Part Workbook 3. Users and Groups
# Table of Contents

1. Linux Users and the /etc/passwd File ................................................................. 4
   Discussion ............................................................................................................. 4
   Linux Users and the /etc/passwd file. ................................................................. 4
   User Passwords and the /etc/shadow file. ............................................................ 5
   Three types of users: normal, root, and system ................................................... 6
   Examples .............................................................................................................. 7
   Example 1. Examining process userids ................................................................. 7
   Example 2. Examining File Owners by username and userid ......................... 8
   Example 3. Changing a username, as root ......................................................... 8
   Online Exercises ................................................................................................. 9
   Online Exercise 1. Determining User Information ............................................. 9
      Specification ................................................................................................... 9
      Deliverables .................................................................................................. 9
   Questions .......................................................................................................... 10
2. Linux Groups and the /etc/group File ..................................................................... 12
   Discussion .......................................................................................................... 12
   Linux Groups ..................................................................................................... 12
   The /etc/group file. ........................................................................................... 12
   Why groups? ...................................................................................................... 13
   Primary and Secondary Groups ......................................................................... 13
   How do I change my group memberships? ....................................................... 14
   Examples ........................................................................................................... 14
   Example 1. Who are members of that group? ..................................................... 14
   Example 2. What groups does that user belong to? .......................................... 15
   Online Exercises ............................................................................................... 15
   Online Exercise 1. Determining group memberships ....................................... 15
      Specification ................................................................................................. 15
      Deliverables ................................................................................................. 16
   Online Exercise 2. Determining a user's subscribed groups (the hard way) .... 16
      Specification ................................................................................................. 16
      Deliverables ................................................................................................. 16
   Questions .......................................................................................................... 16
   Group memberships ......................................................................................... 16
3. Examining User Information .................................................................................. 20
   Discussion .......................................................................................................... 20
   Identifying users: the id command. ................................................................. 20
   The whoami command. .................................................................................... 21
   Who is currently logged on? The users, w, and who commands. .................. 21
   Checking up on users: the finger command. .................................................. 22
      Using the finger command. ........................................................................... 22
      Customizing finger command output. .......................................................... 23
      Using the finger command over the network. ............................................. 24
   Examples ........................................................................................................... 24
   Example 1. Using the id command to determine group memberships. ........ 24
   Example 2. Catching up with elvis. ................................................................. 24
   Online Exercises ............................................................................................... 25
   Online Exercise 1. Listing groups with the id command. ............................... 25
      Specification ................................................................................................. 25
      Deliverables ................................................................................................. 25
   Questions .......................................................................................................... 25
   Determining user information ........................................................................ 25
### 4. Changing Identity

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discussion</td>
<td>27</td>
</tr>
<tr>
<td>Switching identity</td>
<td>27</td>
</tr>
<tr>
<td>Becoming root</td>
<td>28</td>
</tr>
<tr>
<td>Switching primary group with the <code>newgrp</code> (or <code>sg</code>) command</td>
<td>28</td>
</tr>
<tr>
<td>Examples</td>
<td>29</td>
</tr>
<tr>
<td>Example 1. The <code>su</code> command and sessions</td>
<td>29</td>
</tr>
<tr>
<td>Online Exercises</td>
<td>29</td>
</tr>
<tr>
<td>Online Exercise 1. Using the <code>newgrp</code> command to change primary groups</td>
<td>29</td>
</tr>
<tr>
<td>Specification</td>
<td>29</td>
</tr>
<tr>
<td>Deliverables</td>
<td>30</td>
</tr>
<tr>
<td>Possible Solution</td>
<td>30</td>
</tr>
<tr>
<td>Questions</td>
<td>30</td>
</tr>
<tr>
<td>Switching userid and group</td>
<td>30</td>
</tr>
</tbody>
</table>
Chapter 1. Linux Users and the /etc/passwd File

Key Concepts

- At a low level, users are represented by an integer called a User Id (uid).
- Every process that runs on the system runs as a given uid.
- Every file in the filesystem is owned by a uid.
- The /etc/passwd file maps uids to user accounts.
- User accounts map uids to a username, password, Group Id(s), a home directory, and a login shell.
- Passwords are changed with passwd command.

Discussion

Linux Users and the /etc/passwd file.

When using a Linux system, you first identify yourself by logging on with a particular username. Your username represents you. Your username is associated with the things that you do: every process that runs on the system has an associated username. Your username is associated with the things that you save: every file on the system is labeled as owned by a particular username. Your username is associated with the things that you use: the amount of disk space that you use, or the amount of processor time that you use, can be tracked by username.

Not only does every user on the system have a unique username, but they normally have a unique userid, often abbreviated uid. Linux tracks userids as a 32bit integer, meaning that there can be up to $2^{32}$, or about 4 billion, distinct users. While people like to think in terms of words (usernames), the Linux kernel finds it simpler to think in terms of numbers (uids). When the kernel keeps track of who owns a process, or who owns a file, it remembers the uid instead of the username. Only when some command produces output for people to read does the uid get converted into a username.

The system maintains a database that maps usernames to userids. This database is stored in the /etc/passwd configuration file. Linux, like Unix, has a fortunate tradition: even the core configuration files on the system are maintained in human readable text, and editable by a text editor. Users, and administrators, can use simple tools for handling text, such as pagers, to examine the database. Most users on the system have permissions to read, but not modify, this file. The following shows a few lines from a typical /etc/passwd file.

```
$ tail /etc/passwd
apache:x:48:48:Apache:/var/www:/bin/bash
postfix:x:89:89::/var/spool/postfix/sbin/nologin
elvis:x:501:501::/home/elvis:/bin/bash
prince:x:502:502::/home/prince:/bin/bash
madonna:x:504:504::/home/madonna:/bin/bash
blondie:x:505:505::/home/blondie:/bin/bash
sleepy:x:507:507::/home/sleepy:/bin/bash
grumpy:x:509:509::/home/grumpy:/bin/bash
doc:x:510:510::/home/doc:/bin/bash
```
The /etc/passwd file is a line based configuration file, where each line defines a single user on the system. Lines are internally broken down into seven fields, with each field separated by a colon. The following table explains the use of each of these fields.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Num</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
<td>1</td>
<td>The username is used to give a human readable name to the user.</td>
</tr>
<tr>
<td>Password</td>
<td>2</td>
<td>On older Unix systems, this field contains the user's encrypted password. By default, Red Hat Enterprise Linux does not make use of this field, for security reasons.</td>
</tr>
<tr>
<td>User Id (uid)</td>
<td>3</td>
<td>The integer that the Linux kernel uses to identify the user.</td>
</tr>
<tr>
<td>Primary Groupid (gid)</td>
<td>4</td>
<td>The integer that the Linux kernel uses to identify the user's primary group. Group memberships are discussed in the next lesson.</td>
</tr>
<tr>
<td>GECOS</td>
<td>5</td>
<td>This oddly named field no longer serves its original purpose, which was relevant to Unix's original development environment. These days, the field is used to store simple text that helps identify the user, usually just a full name, but sometimes including a phone number or office address as well.</td>
</tr>
<tr>
<td>Home Directory</td>
<td>6</td>
<td>When a user logs in, his login shell will use this as its current working directory. It's one of the few directories that standard users can write in, and it's usually private to the user.</td>
</tr>
<tr>
<td>Login Shell</td>
<td>7</td>
<td>The login shell is the user's default shell when they login. In Red Hat Enterprise Linux, it is usually /bin/bash.</td>
</tr>
</tbody>
</table>

