Schools Workshop

7th May, 2025

Presented By: Derrick Ayesigye

Operating System Lab

Knowledge | Community | Solutions

Outline

- ssh
- User Management
- User Permissions & Ownership
- Package Management
- Process Management

ssh

- SSH (**Secure Shell**) is a network protocol that provides secure remote login and file transfer over a network.

```
$ ssh <username>@<server-ip>
```

Username: **ict**

- **137.63.148.90**
- **137.63.148.91**
- **137.63.148.89**
- **137.63.148.88**
- **137.63.148.87**
- **137.63.148.66**

User Management - Users

- User management includes everything from creating a user to deleting a user.
- Command line tools used include

```
$ adduser
$ userdel
$ passwd
$ usermod
```

- The root user is the superuser and always has a user ID of 0.

User Management - User groups

- User groups in Linux are used in scenarios when you need to manage permissions for a certain group of users.
- Every Linux user exists in at least one group.
- Command line tools include:

```
$ addgroup/groupadd
$ groupdel
$ usermod
```

User Management - Using sudo

- The sudo command allows you to run programs as another user (provided you have that user's password), by default the root user.
- To add a user to the sudo group, run:

```
$ usermod -aG sudo <username>
```

- To run a command as another user, run:

```
$ sudo -u <another_user> <command(s)>
```

User Management - Using sudo

- If you want to allow a specific user to run only certain programs as sudo, instead of adding the user to the sudo group, add the users to the sudoers file with the following command:
- For example, to allow the user <user> to run only the mkdir command as sudo, type:

```
$ sudo visudo
```

<user> ALL=/bin/mkdir

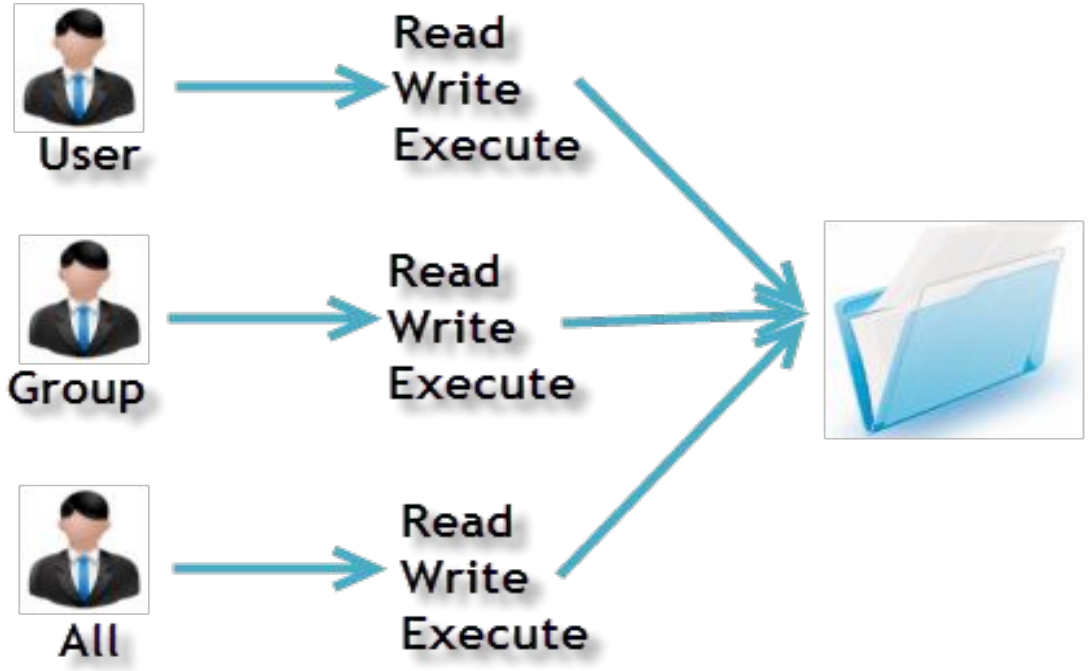
User Permissions and File ownership

- Every file on your Linux system is assigned three kinds of owners, namely:
  - **User:** A user is the owner of the file. By default, the person who created a file becomes its owner.
  - **Group:** A group can contain multiple users.
  - **Other:** Any other user who has access to a file. Practically, it means everybody else.

User Permissions and File ownership

- Every file and directory in the Linux system has the following 3 permissions defined for all the 3 file owners above:
  - **read:** This permission gives you the authority to open and read a file. Read permission on a directory gives you the ability to list its content.
  - **write:** The write permission gives you the authority to modify the contents of a file. The write permission on a directory gives you the authority to add, remove, and rename files stored in the directory.
  - **execute**: In Unix/Linux, you cannot run a program unless the execute permission is set.

Owners assigned Permission On Every File and Directory



File permissions: Symbolic mode

- In the symbolic mode, you can modify permissions of a specific owner. It makes use of mathematical symbols to modify the file permissions.

