Exercise 04 - Creating Local Users - 03

Goal:

The goal of this exercise is to create a shell script that adds users to the same Linux system as the script is executed on. Additionally this script will conform to Linux program standard conventions.

Scenario:

The help desk team loves the script you created for them. However, they have just one more request. They want the script to only display the details that they need to send to the user after they create their account.

You decide that's an easy enough change. Because you know you'll have to use redirection to accomplish the task, you decide add redirection in other places in the script to make it conform to standard UNIX/Linux program conventions. Namely, sending errors to STDERR.

Shell Script Requirements:

You have your requirements from the "`add-new-local-user.sh`" script you created. You decide to use those as the basis for your new requirements. You come up with the following list.

The script:

- Is named "`add-newer-local-user.sh`". (You change the name slightly to distinguish it from the previous script. NOTE: In the real world you could have easily kept the same script name. I just want to use a different name for the purpose of discussing specific scripts in the class.)
- 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.
- Uses the first argument provided on the command line as the username for the account. Any remaining arguments on the command line will be treated as the comment for the account.
- Automatically generates a password for the new account.
- Informs the user if the account was not able to be created for some reason. If the account is not created, the script is to return an exit status of 1. All messages associated with this event will be displayed on standard error.

- Displays the username, password, and host where the account was created. This way the help desk staff can copy the output of the script in order to easily deliver the information to the new account holder.
- Suppress the output from all other commands.

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.

NOTE: This script is very similar to the script you created in the previous exercise. You can use it as the starting point for this script and make the required changes or you can start from scratch to give you even more practice.

Test Your Script

Once you've finished writing the script, test it by creating the following accounts:

- Username: turing
  - Real Name: Alan Turing
- Username: woz
  - Real Name: Steve Wozniak
- Username: moore
  - Real Name: Gordon Moore

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

```
chmod 755 add-newer-local-user.sh
```

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

```
sudo ./add-newer-local-user.sh turing Alan Turing
username:
turing

password:
d4c708512e5a116b327197cd9d59d350ca4ab5938fc463f1

host:
localusers
```

Make sure the accounts have been created by examining the last 3 lines of the `/etc/passwd` file.

```
cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
... # additional accounts will be displayed
turing:x:1007:1007:Alan Turing:/home/turing:/bin/bash
woz:x:1008:1008:Steve Wozniak:/home/woz:/bin/bash
moore:x:1009:1009:Gordon Moore:/home/moore:/bin/bash
```

(NOTE: You could have also used `tail -3 /etc/passwd` to display just the last 3 lines of the file.)

Switch to the `turing` user. Because the script forces a password change upon first login, create a new password for the `turing` user. (Suggested password: `"EnigmaBreaker"`)

```
su - turing
```

Once you've changed the password of the user, exit out of the session to return to the vagrant user.

```
exit
```

Test to make sure that the script exits with a non-zero exit status if the user does not use superuser (root) privileges. (Portions typed are in bold.)

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

Test to make sure that the error message is displayed on standard error. (Portions typed are in bold.)

```
./add-newer-local-user.sh 1>std.out 2>std.err
cat std.out
cat std.err
Please run with sudo or as root.
rm std.out std.err
```

The contents of the file "std.out" should be blank while the contents of the file "std.err" should contain the error message.

Test to make sure that the script exits with a non-zero exit status if the user does not supply a username. (Portions typed are in bold.)

```
sudo ./add-newer-local-user.sh
Usage: ./add-newer-local-user.sh USER_NAME [COMMENT]...
Create an account on the local system with the name of USER_NAME and a
comments field of COMMENT.
echo ${?}
1
```

Test to make sure that the error message is displayed on standard error. (Portions typed are in bold.)

```
sudo ./add-newer-local-user.sh 1>std.out 2>std.err
cat std.out
cat std.err
Usage: ./add-newer-local-user.sh USER_NAME [COMMENT]...
Create an account on the local system with the name of USER_NAME and a
comments field of COMMENT.
rm std.out std.err
```

The contents of the file "std.out" should be blank while the contents of the file "std.err" should contain the error message.

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.

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

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

# The first parameter is the user name.
```

```
# The rest of the parameters are for the account comments.

# Generate a password.

# Create the user with the password.

# Check to see if the useradd command succeeded.

# Set the password.

# Check to see if the passwd command succeeded.

# Force password change on first login.

# Display the username, password, and the host where the user was
created.
```