Users seldom, if ever, modify this file directly, although several commands will be introduced that allow users to change certain fields. If you ever need to refresh your memory, the fields are documented in the passwd(5) man page.

**User Passwords and the /etc/shadow file.**

As mentioned above, a user's encrypted password used to be stored in the second field of the /etc/passwd file. Because the /etc/passwd file contains much more information than just passwords, everyone needs to be able to read it. With modern computing power, however, even exposing the encrypted form of your password is dangerous. Without too much effort, modern machines can try to figure out your password by encrypting every combination of every letter until a match is found. This is known as a "brute force" attack.

Instead, modern Linux and Unix systems store passwords using a newer technique called "Shadow Passwords", where users' passwords are stored in a dedicated /etc/shadow file. Because the file contains only password related information, its permissions do not allow people to view its contents. Curious readers can refer to the shadow(5) man page for details.

Users can change their password with a simple command called passwd. If you are not the user root, the passwd command takes no arguments, and accepts no command line switches. Its single use is to allow a user to change her own password:

```
[madonna@station madonna]$ passwd
Changing password for user madonna.
Changing password for madonna
(current) UNIX password:
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```
Linux Users and the /etc/passwd File

Notice that users need to supply their current password before they can change it. This prevents somebody from taking advantage of a momentarily unattended terminal.

**Remember your password!**

If you change your Linux password, make sure that you remember it! Your password is never stored on the system in human readable plaintext, so even your system administrator can’t know your password. If you do forget your password, someone with root privileges can reset your password, and then tell you what it was reset to.

**Choosing a "strong" password**

When choosing a new password, users are often admonished with a message beginning BAD PASSWORD. Traditionally, passwords are susceptible to a type of attack known as a "dictionary" attack, whereby an attacker encrypts an entire dictionary (such as /usr/share/dict/words), and compares the encrypted output with the contents of the /etc/shadow file.

To help prevent successful dictionary attacks, the `passwd` command will force users to avoid passwords which are too simple or might be found in a dictionary.

**Three types of users: normal, root, and system**

Linux users can usually be grouped into three classes.

| Normal Users | Normal users represent real people who use the system. Normal users usually have /bin/bash as a login shell, and a home directory within the /home directory. Generally, normal users may create files only within their home directories and system wide temporary directories, such as /tmp and /var/tmp. In Red Hat Enterprise Linux, normal users usually have uids greater than 500. |
| The root User | The uid 0 is reserved for the user root, sometimes called the superuser. The root user has a free reign on the system: she may modify or remove any file; she may run any command; she may kill any process. The root user is in charge of adding and maintaining other users, configuring hardware, and adding system software. Although the root user may create files anywhere on the system, she usually uses /root as her home directory. |
| System Users | Most Linux systems reserve a range of low valued uids to act as system users. System users don't represent people, but components of the system. For example, the processes that handle email often run as the username mail. The processes that run the Apache web server run as the user apache. System users usually do not have a login shell, because they don't represent people who actually log in. Likewise, the home directories of system users seldom reside in /home, but are usually system directories that pertain to the relevant application. For example, the user apache has a home directory of /var/www. In Red Hat Enterprise Linux, system users have uids ranging from 1 - 499. |

**Table 1.1. Red Hat Enterprise Linux User Ids**

<table>
<thead>
<tr>
<th>uid range</th>
<th>Type of user</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>the user root</td>
</tr>
</tbody>
</table>
Examples

Examining process userids

The user elvis is curious what other people are currently using the Linux system he's on, and what they are up to. He lists all of the processes currently running on the machine.

```
[elvis@station elvis]$ ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY  STAT START   TIME COMMAND
root         1  0.0  0.0  1380   76 ?    S  03:33   0:04 init [   
root         2  0.0  0.0       0    0 ?    SW  03:33   0:00 [keventd]
root         3  0.0  0.0       0    0 ?    SW  03:33   0:00 [kapmd]
root        872  0.0  0.1  5932  440 ?    S  03:34   0:00 [sendmail]
smmsp      881  0.0  0.1  5732  312 ?    S  03:33   0:00 [sendmail]
root        891  0.0  0.0  2688  388 ?    S  03:37   0:00 gpm -t ps/2 -m /d
root        900  0.0  0.0  1572  128 ?    S  03:34   0:00 crond
xfs        973  0.0  0.0  4812  236 ?    S  03:34   0:00 [xfs]
root        991  0.0  0.0  1420   56 ?    S  03:34   0:00 gpm -t ps/2 -m /d
root        999  0.0  0.0  1356   31 ?    S  03:34   0:00 /sbin/mingetty tt1
...
```

Some of the lines in this rather long listing were edited away, and replaced with "...".

The first column of this listing shows the username that a process is running as. In addition to prince, madonna, and elvis, whom elvis assumes are usernames associated with actual people, elvis notes that many of the processes on the system are running as the user root, and also as the system users smmsp and xfs.

### Examining File Owners by username and userid

The user blondie is examining the /home directory, and noticing that each user’s home directory is owned by appropriate username. She then uses the `ls -ln` command, to list the directory owners “numerically”, or by userid instead of by username. Pay close attention to the 3rd column in the following listing, which shows a file’s owner.

