13.1 Introduction

User accounts are designed to provide security on a Linux operating system. Each person on the system must log in using a user account and the user account will either allow the person to access a specific file/directory or disallow such access. This is accomplished by file permissions, a topic discussed in a later chapter.

User accounts also belong to groups which can also be used to provide access to files/directories. Each user belongs to at least one group (often many) to allow users to more easily share data that is stored in files with other users.

User and group account data is stored in database files. Knowing the content of these files is important, since it will allow you to better understand which users have access to files and directories on the system. These database files also contain vital security information that may affect the ability of a user to access the system (login).

There are several commands that will provide you with the ability to see user and group account information, as well as allow you to switch from one user account to another (provided you have the appropriate authority to do so). These commands are valuable for investigating usage of the system, troubleshooting system problems and for monitoring unauthorized access to the system.

13.2 Linux Essentials Exam Objectives

This chapter will cover the topics for the following Linux Essentials exam objectives:

Topic 5: Security and File Permissions (weight: 7)

- **5.1: Basic Security and Identifying User Types**
  - Weight: 2
  - Description: Various types of users on a Linux system.
  - Key Knowledge Areas:
    - Root and Standard Users
    - System Users
  - The following is a partial list of the used files, terms, and utilities:
    - /etc/passwd, /etc/group
    - id, who, w
    - sudo
  - Things that are nice to know:
    - su

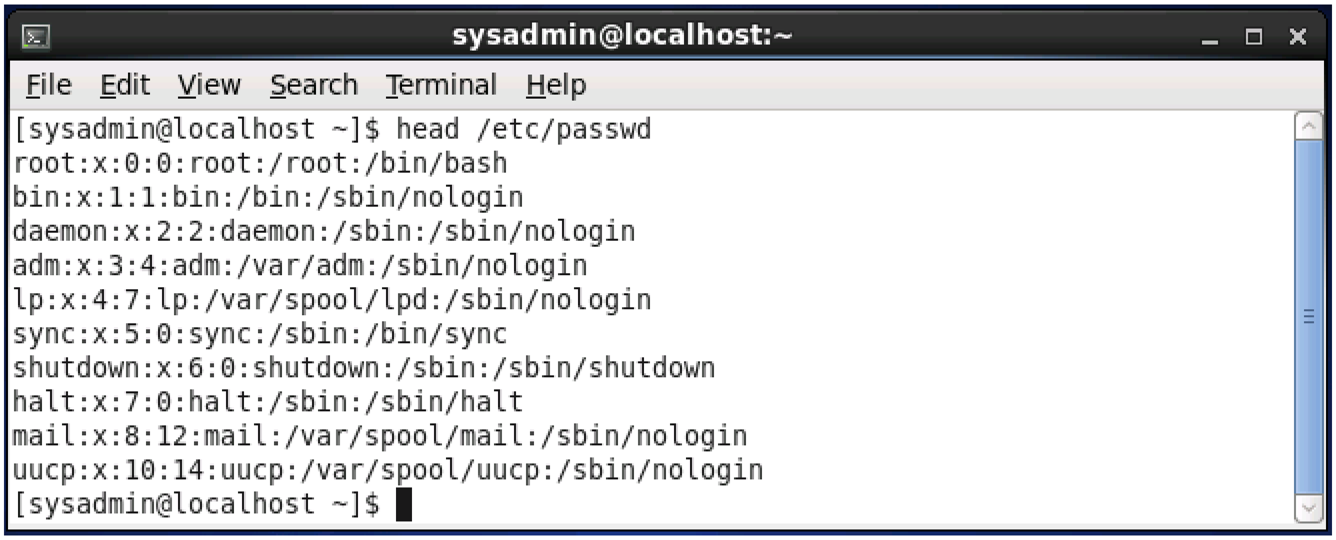
13.3 User Accounts

There are several text files in the `/etc` directory that contain the account data of the users and groups that are defined on the system. For example, if you wanted to see if a specific user account has been defined on the system, then the place to check is the `/etc/passwd` file.

The `/etc/passwd` file defines some of the account information for user accounts. Oddly enough, the passwords for the accounts are not stored in the `/etc/passwd` file, as the file name implies, but rather the `/etc/shadow` file.

13.3.1 /etc/passwd File

Each line of the `/etc/passwd` file relates to a user account. The following graphic shows the first ten lines of a typical `/etc/passwd` file:



```
[sysadmin@localhost ~]$ head /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
[sysadmin@localhost ~]$
```

Each line is separated into fields by colon characters. The fields from left to right are as follows:

```
name:password placeholder:user id:primary group id:comment:home directory:shell
```

The following table describes each of these fields in detail, using the first line of the output of the previous graphic (**root:x:0:0:root:/root:/bin/bash**):

Field	**Example**	**Description**
name	root	This is the name of the account. This name is used by the person when they log in to the system and when file ownership is provided with the `ls -l` command. Typically, the system uses the *user ID* (see below) internally and the account name is provided to make it easiest for regular users to refer to the account.

