# Lab_03_2: Users & Groups

Otago Polytechnic, IN616 Operating Systems Concepts,

Semester 1 – 2020

## 1 Objectives

- Setting up user accounts
- Setting up primary and secondary groups

# 2 Creating and Configuring New Users

When we first installed Ubuntu Server, we created a user named `student`. However, in addition to the **student** account, you have also run commands using a second user named **root**. This has been achieved by using the `sudo` command, allowing us to run specific commands as the **root** user.

Linux is a true multi-user operating system. Think of the Kate server at Otago Polytechnic... this system has an account for every BIT student. Multiple students can simultaneously log into Kate and perform a variety of tasks: upload files, download files etc.

## 2.1 Determining Active Users

Imagine you are the administrator of the Kate server. It would be useful to know who was logged into the system at any given time. There are a variety of different commands to determine this information. Specifically, there are three useful commands:

```
users
who
w
```

The `users` command is very simple - it will output a list of any logged in user accounts. Try the command on your Linux system:

```
users
```

After running this command, you will only see one user who is logged in: `student`. This is because you are the user on the system. Furthermore, the `users` account does not inform us of all the users that are available on the system, only the users that are **logged in**. Now try the following two commands:

```
who
w
```

Review the output from these commands. Note that there is more information displayed when compared to the `users` command. To determine the exact purpose of each command have a look at the manual for each command:

```
man who
man w
```

**Q1.** What is the difference between the `who` and `w` commands?

The `users`, `who` and `w` commands all show logged in users - with varying levels of information. However, they are limited because they only show logged in users. To determine all users on the system you need to view the password file. We looked at this file briefly last class. Try viewing the file using the following command:

```
cat /etc/passwd
```

As you may have noticed in the following classes, in Linux nearly everything is represented in files. This makes information quickly accessible and easily configurable (just edit the text file). The `/etc/passwd` file is the place where the most general user configuration is stored. The following line is the configuration for the `student` account.

```
student:x:1000:1000:student,,,:/home/student:/bin/bash
```

We discussed the structure of the `/etc/passwd` file in the lecture today. Using this information, and the `student` account line from above, answer the following questions:

**Q2.** What is the User ID (UID) of the `student` account?

**Q3.** What is the Group ID (GID) of the `student` account?

**Q4.** What is the home directory of the `student` account?

**Q5.** What is the default shell of the `student` account?

**Q6.** The second column of the `student` account configuration has a lowercase `x`. What does this configuration mean?

Although the information in the `/etc/passwd` file is basic, it shows the flexibility of how user accounts to be customized to specific requirements.

## 2.2 Creating a New User Account

We want to practice adding a new user account on our system. This is a fundamental administration task which is essential to perform. It is also important that we add and configure users using best practices. In the following task we will create a new user named: `gandolf`.

We can add a user with no configuration using the following command:

```
useradd gandolf (DO NOT RUN THIS COMMAND!)
```

However, this is not best practice. When creating a user we will usually want to specify certain configurations. For example: what groups is a user part of, what should the default shell of the user be, and should the user have a home directory.

The following list specifies a full command that can be used to create a user while also specifying some basic configuration.

```
useradd -d /home/gandolf -m -s /bin/bash -G sudo gandolf
```

Run the command above to create the `gandolf` user. Now, we are going to use the Linux help system to see what we have configured for the new user account. Remember, the two methods to get help:

```
useradd --help
man useradd
```

It is recommended to use the `-help` to find information about command arguments, and it is recommended to use the `man` command to find detailed information, command examples and detailed descriptions about a command. Since we want to know what argument (options) we have specified in the `useradd` command, it is recommended to use:

```
useradd --help
```

**Q7.** What does the `-d /home/gandolf` command argument do?

**Q8.** What does the `-m` command argument do?

**Q9.** What does the `-s /bin/bash` command argument do?

**Q10.** What does the `-G sudo` command argument do?

**Q11.** Why do we put the string `gandolf` at the end of the `useradd` argument?

Make sure you have executed the `useradd` command specified above to actually create the user. We can check that the user has been added correctly by searching the `/etc/passwd` file. Try the following command:

```
grep "gandolf" /etc/passwd
```

You should see the following output:

```
gandolf:x:1002:1002::/home/gandolf:/bin/bash
```

Success! A new user has been added! We should check that we can login to the `gandolf` account. Instead of logging out of our `student` account, we are going to open a new terminal. Open a terminal using one of the following two methods:

- **If using VMWare Workstation**: switch to a different terminal using the keyboard shortcut `Alt + F2`
- **If using PuTTY**: duplicate the terminal by right clicking the title bar of PuTTY and selecting `Duplicate Session`

It is good practice to learn how to open multiple terminals so that you can have multiple Bash shells open. This is like tabbed browsing!

Try log in using the new account, **gandolf**

**Q12.** What happened when you tried to log in as `gandolf`? ... Why can you not log on?

On a side note... In Linux, some users can be created without a password. However, these are daemon users – user accounts that are in charge of running specific system services. For example, a MySQL database would be run by a daemon user account, usually with the **mysql** username. This `mysql` user would not be able to login to a Bash shell, and would have no password.

**Q13.** Why aren't services (like MySQL) simply run using the root user? Why do they (usually) have a specific user account with no privilege?