```
[blondie@station blondie]$ ls -l /home/
total 48
drwx------ 4 blondie blondie 4096 May 14 06:35 blondie
drwx------ 4 doc doc 4096 May 14 06:32 doc
drwx------ 4 elvis elvis 4096 May 14 06:31 elvis
drwx------ 4 grumpy grumpy 4096 May 14 06:32 grumpy
drw-x------ 4 madonna madonna 4096 May 14 06:31 madonna
drw-x------ 4 prince prince 4096 May 14 06:31 prince
drw-x------ 4 sleepy sleepy 4096 May 14 06:32 sleepy
```

In the `ls -l` listing, the file's owners are shown by username. In the `ls -ln` listing, the file's owners were shown by userid.

### Changing a username, as root.

The machine’s administrator, acting as root, wants to edit the /etc/passwd file. First, root will take an `ls -l` of the files in the /home directory. Then, root will change sleepy’s username in the user database, and lastly look at the output of the `ls -l` command again.

```
[root@station root]# ls -l /home/
total 48
drwx------ 4 blondie blondie 4096 May 14 06:40 blondie
drwx------ 4 doc doc 4096 May 14 06:32 doc
drwx------ 4 elvis elvis 4096 May 14 06:31 elvis
drwx------ 4 grumpy grumpy 4096 May 14 06:32 grumpy
drwx------ 4 madonna madonna 4096 May 14 06:31 madonna
drwx------ 4 prince prince 4096 May 14 06:31 prince
drw-x------ 4 sleepy sleepy 4096 May 14 06:32 sleepy
```

(root edits the /etc/passwd file, so that the line ...)

```
sleepy:x:507:507::/home/sleepy:/bin/bash
```

... now reads ...

```
sleeper:x:507:507::/home/sleepy:/bin/bash
```

... )

```
[root@station root]# ls -l /home/
```
Linux Users and the /etc/passwd File

In the first case, the user owner of sleepy's home directory, /home/sleepy, is listed as sleepy.
In the second case, the user owner of sleepy's home directory is now listed as sleepier.

What can be learned from this example? root did not change anything about the /home/sleepy directory, only the user database. However, as soon as the modified /etc/passwd file was saved, the ls -l command started reporting the new information. This suggests the following:

1. The Linux kernel does not store the username of the user that owns a file, but instead the integer userid. (Can you tell from the previous example which userid owns the /home/sleepy directory?)

2. Every time the ls command runs, it must look up username/userid mappings in the /etc/passwd file to attach usernames to the userid owners supplied by the kernel.

Online Exercises

Determining User Information

Lab Exercise

Objective: Determine user related parameters, such as userid and login shell.

Estimated Time: 10 mins.

Specification

By examining the first, third and last (seventh) fields of the /etc/passwd file, determine the userid and login shell for your username, the user root, and the user nobody (yes, there is a system user with the username "nobody"). Create simple files in your home directory, called my.uid, my.shell, root.uid, root.shell, nobody.uid, and nobody.shell, which contain only the appropriate information on a single line.

For example, if the user nobody's login shell were /bin/bash, the following command would easily create the appropriate file.

```
[student@station student]$ echo /bin/bash > nobody.shell
[student@station student]$ cat nobody.shell
/bin/bash
```

Deliverables

1. When completed, the following six files should be in your home directory, containing only the following information:

<table>
<thead>
<tr>
<th>filename</th>
<th>content</th>
</tr>
</thead>
<tbody>
<tr>
<td>my.uid</td>
<td>Your account's integer userid</td>
</tr>
</tbody>
</table>
Linux Users and the /etc/passwd File

<table>
<thead>
<tr>
<th>filename</th>
<th>content</th>
</tr>
</thead>
<tbody>
<tr>
<td>my.shell</td>
<td>Your login shell</td>
</tr>
<tr>
<td>root.uid</td>
<td>The user root's integer userid</td>
</tr>
<tr>
<td>root.shell</td>
<td>The user root's login shell</td>
</tr>
<tr>
<td>nobody.uid</td>
<td>The user nobody's integer userid</td>
</tr>
<tr>
<td>nobody.shell</td>
<td>The user nobody's login shell</td>
</tr>
</tbody>
</table>

**Questions**

1. Which file contains the user information database?
   a. /etc/users.dat  
   b. /etc/passwd.dat 
   c. /etc/users    
   d. /etc/passwd

2. Which of the following information is not associated with a uid in the user information database?
   a. username  
   b. login terminal  
   c. login shell  
   d. home directory

3. Which of the following is true?
   a. Standard users may view /etc/passwd file.  
   b. Standard users may edit /etc/passwd file.  
   c. Both A and B.  
   d. None of the above.

4. What command would the user elvis run to change his password?
   a. `password -u elvis`  
   b. `passwd -u elvis`  
   c. `password`  
   d. `passwd`

5. Which of the following best describes why the /etc/passwd file is inappropriately named?
   a. That's not how you spell password.  
   b. The file would more appropriately be stored in a user's home directory.
c. On most modern Linux and Unix systems, the file does not contain passwords.
d. The filename is too difficult to remember.

6. In modern Linux and Unix systems, which file contains users locally stored passwords?
   a. /etc/passwd
   b. /etc/password
   c. /etc/shadow
   d. /etc/shdw

7. Which of the following is true?
   a. Standard users may view /etc/shadow file.
   b. Standard users may edit /etc/shadow file.
   c. Both A and B.
   d. None of the above.

8. Which best describes why the /etc/passwd must be readable by all users?
   a. So users can easily lookup forgotten passwords.
   b. Because the file contains passwords, it should not be world readable.
   c. So that processes can associate usernames with kernel provided userid owners of files, processes, and other resources.
   d. So that users may change their login shell, if they choose to do so.
Chapter 2. Linux Groups and the /etc/group File

Key Concepts

- Fundamentally, a group is just an integer group id (gid).
- Every process that runs on the system runs under a collection of groups (gids).
- The /etc/group file maps gids to group names and group memberships.
- Every file in the filesystem is owned by a single gid.
- Users have a single primary group defined in the /etc/passwd file.
- Users may be members of multiple secondary groups, defined in the /etc/groups file.

Discussion

Linux Groups

The previous lesson introduced the fact that every process runs under the context of a given user. In addition, users, and the processes that they run, and the files that they own, belong to a collection of groups. Group memberships allow system administrators to efficiently manage collections of users with similar objectives.

Every user is a member of one primary group. Additionally, users can be a member of zero or more secondary groups. The implications of primary and secondary groups are discussed below.

The /etc/group file.

To the Linux kernel, a group is referenced using a 32 bit integer group id (GID). (Note the similarity to how the kernel references users.) The /etc/group file associates group names with GIDs (for humans), and defines which users belong to which groups. The /etc/group file plays a similar role for groups as the /etc/passwd file plays for users, has a similar structure, and a more reasonable name. It is a line based configuration file, with each line consisting of colon separated fields, as demonstrated in the following extracted lines:

```
wrestle:x:201:ventura,hogan,elvis
physics:x:202:einstein,maxwell,elvis
emperors:x:203:nero,julius,elvis
governor:x:204:ventura,pataki
music:x:205:elvis,blondie,prince,madonna
dwarfs:x:206:sleepy,grumpy,doc
elvis:x:501:
prince:x:502:
madonna:x:504:
blondie:x:505:
```

The four fields supply the following information:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Num</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Groupname</td>
<td>1</td>
<td>The group name is used to give a human readable name to the group.</td>
</tr>
</tbody>
</table>
### Linux Groups and the /etc/group File

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Num</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group Password</td>
<td>2</td>
<td>Groups can be supplied with a group password, though this is rarely done.</td>
</tr>
<tr>
<td>Group ID (GID)</td>
<td>3</td>
<td>The integer group id.</td>
</tr>
<tr>
<td>Group members</td>
<td>4</td>
<td>A comma separated list of usernames defining the group members.</td>
</tr>
</tbody>
</table>

#### Why groups?

Group memberships are most often used to determine who has access to what in the filesystem. The dwarfs found in the above example of the /etc/group file, doc, grumpy, and sleepy, might all be working on mine prospects, and they don't want other users to have access to their information. The files they are using (and the directories they are storing them in) can be arranged so that any member of the group dwarfs has access to them, but other users do not.

The fact that every file on the system is owned by a user has already been mentioned. Every file on the system is also owned by a group, which is referred to as the “group owner” of the file. In the following ls -l listing of files in the /var/prospects directory, the third column lists the user owner of the files, while the fourth column lists the group owner.

```
[doc@station prospects]$ ls -l /var/prospects/
total 12
-rw-rw---- 1 doc dwarfs 143 May 15 07:38 doc.txt
-rw-rw---- 1 grumpy dwarfs 29 May 15 07:38 grumpy.txt
-rw-rw---- 1 sleepy dwarfs 2027 May 15 07:38 sleepy.txt
```

While the individual users own their own files, each file has the group dwarfs as its group owner, or in other words, is “group owned” by the group dwarfs. The first column specifies the permissions that the user owner, and members of the group that owns the file, have: can they read it? can they modify it? How to interpret and manage these permissions is the topic of the next workbook. For now, just realize: every file in the filesystem has a user owner and a group owner.

#### Primary and Secondary Groups

Because every file must have a group owner, there must be a default group associated with each user. This default group becomes the group owner of newly created files. This group is known as a user's primary group. A user's primary group is defined in the fourth field of a user's /etc/passwd entry.

In addition to a primary group, users may optionally belong to other groups as well. These groups are termed secondary groups, and are defined (coincidentally) in the fourth field of the /etc/group file.

Consider the following excerpts from a /etc/passwd file, and a /etc/group file.

```
[elvis@station elvis]$ tail /etc/passwd
grumpy:x:509:509::/home/grumpy:/bin/bash
doc:x:510:510::/home/doc:/bin/bash
student:x:2299:2299::/home/student:/bin/bash
ventura:x:511:511::/home/ventura:/bin/bash
hogan:x:512:512::/home/hogan:/bin/bash
pataki:x:513:513::/home/pataki:/bin/bash
einstein:x:514:514::/home/einstein:/bin/bash
maxwell:x:515:515::/home/maxwell:/bin/bash
nero:x:516:516::/home/nero:/bin/bash
julius:x:517:517::/home/julius:/bin/bash
[elvis@station elvis]$ cat /etc/group
...
```

```
wrestle:x:201:ventura,hogan,elvis
```
In the third field, we find that the user hogan has a userid of 512.

In the fourth field, we find that the user hogan has a primary group id of 512. (Note hogan's primary group id is the same as his user id. This does not have to be the case, but because of the way users are added in Red Hat Enterprise Linux, often is.)

Although we know that hogan's primary group is 512 from the /etc/passwd file, we cannot know the name of the group until we examine the /etc/group file. Here we find that the group name of group 512 is hogan. In Red Hat Enterprise Linux, a user's primary group almost always has the same name as the username.

Here we find that hogan is also a member of the group wrestlers. The group wrestlers is a secondary group.

In summary, the following can be said about primary and secondary groups:

- A user's primary groupid is defined in the fourth field of the /etc/passwd file. The user's primary group name is then associated with the groupid in the /etc/group file.

- A user's secondary groups are defined by adding the username to the fourth field of the appropriate lines in the /etc/group file.

How do I change my group memberships?

Standard users do not have permission to edit the /etc/passwd file or the /etc/group file, and therefore cannot change their group memberships. Instead, only the administrative user, root, may change group memberships.

If you do have access to a machine as root, group memberships may be modified using the usermod and groupmod command line utilities, or the the system-config-users graphical utility. The use of these commands are covered in another course.

Examples

Who are members of that group?

In the following long listing of the /var/spool directory, prince notices that the group "sys" owns the /var/spool/cups directory.

[prince@station prince]$ ls -l /var/spool/
total 64
...
Linux Groups and the /etc/group File

The user prince is now wondering which users on the system are members of the group sys. In order to find out, he examines the /etc/group file.

```
[prince@station prince]$ head /etc/group
root:x:0:root
bin:x:1:root,bin,daemon
daemon:x:2:root,bin,daemon
sys:x:3:root,bin,adm
adm:x:4:root,adm,daemon
tty:x:5:
disk:x:6:root
lp:x:7:daemon,lp
mem:x:8:
kmem:x:9:
```

By examining the fourth line, prince determines that the users root, bin, and adm are the current members of the group sys.

**What groups does that user belong to?**

The user prince wants to find out which groups the user root belongs to. Because of the format of the /etc/group file, he cannot determine the answer by examining a single line. He decides to search through the file for the word "root".

