COURSEWARE

- Professional Skills
- Agile Fundamentals
- Jira
- Git
- Databases Introduction
- Java Beginner
- Maven
- Testing (Foundation)
- Java Intermediate
- HTML
- CSS
- Javascript
- Spring Boot
- Selenium
- Sonarqube
- Advanced Testing (Theory)
- Cucumber
- MongoDB
- Express
- NodeJS
- React
- Express-Testing
- Networking
- Security
- Cloud Fundamentals
- AWS Foundations
- AWS Intermediate
- Linux
  - Linux Introduction

User Administration

Contents

Overview

In Linux we can control which users have access to different files and processes.
The super user of the system is automatically configured by the system and has access to everything.

Users and Groups

Authentication vs Authorisation

There are two main security points to consider when working with users on a system, authentication and authorisation.

- **Authentication** is the process of determining if someone is who they say they are.
  This can be done by getting a user to log on with a password, if they know the username and the password, then they are probably who they say they are.
- **Authorisation** is the process of verifying if someone has access to something. This is usually built in to either the user or the group identity.

There are three files which are very important when considering *authorisation* and *authentication*:

- `/etc/passwd` contains a list of all users which have been created on the system and some information about them.
  Some of these were created manually by an admin user, others were created automatically by the system.
- `/etc/group` contains a list of all groups which have been created on the system.
- `/etc/shadow` or `/etc/gshadow` contains the sensitive information about users and groups respectively.
  This information could be passwords, first names etc.
  The information in both files is encrypted by default so cannot be read by people.

Adding a user

Any user with authorisation to run `sudo` commands can add another user.

```
$ sudo adduser john
```

This sets up a new user called john and creates a home directory which is populated with profile files.
You will be prompted to set a password, and other details for that user.

The `-s` flag will allow you to set the shell for a user. Whether this is on or after creation.

```
$ sudo useradd -m -s /usr/bin/python3 john
```

Adding a group

Similar to users, any user with authorisation to run `sudo` commands can create a group.

```
$ sudo addgroup theJohns
```

This sets up a new group call theJohns.
You can set up a password for theJohns by running the command:

```
$ sudo gpasswd theJohns
```

Then add a password when promted.

Adding users to groups

Once we have a user and a group, we can add a user to a group by modifying the user and adding the `-G` option

```
$ sudo usermod -aG theJohns john
```

This is useful if you want a group of people to have authorisation to do something.
Rather than give each person authorisation for that, we can give the group authorisation and add people to the group.

The `-a` means `append`.

The Super User

The **Super User** (UID=0) has unrestricted access to the Linux system.
In most Linux distributions the **Super User** is called `root`.

Working as the Super User

Nearly all Linux administration must be done as the **Super User**, since most administrative files are not accessible by standard users.

Administration should only be done by one user at a time.
You can check if another person is logged in as the root user by running:

```
$ who
```

This will display a list of the users that are currently logged in, if `root` is there you should find out who is logged in as root and what they are doing, before trying to make any administrative changes yourself.

Switching User

The su command

The `su` command stands for **S**ubstitute **U**ser.
This switches to another user's account with all the relevant permissions of that user.
It also starts a new shell for the new user.

Root can switch user without providing a password, other users must provide the password for the user to whom they are switching.

There are two options to the `su` command that we need to know about (but a full list can be found by running the command `$ man su`).

- `-` the minus sign without a letter will change directory to the home directory of the user you are switching to.

```
QA-JaneDoe: ~ $ su JohnDoe
Password:
QA-JohnDoe: /home/JaneDoe/ $ exit
exit
QA-JaneDoe: ~ $ su - JohnDoe
Password:
QA-JohnDoe: ~ $
```

Notice, if you don't include the `-` option, you will switch user and load that users environment.

- `-c` this option runs one specified command as the selected user in a new shell, then exits out of that shell automatically.

  ```
  QA-JaneDoe: ~ $ whoami
  JaneDoe
  QA-JaneDoe: ~ $ su - JohnDoe -c "whoami"
  JohnDoe
  QA-JaneDoe: ~ $
  ```

  Notice, in the second command, the `-c` option runs the command `whoami` in a shell which is opened as the user JohnDoe in thier home directory, then closes the shell immediately after.

If no user is specified when the `su` command is used, Linux defaults to the **super user**.
So running the command '`$ su -`' would switch to the root user in the root directory.

Sudo

The `sudo` command is an alternative to the `-c` option.
`sudo` stands for **S**uper **U**ser & **DO**.

Similarly to the `su` command, `sudo` defaults to the super user if no user is specified.
However, for `sudo` you must use the option `-u` to specify the user to use.

The most common use for `sudo` is to run commands as the **super user**, so this is the area we shall focus on.

Sudo Sessions

Whenever you run a command using `sudo`, whether the command was successful or not, it will be recorded in the logs.

When you run a command using `sudo` for the first time, you will be asked for your password.
Upon entering the correct password, and assuming you have the correct authorisation, you will start a *sudo session*.

A *sudo session* is a set length of time, during which you will be able to use the `sudo` command without entering your password.
For Ubuntu, *sudo sessions* last for 15 minutes.

Sudoers

Only certain users will have authorisation to use the `sudo` command, and what they can do with it will often be limited.

The list of users/groups who can use the `sudo` command is kept in a file `/etc/sudoers`.

Sudo users are integral to the security and general functionality of a Linux system.
As a result, we do not edit the *sudoers* file directly, as mistakes can result in problems for all *sudoers*.

To edit the `/etc/sudoers` file, we must use an intermediary file called `visudo`.

To add a user called 'jane' to the sudo users, we would first have to edit the file `visudo`.

```
$ sudo visudo
```

This opens the file in a *nano* text editor.
The person doing the system administration must be a *sudoer* themselves in order to edit the file.

Next we add the following line to the file, note that there are tabs between each block of text, not spaces.

```
jane	ALL=(ALL:ALL) ALL
```

- The first entry: `jane` ALL=(ALL:ALL) ALL defines the name of the user/group which we are adding to the file.
  In this case, it is a user called 'jane'.
- The second entry: jane `ALL`=(ALL:ALL) ALL defines which host the jane can be logged in from.
- The third entry: jane ALL=(`ALL`:ALL) ALL defines which users jane can run commands as.
- The fourth entry: jane ALL=(ALL:`ALL`) ALL defines which groups jane can run commands as.
- The last entry: jane ALL=(ALL:ALL) `ALL` defines which commands the rule applies to.

If the entry is incorrect when you try to save and quit, `visudo` won't let you save the file as this can cause issues for your *sudo users*.
So you have to go back in to the file and correct the mistakes you made.

Tutorial

▶ Create a new user

Exercises

There are no exercises for this module.