User Management
Lab 4: User Management
East Tennessee State University

CSCI 4417/5417

Spring 2016

Jack Ramsey, Lecturer

DUE – 3/1 by 2:45 pm to D2L

CSCI 4417 System Administration

User Management

Purpose

To explore the process of adding users, deleting users, and elevating privileges on Windows and Ubuntu.

Required

1) AWS Server 2012 R2 Instance
2) AWS Ubuntu 14.04 Instance
3) Provided files in Lab4.zip
    a. museradd.sh
    b. users.txt
    c. users.csv

Hand-in

Please write up this lab in accordance to the Lab Manual on D2L. Remember to take an adequate number of screen captures and explain what is being demonstrated. It would be prudent to take your screen shots (the Snipping Tools works very well for this) while you progress through the lab and save them to your external drive. I usually use '00x.jpg' as my naming policy, 001.jpg, 002.jpg, 003.jpg, and so on, so I can easily determine what order the images need to appear in my report. This week there is also a scripting assignment. Your Dropbox submission should include both your completed report and the assigned bash script

Steps

Windows PowerShell

1. Start your Windows DC and Ubuntu instances
2. We've already added a user to our instance using the GUI, so let's try something new
3. Launch PowerShell -  - and ensure that you're running as Administrator (The title bar will read: 'Administrator: Windows PowerShell')
4. One thing you should do when you first run PowerShell is update the help files. You can accomplish this easily enough by entering: update-help (all commands in PowerShell, referred to as 'cmdlets', use a verb-noun format. The help files are extensive and provide examples. You can often figure out how to perform a given operation simply from the help files). PowerShell also includes auto-complete functionality, so you can, for example type 'Show-pr' and hit the Tab key. If more than one command (cmdlet) matches what you've typed in so far, hitting Tab again will cycle through the matches. The interface is case-insensitive, so 'get-help' and 'Get-Help'

both work, though using auto-complete will capitalize using Microsoft's naming convention