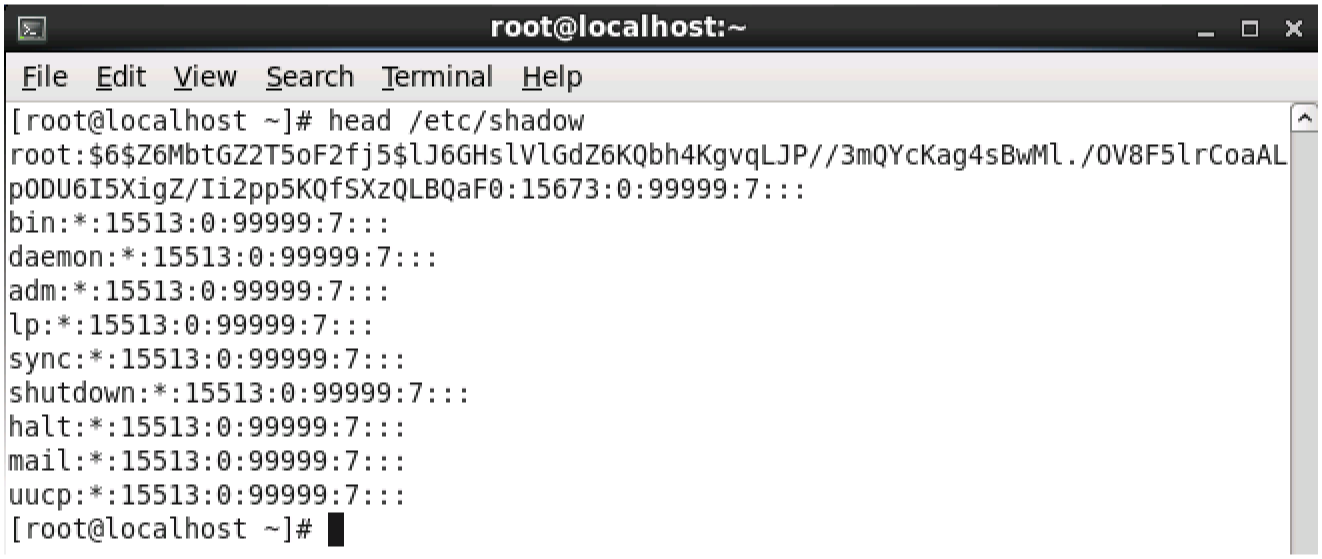
Field	Example	Description
		The *root* account is normally a special administrative account. However, it is important to note that not all systems have a root account and that it is really the user id of 0 (zero) that provides administrative privileges on the system.
password placeholder	x	At one time, the password for the user was actually stored in this location, but now it is stored in the `/etc/shadow` file. The **x** in the password placeholder field indicates to the system that the password is not stored here, but rather in the `/etc/shadow` file.
user id	0	Each account is assigned a user ID (UID). The UID is what really defines the account as the user name is normally not directly used by the system. For example, files are owned by UIDs, not by user names. Some UIDs are special. For example, the UID of 0 provides that user account with administrative privileges. UIDs below 500 (on some Linux distributions 1,000) are reserved for system accounts. System accounts will be covered in more detail later in this chapter.
primary group id	0	Each file and directory is owned by a user account. Normally the person who creates the account owns the file. In addition, each file is owned by a group, normally the user's *primary group*. Groups are assigned numeric IDs just like users are. When a user creates a file, the file is owned by the user's UID and also owned by a group id (GID), the user's primary GID. This field defines which GID is the user's primary GID. Besides, providing default group ownership on a file, this field also indicates that the user is a member of the group, which means the user will have special permissions on any file that is owned by this group. Permissions will be covered in detail in a later chapter.
comment	root	This field can contain any information about the user, including their real (full) name and other useful information. This field is also called the *GECOS* (General Electric Comprehensive Operating System) field. GECOS is a rarely used predefined format for this field that defines a comma-separated list

Field	Example	Description
		of items, including the user's full name, office location, phone number and additional information. An administrator can modify GECOS information with the `chfn` command and users can display this information with the `finger` command.
home directory	/root	This field defines the location of the user's home directory. For regular users, this would normally be `/home/username` where *username* is replaced with the user's username. For example, a username of bob would have a home directory of `/home/bob`. The root user normally has a different place for the home directory: `/root`. System accounts rarely have home directories as they typically are not used to create or store files.
shell	/bin/bash	This is the location of the user's login shell. By default, the user is "placed in" this shell whenever the user logs into a command line environment or opens a terminal window. The user could then switch to a different shell by typing the name of the shell, for example: `/bin/tcsh`. The bash shell (`/bin/bash`) is the most common shell for Linux users.

Note that while this chapter describes the contents of the user and group files, the next chapter will describe the commands and tools used to modify user and group account information.

13.3.2 /etc/shadow File

As previously mentioned, the `/etc/shadow` file contains account information related to the user's password. A typical `/etc/shadow` file would look like the following graphic:



```
[root@localhost ~]# head /etc/shadow
root:$6$Z6MbtGZ2T5oF2fj5$lJ6GHslVlGdZ6KQbh4KgvqLJP//3mQYcKag4sBwMl./OV8F5lrCoaAL
pODU6I5XigZ/Ii2pp5KQfSXzQLBQaF0:15673:0:99999:7:::
bin:*:15513:0:99999:7:::
daemon:*:15513:0:99999:7:::
adm:*:15513:0:99999:7:::
lp:*:15513:0:99999:7:::
sync:*:15513:0:99999:7:::
shutdown:*:15513:0:99999:7:::
halt:*:15513:0:99999:7:::
mail:*:15513:0:99999:7:::
uucp:*:15513:0:99999:7:::
[root@localhost ~]# 
```

