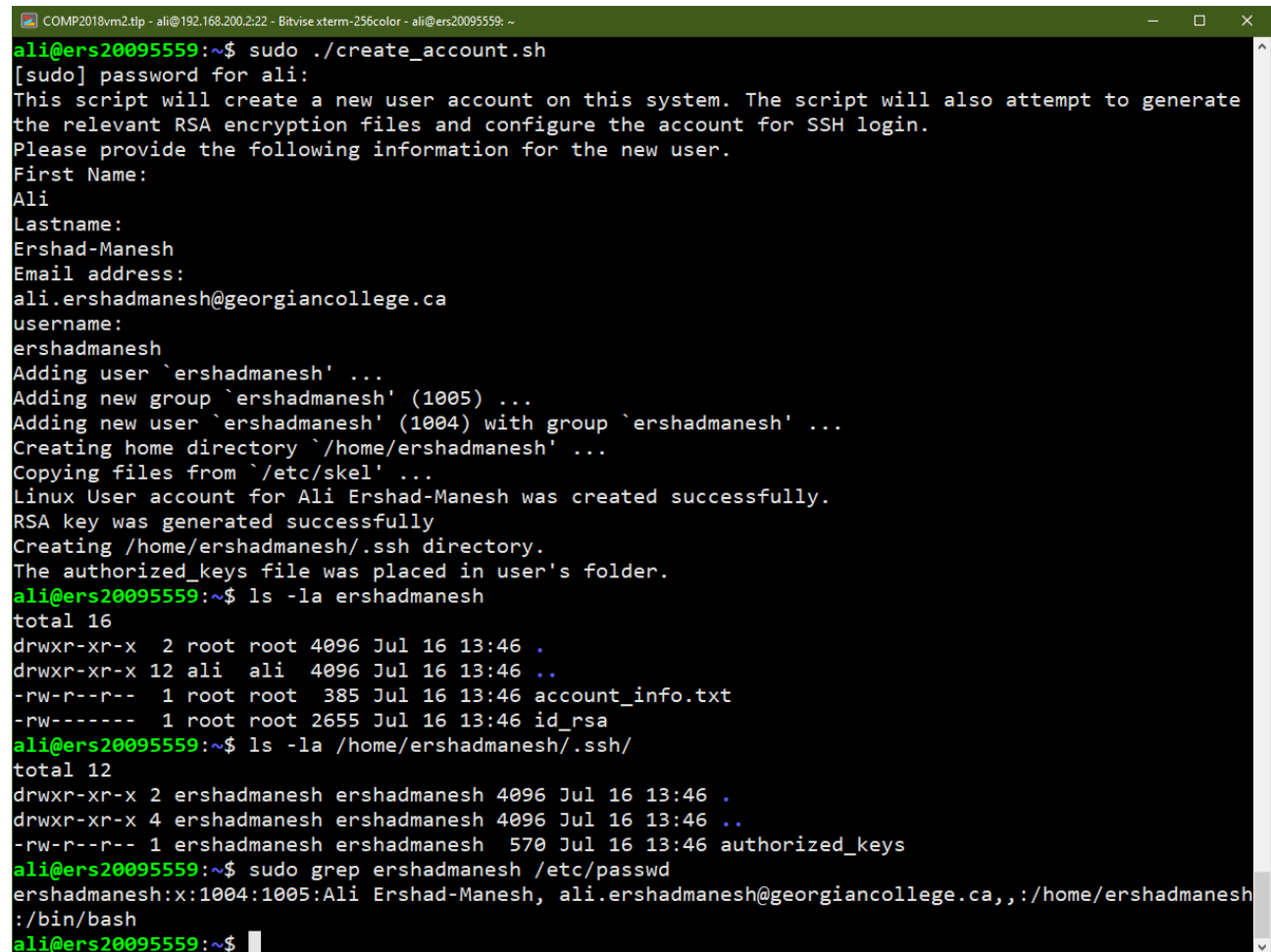


Assignment 8 – Creating script file for automation

Objectives

Create a script to automatically create user accounts, generate the appropriate SSH encryption keys, and configure the ssh authorized_keys file for the user. The script should also create directory to store the private key and a message to the user that contains the passphrase. Here is what the final result should look like:

