This is Worksheet and Assignment 11

This is a combined Worksheet and Assignment.. Quizzes and tests may refer to work done in this Worksheet and Assignment; save your answers. You will use a checking program to verify the correctness of your work, as described on-line in Assignment #11. You must upload the check program results before the due date.

1 Before you get started - REMEMBER TO READ ALL THE WORDS

You must have your own **Fedora 12** virtual machine (with **`root`** permissions) running to do this lab. You cannot do the lab on the **Course Linux Server** because you do not have **`root`** permissions on that machine. You must follow the on-line in Assignment #11 instructions for checking your work and uploading a mark.

Linux User and Group Management

2 Commands, topics, and features covered

Use the on-line help (**man** command) for the commands listed below for more information.

- **`chown`** – (change owner) Change the owner and/or group of an existing inode (needs **`root`** privilege)
- **`gpasswd`** – administer groups: set group administrator users, set group members, add and remove users from a group, change or remove the group password
- **`groupadd`** – add a new group to the **`/etc/group`** file
- **`groupdel`** – remove a group from the **`/etc/group`** file
- **`groupmod`** – modify group name, number, password account information in the **`/etc/group`** file
- **`groups`** – list the groups you (or another account) are in (from **`/etc/group`**)
- **`id [user]`** – display current account, current groups, and SELinux security context information
- **`newgrp`** – start a new shell with the permissions of a different group (similar to **`su`**)
- **`su [-] [user]`** – (substitute user) Become another user (default **root**), with that user's permissions
- **`sudo`** – do a command or start a shell (default with **root** permissions), configured via **`/etc/sudoers`**
- **`useradd`** – add a new user account and home directory
- **`userdel`** – delete an account (and possibly the home directory as well)
- **`usermod`** – modify account information (and possibly home directory as well)
- **`whoami`** – (who am I?) Display current account userid

3 Correct user, command lines, and command output

- Parts of this lab are done as different **ordinary**, non-**`root`** users. Other parts are done as the **`root`** user. Pay attention to which part is done by which user. Your prompt will tell you if you are the **`root`** user by changing to include a **`#`** characcter instead of a **`$`** character. You can also use the commands **`id`** or **`whoami`** to show your current userid.
- Some answer blanks require you to enter **command lines**. Do **not** include the shell **prompt** with your command lines. Give only the part of the command line that you would type yourself.
- Make sure you know the difference between a command **line** (which is what you type into the shell) and command **output** (which is what the command displays on your screen).

4 Backup and Recovery

a) Take a snapshot of your virtual machine before you begin each section of this lab so that you can recover back to the snapshot if needed.

b) You may find it useful to also make a backup copy in a safe place of the **`/etc/passwd`** file and its ***shadow*** and the **`/etc/group`** file and its ***shadow***. You can compare the old and new files to know what has changed during this lab, and you can recover these files without reverting to a snapshot.

5 Obtain a root (super-user) prompt

To do account management, you will need to obtain a **root** (super-user) prompt so that you'll have the required **privilege** level to run the account creation commands. The **root** account is the only account with sufficient **permissions** to use these commands. To obtain a **root** prompt, use the **Substitute User** command, as follows:

a) Log in to Fedora Linux as your **regular** user account (non-root).
b) Open a terminal window running a shell (**Applications-->System Tools-->Terminal**).
c) On the shell command line, issue the Substitute User command **`su`** followed by a space and the option **`--login`** (there is a shorter synonym for **`--login`** that you can also use if you **RTFM**):

```
[user@host ]$ su --login
```

Enter the **root** password for your Fedora machine **root** account when prompted. Your shell prompt will change from dollar "**$**" to number sign "**#**", indicating you now have **root** ***super-user*** privileges. After a full login, your home directory will also change to be the **root HOME** directory; type **`pwd`** to confirm. Note that the **root** HOME directory is not stored under the **`/home`** directory used for regular accounts!

d) Type the **`whoami`** or **`id`** command to confirm that you are now the **root** user; the output should be: **root**
e) Record the absolute path of the **root** account **HOME** directory: _____

6 Creating accounts – `useradd` *and* `passwd`

You will need **root** privileges to run account management commands.

The **`useradd`** utility creates a new account, storing information about the account in the **`/etc/passwd`** file and about the account groups in the **`/etc/group`** file. (On some versions of Linux - Debian, Ubuntu, etc. - a different command **`adduser`** is used. On Fedora, they are the same command with two different names.) The **`passwd`** utility sets a password for an account, storing the password in the ***shadow*** password file named **`/etc/shadow.`** An account cannot be used until a password has been set. Group passwords (rarely used) are stored in the ***shadow*** group file **`/etc/gshadow`**.