Operator	Description
+	Adds a permission to a file/directory
-	Removes a permission from a file/directory
=	Sets the permission and overrides the permission set earlier

- The various owners are represented as: u - user, g - group, o - other, a - all

File permissions: Numeric mode

Number	Permission type	Symbol
0	no-permission	---
1	execute	--x
2	write	-w-
3	execute and write	-wx
4	read	r--
5	read and execute	r-x
6	read and write	rw-
7	read,write and execute	rwx

Managing Permissions and File ownership

- This can be achieved using the "**chmod**" command, which stands for change mode. With this command, we can set permissions (read, write, execute) on a file/directory for the owner, group, or the others.
- The general syntax is shown below:

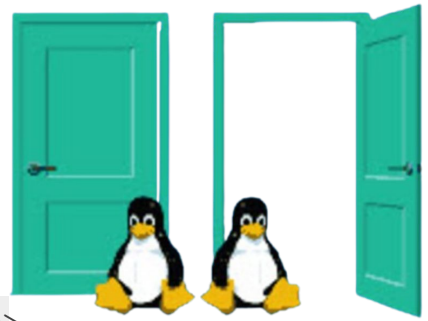
```
$ sudo chmod <permission> <directory/file>
```

- The permissions setting can be done in either symbolic or numeric/absolute mode.

Managing file ownership and groups

- To change the ownership (user) of a file/directory, you can use the following command:

```
$ sudo chown <user> <file/directory>
```



- Changing file ownership

```
$ sudo chown <user>:<group> <file/directory>
```

- To do the above for both the directory and all the files/subdirectories under it, run:

```
$ sudo chown -R <user>:<group>   <directory>
```

- Where R stands for Recursive

Package Management

Process Management

- Package management is a method of installing and maintaining (which includes updating & removing) software on a Linux system.
- If a certain package requires a certain resource, such as a shared library or another package, it is said to have a dependency.

Packaging Systems

- Software required for a Linux system can either be provided by the distribution vendor through central repositories or be available in source code that can be downloaded and installed manually.
- Because of the different distribution families, a package intended for one distribution will not be compatible with another distribution, e.g., Debian uses ***.deb** and CentOS uses ***.rpm**

Package Managers

- **dpkg:** It's a low-level package manager for Debian-based systems. It can install, remove, provide information about, and build *.deb packages, but it can't automatically download and install their corresponding dependencies.
- **apt-get/apt:** It is a high-level package manager for Debian and its derivatives and provides a simple way to retrieve and install packages, including dependency resolution, from multiple sources using the command line. **yum** is used instead of **apt** on Red Hat distros.
- **snap:** It's a software deployment and package management system built by **Canonical. snapd** is the service that runs on your machine and keeps track of your installed snaps, interacts with the store, and provides the snap command for you to interact with it.

Software Installation

- **Installation using dpkg:** To install any ".deb" package, use the **dpkg** command with "-i" option

```
$ dpkg -i <package.deb>
```

- **Installation using apt-get**: The **'install'** subcommand is followed by one or more packages you wish to install or upgrade.

```
$ apt-get install <package name>
```

e.g `$ apt-get install apache2`

- **Removing installed packages** (with dpkg)

```
$ dpkg -r <package name>
```

- **Updating and upgrading packages**

```
$ apt-get update
```

```
$ apt-get upgrade
```

Removing Installed software

- The command is of the format **apt-get <sub-command> <package name>**
- The basic supported sub-commands include:
  - **remove:** To uninstall software packages without removing their configuration files. For later re-use of the same configuration, use the ‘remove‘ sub-command.
  - **purge:** To remove software packages, including their configuration files, use the ‘purge‘ sub-command
  - **autoremove:** The ‘autoremove‘ sub-command is used to auto remove packages that were certainly installed to satisfy dependencies for other packages, but they are now no longer required.

Process Management

Process Management

- When you execute a program on your Unix system, the system creates a special environment for that program.
- This environment contains everything needed for the system to run the program as if no other program were running on the system.
- Whenever you issue a command in Unix, it creates, or starts, a new process. A process, in simple terms, is an instance of a running program.
- The operating system tracks processes through a five-digit ID number known as the pid or the process ID. Each process in the system has a unique pid.

Types of processes

- **Foreground processes** (also referred to as interactive processes)—these are initialized and controlled through a terminal session.
- **Background processes** (also referred to as non-interactive/automatic processes) are processes not connected to a terminal; they don't expect any user input.

Viewing Active processes

- **ps command:** It displays information about a selection of the active processes on the system.

```
$ ps
```

- **top command** - system monitoring tool: top is a powerful tool that offers you a dynamic real-time view of a running system.

```
$ top
```

- **glances command - system monitoring tool:** It is a relatively new system monitoring tool with advanced features.

```
$ glances
```

- **htop command:** Same as top, though it has to be installed first.

```
$ htop
```

Controlling Processes

- **kill command:** It is a built-in command, located in /bin/kill, the terminal, that is used to terminate processes manually. kill command sends a signal to a process, which terminates the process.
- **start/restart and stop command:** The start/restart command is used to start or restart a service/process. The service is stopped when the stop command is executed.
- **Reload command:** This does not stop/restart the service but rather reloads the configurations without first restarting the service.
- **Enable/Disable:** Enable makes a service start at boot up. For example, to have a service run at startup, use the command "systemctl enable <service>" or "service <service_name> enable"

```
$ systemctl enable <service>
```

THE END

Discussion