The fields of the `/etc/shadow` file are:

```
name:password:last change:min:max:warn:inactive:expire:reserved
```

The following table describes the fields of the */etc/shadow* file in more detail, using the following account that describes a typical user account:

```
sysadmin:$6$lS6WJ9O/fNmEzrIi$kO9NKRBjLJJTlZD.L1Dc2xwcuUYaYwCTS.gt4elijSQW8ZDp6GLYAx.T
RNNpUdAgUXUrzDuAPsYs5YHZNAorI1:15020:5:30:7:60:15050:
```

Field	**Example**	**Description**
name	sysadmin	This is the name of the account, which matches the account name in the `/etc/passwd` file.
password	$6$.........rI1	The password field contains the encrypted password for the account. This very long string (which was truncated in the example to the left of this cell) is a one-way encryption, meaning that it can't be "reversed" to determine the original password. While regular users have encrypted passwords in this field, system accounts will have an * character in this field. See more details about system accounts later in this chapter.
last change	15020	This field contains a number that represents the last time the password was changed. The number 15020 is the number of days since January 1, 1970 (called the Epoch) and the last day the account

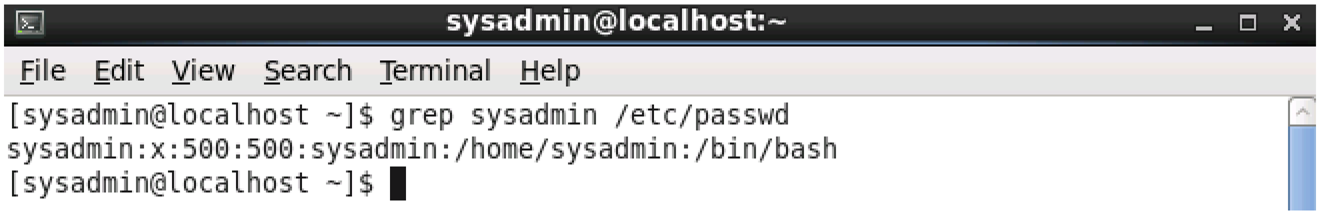
Field	Example	Description
		password was changed. This value is automatically generated when the user's password is modified. This value is important as it is used by the *password aging* features provided by the rest of the fields of this file.
min	5	This is one of the *password aging* fields; a non-zero value in this field indicates that after a user changes their password, the password can't be changed again for the specified number of days (5 days in this example). This field is important when the **max** field is used (see below). A value of zero in this field means the user can always change their password.
max	5	This field is used to force users to change their passwords on a regular basis. A value of 30 in this field means the user must change their password at least every 30 days to avoid having their account "locked out". Note that if the **min** field is set to 0, the user may be able to immediately set their password back to the original value, defeating the purpose of forcing the user to change their password every 30 days. So, if the **max** field is set, the **min** field is normally set as well. For example, a min:max of 5:60 means the user must change their password every 60 days and, after changing, the user must wait 5 days before they can change their password again. If the **max** field is set to 99999, the maximum possible value, then the user essentially never has to change their password (because 99999 days is approximately 274 years).
warn	7	If the **max** field is set, the **warn** field indicates that the user would be "warned" when the **max** timeframe is approaching. For example, if **warn** is set to 7, then any time during the 7 days before the **max** timeframe is reached, the user would be warned to change their password during the login processes. The user is only warned at login, so some administrators have taken the approach to set the **warn** field to a higher value to provide a greater chance of having a warning issued. If the **max** timeframe is set to 99999, then the **warn** field is

Field	Example	Description
		essentially useless
inactive	60	If the user ignores the warnings and they exceed the **max** password timeframe, their account will be locked out. In that case, the **inactive** field provides the user with a "grace" period in which their password can be changed, but only during the login process. If the **inactive** field is set to 60, the user has 60 grace days to change to a new password. If they fail to do so, then the administrator would be needed to reset the password for the user.
expire	15050	This field represents the number of days from January 1, 1970 and the day the account will "expire". An expired account will be locked, not deleted, meaning the administrator can reset the password to unlock the account. Accounts with expiration dates are normally provided to temporary employees or contractors. The account will automatically expire after the user's last day of work. When an administrator sets this field, a tool is used to convert from a real date to an "Epoch" date. There are also several free converters available on the Internet.
reserved		Currently not used, this field is reserved for future use.

Regular users can't view the contents of the `/etc/shadow` file for security reasons. In order to view the contents of this file, you must log in as the administrator (the root account).

13.3.3 Viewing Account Information

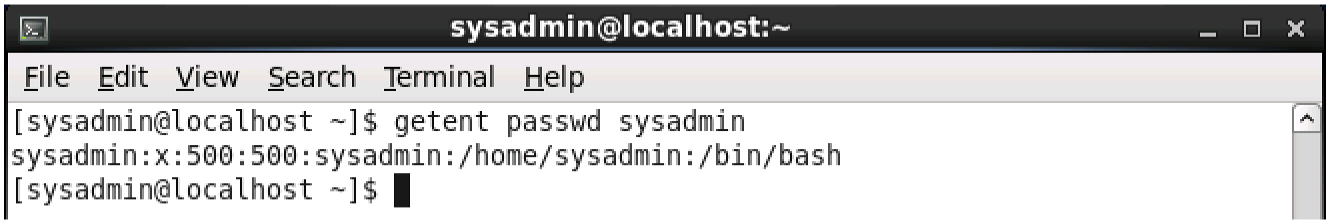
A good way to view account information from the `/etc/passwd` file is to use the `grep` command to output just the line containing the account that you are interested in. For example, to see the account information for the user name named "sysadmin", use the `grep sysadmin /etc/passwd` command:



```
[sysadmin@localhost ~]$ grep sysadmin /etc/passwd
sysadmin:x:500:500:sysadmin:/home/sysadmin:/bin/bash
[sysadmin@localhost ~]$ 
```

Another technique for retrieving user information that is normally contained in the `/etc/passwd`and `/etc/shadow` files is to use the `getent` command. One advantage to using the `getent` command is that it can retrieve account information that is defined locally (`/etc/passwd` and`/etc/shadow`) or on a network directory server.