For this section you will require two more ordinary user (non-root) accounts. To create the two accounts follow these steps below (you need **root** privileges to create accounts - become the **root** user first):

a) `[root@host ]# useradd homer`
The above creates a new "**`homer`**" login account and home directory. The account has no password yet.
b) `[root@host ]# passwd homer`
The above sets **`homer`**'s password. If you do not type the username after the **`passwd`** command, you are changing the password of the account that you are signed in with (i.e. **root!)**. Do **not** change your **root** password! Change **`homer`**'s password.
c) Repeat the above steps to create another account named **`flanders`** and give it the same password.
d) Record the account information for the two new accounts by typing: **`id homer ; id flanders`**

_____

_____

e) Give the absolute pathname of the **`flanders`** account home directory: _____
f) Give the **numeric** permissions of the above home directory: _____

7 Creating a Public Directory in the system ROOT

We will create a **`/public`** directory in the **ROOT** directory in which **any user** can create files. The directory will allow any user to create names in it (or remove names). Recall that the permissions on a directory are not the same as the permissions on the inodes named in the directory. Permission to change names does not grant permission to change content. **Pay attention to the use of absolute pathnames in this section!**

a) With **`root`** privileges create a directory called **`/public`** under the top-level **ROOT** directory: **`/public`** (***NOT*** **`/root/public`** and ***NOT*** **`./public`** !) and record the command line you used to **create** it:

_____

b) Give a **command line** that will show the permissions of **only** the new **`/public`** directory:

_____

c) What are the current **numeric** permissions for the **`/public`** directory: _____
d) Record the **owner** and **group** of the **`/public`** directory: _____
e) Give **`/public`** **full access permissions** for **everybody** and record the exact **command** line you used:

_____

f) What are the resulting changed **numeric** permissions for **`/public`**: _____

8 Using the Public Directory

In the next steps, where command lines are required, **do** each command and **record** the command line used:
Pay attention to the use of absolute pathnames in this section!

a) What command line lets you become the **`flanders`** user: _____
b) What command verifies that you are currently the **`flanders`** user: _____
c) What command line creates a new file **`/public/flanfile`**: _____
d) Record the **owner** and **group** of the new **`flanfile`** file: _____
e) What are the current **numeric** permissions for **`flanfile`**: _____
f) What command line removes (only) all **other** permissions from **`/public/flanfile`** and *does not change any existing* ***user*** *or* ***group*** *permissions*: _____
g) What are the resulting **numeric** permissions for **`flanfile`**: _____
h) As user **`flanders`**, append the **`date`** to the new **`flanfile`** file. Record the full **command line** here:

_____

i) What **command line** shows that the size of **`flanfile`** is **29** bytes: _____
j) As the **`homer`** user, try to display the contents of the **`flanfile`** file and record the **error message**:

_____

k) As the **`homer`** user, rename the **`flanfile`** file owned by **`flanders`** to have the new name **`foo`**, and give the output of **`ls -il /public/foo`** showing that the renamed **`foo`** file is still owned by **`flanders`** :

_____

l) As the **`homer`** user, remove the name **`foo`** for the file owned by **`flanders`**. Why can you both **rename** and then **delete** this file that you don't own and can't read? (*Hint*: Names store separately from content.)

_____

_____

9 Changing ownership with `chown`

Pay attention to the use of absolute pathnames in this section!

a) With **`root`** privileges, create an empty file **`/public/foo`** and then change the **owner** and **group** to **`homer`** and **`homer`**. Record the two commands you used to do this:

_____

_____

b) Give the output of **`ls -il /public/foo`** showing the **`homer homer`** owner and group:

_____

c) Become the **`flanders`** user and try to append the **`date`** to **`/public/foo`**. Can you do it? _____
d) Become the **`homer`** user and try to append the **`date`** to **`/public/foo`**. Can you do it? _____
e) As **`root`**, set (only) the **group** and group **permissions** so that *both* **`homer`** *and* **`flanders`** can read and write **`foo`** but **others** cannot. The idea is that the owner of the file will read and write the file using

the **owner** permissions, and the non-owner will be in the **group** of the file and so **group** permissions will apply, allowing access. Other users will be neither the owner of the file nor in the group of the file, so "**other**" permissions will apply to them. Test it as *both* users.
Give the output of **`ls -il /public/foo`**:

_____

10 More account management

Take a snapshot of your virtual machine. Create another new account by doing all of these steps:

```
[root@host ~]# useradd luke                    (create a new luke user and home directory)
[root@host ~]# passwd luke                     (give the new account a password - remember it!)
[root@host ~]# su - luke                       (become the luke user  - dash ensures a full login)
[luke@host ~]$ pwd                             (verify your current directory - the home directory)
[luke@host ~]$ whoami                          (verify your current user)
[luke@host ~]$ groups                          (verify your current groups)
[luke@host ~]$ id                              (verify your current user, groups, and security context)
[luke@host ~]$ exit                            (exit the luke shell and return to the previous user)
[root@host ~]# grep 'luke' /etc/passwd /etc/shadow      (lines containing luke)
[root@host ~]# grep 'luke' /etc/group /etc/gshadow      (lines containing luke)
```

g) Record the **one** line of **password** file **output** from the **grep** command above:

_____

h) Use **`ls -lid`** on the new **home** directory of the new **luke** account and record the output here:

_____

i) Use a command to **find** all pathnames owned by the **luke user**, located under the **`/var`** directory and record the **command line** you used here (**do not** include the shell **prompt** with a command line):

_____

j) Use **`ls -li`** on the ***mail spool file*** output shown by the above command and record the output here:

_____

k) Who owns the ***mail spool file***: _____ What is its group: _____

11 Modifying a user account and group - `usermod` and `groupmod`

- This section depends on the existence of an account named **`luke`**, with an existing home directory, and a group named **`luke.`** Create this account and group if it does not yet exist. Do not proceed until you have a **`luke`** account created. Verify that **`luke`** exists in all four account files:
  `[root@host ]# grep 'luke' /etc/{passwd,shadow,group,gshadow}`
- This section uses the **`usermod`** and **`groupmod`** commands. Use **only** these commands to make the following section's account and group changes. Do **not** use any other commands to make these changes unless told to do so. Do not text-edit any account files; use the commands designed for the purpose.
- The **`usermod`** command modifies account attributes, as recorded in the **password** and **group** files. RTFM and record below the option letters that let **`usermod`** perform the following functions:

1. **login name** - modified with **`usermod`** _____
2. **password** - (must be encrypted already) - modified with **`usermod`** _____
3. **UID**, or **user id number** - modified with **`usermod`** _____
4. **GID** or **group id number** - modified with **`usermod`** _____
5. comment or additional information such as full name - modified with **`usermod`** _____
6. **login program** - shell run when a user logs in - modified with **`usermod`** _____
7. **home directory** - modified (but not actually moved) with **`usermod`** _____
8. actually **moving** a home directory when modifying it requires **`usermod`** _____

- Modifying account information does **not** always automatically move or modify all the files **owned** by the account. If you change some account information, you may have to walk the entire file system to find files owned by the account and change them to match the new values you have set in the password and group files. One exception is moving home directories using **`usermod`**:
- Using both the **`-d`** and **`-m`** options, the **`usermod`** command is able to both change and move a home directory. Follow the syntax shown in the **SYNOPSIS** section of the man page **exactly**.
- The **`groupmod`** command modifies **group name**, number, and password, as recorded in the **group** file.

a) Modify the **login** name of the **`luke`** account to be **`darth`** and record the **command line** you used here:

_____

b) Modify the **group** name of the **`luke`** group to be **`darth`** and record the **command line** you used here:

_____

c) The new **`darth`** account still uses a home directory of **`/home/luke`**. Modify and move (in one command line) this old home directory from its current **`luke`** name to the new name **`sith`** (use the absolute path!) and record the one command line you used here (RTFM and learn how to use **`-d`** and **`-m`**):

_____

If you don't get this command correct on the first try, using both options correctly, you may have to restore your snapshot or restore your backup password and group files and try again. You might want to take a special snapshot before trying this command! **Follow the syntax shown in the SYNOPSIS section of the man page exactly.**

d) Check your work! Use a command to search in the **password** and **group** files and make sure the word **`luke`** does **not** appear anywhere in those files. Look in the **`/home`** directory and make sure that the old **`luke`** directory has been correctly moved to **`sith`**. Do **not** proceed until you check your work!
When you have verified that the account has been moved, use **`su - darth`** to login as the new account and record the **output** of typing the two commands **`pwd`** and then **`id`** in the new **`darth`** account:

_____

_____

_____

e) Use the appropriate option to the **`chsh`** command to print the list of shells. Now change the shell for **`darth`** to be the one that prevents logins ("***no logins***"). Record the **command line** you used to change the shell for **`darth`**, followed by the **output** of **`su - darth`** showing the **disabled account message**:

_____

_____

f) **Repeat** the above steps and completely move the new **`darth`** account and group to be the new name **`yoda`** with home directory under the usual place with new name **`master`**. Check your work carefully after you have followed all the steps! Login to the **`yoda`** account as before and again record the **output** of typing the two commands **`pwd`** and then **`id`** in the new **`yoda`** account (if you can't log in because the account is disabled, you should know why - reset the login shell to **`/bin/bash`** and try again):

_____

_____

_____

g) Check your work! Use a command to search in all four **password** and **group** files and make sure the word "**`darth`**" does **not** appear anywhere in those files. Record that command line you used here:

_____

h) Make sure the **`yoda`** home directory is in the correct location and has the correct owner and group. Copy the full **output** of the command that shows its inode, permissions, owner, group, modify date, etc. here:

_____

12 Deleting an account - `userdel`

Take a snapshot of your virtual machine. Create another new account by doing all these steps below:

```
[root@host ~]# useradd redshirt          (create an expendable account redshirt and home dir)
[root@host ~]# su redshirt               (become [login as] the new redshirt user)
[redshirt@host ~]$ su root               (become root on top of the logged in redshirt user)
[root@host ~]# userdel redshirt          (try, and fail, to delete logged-in user - does not work)
[root@host ~]# exit                      (exit the root shell and return to the redshirt shell)
[redshirt@host ~]$ exit                  (exit the redshirt shell - redshirt no longer logged in)
[root@host ~]# userdel redshirt          (delete the redshirt account info, but not the home dir)
[root@host ~]# grep 'redshirt' /etc/{passwd,shadow,group,gshadow}   (no output)
[root@host ~]# su - redshirt             (try, and fail, to become a nonexistent redshirt user)
```

a) Use **`ls -lid`** on the existing **home** directory of the deleted **`redshirt`** account and record the output:

_____

b) Note the **numeric** owner and group numbers in the above output, due to the deleted **`redshirt`** account and group. The directory still exists and has its previous **numeric** owner and group IDs, but no accounts or groups exist for those IDs so they print as simple **numbers**. If you now create a **new** account, and the new account is assigned **those** IDs, the files formerly owned by **`redshirt`** will now be owned by the new account. This is almost never what you want. We will show this in the next step. But first:

Record the numeric owner and group of the old **`redshirt`** HOME here:

_____

c) Create a new account named **`newguy`** and then repeat the above **`ls -lid`** on the former **home** directory of the deleted **`redshirt`** account and record the new output here, showing the how the old **`redshirt`** HOME directory is now owned by the new **`newguy`** owner and group:

_____

d) As the above shows, you must make sure you fully delete an account **and all its files** no matter where the files are in the file system. The **`userdel`** command can remove **home** directories using an **option**. Use that option to **fully** remove the **`newguy`** account you just created and record the command line you used here:

_____

e) Removing the **`newguy`** account and its home directory did *not* remove the old **`redshirt`** files, even though they were owned by the **`newguy`** account. Give a command that will **find** and display **every** file and directory owned by the **numeric ID** of the former **`redshirt`** account (command name hint: *find* using its option *user*). Pick the correct starting directory for the search, so that the command finds *all* the files, no matter where they are! Some error messages will also print with the output of the command - redirect just the error messages to **`/dev/null`**. Command used to find all the former **`redshirt`** files and directories:

_____

f) Using ordinary commands (**not** account commands learned in this lab), completely and recursively remove all files and directories that still belong to the deleted **redshirt** account (listed in the previous question) and record the one or two **command lines** used (you can do it all in one command line with two pathnames):

_____

_____

g) Re-run the command that searches for files owned by the former **`redshirt`** numeric ID. All gone, yes?

Group management: The Megadeth Project

- This section uses some commands you have not used before.
- Every command you need to use is mentioned in the opening page of this lab document and is described in the lecture notes. Become familiar with this list of commands before you continue, so that you know what command name to choose below. Every command has a man page.
- You do not need to text-edit any files in this section. **RTFM**!

1 Requirements for Group Management:

- The four-person band **Megadeth** (note the unusual spelling of **Megadeth** and make sure you use this exact spelling) uses the following work approach and has the following **Requirements**:

  a) Song files are created by one band member who is the single **group administrator** account. Only the one **group administrator** account can create, delete, modify and write song files.
  b) Files are readable (not writable or removable) by all other (non-administrator) group (band) members. Ordinary band members can **only** read the files, not change or rename them.
  c) Anybody who is not a **band member** is not allowed to view song files. No public access.