```
[prince@station prince]$ cat /etc/group
root:x:0:root
bin:x:1:root,bin,daemon
daemon:x:2:root,bin,daemon
sys:x:3:root,bin,adm
adm:x:4:root,adm,daemon
tty:x:5:
disk:x:6:root
lp:x:7:daemon,lp
mem:x:8:
kmem:x:9:
wheel:x:10:root
...
```

After some tedious inspection of the output, prince decides that root belongs to the following groups: root, bin, daemon, sys, adm, disk, and wheel. prince thinks to himself, "there must be an easier way...". (Hopefully, prince will read the next lesson.)

**Online Exercises**

**Determining group memberships**

**Lab Exercise**

**Objective:** Determine group memberships of locally defined groups.

**Estimated Time:** 10 mins.

**Specification**

Create a file in your home directory, called physics.txt, which contains a list of the members of the group physics, separated by spaces.

For example, if elvis, blondie, and prince were the members of the group physics, the following command would easily create such a file.
[student@station student]$ echo "blondie elvis prince" > physics.txt
[student@station student]$ cat physics.txt
blondie elvis prince

Deliverables

1. A ~/physics.txt file that contains a list of the members of the group physics, separated by spaces.

Determining a user's subscribed groups (the hard way)

Lab Exercise

Objective: Determine the groups that a locally defined user belongs to.

Estimated Time: 10 mins.

Specification

Create a file in your home directory, called elvisgrp.txt, which contains a list of the groups that the user elvis belongs to, separated by spaces.

For example, if elvis belonged to the groups sys, music, and elvis, then the following command would easily create such a file.

[student@station student]$ echo "elvis music sys" > elvisgrp.txt
[student@station student]$ cat elvisgrp.txt
elvis music sys

Deliverables

1. The file ~/elvisgrp.txt that contains a list of the groups that the user elvis belongs to, separated by spaces.

Questions

Group memberships

Use the output of the following commands to answer the questions below.

[student@station student]$ tail /etc/passwd
sleepy:x:507:507::/home/sleepy:/bin/bash
grumpy:x:509:509::/home/grumpy:/bin/bash
doc:x:510:510::/home/doc:/bin/bash
ventura:x:511:511::/home/ventura:/bin/bash
hogan:x:512:512::/home/hogan:/bin/bash
pataki:x:513:513::/home/pataki:/bin/bash
einstein:x:514:514::/home/einstein:/bin/bash
maxwell:x:515:515::/home/maxwell:/bin/bash
nero:x:516:516::/home/nero:/bin/bash
julius:x:517:517::/home/julius:/bin/bash
[student@station student]$ cat /etc/group
...
wrestle:x:201:ventura,hogan,elvis
physics:x:202:einstein,maxwell,elvis
emperors:x:203:nero,julius,elvis
Linux Groups and the \texttt{/etc/group} File

\begin{verbatim}
governor:x:204:ventura,pataki
music:x:205:elvis,blondie,prince,madonna
dwarfs:x:206:sleepy,grumpy,doc
...
hogan:x:512:
pataki:x:513:
einstein:x:514:
maxwell:x:515:
nero:x:516:
 julius:x:517:
\end{verbatim}

\textit{Note that some lines have been deleted from the output of the \texttt{cat} command, and replaced with \texttt{"..."}.}

1. Which of the following is the group id for the group dwarfs?
   \begin{itemize}
   \item[a.] 201
   \item[b.] 215
   \item[c.] 510
   \item[d.] 206
   \end{itemize}

2. What is the group id of the user maxwell's primary group?
   \begin{itemize}
   \item[a.] 201
   \item[b.] 215
   \item[c.] 515
   \item[d.] 517
   \end{itemize}

3. Which of the following groups is elvis not a member of?
   \begin{itemize}
   \item[a.] physics
   \item[b.] wrestle
   \item[c.] music
   \item[d.] dwarfs
   \end{itemize}

4. What is the name of the user ventura's primary group?
   \begin{itemize}
   \item[a.] wrestle
   \item[b.] governor
   \item[c.] ventura
   \item[d.] The name of the user ventura's primary group cannot be determined from the information provided.
   \end{itemize}

5. Which of the following best explains why the group einstein has no members listed in the \texttt{/etc/group} file?
   \begin{itemize}
   \item[a.] einstein is a username only, and not a group name.
   \item[b.] The group einstein is being used as a primary group, with no secondary group members.
   \end{itemize}
c. The group einstein has no members.
d. The file is misconfigured.

6. Select all of the groups that contain the user elvis as a member.
   a. wrestle
   b. physics
   c. emperors
   d. governor
   e. music
   f. einstein

7. Select all of the groups that contain the user ventura as a member.
   a. wrestle
   b. physics
   c. emperors
   d. governor
   e. music
   f. einstein

8. Which file defines secondary group memberships?
   a. /etc/group
   b. /etc/secondary
   c. /etc/passwd
   d. /etc/shadow

9. Which file(s) must be consulted to determine the group name of a user's primary group?
   a. /etc/passwd
   b. /etc/shadow
   c. /etc/group
   d. Both A and C.

10. Who of the following is not a member of the group wrestle?
    a. nero
    b. ventura
    c. hogan
d. elvis
Chapter 3. Examining User Information

Key Concepts

• The id command displays user and group information.

• The whoami reports the current username

• The who, users, and w commands report users with active sessions.

Discussion

Identifying users: the id command.

The previous lesson introduced the concepts of groups, and how the /etc/passwd and /etc/group files define group memberships. Because determining group memberships from these files is not straightforward, users often use the id command to determine a user's membership information.

`id [(-g) | (-G) | (-u)] [-n] [USERNAME]`

Print information for USERNAME, or the current user.

<table>
<thead>
<tr>
<th>Switch</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>-g, --group</td>
<td>Print only effective group id</td>
</tr>
<tr>
<td>-G, --groups</td>
<td>Print all group ids</td>
</tr>
<tr>
<td>-u, --user</td>
<td>Print only effective user id</td>
</tr>
<tr>
<td>-n, --name</td>
<td>Print user or group name instead of number.</td>
</tr>
</tbody>
</table>

If called without arguments, the id command returns a summary of group memberships for the user who ran it. If supplied with a username, the id command reports group information for the specified user.

```
[elvis@station elvis]$ id
uid=501(elvis) gid=501(elvis) groups=501(elvis),203(emperors),205(music)
[elvis@station elvis]$ id blondie
uid=505(blondie) gid=505(blondie) groups=505(blondie),205(music)
```

By default, the command reports the user's user id, primary group and primary group id, and all groups (primary and secondary) to which the user belongs, both by group name and group id.

The output of the id command can be qualified with command line switches, as summarized in the table above. For example, if a user only wanted to know the primary group id of a user, the command could be invoked with the -g switch. If the user wanted to know the group name instead of number, the -n switch could also be specified. Likewise, the -G command line switch will report all groups, not just the primary group. These command line switches are very helpful when the id command is used in scripts.

