Exercise 05 - Deleting Local Users - Script 4

Goal:

The goal of this exercise is to create a shell script that allows for a local Linux account to be disabled, deleted, and optionally archived.

Scenario:

Today the help desk team received a request to delete an account for a person who has changed departments. This person doesn't have time to log into Linux systems -- they're in management now!

The help desk team also wants to save the contents of this person's home directory just in case this person returns to their original department and job.

They realize this is just going to be the first of these types of requests. They know they're eventually going to receive requests to disable accounts for when people are on extended leave.

Of you course you know it's going to be way easier to write a script and hand it off than it is for you to do all the work.

Shell Script Requirements:

You think about what the shell script must do and how you would like it operate. You come up with the following list.

The script:

- Is named "`disable-local-user.sh`".
- Enforces that it be executed with superuser (root) privileges. If the script is not executed with superuser privileges it will not attempt to create a user and returns an exit status of 1. All messages associated with this event will be displayed on standard error.
- Provides a usage statement much like you would find in a man page if the user does not supply an account name on the command line and returns an exit status of 1. All messages associated with this event will be displayed on standard error.
- Disables (expires/locks) accounts by default.
- Allows the user to specify the following options:
    - -d Deletes accounts instead of disabling them.
    - -r Removes the home directory associated with the account(s).

    - -a Creates an archive of the home directory associated with the accounts(s) and stores the archive in the /archives directory. (NOTE: /archives is not a directory that exists by default on a Linux system. The script will need to create this directory if it does not exist.)
    - Any other option will cause the script to display a usage statement and exit with an exit status of 1.
- Accepts a list of usernames as arguments. At least one username is required or the script will display a usage statement much like you would find in a man page and return an exit status of 1. All messages associated with this event will be displayed on standard error.
- Refuses to disable or delete any accounts that have a UID less than 1,000.
    - Only system accounts should be modified by system administrators. Only allow the help desk team to change user accounts.
- Informs the user if the account was not able to be disabled, deleted, or archived for some reason.
- Displays the username and any actions performed against the account.

Start the Virtual Machine and Log into It:

In a previous exercise you created a vagrant project called localusers. Use the VM created in that project for this exercise.

First, start a command line session on your local machine. Next, move into the working folder you created for this course.

```
cd shellclass
```

Change into the localusers directory, start the virtual machine with "vagrant up", and then connect to it with "vagrant ssh".

```
cd localusers
vagrant up
vagrant ssh
```

Navigate to the `/vagrant` Directory

```
cd /vagrant
```

Write the Shell Script

At this point, you can either create the script inside the virtual machine using the `vim`, `nano`, or `emacs` text editors or you can create the file using your favorite text editor on your local operating system. (Atom from https://atom.io/ is a good choice.)

When creating your script, refer back to the shell script requirements. If you want or need more detailed steps to help you write your script, refer to the pseudocode at the end of this document. It was intentionally placed at the end of the document because I want to encourage you to write the script on your own. It's fine if you need the pseudocode. As you get more scripting practice, you'll be able to script without any additional aids.

Test Your Script

Once you've finished writing the script, create some user accounts on the system to give yourself something to test with. (You can manually add them with the useradd command or use one of the previous scripts you wrote to add them.)

- carrief
- markh
- harrisonf
- alecg
- peterm

Test the script by:

- Executing it without super user privileges.
- Executing it with super user privileges, but without any options or arguments.
- Executing it with super user privileges, but with an invalid option.
- Attempting to disable a system account.
- Disabling the carrief account.
- Deleting the markh account.
- Deleting the harrisonf account and the home directory.
- Deleting the alecg and peterm accounts while removing their home directories and making an archive of their home directories.

Remember that the first time you execute the script you'll need to make sure it has executable permissions.

```
chmod 755 disable-local-user.sh
```

Here is an example run of the script. (Portions typed are in bold.)

```
./disable-local-user.sh
Please run with sudo or as root.
echo ${?}
1
```

Make sure the script displays a usage message if we don't supply an account.

```
sudo ./disable-local-user.sh
Usage: ./disable-local-user.sh [-dra] USER [USERN]
Disable a local Linux account.
  -d  Deletes accounts instead of disabling them.
  -r  Removes the home directory associated with the account(s).
  -a  Creates an archive of the home directory associated with the
accounts(s).
echo ${?}
1
```