A terminal window titled 'COMP2018vm2.tlp - ali@192.168.200.2:22 - Bitvise xterm-256color - ali@ers20095559: ~' shows the execution of a script. The user 'ali' runs 'sudo ./create_account.sh'. The script prompts for a password and then asks for user details: First Name (Ali), Lastname (Ershad-Manesh), and Email address (ali.ershadmanesh@georgiancollege.ca). It then asks for a username (ershadmanesh). The script proceeds to add the user, create a group, and generate an RSA key. It also creates a directory for SSH keys and places the authorized_keys file. Finally, it lists the files in the user's home directory and the SSH directory, and checks the password file for the new user.

```
ali@ers20095559:~$ sudo ./create_account.sh
[sudo] password for ali:
This script will create a new user account on this system. The script will also attempt to generate
the relevant RSA encryption files and configure the account for SSH login.
Please provide the following information for the new user.
First Name:
Ali
Lastname:
Ershad-Manesh
Email address:
ali.ershadmanesh@georgiancollege.ca
username:
ershadmanesh
Adding user `ershadmanesh' ...
Adding new group `ershadmanesh' (1005) ...
Adding new user `ershadmanesh' (1004) with group `ershadmanesh' ...
Creating home directory `/home/ershadmanesh' ...
Copying files from `/etc/skel' ...
Linux User account for Ali Ershad-Manesh was created successfully.
RSA key was generated successfully
Creating /home/ershadmanesh/.ssh directory.
The authorized_keys file was placed in user's folder.
ali@ers20095559:~$ ls -la ershadmanesh
total 16
drwxr-xr-x  2 root root 4096 Jul 16 13:46 .
drwxr-xr-x 12 ali  ali 4096 Jul 16 13:46 ..
-rw-r--r--  1 root root  385 Jul 16 13:46 account_info.txt
-rw-----  1 root root 2655 Jul 16 13:46 id_rsa
ali@ers20095559:~$ ls -la /home/ershadmanesh/.ssh/
total 12
drwxr-xr-x 2 ershadmanesh ershadmanesh 4096 Jul 16 13:46 .
drwxr-xr-x 4 ershadmanesh ershadmanesh 4096 Jul 16 13:46 ..
-rw-r--r-- 1 ershadmanesh ershadmanesh 570 Jul 16 13:46 authorized_keys
ali@ers20095559:~$ sudo grep ershadmanesh /etc/passwd
ershadmanesh:x:1004:1005:Ali Ershad-Manesh, ali.ershadmanesh@georgiancollege.ca,,:/home/ershadmanesh
:/bin/bash
ali@ers20095559:~$
```

Step 1: Getting information

Start your script by asking the administrator user to input the following information for the end user they are creating the account for:

- First name for the end user
- Last name
- Email address
- A Linux username for end user account

Refer to the template file provided to get started:

```

echo "This script will create a new user account on this system. The script will
also attempt to generate the relevant RSA encryption files and configure the
account for SSH login. "
# Step 1 : Get Information
#####
echo "Please provide the following information for the new user. "
echo "First Name: "
read FirstName
echo "Lastname: "
read LastName
echo "Email address: "
read email
echo "username: "
read username

```

Step 2: Attempting to create the user account and check for success or failure

In order to create a new user from the shell terminal we use adduser command. The adduser command is interactive by default (meaning it will ask questions and prompt for answers). Unfortunately, interactivity is not desired when trying to automate processes. Imagine if you are going to be using this script to create 100s of user accounts at a time, reading the user information from a file instead of typing them in the shell. Therefore, refer to the man pages for adduser to find the flags and options needed to accomplish using the command properly in one line without the need to interact with the command.

Make sure that the **password is disabled** upon the creation of the account, and **user information** (firstname, last name and the email address) **are also recorded** in the user profile for future reference (Hint: see the adduser option called "gecos" for extra information such as real name and email address).

In order to check to see if the adduser command was successful we can use the following if statement and take appropriate action in case something goes wrong. Note that adduser will generate an exit status code of 0 when successful.

```

# Did adduser work? what to do if there is an error?
# Gives exit status 1 if adduser failed or generated error
if [ $? -eq 0 ]
then
    echo "Linux User account for ${FirstName} $LastName was created
successfully. "
else
    echo '!!!!!! User account was not added. AddUser Returned exit status '$?'
    '!!!!!!'
    exit 1
fi

```

Step 3: Generate a random password

We are going to be generating an RSA SSH key for this user to be able to ssh into our server. However for better security we would like to have a passphrase on the RSA key as well. Although this is optional, but here we want it anyways.

