U.S. DEPARTMENT OF ENERGY'S
CYBERFORCE® PROGRAM

CyberForce® 101

Intro to Linux

October 2023

cyberforcecompetition@anl.gov

Intro to Linux

Overview

Linux is a family of free and open-source operating systems (OS) based on the Linux kernels. The kernel is a program at the heart of the Linux OS that takes care of the fundamentals, like communication between hardware and software. Linux distributions or distros are OS's based on Linux, including Debian, Ubuntu, Fedora, CentOS, etc.

File System

In Linux, everything is a file. Directories are files, files are files, and devices are files. There are three types of files: general files, directory files, and device files. General files are your ordinary files that contain image, video, program, or text, and these are your most commonly used files. Directory files act like Folders in the Windows OS. Device files are your devices represented as files, and they reside in the directory `/dev/`.

Linux distributions are compliant with the Filesystem Hierarchy Standard (FHS), which defines a set of directions, each serving their own special function.

The forward slash (`/`) is used to indicate the root directory in the filesystem. When a user logs in to the shell, they are brought to their own user directory or home directory stored in `/home`/. The root user has its own home directory `/root/`.

All files have permissions, which allow a user to read, edit, or execute them. Access restrictions can be applied for different kinds of users by changing permissions.

Users

There are three types of users in Linux: regular, administrative (root), service.

A regular user is your normal user account. Your files and directories are stored in `/home` which is your home directory. As a regular user you do not have access to directories of other users.

A root user is the superuser who can access restricted files, install software, and has administrative privileges. Whenever you want to install software, make changes to system files, or perform other administrative tasks, you must use the root user.

The root user should be used exclusively for administrative purposes as mistakes made by the root user can be detrimental. The `sudo` command temporarily bestows root’s powers on you, but your account must be on the sudoers list in order to use it. The `sudo` command requires

you to authenticate using your own password while the command `su` (which functions just like `sudo`) requires you to authenticate using the root user's password.

Services like Apache, Squid, and email have their own individual service accounts. Linux can allow or deny access to various resources depending on the service.

To create a new user, use the command `adduser`. You will need to be root or use `sudo` to create a user.

```
sudo adduser username
```

You can change the password of a user by using the command `passwd -l` .

```
sudo passwd -l username
```

You can remove a user using the `userdel` command. The `-r` flag removes the home directory with the user.

```
sudo userdel -r username
```

To add a user to a group, you can use the `usermod` command.

```
sudo usermod -a -G GROUPNAME USERNAME
```

To remove a user from a group, use the command below.

```
sudo deluser USER GROUPNAME
```

If you want to see information from users, you can use the `finger` command. The `finger` command by itself will show information of all the users logged in. If you use `finger username` it will give the information for that particular user.

Navigation

You can create directories (similar to folders in other operating systems) inside other directories, with files existing in any directory. The `pwd` ("print working directory") allows you to see what directory you are currently in.

```
pwd
```

```
# Output
/home/cyberforce
```

The command `mkdir` (“make directory”) lets you create one or more new directories in your current directory. The `ls` command lets you see a list of files and directories that exist in your current directory.

```
mkdir test1 test2
ls
```

```
# Output
test1
test2
```

You can use `ls -R` to show all the files not only in directories but also subdirectories. `ls -al` gives detailed information of the files and `ls -a` shows hidden files.

To navigate into these new directories, you can use the `cd` (“change directory”) command and specify the directory’s name within your current working directory.

```
cd test1
pwd
```

```
# Output
/home/cyberforce/test1
```

The `cd` command only looks for directories within your current one, so we cannot cd directly into the test2 directory we created before. To navigate into any existing directory regardless of your current one, you can use `cd` with the full path of the directory you want to navigate to.

```
cd /home/cyberforce/test2
```

Also note that in Linux, a tilde (`~`) is shorthand for the home directory of the user you're logged in as. So you could also write the previous command like below.

```
cd ~/test2
```

You can also use `..` to change the directory one level up in your path to get back to your original directory

```
cd ..
```

You can also navigate to the root directory by using `cd` with `/`.

Redirection

Redirection is a Linux feature where you can change the standard input or output devices.

Output redirection uses the `>` symbol. Instead of having your output produced to the terminal or standard location, you can redirect it to other locations, like other files.

For instance, if we want to put the output of the command `ls -al` in a file, we can use the command below.

```
ls -al > list
```

Then, if we `cat` the output of list, we will see the results of the command `ls -al` which shows our files with their permissions.

If you do not want to overwrite a file but add more content, then use the `>>` operator.

You can also redirect standard output to devices.

Input redirection uses the `<` symbol. Now, we can use this to have our output come from a file. For instance, we can `cat` from a file like below.

```
cat < file.txt
```

This will print the contents of file.txt to the terminal.