Make sure the script displays usage message if we supply an invalid option

```
sudo ./disable-local-user.sh -z
./disable-local-user.sh: illegal option -- z
Usage: ./disable-local-user.sh [-dra] USER [USERN]
Disable a local Linux account.
  -d  Deletes accounts instead of disabling them.
  -r  Removes the home directory associated with the account(s).
  -a  Creates an archive of the home directory associated with the
accounts(s).
echo ${?}
1
```

Attempting to disable a system account.

```
sudo ./disable-local-user.sh mail
Processing user: mail
Refusing to remove the mail account with UID 8.
```

Disable the carrief account.

```
sudo ./disable-local-user.sh carrief
Processing user: carrief
The account carrief was disabled.
```

Make sure the account is disabled by logging into it.

```
su - carrief
Password: pass123
Your account has expired; please contact your system administrator
su: User account has expired
```

Make sure the user's home directory still exists:

```
ls -ld /home/carrief
drwx------ 2 carrief carrief 4096 Jan 25 12:20 carrief
```

Now, delete the markh account.

```
sudo ./disable-local-user.sh -d markh
Processing user: markh
The account markh was deleted.
```

Make sure the account is deleted and that home directory still exists.

```
id markh
id: markh: no such user
ls -ld /home/markh
drwx------ 2 1009 1009 4096 Jan 25 12:20 /home/markh
```

Delete the harrisonf account and the associated home directory.

```
sudo ./disable-local-user.sh -dr harrisonf
Processing user: harrisonf
The account harrisonf was deleted.
```

Check to see if the account and directory were deleted.

```
id harrisonf
id: harrisonf: no such user
ls -ld /home/harrisonf
ls: cannot access /home/harrisonf: No such file or directory
```

Delete the alecg and peterm accounts. Archive and then remove their home directories.

```
sudo ./disable-local-user.sh -dra alecg peterm
Processing user: alecg
Creating /archive directory.
Archiving /home/alecg to /archive/alecg.tgz
The account alecg was deleted.
Processing user: peterm
Archiving /home/peterm to /archive/peterm.tgz
The account peterm was deleted.
```

Make sure the accounts and home directories are gone:

```
id alecg
id: alecg: no such user
id peterm
id: peterm: no such user
ls -ld /home/alecg /home/peterm
ls: cannot access /home/alecg: No such file or directory
ls: cannot access /home/peterm: No such file or directory
```

Make sure the archives were created:

```
ls -l /archive/
total 8
-rw-r--r-- 1 root root 487 Jan 25 13:06 alecg.tgz
-rw-r--r-- 1 root root 487 Jan 25 13:06 peterm.tgz
tar -ztvf /archive/alecg.tgz
drwx------ alecg/alecg       0 2018-01-25 13:06 home/alecg/
-rw-r--r-- alecg/alecg     231 2016-12-06 18:19 home/alecg/.bashrc
-rw-r--r-- alecg/alecg     193 2016-12-06 18:19 home/alecg/.bash_profile
-rw-r--r-- alecg/alecg      18 2016-12-06 18:19 home/alecg/.bash_logout
tar -ztvf /archive/peterm.tgz
drwx------ peterm/peterm     0 2018-01-25 13:06 home/peterm/
-rw-r--r-- peterm/peterm   231 2016-12-06 18:19 home/peterm/.bashrc
-rw-r--r-- peterm/peterm   193 2016-12-06 18:19 home/peterm/.bash_profile
-rw-r--r-- peterm/peterm    18 2016-12-06 18:19 home/peterm/.bash_logout
```

Reference Material:

Vagrantfile for localusers

Here are the contents of the `shellclass/localusers/Vagrantfile` file with all the comments removed.

```
Vagrant.configure(2) do |config|
  config.vm.box = "jasonc/centos7"
  config.vm.hostname = "localusers"
end
```

Pseudocode

You can use the following pseudocode to help you with the logic and flow of your script.

```
  # Display the usage and exit.

# Make sure the script is being executed with superuser privileges.

# Parse the options.

# Remove the options while leaving the remaining arguments.

# If the user doesn't supply at least one argument, give them help.

# Loop through all the usernames supplied as arguments.

  # Make sure the UID of the account is at least 1000.

  # Create an archive if requested to do so.

    # Make sure the ARCHIVE_DIR directory exists.

    # Archive the user's home directory and move it into the ARCHIVE_DIR

      # Delete the user.

      # Check to see if the userdel command succeeded.
      # We don't want to tell the user that an account was deleted when
it hasn't been.

      # Check to see if the chage command succeeded.
      # We don't want to tell the user that an account was disabled when
it hasn't been.
```