Introduction to users

Users are the way how we interact with operating system. We need to log in, provide user, password, or (what is definitely better way, and strongly recommended) provide user and ssh key.

When logged, we are able to interact with the system, hardware and applications with defined boundaries. For example, we can see this directory, but not another, we can start and stop one aplication, but not another.

We will go through most important files, where the information about users are stored. But first we will learn how to get the information who are we. Let's get started!

Learn, who you are

Ok. Who are we? A user, well, thank you, Captain Obvious, but which one? Sometimes we can read it in our prompt (yes, we can configure prompt to show different things). But how to get this information in a proper way?

whoami

The whoami command shows the user we are logged in.

Another way to learn this information, not only about current user, but any user in the system.

id

We have here a few information. What is the User ID (UID), primary group ID (GID) and other groups to which user belongs to.

Ok, we can use this tool to ask about another user:

id ubuntu

We wish to learn more about user ubuntu. So, we know his UID (1000), primary GID (1000) and all secondary groups, with their names and GIDs.

UID, GID... What are they?

A little theory here. UID is an unique User Identifier, which identify this user through the system and determine which resources this user can access.

There is a good practice, even a standard, when UIDs are associated.

* UID 0 is reserved for root.
* 1-99 are reserved for predefined accounts (like games, mail, www-data, sys, bin and more)
* 100-999 reserved for system and administrative accounts.
* 1000+ are reserved for users

GID (Group IDentifier) has the same meaning, but for group. A group is an abstract which combines many users in similar entity (for the same privileges, purpose, actions, etc).

We can find here similar number associations:

* GID 0 is reserved for root.
* 1-99 are reserved for system and application use.
* 100+ are for users.

Files involved in users configuration

We have 4 files involved in configuration of users. Let's go through them one by one.

passwd

The first file is the passwd file. It is located, like most of config files, in /etc. Let's have a look on it.

clear && cat /etc/passwd

The structure of this file is as follows:

username:password:UID:GID:description:homedir:shell

* username - obvious, it is the name of the user
* password - password. Many years ago this was the place where password actually was stored. But for security reasons it chaned and now passwords are available in different form, somewhere else. We will go there later. x in this field means that password is encrypted and stored in the other file.
* UID - unique User IDentifier.
* GID - unique Group IDentifier.
* description - this field can contain many information. The real first and last name. Address. Phone number. Role in the organization. You name it.
* homedir - the location named as home directory. This is the "home" of the user, where he logs and stores his data.
* shell - simply speaking, the shell is the program which takes the commands sent by user, interprets it, and return the answer

We can use many shells. Today, however, the most popular is bash (**B**ourne **A**gain **SH**ell). It is an evolution of sh. Other examples are ksh, zsh (main shell on MacOS) or tcsh.

How to find the available shells? Execute

cat /etc/shells

This file contains all available, installed shells. And yes, shell can be installed like any other program.

Sometimes in the shell field we can find something else. it is /sbin/nologin. What is it? Some administrative users or technical users don't need to have the possibility to login. Another words, there is no need to perform actions interactively. Therefore, there is no need to login to this user. So, by setting shell as /sbin/nologin we ensure, that no one will log in as this user.

As we can see here, sometimes it is /usr/sbin/nologin.

shadow

The "magic" file. This is "the other file" I reffered to minutes ago. This file contains sensitive information about the user, like password, and other configurations. We will talk about these settings later, now we will focus on passwords.

clear && cat /etc/shadow

The part of the file, which we want to understand now are the first and second element. It is a username and password.

* if password is set for the user, it is hashed and not retrievable from the file (well, let's say "not retrievable"... by staring on it. There are tools which allow us to crack passwords).
* \* there is no password set (and very good, as we want to use passwordless approach with access keys, no passwords).
* ! password was never set.

group

The file /etc/group contains information about groups.

clear && cat /etc/group

The structure is as follows:

* group name
* password
* GID
* users belong to the group

Why password? Well, there is a possibility to have password set for groups. Honestly, during more than 20 years of my work with Linuxes, Unixes and other systems, I never saw password set for group, but, well, there is the possibility. And the principles for these passwords are the same like for users.

gshadow

The last file is /etc/gshadow.

clear && cat /etc/gshadow

This file contains encrypted passwords for groups.

Create user

We already know how to check users. It is time to create one or two.

In order to create user we can use one of two ways. First is useradd and second is adduser. What is the difference?

Let's figure it out.

whatis useradd  
whatis adduser

Well, this doesn't explain much.

Let's try another way. We will use two new commands. file and which.

which useradd  
which adduser

Command which show us where the executable is located. We asked where is the executable for useradd and the anser is "in /usr/bin/useradd".

file gives us information about the object. We can use it in simple way - file <object>, but we can combine these two commands together.

command $(another\_command)

What bash will do? First, the $(another\_command) will be executed. We enforce this by using $() then output of this command will be executed as parameter for command.

Let's see it in the real example:

file $(which useradd)

How bash will interpret it? which useradd will be executed and it will be translated (well, broadly speaking) into which /usr/bin/useradd and executed.

so, let's do it!

file $(which useradd)  
file $(which adduser)

So, there is a difference in the output. In fact, useradd is a binary, compiled with the system. adduser is a Perl script, which uses useradd in its backend. Another words, adduser uses useradd when creates user.