## 2.3 Setting Passwords for Users

We need to create a password for `gandolf`. To assign a password to the recently created user, switch back to the terminal where you are already logged in as the `student` account, using one of the method below:

- **If using VMWare Workstation**: switch to your original terminal using the keyboard shortcut `Alt + F1`
- **If using PuTTY**: switch back to the original PuTTY window

Now, using the `student` account, we will set a password for `gandolf` so we can log in using the account:

```
passwd gandolf
```

Pick a password for `gandolf`. We shouldn't document passwords – primarily for security reasons. However, for a throwaway virtual machine, this is not a primary concern. With that being said ...

**Q14.** What password did you set for the **gandolf** account?

> **PRO TIP:** If you run `passwd` without a username, it will set the password for the account you are currently logged in as. In this case the `student` account.

Now switch back to your other terminal - the one where you tried to log in as `gandolf` before. Try to log in as the **gandolf** account using the password you just created. You should be logged in and ready to go! Nicely done! Make sure you log out of the `gandolf` account using the following command:

```
logout
```

## 2.4 Deleting User Accounts

In the last exercise we create a user account named `gandolf`. Now we want to delete the account (just to get experience with removing accounts). Try the following command:

```
userdel -r gandolf
```

**Q15.** What does the `-r` option do?

Now try to search the `/etc/passwd` file to determine if the `gandolf` user account has been deleted from the system.

**Q16.** Enter the full `grep` command you used to search the `/etc/passwd` file.

When you run the `grep` command, you should expect to see no output! This is because the `gandolf` account has been deleted and does not exist any more.

Just for practice, we should try adding two more users. We should get used to constructing `useradd` commands and how to modify different configurations. Try adding another user with the following configuration:

- Username: `frodo`
- Home directory: `/home/frodo`
- Secondary groups: `none`
- Default shell: `/bin/bash`

If you are stuck, try looking at the first `useradd` command we used for creating the `gandolf` user account. Remember, if you make a mistake with the command, you can always delete the user account using the `userdel` command.

**Q17.** Enter the full `useradd` command you used to create the `frodo` user account.

Make sure you set a password for the `frodo` account using the following command:

```
passwd frodo
```

Make sure you remember the password you set, because we will use this account in the next exercise: working with groups.

Now create another user. The details of this user are:

- Username: `samwise`
- Home directory: `/home/samwise`
- Secondary groups: `sudo`
- Default shell: `/bin/bash`

Make sure you set a password for the `samwise` account using the following command:

```
passwd samwise
```

Make sure you check the `/etc/passwd` file to ensure the two users have been successfully created.

# 3 Working with Groups

In all operating systems that manage multiple user accounts, groups are essential. They allow user accounts to be combined based on common properties. For example, as a student at Otago Polytech your student account is added to a general `student` group, and a `BIT` group (as well as numerous other groups). Each group provides specific access and permission to do tasks. For example, to access specific shared drives.

## 3.1 Identifying Group Associations

All users are associated with groups by default. Especially for larger groups this makes the management of permissions easier. But how do we determine information about groups?

**Make sure you are logged in as the `frodo` user account** (the user that you created in the previous exercise).

Try the following command:

```
groups
```

The `groups` command will identify group membership of the user that is currently logged in.

**Q18.** List the groups that the **frodo** account is a member of?

As you can see from the output, most Linux distributions create a default group for each user to manage its permissions. For example, when creating the `frodo` user, a group named `frodo` is also created. This is the **primary group** of the `frodo` account - and currently, the only group that the user is associated with. However, you can add users to multiple groups and create new groups for specific purposes.

To identify all groups in a system inspect the following file:

```
cat /etc/group
```

This file has a similar structure to the `/etc/passwd` file, however, does not contain as much information.

## 3.2 Creating Groups and Adding Users to Groups

Since users can be members of multiple groups (one primary group and multiple secondary groups), we need to be able to associate users with different groups as well as create new groups. We will start with creating a new group. This group will be called `fellowship`. The

command we need to create a new group is the `groupadd` command. Perform the following task:

- Use `groupadd` to create a new group called **fellowship**
- Remember, use `groupadd -help` to determine how to use the `groupadd` command correctly.

**Q19.** Document the command you used to create the **fellowship** group.

HINT: You might not need any arguments or options.

Now we want to add users to the `fellowship` group. To add users to a group we can use the `usermod` command. Note: we can also add users to secondary groups when we create the user - however, we have already created the user accounts so cannot use this method. The general syntax of the `usermod` command is listed below:

```
usermod -G <group-name> -a <username>
```

Firstly, we should determine what the options in this command are doing.

**Q20.** What does the option `-a` do when used with the `usermod` command?

**Q21.** What does the option `-G <groupname>` do when used with the `usermod` command?

Our goal is to add `frodo` and `samwise` to the `fellowship` group. Construct two command (one for each user) to add each to the `fellowship` group. The group must be added as a secondary, or supplementary, group.

**Q22.** Document the two commands you used to add `frodo` and `samwise` users to the **fellowship** group.

For both users check that they are properly associated with their groups. First check the `/etc/group` file. Try to use `grep` to search the file. Thes check using the `groups` command. Note that the `groups` command will only show all group associations after logging on again, even though it is updated immediately in the system. Remember, you can logout using `exit`.

## 3.3 Modifying Groups

Similar to users you can modify groups, specifically their name and group id. This can be done with the `groupmod` command. However, in practice you should apply this with care, since changes may affect other users on the system and beyond. You should use this only if you are a single user of a system or very sure what you are doing. A preferable strategy is to create a new group and move users into that group, dissociate them from the old groups, and test whether everything works as before.

## 3.4 Deleting Groups

Similar as with users, groups can be deleted using the `groupdel` command. Have a look at the manual page for the command.