There is a random number generator device in Linux (/dev/urandom) that we will use for this purpose. We must translate (tr) the output of the random number generators to ASCII letters, numbers, and symbols, then only keep a desired number of them to limit the length. For the sake of time and scope of this lab I am not going to ask you to research this. Below is the command that accomplishes this. Notice the “back quotes” and “;”. You can also try inputting this command into your shell to see what it gives you, and subsequently modify the command or play with various parameters to learn and explore.

```
# Generate a strong, 16 characters, random password
Password=`< /dev/urandom tr -dc _A-Z-a-z-0-9%\%&\*\$#\@\! | head -c${1:-16};echo;`
```

Step 4: Generate a message for the user

Let's store the passphrase and username in a directory located in the pwd where the script is running from. We are naming the directory the same as the username we are creating:

```
# Create a directory to store the information we will need to provide to the user
later
mkdir $username
# Generate a message for the user and store in account_info.txt
echo "Hello $FirstName" > ./${username}/account_info.txt
echo "You have been granted a user account on our system. " >>
./${username}/account_info.txt
echo "Your User Name is: " $username >> ./${username}/account_info.txt
echo "Your randomly generated Passphrase is: " $Password >>
./${username}/account_info.txt
echo "You are being provided with your own RSA private key. The server will be
configured with your public key in order to allow you access to server via SSH. "
>> ./${username}/account_info.txt
echo "Make sure to protect and guard your private key and passphrase at all costs!
" >> ./${username}/account_info.txt
```

Step 5: Generate RSA keys, and check for errors

In order to give user SSH access to our server we need to generate an RSA key pair (public and private key). We will need to give the private key, username, and passphrase to the user. The public key will need to be stored in the /home/\${username}/.ssh/authorized_keys file.

First use the man pages for ssh-keygen command to find out what are the flags you will need:

- Specify a location for the files to be stored (we want key to be stored in ./\${username}/id_rsa)
- Specify a passphrase
- Do not prompt us and do not show output to console since we do not need to see this

Once you have the command with appropriate flags constructed you can again have an error handling if statement as follows. This time we are giving exist status 2 for our script in case ssh-keygen fails:

```
# Did ssh-keygen work? if failed, delete the user account and home directory, also
give exit Status 2
if [ $? -eq 0 ]
then
    echo "RSA key was generated successfully"
else
    echo '!!!!!! RSA Key was not created, ssh-keygen exit status '$?' !!!!!!'
    echo "Undoing the changes made to system so far"
    deluser $username && rm -r /home/$username
```

```
exit 2
fi
```

Step 6: Move the public key to appropriate location

Now make sure to use mv command to move the public key to the user's .ssh/authorized_keys file. You may notice that by default ~/.ssh directory is not created. We can create .ssh directory in /etc/skel since adduser will copy that directory structure to every new user home directory. Since we may run this script on many systems, we cannot be sure the .ssh directory exists in /etc/skel in every system.

However, we can deal with this in our own script without having to depend on /etc/skel:

1. Check to see if /home/\${username}/.ssh directory doesn't exist, create the directory with appropriate permissions
2. Then move the public key to /home/\${username}/.ssh/authorized_keys file

NOTE: Don't forget to use chmod command to give ownership of these files to the user and its group every time you create a file under the user's home directory. Otherwise the files are going to belong to root:root and will prevent user from accessing these files.

Testing your script and cleanup

You can simply run your script to see if it works. You can also run individual lines or even multiple lines of commands in your shell to make sure your syntax is correct. However, chances are you have to do a lot of trial and error to find all the bugs and to make sure the script is working properly. That may leave your system with a lot of user accounts and a ton of garbage files under /home.

You can undo the changes by running the commands manually each time. If your script did something that was unintended you may have to undo the changes manually yourself. On the other hand, you can create another script such as the following to clean up your system and to undo the changes that have been made by your main script:

```
#!/bin/bash
# This script is to undo the changes made to the system when using the assignment 8
script to create user accounts, and giving them SSH access
echo "What was the username for the account you need to remove? "
read username
deluser $username
rm -r /home/$username
rm -r ./ $username
```

Assignment submission

Please name your script according to the formula: create_account_<studentID>.sh

NOTE: Make sure to follow the naming scheme exactly since I may be using a script to test your file and its function!

Submit your script file via Blackboard as an attachment (NOT in comments or text submission).