```
[elvis@station elvis]$ id -g blondie
505
[elvis@station elvis]$ id -gn blondie
blondie
[elvis@station elvis]$ id -Gn blondie
blondie music
```
SE Linux Context

The id command can report one more piece of information, starting with context=, known as the SE Linux context. SE Linux, or "Security Enhanced Linux", involves topics which will be covered in more detail in a later course, and for now can be safely ignored.

The whoami command.

The whoami command simply reports the current user’s username.

whoami

Print the current user’s username.

In Red Hat Enterprise Linux, there is seldom a need to use the whoami command, because the default bash prompt displays the current user’s username. The prompt is configurable, however, and on other Linux or Unix systems, or in other environments (such as rescue shells), a user’s identity might not be as obvious. Also, the whoami command can be embedded in a script, which could take on different behavior depending on who’s running it.

Who is currently logged on? The users, w, and who commands.

Often, a user would like to know who else is using a particular Linux or Unix machine. Red Hat Enterprise Linux provides three commands which report current users, each at a different level of detail. The simplest of these is the users command.

users

Print a simple list of who is currently logged in.

The user elvis might be curious who else is logged into the machine.

[elvis@station elvis]$ users
blondie elvis elvis elvis elvis elvis prince root

The command reports that blondie, elvis, prince, and root are currently logged into the machine. Why is elvis listed five times? Unix traditionally associates all activity from a single terminal with what is technically called a "session". When using the X graphical environment, each terminal window that is opened is considered a distinct terminal, and thus a distinct session. The user elvis is probably using multiple terminals within X windows.

w [-h] [-s] [-f] [USERNAME]

Print detailed information about who is currently logged in, or for the user USERNAME, if provided.

<table>
<thead>
<tr>
<th>Switch</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>-h</td>
<td>skip header</td>
</tr>
<tr>
<td>-s</td>
<td>short listing</td>
</tr>
<tr>
<td>-f</td>
<td>don't display &quot;FROM&quot; field.</td>
</tr>
</tbody>
</table>

Like the users command, the w command lists currently logged on users, but provides much more detailed information, as demonstrated by the user elvis:

[elvis@station elvis]$ w
Examining User Information

The top line gives summary information: the current time, the uptime (how long since the machine was booted in days, hours, and minutes), the number of users, and the current load average (as a 1, 5, and 15 minute average) for the machine. The command then displays a table of the following information.

<table>
<thead>
<tr>
<th>Column</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER</td>
<td>The username of the logged on user.</td>
</tr>
<tr>
<td>TTY</td>
<td>The terminal that the user logged in from. (Interpreting terminals is discussed in Workbook 1 (Quick Tour)).</td>
</tr>
<tr>
<td>FROM</td>
<td>If the user is logged in over the network, the hostname that the user connected from.</td>
</tr>
<tr>
<td>LOGIN@</td>
<td>The time that the user started the session.</td>
</tr>
<tr>
<td>IDLE</td>
<td>How much time has expired since the terminal has seen interaction.</td>
</tr>
<tr>
<td>JCPU</td>
<td>The amount of CPU time consumed by all currently running processes associated with the terminal session.</td>
</tr>
<tr>
<td>PCPU</td>
<td>The amount of CPU time consumed by only the current process only, as named in the &quot;WHAT&quot; field.</td>
</tr>
<tr>
<td>WHAT</td>
<td>The user's currently running process.</td>
</tr>
</tbody>
</table>

The last of these three similar commands, who, also provides detailed information about who is currently logged on. Called without arguments, it behaves similarly to the w command. The who command is also designed to report system events, such as the current runlevel, system clock adjustments, and processes automatically spawned by the /sbin/init process. Much of this information is beyond the scope of this course.

Checking up on users: the finger command.

Using the finger command.

The finger command can be used to "check up" on a user, to see not only if that user is logged on, but if they have read their mail, what time they last logged in, and other information as well.

finger [-s] [-l] [-p] [USER ...] [USER@HOST ...]

Print summary information about the specified user, including if they are logged in and if they have checked their mail, or a list of all currently logged in users if none are provided.

<table>
<thead>
<tr>
<th>Switch</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>-s</td>
<td>Output in summary format</td>
</tr>
<tr>
<td>-l</td>
<td>Output in long format</td>
</tr>
<tr>
<td>-p</td>
<td>Don't display &quot;.plan&quot;, &quot;.project&quot;, etc., information</td>
</tr>
</tbody>
</table>

As an example, the user elvis uses the finger command to see who's logged onto the machine.
Examining User Information

[elvis@station elvis]$ finger
Login     Name       Tty      Idle  Login Time   Office     Office Phone
madonna             *:0             May 26 06:40
madonna              pts/0      28  May 26 06:40 (:0.0)
madonna              pts/1      20  May 26 06:41 (:0.0)
madonna              pts/2          May 26 06:50 (:0.0)
madonna              pts/3       3  May 26 07:04 (:0.0)
elvis                tty1           May 26 07:09
prince               tty2           May 26 07:10

Without arguments, the **finger** command acts much like the **who** command, giving summary information including a user's terminal (Tty), login time, and idle time.

elvis now realizes that his friend blondie is not logged on, and decides to find out more information about her.

[elvis@station elvis]$ finger blondie
Login: blondie           Name: (null)
Directory: /home/blondie             Shell: /bin/bash
Never logged in.
No mail.
No Plan.

In this case, the **finger** command returns summary information about the user blondie, including the fact that she has never logged in, and she has no mail waiting.

**Customizing finger command output.**

When elvis decides to check up on the user prince, the finger command gives much more information.

[elvis@station elvis]$ finger prince
Login: prince            Name: (null)
Directory: /home/prince              Shell: /bin/bash
On since Mon May 26 07:10 (EDT) on tty2  10 minutes 30 seconds idle
New mail received Mon May 26 07:17 2003 (EDT)
Unread since Wed May 14 06:31 2003 (EDT)
Plan:
12:00 - 1:00 lunch
1:00 - 3:00 in class
3:30 - 4:30 ultimate frisbee

1 Because prince is currently logged on, the finger command returns the terminal, the login time, and the idle time for the user.

2 The finger command reports the last time that the user received mail, and the last time that the user read mail. (Note, this only works if the machine is configured to receive mail directly, rather than using, for example, a POP account provided by an Internet Service Provider.)

3 The user prince has created a `~/.plan` file, which is reported by the **finger** command.

A user can customize the information someone sees when they are **fingered**, by creating specially named hidden files in their home directory. The following files are known by the **finger** command.

<table>
<thead>
<tr>
<th>Filename</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>.plan</td>
<td>Displayed by the finger command, under the heading &quot;Plan:“.</td>
</tr>
<tr>
<td>.project</td>
<td>Displayed by the finger command, under the heading &quot;Project:“.</td>
</tr>
<tr>
<td>.pgpkey</td>
<td>Displayed by the finger command, under the heading &quot;PGP Key:“. This is a convenient way for a user to publish the public key portion of a public/private key pair used in public key encryption.</td>
</tr>
</tbody>
</table>
Allowing access to your home directory

By default in Red Hat Enterprise Linux, a user's home directory is completely inaccessible to other users on the system. In order for the `finger` command to find these files, a user must enable execute permissions on their home directory, using a command similar to the following:

```
[prince@station prince]$ chmod o+x ~
```

This topic will be covered thoroughly in Workbook 4.

Using the finger command over the network.

The finger command can report user activity on remote machines, as well. The user elvis wants to see what the user pataki on the machine nimbus.example.com is up to.

```
[elvis@station elvis]$ finger pataki@nimbus.example.com
Login: pataki            Name: (null)
Directory: /home/pataki              Shell: /bin/bash
Never logged in.
New mail received Mon May 26 07:41 2003 (EDT)
    Unread since Wed May 14 15:34 2003 (EDT)
Plan:
Balance the budget.
```

For this command to work, the remote machine nimbus.example.com must elect to run the finger service. In the early days of the Internet, Unix machines commonly ran this service. Now that the Internet has grown, and the demands of security often outweigh the spirit of community, most administrators choose not to run the finger service. The service is available in Red Hat Enterprise Linux, but disabled by default.

Examples

Using the id command to determine group memberships.

The user blondie had sent her administrator email, asking to add her to the group music. Upon logging in the next day, she wants to know if the administrator did it. She runs the `id` command to find out.

```
[blondie@station blondie]$ id
uid=505(blondie) gid=505(blondie) groups=505(blondie),205(music)
```

blondie discovers that she has been added to the group music.

Catching up with elvis.

The user blondie wants to catch up with the user elvis. She first determines if elvis is logged onto the machine, using the `users` command.

```
[blondie@station blondie]$ users
blondie blondie blondie blondie blondie elvis prince
```

The user elvis is logged in but no one has seen him in the office. The user blondie decides to get more information using the `w` command.

```
[blondie@station blondie]$ w
  08:05:50  up  1:28,  7 users,  load average: 0.00, 0.15, 0.34
 USER      TTY           FROM      LOGIN@   IDLE  JCPU  PCPU WHAT
prince    tty2          -         9:10am 55:48 0.05s 0.05s   -bash
```
Examining User Information

Looking at the output, blondie notices that elvis is working remotely today.

Online Exercises

Listing groups with the id command.

Lab Exercise

Objective: List group memberships with the id command.

Estimated Time: 5 mins.

Specification

Redirect the output of the id command to create a file in your home directory called mygroups.txt. Use the appropriate command line switches so that your subscribed groups (primary and secondary) should be listed on a single line, separated by a single space.

For example, if the exercises has been completed correctly, the file should have a similar format to the following.

```
[student@station student]$ cat mygroups.txt
student wrestle physics
```

Deliverables

1. A ~/mygroups.txt file that contains your subscribed groups, separated by a space on a single line.

Questions

Determining user information

1. Which command is used to determine the group memberships of a user?
   a. id
   b. who
   c. w
   d. finger

2. Which of the following commands can be used to find out who is currently logged on?
   a. who
b. users

c. groups

d. w

3. Which of the following statements is true?
   a. The `w` command shows who is logged into a remote machine.
   b. The `w` command can show information about users not currently logged on.
   c. The `users` command shows who is logged into a remote machine.
   d. The `id` command can show information about users not currently logged on.

4. Which invocation of the `id` command lists all group memberships by name?
   a. `id -g`
   b. `id -gn`
   c. `id -G`
   d. `id -Gn`

5. Which of the following information can not be determined by the `id` command?
   a. username
   b. user id
   c. login shell
   d. primary group id
Chapter 4. Changing Identity

Key Concepts

• The **su** command allows a user to switch user id.

• The **su** command is usually called with a hyphen, such as **su -**, to specify that the user should effectively log in as the new user.

• The **newgrp** command, also called **sg**, allows a user to switch primary group id.

Discussion

Switching identity

Often in Linux, a user would like to temporarily become another user. You might need to change the permissions on a file that you don't own, or a friend might want to walk up and borrow your terminal temporarily. The command that allows you to temporarily switch user id is called simple **su**.

```
su [[-] | [-c] | [-m, -p] | [-s, --shell=SHELL]] [USER [ARG]]
```

Switch userid to specified USER, or to root if now USER is specified.

<table>
<thead>
<tr>
<th>Switch</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>make shell a login shell</td>
</tr>
<tr>
<td>-c</td>
<td>run the specified command, and return</td>
</tr>
<tr>
<td>-m, -p</td>
<td>preserve the current environment</td>
</tr>
<tr>
<td>-s, --shell=SHELL</td>
<td>use SHELL as the new user's shell</td>
</tr>
</tbody>
</table>

Temporarily becoming another user can be as simple as "su"ing to that user:

```
[elvis@station elvis]$ su prince
Password: 
[prince@station prince]$
```

Note the obvious security concern. User ownerships and permissions would be meaningless if users could **su** at will. In order to become the user prince, elvis needed to know prince's password.

When using the **su** command as shown, the new shell becomes prince's shell, and any processes started from the shell would be owned by prince. However, the shell is called a **non-login** shell. prince didn't fully log in - the prompt indicates he's still sitting in elvis's home directory. Although the shell, and all that it starts, belongs to prince, much of the shell's environment was inherited from elvis.

In order to completely log in as a new user, the **su** command can be called with a **-** (a bare hyphen). This specifies that the new user's shell should be a **login** shell.