All right, after the lenghty intro, we can create users. We use both ways in simplest way, compare differences, and then we will use adduser to create final user.

useradd

useradd testuser1

adduser

adduser testuser2

Comparison

First difference is that adduser is more verbose. We see what is happening.

Second, very significant difference is that adduser is interactive. It asks for password for user and other information. Provide some test data during the execution of the binary. Let's have something to compare.

Let's compare the users.

grep testuser /etc/passwd

We see something similar to this:

testuser1:x:1001:1001::/home/testuser1:/bin/sh

testuser2:x:1002:1002:Some test user,,,,other info:/home/testuser2:/bin/bash

grep testuser /etc/shadow

and output:

testuser1:!:19002:0:99999:7:::

testuser2:$6$4MrAXI/aJY1cfwDU$kVdzcVnA1kT3CtZa4U/R0A0IiIta0QMG8psxc1PKk48DvZllXrecwNJ7Yfxr5xpJ4GAyAb/vjBYMAwnquYyKS.:19002:0:99999:7:::

What we see here?

* password is not set for testuser1
* description is added for testuser2
* testuser1 has sh set and testuser2 - bash as shell.

One more check:

ls -l /home

Well, testuser1 has the home directory defined, but it is not created.

Let's become an **admin**

Right, the true admin will use useradd :) I am kidding here, use what is better for you, but in my history I didn't use adduser too much. useradd is available on every Linux and works in the same way. And we, admins, have our ways with it.

Let's go through most important arguments, which we should consider to use during create of new user.

-d - create home directory in specified location, if we want to change

-m - create the home directory

-p - password

-s - provide shell

-c - comments, or description of the account

And generally that's it.

If you wish to see more options, execute

useradd -h .

Ok, now we know the basis, let's create our final user.

useradd testuser3 -s /bin/bash -m

Let's see all three accounts now.

grep testuser /etc/passwd

grep testuser /etc/shadow

ls -l /home

Modify existing user

There are some functionalities which we do not touch in the fundamentals course. These will be touched later in more advanced one. Here we go through two functionalities which allow us modify existing users.

Of course, many details can be changed through files manipulations, but... let's not go this way.

passwd

The most common reality of many admins is... changing the passwords. Unfortunatelly, this is something what still exists and we need to deal with it. In order to change the password we use command passwd <user>. If user is not provided, the password will be changed for current user.

We have our testuser3, created in previous chapter. We will modify this user.

grep testuser3 /etc/shadow

passwd testuser3

Provide the new password twice.

grep testuser3 /etc/shadow

As you can see, the record changed.

usermod

So, we know how to change password, it is time to modify the user.

With usermod testuser3 -g testuser2

we modified the primary group for testuser3. Right now it is testuser2.

grep testuser3 /etc/passwd

grep testuser3 /etc/group

The change is visible.

We can use GID also, not just name:

usermod testuser3 -g 1001

grep testuser3 /etc/passwd

grep testuser3 /etc/group

Did you observed how the GID changed from testuser3 to testuser2 and then to testuser1?

Let's come back to original setting usermod testuser3 -g 1003

Ok, now it is time to change or add secondary group.

grep testuser3 /etc/group

We just confirmed, testuser3 is attached to testuser3 group only.

usermod testuser3 -G testuser1

grep testuser3 /etc/group

Yep, we added the secondary group 1001. Ok, let's **add** one **more**.

usermod testuser3 -G testuser2

grep testuser3 /etc/group

Hm... We should have **two** secondary groups, not one!

This happened because we didn't do the operation correctly. It is a good habit to add a argument to usermod -G. a means here - append.

Ok, let's correct this.

usermod testuser3 -aG 1001

grep testuser3 /etc/group

It works as expected now.

With usermod -d we can create a home directory

usermod testuser3 -d /home/anotherdir

Let's see:

grep testuser3 /etc/passwd ok, here it looks fine.

ls -l /home Hmm... There is no directory...

All right, let's come back

usermod testuser3 -d /home/testuser3

In order to properly **move** home directory (not just create if it doesn't exist, remember adduser without parameters?) we need to add another argument.

usermod testuser3 -d /home/newdir -m

Let's check:

grep testuser3 /etc/passwd ok, here it looks good.

ls -l /home ok, we have a directory! Let's check the content of the directory and owner of the files:

ls -l /home/newdir Yep, it works now.

Shell

Let's modify the shell.

usermod testuser3 -s /bin/sh

grep testuser3 /etc/passwd

Delete user

We will learn more possibilities, like locking account, etc later. Now it is time to remove our user. In order to do that, execute

userdel testuser1 .

Well... Surprise.. We cannot remove the user's group because the group cannot be removed at this point. First, we need to clear other users attached to the group. Let's leave it, you know what to do :)

The user itself, however, was removed.

grep testuser /etc/passwd

grep testuser /etc/group

In order to remove secondary groups, we can run

usermod testuser3 -G ""

Another words, we set empty as secondary group.

grep testuser /etc/group

Ok, let's continue.

userdel testuser2

grep testuser /etc/passwd

grep testuser /etc/group

User and group are removed.

ls -l /home hm... the home directory wasn't removed...

In order to remove the files, it is a good practice (if we know what we are doing :) ) to add two arguments:

* r - remove files
* f - force removal (in case if files don't belongs to the user)

So, userdel -rf testuser3

Don't be alarmed, all files of the user means all files of the user. Mailbox too.

grep testuser /etc/passwd

ls -l /home

That works.