User groups play an important role on Linux systems. They provide an easy way for a select groups of users to share files with each other. They also allow sysadmins to more effectively manage user privileges, since they can assign privileges to groups rather than individual users.

While a user group is generally created whenever a user account is added to a system, there’s still a lot to know about how they work and how to work with them.

**[ Two-Minute Linux Tips:**[**Learn how to master a host of Linux commands in these 2-minute video tutorials**](https://www.youtube.com/playlist?list=PL7D2RMSmRO9J8OTpjFECi8DJiTQdd4hua)**]**

**One user, one group?**

Most user accounts on Linux systems are set up with the user and group names the same. The user "jdoe" will be set up with a group named "jdoe" and will be the only member of that newly created group. The user’s login name, user id, and group id will be added to the **/etc/passwd** and **/etc/group** files when the account is added, as shown in this example:

$ sudo useradd jdoe

$ grep jdoe /etc/passwd

jdoe:x:1066:1066:Jane Doe:/home/jdoe:/bin/sh

$ grep jdoe /etc/group

jdoe:x:1066:

The values in these files allow the system to translate between the text (jdoe) and numeric (1066) versions of the user id — jdoe is 1066 and 1066 is jdoe.

The assigned UID (user id) and GID (group id) for each user are generally the same and configured sequentially. If Jane Doe in the above example were the most recently added user, the next new user would likely be assigned 1067 as their user and group IDs.

**GID = UID?**

UIDs and GIDs can get of out sync. For example, if you add a group using the **groupadd** command without specifying a group id, your system will assign the next available group id (in this case, 1067). The next user to be added to the system would then get 1067 as a UID but 1068 as a GID.

You can avoid this issue by specifying a smaller group id when you add a group rather than going with the default. In this command, we add a new group and provide a GID that is smaller than the range used for user accounts.

$ sudo groupadd -g 500 devops

If it works better for you, you can specify a shared group when you create accounts. For example, you might want to assign new development staff members to a devops group instead of putting each one in their own group.

$ sudo useradd -g staff bennyg

$ grep bennyg /etc/passwd

bennyg:x:1064:50::/home/bennyg:/bin/sh

**Primary and secondary groups**

There are actually two types of groups — primary and secondary.

The **primary group** is the one that’s recorded in the **/etc/passwd** file, configured when an account is set up. When a user creates a file, it’s their primary group that is associated with it.

$ whoami

jdoe

$ grep jdoe /etc/passwd

jdoe:x:1066:1066:John Doe:/home/jdoe:/bin/bash

^

|

+-------- primary group

$ touch newfile

$ ls -l newfile

-rw-rw-r-- 1 jdoe jdoe 0 Jul 16 15:22 newfile

^

|

+-------- primary group

**Secondary groups** are those that users might be added to once they already have accounts. Secondary group memberships show up in the /etc/group file.

$ grep devops /etc/group

devops:x:500:shs,jadep

^

|

+-------- secondary group for shs and jadep

The **/etc/group** file assigns names to user groups (e.g., 500 = devops) and records secondary group members.

**Preferred convention**

The convention of having each user a member of their own group and optionally a member of any number of secondary groups allows users to more easily separate files that are personal from those they need to share with co-workers. When a user creates a file, members of the various user groups they belong to don't necessarily have access. A user will have to use the **chgrp** command to associate a file with a secondary group.

**There’s no place like /home**

One important detail when adding a new account is that the **useradd** command does not necessarily add a home directory for a new user. If you want this step to be taken only some of the time, you can add **-m** (think of this as the “make home” option) with your useradd commands.

$ sudo useradd -m -g devops -c "John Doe" jdoe2

The options in this command:

* **-m** creates the home directory and populates it with start-up files
* **-g** specifies the group to assign the user to
* **-c** adds a descriptor for the account (usually the person’s name)

If you want a home directory to be created *all* of the time, you can change the default behavior by editing the **/etc/login.defs** file. Change or add a setting for the CREATE\_HOME variable and set it to “yes”:

$ grep CREATE\_HOME /etc/login.defs

CREATE\_HOME yes

Another option is to set yourself up with an alias so that **useradd** always uses the -m option.

$ alias useradd=’useradd -m’

Make sure you add the alias to your ~/.bashrc or similar start-up file to make it permanent.

**Looking into /etc/login.defs**

Here’s a command to list all the setting in the /etc/login.defs file. The **grep** commands are hiding comments and blank lines.

$ cat /etc/login.defs | grep -v "^#" | grep -v "^$"

MAIL\_DIR /var/mail

FAILLOG\_ENAB yes

LOG\_UNKFAIL\_ENAB no

LOG\_OK\_LOGINS no

SYSLOG\_SU\_ENAB yes

SYSLOG\_SG\_ENAB yes

FTMP\_FILE /var/log/btmp

SU\_NAME su

HUSHLOGIN\_FILE .hushlogin

ENV\_SUPATH PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

ENV\_PATH PATH=/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games

TTYGROUP tty

TTYPERM 0600

ERASECHAR 0177

KILLCHAR 025

UMASK 022

PASS\_MAX\_DAYS 99999

PASS\_MIN\_DAYS 0

PASS\_WARN\_AGE 7

UID\_MIN 1000

UID\_MAX 60000

GID\_MIN 1000

GID\_MAX 60000

LOGIN\_RETRIES 5

LOGIN\_TIMEOUT 60

CHFN\_RESTRICT rwh

DEFAULT\_HOME yes

CREATE\_HOME yes <===

USERGROUPS\_ENAB yes

ENCRYPT\_METHOD SHA512

Notice the various settings in this file determine the range of user ids to be used along with password aging and other setting (e.g., umask).

**How to display a user’s groups**

Users can be members of multiple groups for various reasons. Group membership gives a user access to group-owned files and directories, and sometimes this behavior is critical. To generate a list of the groups that some user belongs to, use the **groups** command.

$ groups jdoe

jdoe : jdoe adm admin cdrom sudo dip plugdev lpadmin staff sambashare

You can list your own groups by typing “groups” without an argument.

**How to add users to groups**

If you want to add an existing user to another group, you can do that with a command like this:

$ sudo usermod -a -G devops jdoe

You can also add a user to multiple groups by specifying the groups in a comma-separated list:

$ sudo usermod -a -G devops,mgrs jdoe

The **-a** argument means “add” while **-G** lists the groups.

You can remove a user from a group by editing the **/etc/group** file and removing the username from the list. The usermod command may also have an option for removing a member from a group.

fish:x:16:nemo,dory,shark

|

V

fish:x:16:nemo,dory

**Wrap-up**

Adding and managing user groups isn't particularly difficult, but consistency in how you configure accounts can make it easier in the long run.