The general syntax of a `getent` command is: `getent` *`database record`*. For example, the `getent passwd sysadmin` command would retrieve the passwd account information for the sysadmin user:

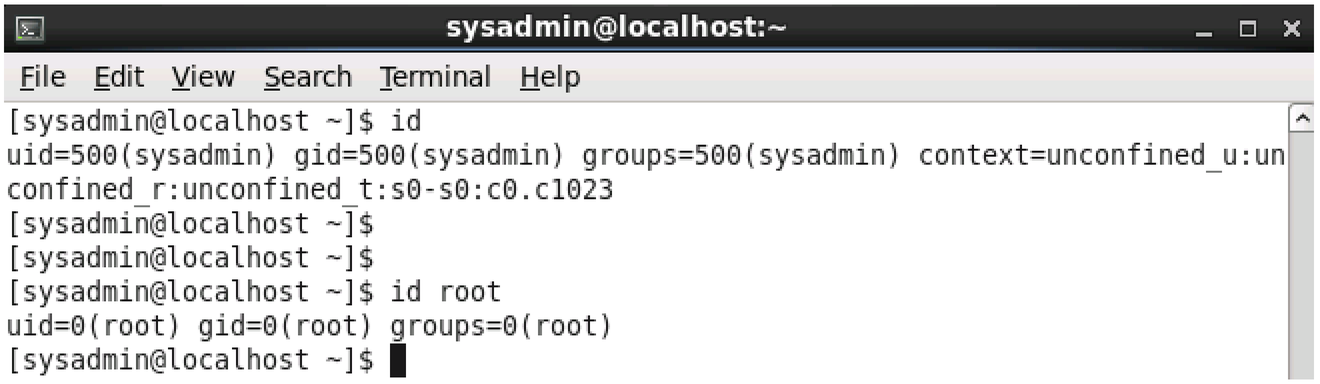


```
[sysadmin@localhost ~]$ getent passwd sysadmin
sysadmin:x:500:500:sysadmin:/home/sysadmin:/bin/bash
[sysadmin@localhost ~]$ 
```

13.3.4 Viewing Login Information

When you log into different user accounts, it can be confusing as to who you are currently logged in as. To verify your identity (view whose account you are currently using) you can execute the `id` command.

The `id` command will report the current identity, both by user name and user ID. In addition to providing the user account information, the group membership is also displayed. With no argument, the `id` command will display your identity. Given a user name as an argument, such as `id root`, the command will display other account information:



```
[sysadmin@localhost ~]$ id
uid=500(sysadmin) gid=500(sysadmin) groups=500(sysadmin) context=unconfined_u:un
confined_r:unconfined_t:s0-s0:c0.c1023
[sysadmin@localhost ~]$
[sysadmin@localhost ~]$
[sysadmin@localhost ~]$ id root
uid=0(root) gid=0(root) groups=0(root)
[sysadmin@localhost ~]$ 
```

The output includes Security Enhanced Linux context (the **content=** part of the output), a topic which is beyond the scope of this course.

13.3.5 System Accounts

Users will log into the system using regular user accounts. Typically, these accounts have UID values of greater than 500 (on some systems 1,000).

The *root* user has special access to the system. As previously mentioned, this special access is actually provided to the account with a UID of 0.

There are additional accounts that are not designed for users to log into. These accounts, typically from UID 1 to UID 499, are called system accounts and they are designed to provide accounts for services that are running on the system.

System accounts have some fields in the `/etc/passwd` and `/etc/shadow` files that are different than other accounts. For example, in the `/etc/passwd` file, system accounts will have a non-login program in the "login shell" field:

```
bin:x:1:1:bin:/bin:/sbin:/sbin/nologin
```

In the `/etc/shadow` file, system accounts will typically have an * character in place of the password field:

```
bin:*:15513:0:99999:7:::
```

There are a few important things that you should remember about system accounts:

- Most are necessary for the system to function correctly.
- You should not delete a system account unless you are absolutely certain that removing the account won't cause problems.
- As you gain more experience, you should learn what each system account does. System administrators are tasked with ensuring the security on the system and that includes properly securing the system accounts.

13.4 Group Accounts

Your level of access to a system is not determined solely by your user account. Each user can be a member of one or more groups, which can also affect your level of access to the system.

Traditionally, UNIX systems limited users to belonging to no more than a total of sixteen groups, but the recent Linux kernels support users with over sixty-five thousand group memberships.

As you have already seen, the `/etc/passwd` file defines the primary group membership for a user. Supplemental group memberships (or secondary group memberships), as well as the groups themselves, are defined in the `/etc/group` file.

