Student Name: _____ **Lab Section:** _____

/3 marks

Linux User and Group Management

1 Due Date - Upload to Blackboard by 8:30am Monday April 2, 2012

Submit the completed lab to Blackboard following the Rules for submitting Online Labs and Assignments. You must upload **two** files for this submission. You must always upload **both** files when you submit. Both.

2 Commands, topics, and features covered

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

- **`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)
- **`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
- **`newgrp`** – start a new shell with the permissions of a different group (similar to **`su`**)
- **`gpasswd`** – administer groups: set group administrator users, set group members, add and remove users from a group, change or remove the group password
- **`groups`** – list the groups you (or another account) are in (from /etc/group)
- **`id`** – display current account, current groups, and SELinux security context information

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.
- 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

- Take a snapshot of your virtual machine before you begin this lab so that you can recover if needed.
- Make a backup copy of the **/etc/passwd** file and its ***shadow*** and the **/etc/group** file and its ***shadow***.

5 Creating an account – `useradd` and `passwd`

You will need **root** privileges to run these 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 are stored in the ***shadow*** group file **`/etc/gshadow`**.

```
[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     (display lines containing luke)
[root@host ~]# grep 'luke' /etc/group /etc/gshadow     (display lines containing luke)
```

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

_____

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

_____

c) 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):

_____

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

_____

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

6 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 (see Section 1, above) 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.
- The **`usermod`** command modifies account attributes, as recorded in the **password** file. Some are:
  - **login name** - modified with **`usermod -l`**
  - **password** - encrypted and stored in **`/etc/shadow`** file;
    - modified with **`usermod -p`** *or* **`passwd`**
  - **UID**, or **user id number** - modified with **`usermod -u`**
  - **GID** or **group id number** - modified with **`usermod -g`**
  - additional information, such as full name; modified with **`usermod -c`**
  - **home directory** - modified with **`usermod -d ... -m`**
  - **login program** - program (shell) run when a user logs in - modified with **`usermod -s`** *or* **`chsh`**
- Modifying account information does **not** automatically move or modify files **owned** by the account. If you change account information, you may have to walk the entire file system to find files owned by the account and change them. One exception is moving home directories using **`usermod`**:
- Using the **`-d`** and **`-m`** options, the **`usermod`** command is able to move a home directory.
- 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 **`lord`** (use the absolute path!) and record the one command line you used here (RTFM and learn how to use **`-d`** and **`-m`**):

_____

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 **lord**. 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 subdirectory name **jedi**. 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:

_____

7 Deleting an account - `userdel`

```
[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 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. Create a new account named **bogus** 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 new bogus owner and group:

_____

c) 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 **`bogus`** account you just created and record the command line you used here:

_____

d) Removing the **`bogus`** account and its home directory did *not* remove the old **`redshirt`** files. 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:

_____

e) 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):

_____

_____

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

8 Group management: The Megadeth Project

- Take a VM **snapshot** before you begin this section, so you can return here if you make many mistakes.
- This section uses commands you have not used before. Every command you need to use is mentioned in the opening page(s) of this lab document. Every command has a man page. RTFM!
- **Requirements for Group Management**:
  The band **Megadeth** (note the unusual spelling of **Megadeth** make sure you use the correct spelling) uses the following work approach and has the following **Requirements**:

  1. Song files are created by a single **group administrator** account. Only the one **group administrator** account can create, delete, modify and write song files.
  2. 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.
  3. 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 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`**

8.1 Creating and configuring the Megadeth Working Group

a) 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.

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

_____

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

_____

d) 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.

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

_____

f) The band wants to store songs under the **`songs`** directory, matching the **Requirements** given above. Set ownership and permissions for the **`songs`** directory to implement the given **Requirements:**

1. Which account should become the **owner** of **`songs`**? _____
2. Which group should become the **group** of **`songs`**? _____
3. Which **permissions** (symbolic) should be set on **`songs`**? _____

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

_____
_____
_____
_____

8.2 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 **`songs`** directory. Display the file on your screen to make sure it has content you can see. Record the output of command **`ls -li`** using the **absolute** pathname to the **`test`** file in the **`songs`** directory:

_____

a) Also record the same information for just the **`songs`** 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.):

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)	Loged in as any non-band user account (not root!)
List the contents of the **`songs`** directory?	**Yes or No?** _____	**Yes or No?** _____
Change into the **`songs`** 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**? _____

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	____	____	_____	_____

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	____	_____

8.3 Finishing touch - four symbolic links

a) Log in as each band member (four times) and create a **soft link** (symbolic link) named **`tunes`** in the home directory that links up to **`/home/songs`** so that each member can then use the soft link to access the **`/home/songs`** directory instead typing the entire pathname. Record the **command line** used to create this symbolic link named **`tunes`**:

_____

Lab Check and Upload - `lab07marks.txt`

This is the section that tests and marks the work you did above. The Lab Check program below will do the checking to make sure you got things right. (Did you correctly move **luke** to **darth**? Did you correctly move **darth** to **yoda**? Did you fully delete the **redshirt** account? Is the **songs** directory configured correctly?)

Download (right-click and **Save Link As**) the **`lab07check`** program from the Class Notes into **your** home directory, make it executable, and run it as the **root** user with the **HOME** variable (all **upper**-case) set to the **absolute path** of **your** own **HOME** directory:

```
[user@host ]$ cd
[user@host ~]$ su root                    (do not use the dash that means a full login)
[root@host user]# chmod u+x lab07check
[root@host user]# HOME=/home/user         (use your own HOME directory name here)
[root@host user]# ./lab07check
```

This program will check your work for this lab, assign you a mark, and put the mark and status information into a new file **`lab07marks.txt`** so that you can upload it to Blackboard as part of this assignment. **Do not print this file - it contains dozens of pages of status information!** You may run the **`lab07check`** program *as many times as you wish*, to correct mistakes and get the best mark, before you upload the final marks file.

You will upload **two** files for this assignment: this ODT document and the **`lab07marks.txt`** output file. Use the **exact** names given for uploading. Do not change the names.

Do not print the `lab07marks.txt` file - it contains dozens of pages of status information!
You will upload two files for this assignment. Two. Two files. You will use the correct names. Yes.