File Manipulation

One way to create a file is using the `touch` command. Below, we create an empty file with the name `file.txt` in our current working directory. The file is empty.

```
touch file.txt
```

We can rename the file with the `mv` command like shown below.

```
mv file.txt newfile.txt
```

`mv` stands for "move" and will move a file or directory from one place to another. Since we specified it above, we "moved" it to a new location in the current working directory, thereby renaming it.

You can also copy a file to a new location using the command `cp`. Below, we create a new file `thirdfile.txt` that has the same contents as `newfile.txt`.

```
cp newfile.txt thirdfile.txt
```

To edit files, you need to use a file editor. These include `vim`, `nano`, and `pico`, among others. For instance, if you want to open `newfile.txt` in nano, you would use the command below. Then, you can add, modify, and remove the text in the file.

```
nano newfile.txt
```

You can view text using `cat` or `less`. The `cat` command prints the entire contents of a specified file to the terminal. `less` will also print the contents but only one terminal page at a time beginning at the start. You can use the spacebar to advance a page or the arrow keys to go up and down one line at a time. `q` quits `less`.

```
cat newfile.txt
```

You can delete files using the `rm` command.

```
rm newfile.txt
```

`rm` cannot delete directories without the `-d` flag added to it, which allows you to delete empty directories. You can also remove empty directories with the `rmdir` command.

To delete a non-empty directory, you can use the rm command with the -r flag, which will delete the specified directory and all of its contents.

```
rm -r test1
```

File Permissions

As Linux is a multi-user system, it uses permissions and ownership for security.

There are three permissions: read, write, and execute. Read allows you to open and read a file. Read permission on a directory allows you to list its content. Write lets you modify a file's contents. For a directory, write lets you add, remove, and rename files stored in it. To run a program, the execute permission must be set.

You can see file permissions with the `ls -l` command. Using this command you may see something like `-rw-rw-r--` as output. If it is a directory, their will be a `d` in the front of it.

The `r` represents the read permission, `w` for write permission, and `x` for execute permission. `-` means that there is no permission.

We can divide the output for every three letters. The first three letters, in this case `rw-` shows what the user has permission to do, which is read and write. The second set of three letters, also `rw-`. shows what the group can do. This means any group members can also read and write. The last set, `r--`, means that everyone can read the file, regardless of the group or user.

Changing File Permissions

We can use the `chmod` ("change mode") command to change permissions. This lets us set permissions (read, write, and execute) on a file or directory for the owner, group, and world.

You can use absolute mode or symbolic mode to do this. Absolute, or numeric, mode is where permissions are represented as a three-digit octal number.

Number	Permission Type	Symbol
0	No Permission	--
1	Execute	-x
2	Write	-w-
3	Execute + Write	-wx
4	Read	r--
5	Read + Execute	r-x
6	Read + Write	rw-
7	Read + Write + Execute	rwx

To let the user have read, write, and execute permissions while the group has only read and write permissions and the world has only read permissions, we use the command below:

```
chmod 764 test
```

The user gets the code `7`, the group gets the code `6`, and the world gets the code `4`.

Symbolic mode allows you to modify permissions of a specific owner instead of changing permissions for all three owners. It makes use of mathematical operators instead of numbers.

Operator	Description
+	Adds a permission to a file or directory
-	Removes the permission

Operator	Description
=	Sets the permission and overrides the permissions set earlier

Owners are represented with characters.

User	Denotations
u	user/owner
g	group
o	other
a	all

So to set permissions to the user, you would use a command like below.

```
chmod u-r test
```

For the group:

```
chmod g+x test
```

For the world:

```
chmod o=rwx test
```

Changing Ownership and Groups

To change the ownership of a file/directory, you can use the `chown` command.

```
chown user filename
```

To change the user and the group for a file or directory, use the `chown` command as shown below.

```
chown user:group filename
```

To change only the group owner, use the command `chgrp` like below.

```
chgrp group_name filename
```

Important Notes

The file /etc/group contains all the groups defined in the system. You can also use the command `groups` to find all the groups your user is a member of. To work as a member of a group other than your default, you can use the command `newgrp`.

Two groups cannot own the same file. Nested groups do not exist in Linux. There are other permissions that you can set on files and directories, but the ones covered above are the main ones.

Pipe, Grep, and Sort

The Pipe command in Linux lets you use two or more commands such that output of one command serves as input to the next. It's denoted by the symbol `|`. This helps use mash-up two or more commands at the same time and run them consecutively.

The `grep` command searches for specific information you're looking for it. You can use different options with it.

Option	Function
-v	Shows all the lines that do not match the searched string
-c	Displays only the count of matching lines
-n	Shows the matching line and its number
- i	Match both (upper and lower) case
-l	Shows just the name of the file with the string

Here's an example combining both pipe and grep.

```
cat test | grep -i a
```

This command will show all the words with "a" in the test file.

The sort command helps sort a file's contents alphabetically. There are different options with this command as well.