To view information about a specific group, either the `grep` or `getent` commands can be used. For example, the following commands will display the "mail" group account information:

```
[sysadmin@localhost ~]$ grep mail /etc/group
mail:x:12:mail,postfix
[sysadmin@localhost ~]$ getent group mail
mail:x:12:mail,postfix
```

13.4.1 /etc/group File

The `/etc/group` file is a colon delimited file with the following fields:

```
group_name:password_placeholder:GID:user_list
```

The following table describes the fields of the `/etc/group` file in more detail, using a line that describes a typical group account:

```
mail:x:12:mail,postfix
```

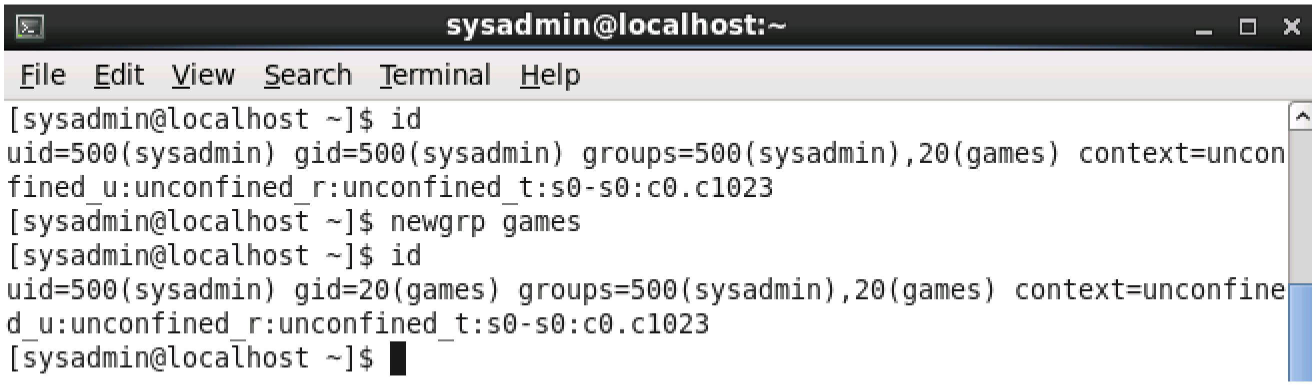
Field	**Example**	**Description**
group_name	mail	This field contains the group name. As with user names, group names are easier for people to remember. The system typically uses Group IDs (GIDs) rather than group names.
password_placeholder	x	While there are passwords for groups, they are rarely used in Linux. In the event that the administrator makes a group password, it would be stored in a different file (`/etc/gshadow`) because the group password is no longer stored in the `/etc/group` file. The "x" in this field is used to indicate that the password is not stored in this file. Group passwords are beyond the scope of this course.

Field	Example	Description
GID	12	Each group is associated with a unique Group ID (GID) which is placed in this field.
user_list	mail,postfix	This last field is used to indicate who is a member of the group. While primary group membership is defined in the`/etc/passwd` file, users who are assigned to additional groups would have their user name placed in this field of the `/etc/group` file. In this case, the **mail** and **postfix** users are secondary members of the **mail** group. It is very common for a user name to also appear as a group name. It is also common for a user to belong to a group with the same name.

13.4.2 Changing Groups

Groups are primarily used for controlling access to files. By default, any new file that a user creates will be owned by the user's primary group.

A user can create a file that is owned by one of their secondary groups by using the `newgrp group_name` command. This command temporarily changes the user's primary group to a different group by opening a new shell with a different primary group. The `id` command can be used to verify that the user's primary group is different:



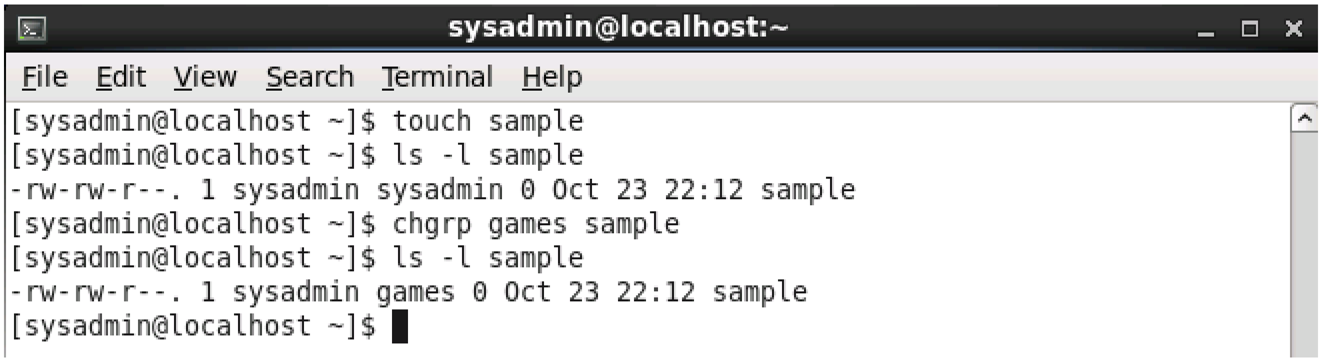
```
[sysadmin@localhost ~]$ id
uid=500(sysadmin) gid=500(sysadmin) groups=500(sysadmin),20(games) context=uncon
fined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[sysadmin@localhost ~]$ newgrp games
[sysadmin@localhost ~]$ id
uid=500(sysadmin) gid=20(games) groups=500(sysadmin),20(games) context=unconfine
d_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[sysadmin@localhost ~]$ 
```

After using the shell started by the newgrp command to create the necessary files, return to your original shell by using the exit command.

