UNIX

Lecture 5: User Management and Access Rights

Filipe Vasconcelos[1]
[1]ESME, Lille, filipe.vasconcelo@esme.fr

Outline

1. User management
2. Permission Management

Intended Learning Outcomes

ILO5

Analyze and explore the file system hierarchy of a Unix-type OS using basic commands, and manage the access rights of directories and files within the hierarchy.

Outline

Users in UNIX

- UNIX is a multi-tasking and multi-user operating system.
- This means that multiple people can work simultaneously on the same OS, including remote connections.
- There is a set of rules that define who has the right to do what on the system.
- Since multiple users can be connected to Linux at the same time, there needs to be a mechanism to manage these users.

Users in UNIX

- Each user must identify themselves to access the machine, and they are designated by a unique number called UID (User Identifier).
- UID: The unique identifier for each user account. Numbers from 0 to 99 are often reserved for system-specific accounts. Values above 100 are reserved for user accounts.
- Users are organized into groups identified by unique numbers called GID (Group Identifier).
- Groups are used to gather users together and assign them common rights.
- Each resource (including files) belongs to:
  - A single user: the one who created the file, making them the owner of the file.
  - One or more groups of other users.

Types of Users

There are two types of users in the Unix system:

- User (normal): Those who use the system to accomplish tasks and view Unix as a means, a tool.
- Administrator (superuser): Responsible for installation, configuration, and ensuring the proper operation of the Unix system.

In a Unix system, Unix users have limited rights. In other words, they are restricted from using certain commands and do not have access to certain parts of the system. System administrators, on the other hand, have full control over the system.

Super User `root`

- On every UNIX system, there is a superuser, typically named `root`, who has full control over the system.
- `root` can freely access all computer resources, including acting on behalf of another user, in other words, assuming their identity.
- Generally, at least on production systems, only the system administrator knows the root password. The user `root` has UID 0.
- This simple protection helps limit the potential damage in case of mishandling,viruses on your PC, etc.

`sudo` Command

`sudo` – *"superuser do" or "substitute user do"*

- This command allows you to execute a command as root or another user.
- You will be prompted for a password to execute the command. This password is the same as the one for your limited user account.
- Example:

```
$ sudo apt-get update
```

```
[sudo] Password for filipe:
```

`sudo su` Command

- This command allows you to become root (superuser) and remain in that state.
- The # symbol (new prompt) at the end of the command prompt indicates that you have become the superuser.
- To exit *root mode*, type `exit`.

adduser Command

- This command allows you to add a new user.
- You need to run it as root (with sudo or by staying as root).

```
$ sudo adduser toto
```

```
Adding user 'toto'...
Adding new group 'toto' (1003)...
Adding new user 'toto' (1003) with group 'toto'...
Creating home directory '/home/toto'...
Copying files from '/etc/skel'...
New password:
BAD PASSWORD: The password is a palindrome
Retype new password:
passwd:  password updated successfully
Changing the user information for toto
Enter the new value, or press ENTER for the default
Full Name []:  Le petit Toto
Room Number []:  42
Work Phone []:  555-4242
Home Phone []:  555-4242
Other []:  Loves telling jokes
Is the information correct?  [Y/n]Y
```

User Management

- The 'toto' directory is created under `/home`.
- Change the password with the command: `passwd`

```
$ sudo passwd toto
```

```
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd:  password updated successfully
```

Group Management

- Each user belongs to a group.
- By default, if nothing is specified, a group with the same name as the user will be automatically created.
- Example:

```
$ ls -l /home/
```

```
drwxr-xr-x 73 filipe filipe 4096 Oct 5 16:41 filipe
drwxr-xr-x 6 git git 4096 Oct 16 2019 git
drwxr-xr-x 17 guillaume guillaume 4096 Jan 2 2018 guillaume
```

Group Management

addgroup Command

- This command allows you to create a new group.
- Example:

```
$ sudo addgroup unix
```

```
Adding group 'unix' (GID 1003)...
Done.
```

- Currently, there are no users in this group.

