
       Mandatory arguments to long options are
mandatory for short options too.
       -a, --all
              do not ignore entries starting
with .
       -A, --almost-all
              do not list implied . and ..
```

(output truncated)

Depending on the command, other options to get help are to append "-h" or just "help" after the command.

But how do you know if it's still running, and how do you get a list of every process running on your system? The **ps** command displays a list of running processes in Linux. This command is often coupled with the **-ef** flag to show every process in the long list format shown below. You'll see right at the top that "/sbin/init" is PID (process identifier) #1, and it's owned by root (the superuser).

```
david@debian:~$ ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root  1     0  0 01:52 ?    00:00:01 /sbin/init
root  2     0  0 01:52 ?    00:00:00 [kthreadd]
root  3     2  0 01:52 ?    00:00:00 [ksoftirqd/0]
root  5     2  0 01:52 ?    00:00:00 [kworker/0:0H]
```

(Output truncated)

You may want to pipe the **ps -ef** command to **less**, like **ps -ef | less**, to see the output page by page.

If you just enter **ps** by itself, you'll see only your running processes, like this:

```
david@debian:~$ ps
  PID TTY          TIME CMD
 1679 pts/1    00:00:00 bash
 1784 pts/1    00:00:00 ps
```

In this case, you can see that this user is running the **bash** shell, which is providing the command prompt and the **ps** command to show what processes are running (also the command that produced this output).

Linux uses the concept of *system services*, which are long-running programs that are run in the background and typically provide some service on behalf of system users. You can start, stop, and check the status of services with the command **systemctl**, like this:

```
david@debian:/opt$ systemctl status
● debian
    State: running
     Jobs: 0 queued
   Failed: 0 units
    Since: Tue 2017-08-08 20:49:02 EDT; 1h 37min ago
   CGroup: /
           ├─1 /sbin/init
           ├─system.slice
           │ ├─avahi-daemon.service
           │ │ ├─469 avahi-daemon: running [debian.local
david@debian:/opt$
```

(**Output truncated**)

Service vs. Systemctl

There is currently a command transition happening, with the **service** command being phased out in favor of the new **systemctl** command. You may see references on websites to the older **service** command. Be aware that the new **systemctl** will soon replace **service** in all Linux distributions.

#8 – Importance of Linux Log Files

To be an effective administrator of any type of Linux system, you need to be able to find and search log files to determine the status of the system, including finding any system and application errors. Although applications can put their log file anywhere they like (such as placing them in the application's directory), most Linux system log files will be found in /var/log.

If you go over to /var/log (with **`cd /var/log`**) and do a **`ls -l`** (to list the files in long format), you'll find that there are quite a few Linux system log files.

```
david@debian:~$ cd /var/log
david@debian:/var/log$ ls -l
total 4924
-rw-r--r-- 1 root root 0 Jul 14 01:57 alternatives.log
-rw-r--r-- 1 root root 40586 May 15 12:12 alternatives.log.1
drwxr-xr-x 2 root root 4096  Jul 14 01:57 apt
-rw-r----- 1 root adm  1471  Jul 14 02:17 auth.log
-rw-r----- 1 root adm  24651 Jul 14 01:55 auth.log.1
-rw-rw---- 1 root utmp 0     Jul 14 01:57 btmp
-rw------- 1 root utmp 768   Jul 14 01:53 btmp.1
drwxr-xr-x 2 root root 4096  Jul 14 01:57 cups
```

(**Output truncated**)

The following are the most important system log files:

- **syslog.** Contains the centralized logging system, called syslog, in which you'll find messages related to the kernel, applications, and more. If configured, this could be the centralized log file for all Linux systems (or even all network devices) in your data center.
- **auth.log**. Contains authentication failures and successes
- **messages**. Contains general system messages of all types

A variety of different tools can be used to view and parse log files, such as:

- **`cat`**. Display the contents of a file
- **`less`**. View a file with pagination and scrolling
- **`grep`**. Search for a string in a file where the usage is **`grep PATTERN [FILE]`**
- **`head`**. See the first lines (head end) of a text file
- **`tail`**. View the last lines (tail end) of a text file. A common use case for tail is to watch the status of a log file in real time with the "-f" flag like **`tail -f /var/log/syslog`**

Even if you ignore the rest of the commands in the previous list, learn to use **`grep`**.

#9 – Users and Superusers

Just as you might expect with any multi-user operating system, Linux supports the concept of users with differing levels of access. By default, you'll log in as a common user and be able to view most of what's happening on the system, although you're not allowed to view log files as a standard unprivileged user. To be able to reconfigure the system or view log files, you'll need administrative rights. In Linux, these administrative privileges are referred to as *superuser* privileges and are equivalent to the root user, who has a user ID of 0 (zero).

Adding, Modifying, and Deleting User Accounts

It's important to note that Linux user accounts can be added, modified, and deleted with the commands **`adduser`**, **`moduser`**, and **`deluser`**. To add, modify, or delete users, you must have the correct privileges (which are usually the privileges of the root user).

Take a look at this command sequence:

```
david@debian:/$ id
uid=1000(david) gid=1000(david)
groups=1000(david),24(cdrom),25(floppy),27(sudo),29(audio),30(dip),44(video),46(plu
gdev),108(netdev),110(lpadmin),113(scanner),117(bluetooth)
david@debian:/$ whoami
david
david@debian:/$ sudo id
uid=0(root) gid=0(root) groups=0(root)
david@debian:/$
david@debian:/$ sudo whoami
root
```

Notice in the dialog above how the **`id`** command was used to see that we were "uid" (user ID) 1000, and how the **`whoami`** command was used to see that I am a user called "david." From there, I used the **`sudo id`** command to make sure I was the root user, and the **`sudo whoami`** command verified that I had become root. You'll note that the **`id`** command proves that I have the uid of 0 (zero).

Here's a real-world example. Suppose you'd like to view the latest system logs from the Linux syslog file. Doing so isn't possible with a regular user account. To view the syslog file (using the **`tail`** command, in this case), you must use the **`sudo`** command:

```
david@debian:~$ tail /var/log/syslog
tail: cannot open '/var/log/syslog' for reading: Permission denied
david@debian:~$ sudo tail /var/log/syslog
May 15 10:00:08 debian systemd[1]: Reached target Network is Online.
May 15 10:00:08 debian systemd[1]: Started ACPI event daemon.
May 15 10:00:08 debian systemd[1]: Listening on ACPID Listen Socket.
May 15 10:00:08 debian systemd[1]: Started LSB: RPC portmapper replacement.
May 15 10:00:08 debian systemd[1]: Reached target RPC Port Mapper.
May 15 10:00:08 debian systemd[1]: Activated swap /dev/disk/by-uuid/4a28e383-9cad-
4d88-997e-62cfb508d606.
May 15 10:00:08 debian systemd[1]: Activated swap /dev/sda5.
May 15 10:00:08 debian systemd[1]: Activated swap /dev/disk/by-path/pci-
0000:03:00.0-scsi-0:0:0:0-part5.
May 15 10:06:19 debian systemd[1]: Starting Session 106 of user david.
May 15 10:06:19 debian systemd[1]: Started Session 106 of user david.
```

In the above command sequence, you can see that first there was a permission denied error when trying to view the syslog file, but when the **sudo** command was used (which typically prompts you for the root password, since no other user was specified), the last 10 lines of the log file were shown. Many systems prevent you from becoming the root user with **su** and instead require you to use the **sudo** command.

The privileges for who can run what are determined by the /etc/sudoers file, and that file should be edited using the **visudo** command to ensure safe access to a critically important configuration file. For more information on **sudo**, just use **man sudo** to view the manual page.

#10 – Files and Permissions

The reason that the user "david" was denied access to the file /var/log/syslog in the previous example is that the user "david" doesn't have permission to access to the file. You can see this if you execute **ls -l /var/log/syslog**:

```
david@debian:~$ ls -l /var/log/syslog
-rw-r----- 1 root adm 9074 May 15 10:17 /var/log/syslog
```

The file is owned by the user "root" and the group "adm". The file permissions are "rw" (shorthand for read/write) for the owner and "r" (shorthand for "read") for the group with no permissions for anyone else. Figure 2 shows how file permissions work in Linux.

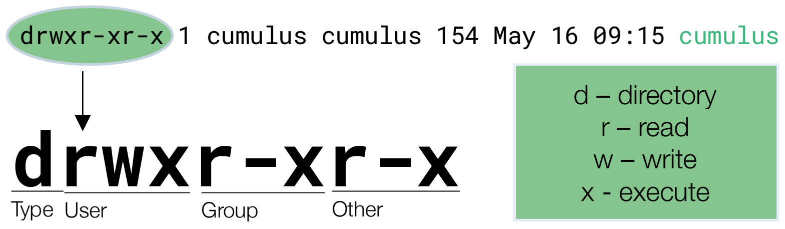

Figure 2. Linux file permissions

In the file permissions graphic (Figure 2), a "d" on the left tells you whether you are looking at a directory (or folder). Then the three sets of permissions "rwx, r-x, r-x" say whether you can read, write, and execute (or start the application) at the user level, the group level, and the "everyone else" level (others). The type indicator shown in Figure 2 identifies the selected object as a directory, hence the "d" as the type. The two most important types of objects in the Linux file system are directories ("d") and files ("-"). There are other possible types as well, but for my purposes here, we'll stick with directories and files.

In Summary

You've learned where to download Linux, how to login, and how to get around Linux, once logged in. Additionally, you learned about Linux applications, man pages, processes, files, and permissions. With these tips in mind, you're ready to jump into Linux! Thanks for reading the top 10 must-knows of basic Linux administration!

Last updated: December 2017