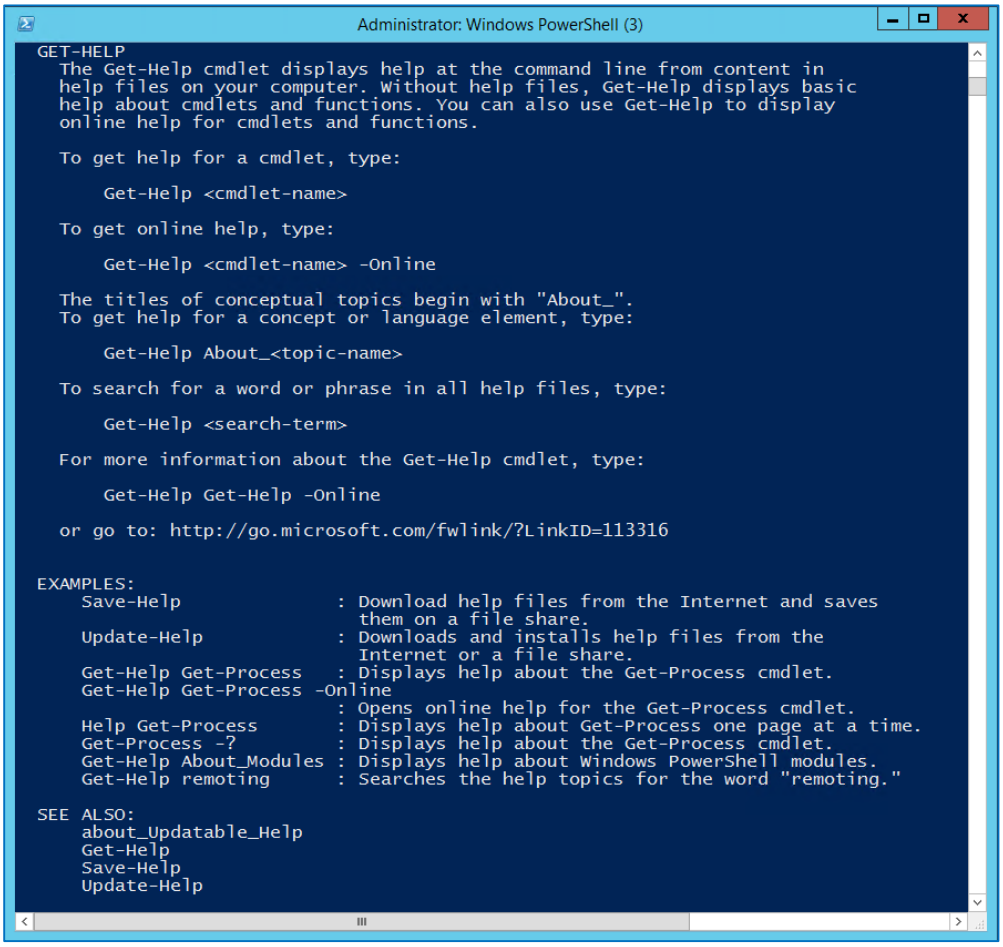

Figure 1: Example output from the Get-Help cmdlet

5. One example of cmdlets in action:
    a. Start Notepad
    b. In your PowerShell window, enter 'Get-Process notep*'
    c. Similar to *top* in Ubuntu, this will display the process information for notepad
    d. Now enter 'Stop-Process' and the 'Id' value displayed for notepad from the previous cmdlet
    e. What happens?
    f. Yeah, I tried 'Stop-Process notepad' too. Didn't work for me, either

PowerShell Integrated Scripting Environment

6. Now for our first script. Right-click on the PowerShell icon on the taskbar. Select 'Run ISE as Administrator.' This will launch the PowerShell Integrated Scripting Environment. If you see this:

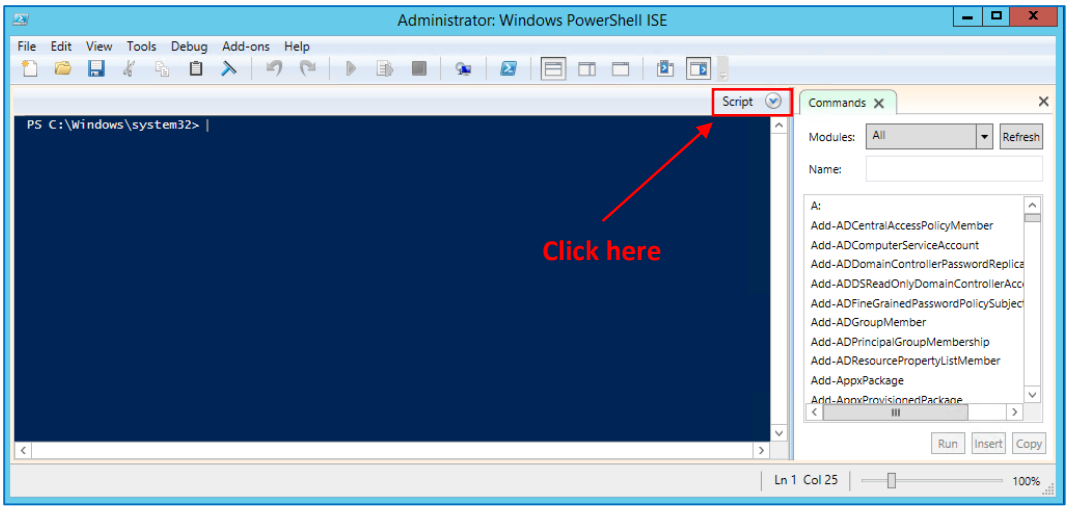

Figure 2: Using PowerShell ISE

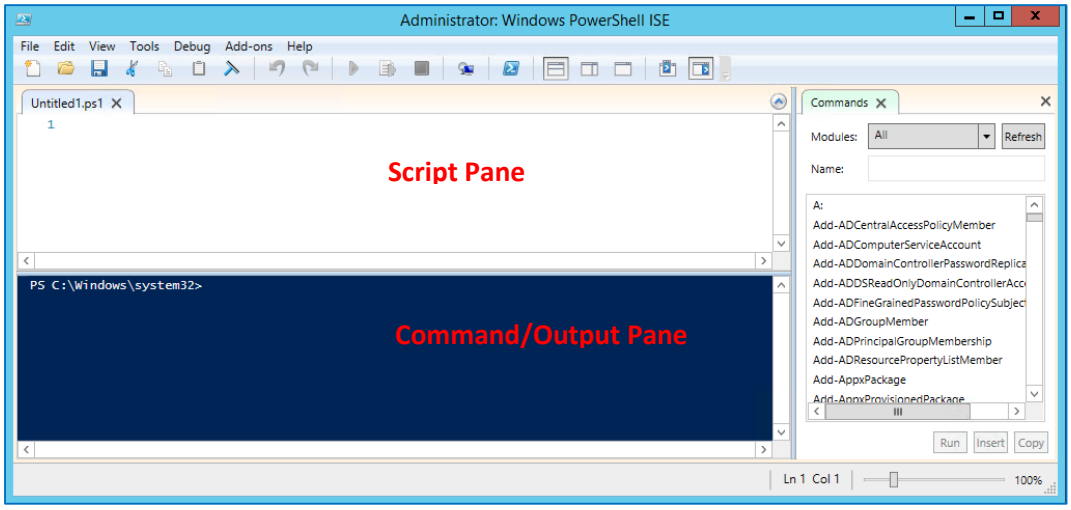

Figure 3: PowerShell ISE panes

Adding Multiple Users

7. Enter the following in the script pane. (Note the backtick - ` - at the end of the first two lines. This allows us to break a long series of piped commands into several lines for readability. The backtick is located in the upper left of your keyboard, under the tilde - ~.):

```
Import-Csv users.csv | New-ADUser -PassThru | Set-ADAccountPassword `
-Reset -NewPassword (ConvertTo-SecureString -AsPlainText 'Passw0rd!' `
-Force) -PassThru | Enable-ADAccount
```

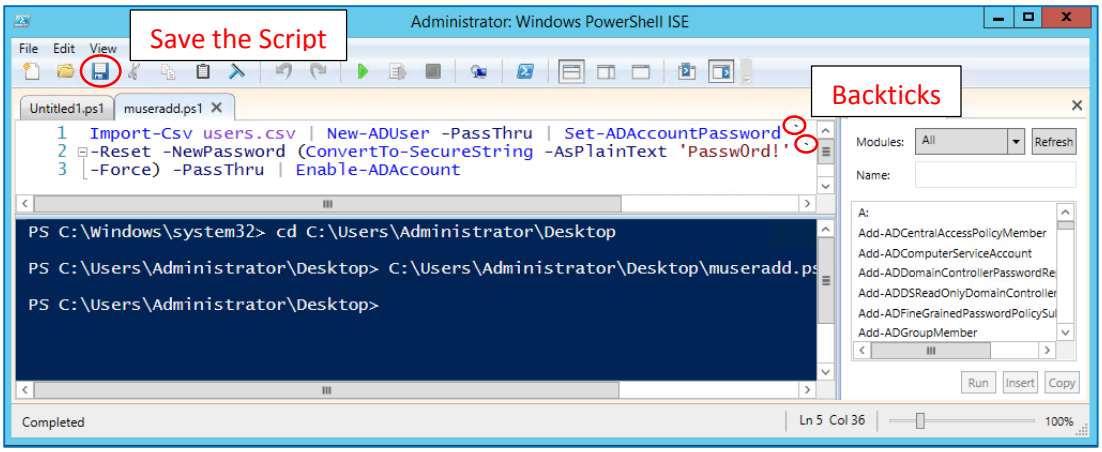

Figure 4: museradd.ps1 script

So what's going on here?

- Import-Csv users.csv - Imports the contents of users.csv as an .NET object
- | - Pipes the object to the next cmdlet
- `New-AdUser` - Runs the command to add a user to Active Directory
- `-PassThru` - Passes the new user object back to the shell for the next cmdlet
- | - Pipes to the next cmdlet
- `Set-ADAccountPassword -Reset` - Forces the user to change password at next login
- `-NewPassword (ConvertTo-SecureString -AsPlainText 'Passw0rd!')` - Encrypts 'Passw0rd!' and sets it to be the new user's password
- `-Force` - Suppresses the "Are you sure you want to do this? [y/N]' prompt
- `-PassThru` - Again passes the user Object back in to the shell
- | - Pipes the result to the next cmdlet
- `Enable-ADAccount` - By default, New-ADUser creates accounts in a disabled state. This will enable the new account