Option	Function
-r	Reverses sorting
-n	Sorts numerically
-f	Case insensitive sorting

Regular Expressions

Regular expressions are special characters which help search data and match complex patterns. There are three types: basic regular expressions, interval regular expressions, and extended regular expressions.

Basic regular expressions can be found in the table below.

Symbol	Descriptions
.	replaces any character
^	matches start of string
$	matches end of string
*	matches up zero or more times the preceding character
\	represent special characters
()	groups regular expressions
?	matches up exactly one character

Interval expressions tell us about the number of occurrences of a character in a string. You need to add the flag `-E` to these expressions

Expression	Description
{n}	matches the preceding character appearing 'n' times exactly
{n, m}	matches the preceding character appearing 'n' times but not more than m
{n, }	matches the preceding character only when it appears 'n' times or more

An example command is shown below. Here, we check that the character 'p' appears exactly 2 times in a string one after the other.

```
cat test | grep -E p\{2}
```

Extended regular expressions contain combinations of more than one expression.

Expression	Description
\+	matches one or more occurrence of the previous character
\?	matches zero or one occurrence of the previous character

Suppose we want to find words where 'a' precedes the character 'n'. We can use the command below.

```
cat test | grep "a\+t"
```

Brace expansion is when we have a sequence or a comma separated list of items inside curly braces '{}'. The starting and ending items are separated by two periods "..".

Below, we use the echo command that creates strings using brace expansion.

```
echo {aa, bb}
aa, bb
echo a{0..4}b
a0b a1b a2b a3b a4b
```

Environment Variables

Environment variables are dynamic values which can affect the processes or programs on a computer. They can be created, edited, saved, and deleted. They give information about the system behavior.

Variable	Description
PATH	Contains a colon (:)-separated list of directories in which your system looks for executable files
USER	Username
HOME	Default path to user's home directory
EDITOR	Path to the program which edits the content of files
UID	User's unique ID
TERM	Default terminal emulator
SHELL	Shell being used by the user

In order to determine the value of a variable, use the command `echo $VARIABLE`, replacing VARIABLE with the correct name.

The command `env` displays all the environment variables.

You can create new variables with the command below.

```
VARIABLE_NAME= variable_value
```

You can delete a variable using the command `unset`.

You can set a value of an environment variable by using the command below.

```
export Variable=value
```

Process Management

A process is an instance of a program. Any command that you give to your Linux machine starts a new process. It is possible to have multiple processes for the same program. There are two types of processes: foreground processes and background processes.

Foreground processes run on the screen and need input from the user. You can start a foreground process from the terminal. You have to wait until the process runs before starting another process.

Background processes run in the background and do not need user input. You can send a program in the background so that you can still use the terminal.

To do this, you want to first start the program. Once it starts, use the command `^Z` to stop it. Then type `bg` into the terminal. It will then be sent to the background.

You can use the command `fg` to continue a program which was stopped and bring it to the foreground.

The command `top` will tell you about all the running processes on the machine. Use `q` to move out of the process display.

The table below shows key terminology.

Field	Description
PID	Process ID of each task
User	Username of task owner
PR	Priority, can be 20 (highest) or -20 (lowest)
NI	Nice value of a task
VIRT	Virtual memory used (kb)
RES	Physical memory used (kb)
SHR	Shared memory used (kb)
S	Status D = Uninterruptible sleep R = Running S = Sleeping

Field	Description
	T = Traced or Stopped Z = Zombie
%CPU	% of CPU Time
% MEM	Physical memory used
TIME+	Total CPU time
Command	Command name

The `ps` ("process status") command is similar to "Task Manager" in Windows.

`ps ux` shows all the processes running under a user. `ps PID` checks the process status of a single process.

To terminate a process, use the command `kill` with the PID number.

```
kill PID
```

To find the PID of a process, use the command below.

```
pidof Process name
```

Installing Software

There are different tools to install software depending on your distribution of Linux. You can use Advanced Packaging Tool (Apt) for Ubuntu and Debian-based systems or use DNF and Yum for RedHat and Fedora systems.

Before you install any software, you should first update your system.

For APT:

```
sudo apt-get update
```

For DNF and Yum: (simply switch one for the other)

```
sudo yum upgrade
```

Then you can start installing your software.

For APT:

```
sudo apt-get install package_name
```

For DNF and Yum:

```
sudo yum install package_name
```

You can also remove packages.

For APT:

```
sudo apt-get remove package_name
```

For DNF and Yum:

```
sudo yum remove package_name
```

Other Useful Commands

If you need help with a certain command, pass the command as an argument to the `man` command for documentation from the manual. It will show the purpose of the command, how to use it, what options are available, and examples of its use.

```
man command
```

The `history` command shows all the basic commands in Linux that you have used for the current terminal session.

The `clear` command clears all the clutter on the terminal, giving you a clean window to work on.

Sources

1. UNIX / Linux Tutorial for Beginners
2. Unix / Linux Tutorial (Tutorialspoint)
3. Input Output Redirection in Linux