Important: To use the `newgrp` command, a user must be a part of the group that they are switching to. Also, on some systems, the `newgrp` command is disabled for regular user use due to group passwords.

13.4.3 Changing the Group Ownership of an Existing File

To change the group owner of an existing file the `chgrp group_name file` command can be used. Users can only change the ownership of files that they own. The new group owner of the file must also be a group that the user is a member of:



```
[sysadmin@localhost ~]$ touch sample
[sysadmin@localhost ~]$ ls -l sample
-rw-rw-r--. 1 sysadmin sysadmin 0 Oct 23 22:12 sample
[sysadmin@localhost ~]$ chgrp games sample
[sysadmin@localhost ~]$ ls -l sample
-rw-rw-r--. 1 sysadmin games 0 Oct 23 22:12 sample
[sysadmin@localhost ~]$
```

To change the group ownership of all of the files of a directory structure, use the `-R` option to the `chgrp` command. For example, the `chgrp -R games test_dir` command would change the group ownership of the test_dir directory and all files and sub directories of the test_dir directory.

There is also a `chown` command that can be used to change the user owner of a file or directory. However, this command can only be used by the root user. Regular users can't "give" their files to another user.

13.5 Logging In As Root

There are many different ways to execute a command that requires administrative or root privileges. As already mentioned, logging in to the system as the root user allows you to execute commands as the administrator. This is potentially dangerous because you may forget that you are logged in as root and might run a command that could cause problems on the system. As a result, it is not recommended to login as the root user directly.

Because using the root account is potentially dangerous, you should only execute commands as root if administrative privileges are needed. If the root account is disabled, as it is on the Ubuntu distribution, then administrative commands can be executed using the `sudo` command. If the root account is enabled, then a regular user can execute the `su` command to switch accounts to the root account.

When you login to the system directly as root to execute commands, then everything about your session runs as the root user. If using the graphical environment, this is especially dangerous as the graphical login process is comprised of many different executables (programs that run during login) . Each program that runs as the root user represents a greater threat than a process run as a standard user, as those programs would be allowed to do nearly anything, whereas standard user programs are very restricted in what they can do.

The other potential danger with logging into the system as root is that a person that does this may forget to log out to do their non-administrative work. This means that programs such as browsers, email clients, etc. would be run as the root user without restrictions on what they could do. The fact that several distributions of Linux, notably Ubuntu, do not allow users to login as the root user should be enough implication that this is not the preferred way to perform administrative tasks.

13.6 Using the su Command

The `su` command allows you to run a shell as a different user. By default, if a user account is not specified, the su command will open a new shell as the root user. While switching to the root user is what the su command is used for most frequently, it can also switch to other users as well.

One common option that is used with the `su` command is the `-l` option, which results in the new shell being a *login* shell. Using the `su` command with a login shell option is often important to ensuring that any commands executed will run correctly, as the login shell fully configures the new shell with the settings of the new user. A *non-login shell* essentially just changes the UID, but doesn't fully log the user in. The `-l` option can be abbreviated as simply - or spelled out as `--login`.

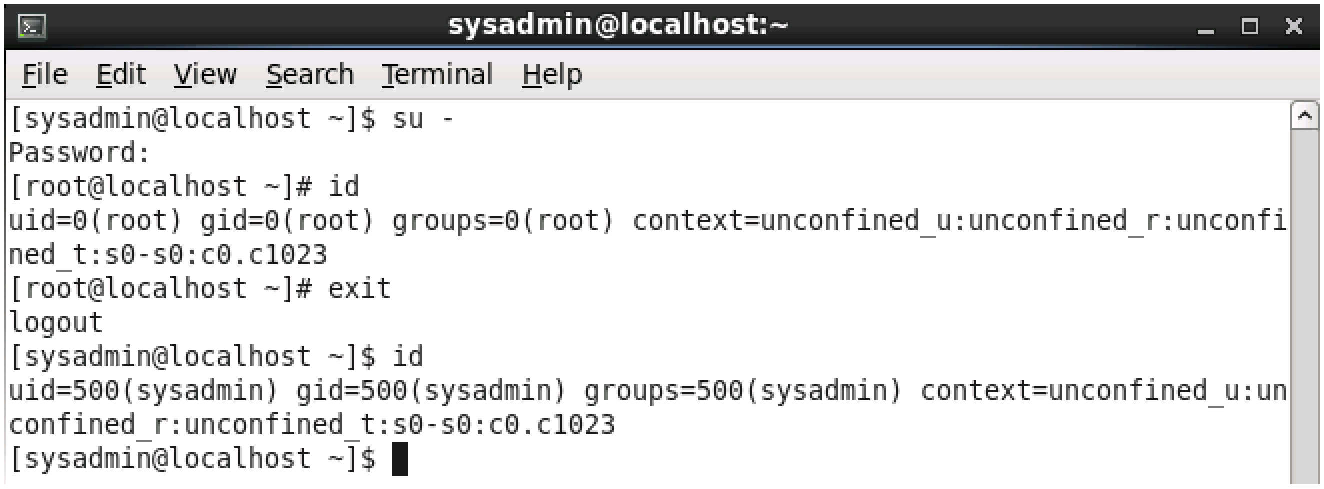
Since the root account is used by default with the su command, the following two commands are equivalent ways to start a shell as the root user:

```
su - root

su -
```

After pressing **Enter** to execute either one of these commands, the user must provide the password of the root user to start the shell as the root user. If you don't know the password of the account that you are shifting to, then the `su` command will fail.