- Follow the directions below to create accounts and directories that implement the above permissions. Some of the work will need to be done as the **root** super-user. (Only the **root** user can create new accounts.) Some group maintenance work can be done as the band member who is assigned to be the **group administrator**.
- The four **Megadeth** band members are (get the name and account spellings correct! Case matters):
  - **Chris Broderick** – login name: **`broderc`**
  - **Dave Mustaine** – login name: **`mustaid`**
  - **Shawn Drover** – login name: **`drovers`**
  - **David Ellefson** – login name: **`ellefsd`**

2 Creating and configuring the Megadeth Working Group

a) Take a VM **snapshot** before you begin this section, so you can return here if you make many mistakes.
b) Use the **`--comment`** option to include the user's **full name** in each account you create (remember to **quote** names containing blanks) and record the **four command lines** used to create ordinary user accounts for all four band members (do not set any special groups yet - just create ordinary accounts):

_____
_____
_____
_____

Confirm that all **four** band members have accounts containing their **full names** by looking in a file.

a) Record the **one** command used to create a new **group** named **`megadeth`** (spelled all lower case):

_____

b) Set **Chris Broderick** as the **group administrator** of the new group and record the **command line** used:

_____

c) Become (**su**) the **`megadeth`** **group administrator**. Record the four **command lines** used by Chris to **add** each of the **four** band members to the **`megadeth`** group (**exit** the Chris shell when you are done):

_____
_____
_____
_____

Confirm that the **group** file contains the new group with all **four** band members listed beside it.

d) Create a new song directory named **`/home/music`** and record the **output** of **`ls -lid`** on the new directory (it will be owned by **root** and in group **root** with default permissions):

_____

e) The band wants to store songs under the **`music`** directory, matching the **Requirements** given above. Set ownership and permissions for the **`music`** directory to implement the given **Requirements:**
   1. Which account should become the **owner** of **`music`**? _____
   2. Which group should become the **group** of **`music`**? _____
   3. Which **permissions** (symbolic) should be set on **`music`**? _____

f) Record all the commands (minimum two) used to implement the above **Requirements**:

_____

_____

_____

_____

3 Test Plan for the Megadeth Group Project

You need to **verify** that the requirements have been **met** using a **Test Plan**. Here it is:

a) Become (**su**) the **group administrator** and **redirect** the current **date** into a file named **`test`** in the **`music`** directory. Display the file on your screen to make sure it has content you can see as the **group administrator**. Record the output of command **`ls -li`** using the **absolute** pathname to the **`test`** file in the **`music`** directory (does this new file have the correct owner, group, and permissions?):

_____

a) Also record the same information for just the **`music`** directory itself (use the **absolute** pathname):

_____

Exit the **group administrator** account when you are done. Refer again to the **Requirements**, listed at the start of this section. Complete the tests below for each type of user logged in (using **su)**, making sure the test **results** match the **Requirements** (Hint: **Neither** of the test accounts below should be able to **modify** or **delete** the file.):

3.1 Table #1 - test results when logged in as different users

Test to perform: Can you...	Logged in as a band member (not the group administrator)	Logged in as any non-band user account (not root!)
List the contents of the **`music`** directory?	**Yes or No?** _____	**Yes or No?** _____
Change into the **`music`** directory?	**Yes or No?** _____	**Yes or No?** _____
Read the file **`test`**?	**Yes or No?** _____	**Yes or No?** _____
Modify the file **`test`**?	**Yes or No?** _____	**Yes or No?** _____
Delete the file **`test`**?	**Yes or No?** _____	**Yes or No?** _____

The Test Plan records the results of your testing. Do the test results meet the original **Requirements**? _____

3.2 Table #2 - excerpt from the /etc/passwd file

Record the entries in the **`/etc/passwd`** file for each user created:

User Name	Password	UID	GID	Home Directory	Login Shell
`broderc`	X	____	____	_____	_____
`mustaid`	X	____	____	_____	_____
`drovers`	X	____	____	_____	_____
`ellefsd`	X	____	____	_____	_____

3.3 Table #3 - excerpt from the /etc/group file

Record the entries in the **`/etc/group`** file for these group entries:

Group Name	Password	GID	Group Members (if any)
`broderc`	X	____	_____
`mustaid`	X	____	_____
`drovers`	X	____	_____
`ellefsd`	X	____	_____
`megadeth`	X	____	_____

4 Finishing touch - four symbolic links

a) Log in as each band member (four times) and create the shortest relative (not absolute) **soft link** (symbolic link) named **`music`** in the home directory that links up to **`/home/music`** so that each member can then use the soft link to access the **`/home/music`** directory instead typing the entire pathname. Record the **command line** used to create this symbolic link named **`music`**:

_____

Lab Check and Upload

Follow the posted instructions on-line in Assignment #11 for checking your work and uploading a mark to Blackboard by the given due date:

`http://teaching.idallen.com/cst8207/13w/notes/assignment11.html`