```
[elvis@station elvis]$ su - prince
Password: 
[prince@station prince]$
```

In this case, prince has effectively logged in, so that the new shell's current working directory is prince's home directory, and prince's environment has been initialized from his own configuration.
The difference between a login shell and a non-login shell, and the use of environment variables, is discussed in detail in a later workbook. For now, just appreciate that in order to fully become a new user, the su command should be passed a - as well as a user name. Until you know enough to understand the difference, always su - (with the hyphen).

Becoming root

Often, a user must be root in order to configure (or fix a misconfigured) Linux machine. If a user knows the password to the root account, this is often done by "su-ing" to root. If called without an argument, the su command assumes that the user is trying to become root.

[elvis@station elvis]$ su -
Password:
[root@station root]#

Because the su command was called with a -, the new root shell starts out in root's home directory. Also note the subtle change in the bash prompt. Whenever the shell is being run as root, the prompt uses the "#" character instead of the "$".

The "superuser"

In Linux and Unix, the root account is sometimes referred to as the "superuser". This term probably came about because of a misunderstanding of the meaning of the letters of the su command which are so often used to become root. Rather than superuser, the letters stand for switch userid.

Switching primary group with the newgrp (or sg) command.

The previous lessons discussed the role of the primary group. To review, every user has one primary group, as defined in the /etc/passwd file, and newly created files are group owned by a user's primary group. Sometimes, users might want to make one of their secondary groups the primary group, so that newly created files are group owned by that group. The newgrp command, equivalently called the sg command, does just this.

newgrp [[GROUP]]

Spawn a new shell, with GROUP as its primary group. If GROUP is not specified, use the default primary group.

In the following sequence of commands, prince will create a new file, then use the newgrp command to spawn a new shell with a primary group of music. In the new shell, prince will create another file, which will be group owned by music. prince then exits the new shell.

[prince@station prince]$ id
uid=502(prince) gid=502(prince) groups=502(prince),205(music)
[prince@station prince]$ date > foo
[prince@station prince]$ ls -l
total 4
-rw-rw-r-- 1 prince prince 29 May 26 12:26 foo
[prince@station prince]$ newgrp music
[prince@station prince]$ id
uid=502(prince) gid=205(music) groups=502(prince),205(music)
[prince@station prince]$ date > biz
[prince@station prince]$ ls -l
total 8
-rw-r--r-- 1 prince music 29 May 26 12:26 biz

28
Note that, after running the `newgrp` command, the `id` command is reporting prince's primary group as "music".

The file `biz`, which was created when prince's primary group was music, is group owned by music.

Examples

The su command and sessions.

From the second virtual console (tty2), the user elvis uses the `su` command to become the user prince, noting who is logged into the machine both before and after `su`ing.

```
[elvis@station elvis]$ w
12:44:38  up 3:44, 1 user, load average: 0.02, 0.07, 0.07
USER   TTY      FROM              LOGIN@   IDLE   JCPU   PCPU  WHAT
elvis  tty2     -                12:44pm  0.00s  0.08s  0.02s  w
[elvis@station elvis]$ su - prince
Password:
[prince@station prince]$ w
12:44:56  up 3:44, 1 user, load average: 0.02, 0.06, 0.07
USER   TTY      FROM              LOGIN@   IDLE   JCPU   PCPU  WHAT
elvis  tty2     -                12:44pm  0.00s  0.13s  0.14s  login -- elvis
```

To the user's surprise, there is no evidence of prince being logged onto the system. A listing of all processes associated with the second virtual console helps illustrate why.

```
[prince@station prince]$ ps aux | grep tty2
elvis  6115  0.0  0.5  4316 1392 tty2     S    12:44   0:00 -bash
root   6156  0.0  0.3  4112  960 tty2     S    12:44   0:00 su - prince
prince 6157  0.0  0.5  4316 1404 tty2     S    12:44   0:00 -bash
prince 6207  0.0  0.2  2648  696 tty2     R    12:49   0:00 ps aux
prince 6208  0.0  0.2  3576  644 tty2     S    12:49   0:00 grep tty2
```

When elvis ran the `su` command, the `su` command ran as root (this is known as a `suid` executable, which will be covered in more detail later). The `su` command, because it was running as root, could then start a new shell as the user prince. The original shell, as elvis, still exists, and because it is the first process to be started on the terminal tty2, it is still the owner of the `session` associated with that terminal. The `w` command, which lists sessions, is unaware of the shell owned by the user prince.

Online Exercises

Using the newgrp command to change primary groups.

**Lab Exercise**

**Objective:** Use the `newgrp` command to change primary groups and create a file group owned by the new group.

**Estimated Time:** 10 mins.

**Specification**

Use the `id` command to confirm that your account is a member of the secondary group music.

```
[student@station student]$ id
```
uid=2299(student) gid=2299(student) groups=2299(student),201(wrestle),205(music)

In your home directory, create a file titled `music.txt` that is group owned by the group music, and another file, `wrestle.txt`, that is group owned by the group wrestle. In order to do this, you should use the `newgrp` command to switch your primary group. The contents of the files doesn't matter.

If the exercise has been completed correctly, a long listing (`ls -l`) of your home directory should give output similar to the following:

```
[student@station student]$ ls -l
total 8
-rw-r--r-- 1 student music 29 May 27 08:57 music.txt
-rw-r--r-- 1 student wrestle 29 May 27 08:57 wrestle.txt
```

**Deliverables**

1. A file titled `~/music.txt` that is group owned by the group music.
2. A file titled `~/wrestle.txt` that is group owned by the group wrestle.

**Possible Solution**

The following sequence of commands provides one possible solution for creating the file `music.txt`.

```
[student@station student]$ id
uid=2299(student) gid=2299(student) groups=2299(student),201(wrestle),205(music)

[student@station student]$ newgrp music

[student@station student]$ id
uid=2299(student) gid=205(music) groups=2299(student),201(wrestle),205(music)

[student@station student]$ date > music.txt
[student@station student]$ exit
```

**Questions**

**Switching userid and group**

1. Which of the following commands, along with root's password, would allow a user to start a shell as root?
   a. `su - superuser`
   b. `superuser -`
   c. `switchuser root`
   d. `su -`

2. Which of the following commands could be used to start a new shell with the primary group music?
   a. `newgroup music`
   b. `newgrp music`
   c. `newgrp -g music`
d. newgroup -G music

3. Which of the following is the most reasonable explanation for the naming of the `sg` command?
   a. The letters sg stand for super group.
   b. The letters sg stand for switch groupid.
   c. The letters sg stand for sage, the name of the user who is allowed to switch groups at will.
   d. There is no command called `sg`. 