7.1.1 Linux User Overview

Click one of the buttons to take you to that part of the video.

Linux User Overview 0:00-1:06

One of the great things about Linux is the fact that it is a true multi-user operating system. A single Linux system can be configured with multiple user accounts. Each user on the system is provided with his or her own computing environment that is unique to just that user. You've probably noticed that you have to log in before you're allowed to use a Linux system. This is called authentication. In order to authenticate, you have to provide two credentials (by default, anyway). You have to provide a username, and you have to provide a password.

Be aware that a variety of other authentication methods are also available for Linux systems. Instead of manually entering a username and password, you could reconfigure Linux to require a smart card or to use a proximity card. You could configure it to use a biometric reader and so on. This is all possible because Linux uses the pluggable authentication module, or PAM, in order to manage authentication to the system. Essentially, PAM makes Linux authentication very, very flexible. You can use whatever authentication method you want.

User Environment 1:07-2:10

After you authenticate--once you're logged in--then your user's unique system environment is created for you. For example, here I've logged in to my Fedora system as my rtracy user account. As a result, my desktop is customized with my preferences. I'm given access to my user's /home directory, which is /home/rtracy. Notice over here, in the output of the finger command, that my user environment includes the path to my /home directory, which is /home/rtracy. By the way, remember that all user /home directories are within /home with the exception of root. Root's home directory is elsewhere. It is maintained at the root of the file system in a directory named, of all things, /root.

Also, notice that my default shell is configured for me, which is the Bash shell. If another user were to log in to this very same system, then their preferences would be used instead of mine.

Understand that each user account on a Linux system is assigned a unique user ID, which is just called the UID.

User IDs (UID) 2:07-3:25

Here's the important thing you need to remember: no two user accounts on the same system can have the same UID. On some distributions, the first regular user account created on the system is given a UID of 1000. Then the next user account will be given a UID of 1001, and so on. Other distributions may use a different numbering scheme, however, for the UID.

For example, the UID on some distributions may start at 500 instead of 1000. In this example, we are checking the user ID for a user on this system named ksanders. You can see that her user ID, UID, is 1001. This numbering scheme applies only to standard user accounts. Root, our superuser, is not a standard user; it's a superuser. Therefore, the root user, on almost all distributions that I've ever seen is assigned a user ID number of 0. The user ID is very important because the operating system is going to use your user ID number in order to control access to files and directories within the file system.

Understand that Linux is a very flexible operating system. One of its flexible features is its ability to store user accounts in a variety of different locations and in a variety of different systems. For our purposes here today, though, we're just going to focus on storing user accounts locally.

Locally Stored User Account Information 3:21-4:19

This option stores user and group information within the three files in the file system that you see here. Each of these files is located in the /etc directory.

First of all, the passwd file contains user account information for your system. The shadow file contains the passwords for the user accounts in the password file, which is kind of funny because you would think that, being named passwd, that's where your passwords would be. They're not. Only user accounts are in the passwd file. The actual passwords for those user accounts are stored in a different file called shadow.

Then, finally, we have the group file, which contains your system's group accounts.

Let's look at the passwd file first. As we said just a minute ago, the passwd file contains your Linux system's user accounts. Each user account on your system is represented by a single line in the file. For example, we have an account right here for the ksanders user, an account right here for the rtracy user, an account right here for the root user, and so on.

The Passwd File 4:19-9:15

Each user account within this file is composed of several different fields. Each field within the same line is separated by a full colon (:), which you see right here. First, we have the username, and then there's a field for the user's password. We'll talk about how that works here in just a second. We have the user ID; the group ID; the description field, which we typically use for the full name; the /home directory field; and then the default shell field.

An example is shown right here. We have a record here for the ksanders user. There's ksanders' username. This is the username that the user has to supply when they're logging in to the system. Notice that we do have, right here, a password field. That's why it's called the passwd file. Notice there's no password in here. It's just an x. The password field is actually a legacy field. At one time, back before the earth was round and dinosaurs still roamed, the user's password was actually stored within this passwd file. However, today, for security reasons, the password has been removed from the passwd file and is now saved in the shadow file. That's what's indicated by this x character in the password field. It basically tells everyone, "The password isn't here; it's in the shadow file. Don't bother looking here for the password."

The UID field contains the user ID for that user account. In this case, the UID for the ksanders account is 1001. The next field is the group ID field, or GID field. This references the group ID number of the user's default group. In this example, the GID for the ksanders user account is 1001. That means that the group that has a group ID number of 1001 is the ksanders user's default user group. As we said a minute ago, the description field is typically used for the user's full name.

Next, we have the /home directory field, which just contains the path to the user's /home directory. Then, finally, we have the shell field that specifies which shell will be used by default for the user. In this case, the ksanders user is going to use the Bash shell, which is usually default for most Linux distributions.

A minute ago, we looked at the passwd file and we looked at the root, the rtracy, and the ksanders user account. You may have noticed that between root and the rtracy user account, there were a whole bunch of other user accounts that didn't really look very familiar.

This is because these other user accounts are actually system user accounts. They're not used by an actual live human being to log in to the system. An example of three commonly used system user accounts are shown here: sshd, uucp, and wwwrun.

As I said, system user accounts cannot be used to actually log in to the system. Instead, they are used by daemons that are running on the system. Basically, the way it works is that when one of these daemons needs to do something in the Linux file system, it does so using one of these system user accounts from the passwd file. You'll note, over here, that the UIDs assigned to the system user accounts are much lower than the UIDs that are assigned to standard user accounts.