After using the shell started by the `su` command to perform the necessary administrative tasks, return to your original shell (and original user account) by using the `exit` command.



```
[sysadmin@localhost ~]$ su -
Password:
[root@localhost ~]# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfi
ned_t:s0-s0:c0.c1023
[root@localhost ~]# exit
logout
[sysadmin@localhost ~]$ id
uid=500(sysadmin) gid=500(sysadmin) groups=500(sysadmin) context=unconfined_u:un
confined_r:unconfined_t:s0-s0:c0.c1023
[sysadmin@localhost ~]$ 
```

13.7 Using the sudo Command

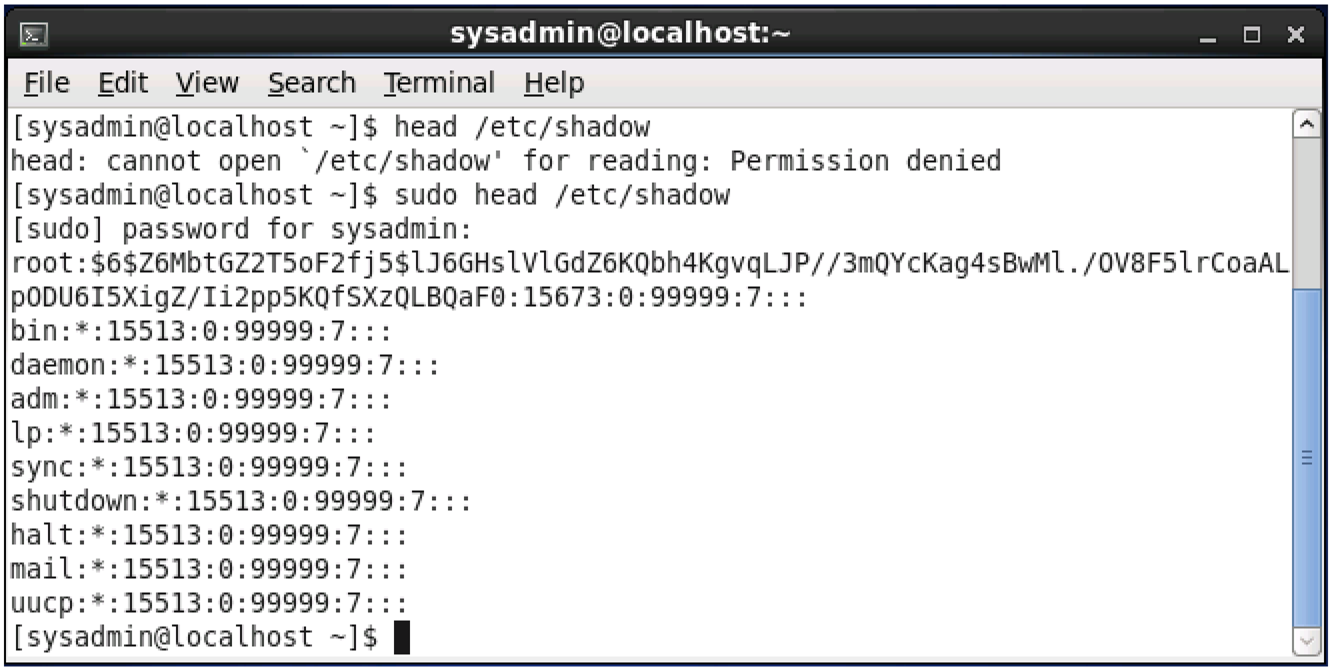
In distributions that do not allow the root user to login directly or via the `su` command, the installation process automatically configures one user account to be able to use the `sudo` command to execute commands as if they were executed by the root user.

Like the `su` command, the `sudo` command assumes by default the root user account should be used to execute commands; to specify a different user account use the `-u` option.

When using the `sudo` command to execute a command as the root user, the command will prompt for the user's own password (not that of the root user). This is a security feature that would prevent unauthorized root access if the user were to leave their computer unattended. The prompt for the password will not appear again as long as the user continues to execute `sudo` commands less than five minutes apart.

Using the `sudo` command to execute an administrative command will result in an entry placed in a log file. This entry will include the user name who executed the command, the command that was executed and the date and time when the comment was executed. This allows for increased accountability, compared to a system where many users might know the root password and can either login directly as root or use the `su` command to execute commands as the root user.