Group Management

`usermod` Command

- This command allows you to edit a user.
- -g: Change the user's group.
- Example: To put 'filipe' in the 'unix' group:

```
$ sudo usermod -g unix filipe
```

`delgroup` Command

This command allows you to delete a group.

User Management

deluser Command

- This command allows you to delete an existing user.
- Do not delete your own user account!! Otherwise, you won't be able to log in when the system starts
- With the `-remove-home` option, the home directory of 'toto' under `/home` will also be deleted.

```
$ sudo deluser –remove-home toto
```

```
Searching for files to backup or delete...
Deleting files...
Removing user 'toto'...
Warning!  Group 'toto' no longer has any members.
Done.
```

Outline

Permission Management

- In Unix, users are divided into three classes:
  - user (u): the file's owner
  - group (g): the group (or groups) associated with the file
  - others (o): everyone else
- Different access rights can be defined for each of these 3 categories.
- The system administrator (usually root) has full rights to manipulate files.

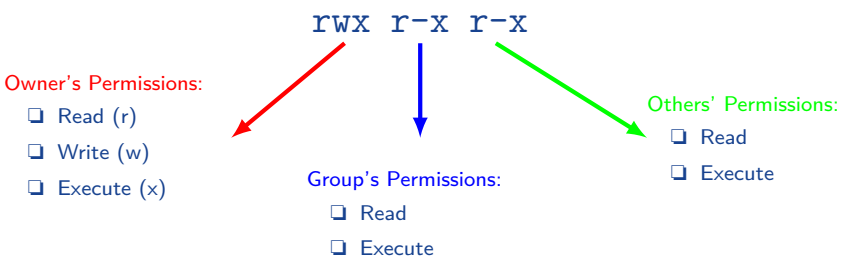
Permission Management

- File or directory name
- Date and time of the last modification
- Size in bytes
- Group ownership
- Owner
- Number of links
- Access rights (nine characters)
- File type (one character)

Permission Management

File Access Permissions

Example: `-rwxr-xr-x 1 filipe filipe 12 Oct 6 12:17 test_access.sh`



Directory Access Permissions

Example:

```
drwxr-xr-x 2 filipe filipe 4096 Oct 6 16:55 dir_test_access
```

rwx r-x r-x

Owner's Permissions:

- Read (to list the contents) (r)
- Write (to create a new file in the directory) (w)
- Execute (to access with cd) (x)

Note: If you assign `w`, you must also assign `x` to the directory

Permission Management

Basic Command: `chown`

- Meaning: **ch**ange **own**er
- Changes the owner of the file/directory
- Syntax: `chown [-R] new_owner [file|directory]`
- -R: Recursive modification of subdirectories
- Example:

```
$ chown toto file.txt
```

Permission Management

Basic Command: `chgrp`

- Meaning: **ch**ange **gr**oup
- Changes the group of the file/directory
- Syntax: `chgrp [-R] new_group [file|directory]`
- -R: Recursive modification of subdirectories
- Example:

```
$ chgrp unix file.txt
```

Permission Management

Basic Command: `chmod`

- Meaning: **ch**ange **mod**e
- Modifies file and directory permissions
- Syntax: `chmod [-R] permissions [file|directory]`
- `-R`: Recursive modification of subdirectories
- `-v`: Verbose
- Permissions can be given in symbolic or numeric form

Symbolic Form of `chmod`

`chmod permissions file`

The `permissions` argument is a combination of three symbols:

- User:
  - `u` (user)
  - `g` (group)
  - `o` (other)
  - `a` (all)
- Operator:
  - `+` (add)
  - `-` (remove)
  - `=` (assignment)
- Permission:
  - `r` (read)
  - `w` (write)
  - `x` (execute)

Examples:

- `chmod a+x test.txt`
- `chmod go-rw test.txt`
- `chmod u=rw test.txt`
- `chmod u+x,go=rw test.txt`

Numeric Form

Permissions	Binary	Decimal
rwx	111	7
rw-	110	6
r-x	101	5
r–	100	4
-wx	011	3
-w-	010	2
–x	001	1
—	000	0

Examples:

```
chmod 755 test.txt
chmod u=rwx,g=r-x,o=r-x test.txt
```