User IDs between 0 and 499 are usually reserved, on most distributions, for system user accounts. Be aware that this could change depending upon which distribution you're using.

For example, let's suppose that I've installed the FTP service on my Linux system. Using FTP, I'm allowed to transfer files back and forth across the network to and from that system. Let's suppose I connect to that FTP service on that Linux system from a different computer using an FTP client, and I log in as an anonymous user. As that user, I tried to upload a file to my FTP server. When that file arrives at the Linux system running the FTP daemon, that file needs to be written to the file system of that computer. Most likely, it's going to be written to the FTP service's default directory.

When it goes to write, it has to do so as a user account. Most likely, it's going to do so as the FTP system user. By using system user accounts, I can actually use permissions that are assigned to those system user accounts to control what a particular daemon can or cannot do in the file system of the Linux computer.

Next, let's look at the shadow file. With most Linux distributions that use local authentications, your users' passwords are going to be stored in this file. They're going to be stored in encrypted format. Otherwise, somebody could just grab ahold and get a copy of the shadow file and know everybody's password. The important thing to remember is that the shadow file is linked to the passwd file. Every user account in passwd must have a corresponding entry in shadow.

Just as with the passwd file, every user account within the shadow file is represented by a single line. An example is shown right here, for the ksanders user. The first field for each user contains the user's username. This has to match up with the same username in the passwd file.

The Shadow File 9:15-13:42

Then we have that user's actual password, right here. As you can see, this is with plain text of this user's password. This is an encrypted form of that user's password. Again, that's to prevent the passwords from being compromised by somebody who happens to look inside this file.

However, be aware that, for system user accounts, unlike standard user accounts, we don't actually configure a password because they can't be used for login anyway.

What you will see in this field instead of a password for system user accounts, you'll see just an asterisk (*) character. This basically indicates that these accounts are not allowed to actually log in to the system. For example, if I tried to log in to my system as my wwwrun system user, I would be denied access even though there does exist an account in my passwd file called wwwrun.

The next field, right here, is the last modified field. This field is somewhat confusing. Okay, I admit it; it's really confusing. Because, for a value, this field displays the number of days since January 1st, 1970, that the password was last changed. In this example, the password was last changed 15,043 days since January 1st, 1970. The next field is the min days field. This field displays the minimum number of days required before a password can be changed. This example is set to 0, which basically means the user is allowed to change the password whenever they want. If you wanted to restrict the user such that they had to wait a certain number of days before they're allowed to change the password, you could enter that number of days in this field.

The next field is the max days field. This field does just the opposite of the min days field. The max days field specifies the maximum number of days before a password has to be changed. In other words, how long can the user have the same password? In this example, it's set to 99,999. Basically, it means that a password change isn't required. The user can keep the same password as long as they want unless they grow to be very, very, very old. If you want to change that such that the user is required to change their password, say, every 45 days, that's the value that would go in this field.

The next field is called the warning days field. This field specifies the number of days prior to the password expiring that the user will receive a warning that the password is going to expire. In this case, it's set to seven days. Seven days before the password is going to expire, the user is going to get a warning that they need to go change their password.

The next couple of fields for this user account are actually not populated. They're just null values. This one is the disabled days field, which configures the number of days to wait after a password has expired to go ahead and then disable the account. Right now, it's set to a null value so that won't happen. But if we did have password expiration configured, say, after 45 days, and then we give the user a week to go ahead and change their password after the expiration date is hit, we would put in this field. This is basically giving the user a grace period if they wait longer than that. Then their user account is going to be automatically disabled.

Finally, we have the expire field, which, again, is set to a null value for this user account. This field configures the number of days since January 1st, 1970, after which the account will be disabled. Because in this example it's set to a null value, it basically means that the account will never expire.

As you can see, it's absolutely critical that the passwd file and the shadow file stay synchronized with each other. If they get out of whack, then it's possible that a user may not be able to log in or that a service will not be able to access the file system appropriately. The good news is these files, in my experience, usually just stay in sync without any intervention on the part of the system administrator. Everything works the way it's supposed to. The only times I've seen these two files get out of whack is when the administrator decides to open up a text editor and then manually edit either the passwd or the shadow file. You really should NOT do this. Your Linux system includes a whole bunch of different utilities that you should use to manage user accounts and passwords. You should always use these utilities and never use a text editor to manage these user accounts. If you use the appropriate utility, then both of these files, passwd and shadow, will be updated appropriately and will stay in sync with each other.

However, if you suspect that something is amiss, that maybe these two files aren't synchronized correctly, you can use the password check command pwck at the shell prompt. This utility will verify each line in the two files and make sure that they're valid, and any errors will be reported on the screen.

---

passwd and shadow File Synchronization 13:41-14:12

If you discover that, for some reason, your passwd and shadow files got out of synchronization, you can then use the pwconv command at the shell prompt to fix the files. Basically, this utility is going to go through and add any missing user accounts from the passwd file to the shadow file.

That's it for this lesson. In this lesson, we talked about Linux user accounts. We first talked about the role and function of user accounts. Then we talked about where user accounts are stored on a Linux system. Then we reviewed the role, function, and structure of the passwd file and the shadow file.

Summary 15:20-14:12

Copyright © 2022 TestOut Corporation All rights reserved.