You can use the `sudo` command to execute a command that requires root privileges. For example, you need to be the root user in order to view the `/etc/shadow` file. The `sudo head /etc/shadow` command would run the `head` command as the root user:



```
[sysadmin@localhost ~]$ head /etc/shadow
head: cannot open `/etc/shadow' for reading: Permission denied
[sysadmin@localhost ~]$ sudo head /etc/shadow
[sudo] password for sysadmin:
root:$6$Z6MbtGZ2T5oF2fj5$lJ6GHslVlGdZ6KQbh4KgvqLJP//3mQYcKag4sBwMl./OV8F5lrCoaAL
pODU6I5XigZ/Ii2pp5KQfSXzQLBQaF0:15673:0:99999:7:::
bin:*:15513:0:99999:7:::
daemon:*:15513:0:99999:7:::
adm:*:15513:0:99999:7:::
lp:*:15513:0:99999:7:::
sync:*:15513:0:99999:7:::
shutdown:*:15513:0:99999:7:::
halt:*:15513:0:99999:7:::
mail:*:15513:0:99999:7:::
uucp:*:15513:0:99999:7:::
[sysadmin@localhost ~]$
```

One big advantage to using `sudo` to execute administrative commands is that it reduces the risk that a user accidently executes a command as root. The intention to execute a command is clear; the command is executed as root if prefixed with the `sudo` command. Otherwise, the command is executed as a regular user.

13.7.1 Setting Up the sudo Command

If the `sudo` command is not automatically configured during installation, it can be configured to work manually post-install. Initially, this would require logging in as the root user (or using the `su` command to switch to the root account), but after it has been configured, specified users will be able to run `sudo` commands. You set up access to the `sudo` command by running the `visudo` command.

The `visudo` command will place you in an editor program, by default the `vi` (or `vim`) editor on most systems, which can be challenging for novice Linux users to use. To configure another editor, export the EDITOR variable with the value of the editor you would prefer. For example, to use the `gedit` editor instead of `vi/vim`, you would execute the `export EDITOR=gedit` command before you would execute the `visudo` command.

Using the `visudo` command will open the `/etc/sudoers` configuration file for editing. Although advanced configurations of the `sudo` command are possible through this file, they are beyond the scope of this course. Basically, entries are made into this file to specify which user(s) on which machines can use the `sudo` command to execute commands as other users.

This default entry...

```
root    ALL=(ALL)       ALL
```

...could be read as, "the root user can on ALL machines act as ALL users to execute ALL commands." To allow a user such as "sysadmin" to execute all commands as all users using the sudo command, an entry like the following could be added:

```
sysadmin ALL=(ALL) ALL
```

13.8 Using the who Command

The `who` command displays a list of users who are currently logged into the system, where they are logged in from and when they logged in. Through the use of options, this command is also able to display information such as the current *run level* (a functional state of the computer) and the time that the system was booted.

For example:

```
[sysadmin@localhost ~]$ who
root             tty2        2013-10-11 10:00
sysadmin tty1        2013-10-11 09:58 (:0)
sysadmin         pts/0       2013-10-11 09:59 (:0.0)
sysadmin         pts/1       2013-10-11 10:00 (example.com)
```

The following table describes the output of the `who` command:

Column	**Example**	**Description**
Username	root	This column indicates the name of the user who is logged in. Note that by "logged in" we mean "any login process and any open terminal window".
terminal	tty2	This column indicates which terminal window the user is working in. If the terminal name starts with "tty", then this is an indication of a *local* login, as this is a regular command line terminal. If the terminal name starts with "pts", then this indicates the user is using a pseudo terminal or running a process that acts like a terminal. This can mean the user has a terminal

Column	Example	Description
		application running in X Windows, such as `gnome-terminal` or `xterm` or they may have used a network protocol to connect to the system, such as `ssh` or `telnet`.
date	2013-10-11 10:00 (example.com)	This indicates when the user logged in. After the date and time the user logged into the system, some location information may appear. If the location information contains a host name, domain name or IP address, then the user has logged in remotely. If there is a colon and a number, then this indicates that they have performed a local graphical login. If no location information is shown in the last column, then this means the user logged in via a local command line process.

If you want to display system status information, the who command can do that by using several options. For example, the `-b` option will show the last time the system started (was booted) and the `-r` option will show the time the system reached a certain run-level:

```
[sysadmin@localhost ~]$ who -b -r
         system boot     2013-10-11 09:54
         run-level 5    2013-10-11 09:54
```

13.9 Using the w Command

The w command provides a more detailed list about the users currently on the system than the who command. It also provides a summary of the system status. For example:

```
[sysadmin@localhost ~]$ w
 10:44:03 up 50 min,  4 users,  load average: 0.78, 0.44, 0.19
USER             TTY     FROM         LOGIN@   IDLE        JCPU     PCPU    WHAT
root             tty2    -           10:00    43:44       0.01s    0.01s    -bash
sysadmin         tty1    :0          09:58    50:02       5.68s    0.16s    pam: gdm-p
assword
sysadmin pts/0   :0.0        09:59    0.00s      0.14s     0.13s   ssh 192.168.1.2
sysadmin         pts/1   example.com 10:00    0.00s        0.03s   0.01s   w
```

The first line of output from the w command is identical to that of the `uptime` command. It shows the current time, how long the system has been running, the total number of current logins (users) and the load on the system averaged over the last 1, 5 and 15 minute time periods. *Load average* is CPU utilization where a value of 100 would mean full CPU usage during that period of time.

The following table describes the rest of the output of the `w` command:

Column	**Example**	**Description**
USER	root	This column indicates the name of the user who is logged in.
TTY	tty2	This column indicates which terminal window the user is working in.
FROM	example.com	Where the user logged in from.
LOGIN@	10:00	When the user logged in.
IDLE	43:44	How long the user has been idle since the last command they ran.
JCPU	0.01s	The total cpu time (s=seconds) used by all processes (programs) run since login.
PCPU	0.01s	The total cpu time for the current process.
WHAT	-bash	The current process that the user is running.