Pipes in PowerShell work similarly to those in Linux, except that instead of passing a stream of output downstream, they pass .NET objects to the downstream cmdlet

8. In the Command pane, change directory (cd) to C:\Users\Administrator\Desktop. Notice how, as you type, a drop-down appears offering relevant matches for what you've typed so far. Once the directory (in this case) is highlighted, you can save yourself some typing by pressing Enter
9. Save the script to your Desktop and name it 'museradd.ps1'
10. Open the file 'users.csv' in Microsoft Excel on your local machine. It is a comma separated values file, so you may have to change the file type to All Files to see it. Using the Replace function, replace all instances of 'ramsey.loc' with your fully qualified domain name (FQDN, i.e., *lastname*.loc)

	A	B	C	D	E
1	Name	GivenName	Surname	SamAccountName	UserPrincipalName
2	Harvey Find	Harvey	Find	findh	findh@ramsey.loc
3	Ivan Peg	Ivan	Peg	pegi	pegi@ramsey.loc
4	Florence Rind	Florence	Rind	rindf	rindf@ramsey.loc
5	Julius Mend	Julius	Mend	mendj	mendj@ramsey.loc
6	Konrad Ping	Konrad	Ping	pingk	pingk@ramsey.loc
7	Alvin York	Alvin	York	yorka	yorka@ramsey.loc
8	Frances Leonard	Frances	Leonard	leonardf	leonardf@ramsey.loc
9	Peggy Potter	Peggy	Potter	potterp	potterp@ramsey.loc
10	Amy Pittman	Amy	Pittman	pittmana	pittmana@ramsey.loc
11	Homer Schneider	Homer	Schneider	schneiderh	schneiderh@ramsey.loc
12	Travis Spencer	Travis	Spencer	spencert	spencert@ramsey.loc
13	Raymond Barnett	Raymond	Barnett	barnettr	barnettr@ramsey.loc
14	Faye Mckenzie	Faye	Mckenzie	mckenzief	mckenzief@ramsey.loc
15	Darrin Mcdaniel	Darrin	Mcdaniel	mcdanield	mcdanield@ramsey.loc
16	Mona Bailey	Mona	Bailey	baileym	baileym@ramsey.loc
17	Ida Washington	Ida	Washington	washingtoni	washingtoni@ramsey.loc
18	Maggie Tate	Maggie	Tate	tatem	tatem@ramsey.loc
19	Stacey Chambers	Stacey	Chambers	chamberss	chamberss@ramsey.loc
20	Lucia Kennedy	Lucia	Kennedy	kennedyl	kennedyl@ramsey.loc

Figure 5: Modify 'users.csv' with your FQDN.
Note what information is required by Windows

The **samAccountName is the User Logon Name in Pre-Windows 2000 (this does not mean samAccountName is not being used as Logon Name in modern windows systems). The **userPrincipalName** is a new [as if a decade and a half can be regarded as 'new' -Jack] way of User Logon Name from Windows 2000 and later versions. user Name part can be different for the same user like DomainName\testUser and userTest@DomainName.Com. More…*

11. Save your changes, the copy the file 'users.csv' to your instance's Desktop
12. First, we have to change the execution policy. Microsoft created four PowerShell execution policies to help administrators follow basic scripting security rules and prevent them from unintentionally violating these rules.
    PowerShell's four execution policies are:
    a. *Restricted.* This default execution policy applies to all Windows versions, with the exception of Windows Server 2012 R2*. Restricted is the most secure policy because it allows PowerShell to operate only in an interactive mode. This means that you can only run individual commands. You can't run scripts under this policy, regardless of where the scripts came from (local or downloaded) and whether they're signed.
    b. *AllSigned.* The AllSigned policy allows scripts to be run as long as they've been digitally signed by a trusted publisher. With this policy, when you attempt to run a signed script, you'll receive a prompt asking you to confirm that you trust the publisher.
    c. *RemoteSigned.* The RemoteSigned policy allows scripts to be run but requires that downloaded scripts have a digital signature from a trusted publisher. Scripts that you run from the local computer don't need to be signed. Under this policy, there are no prompts when you attempt to run a script. RemoteSigned is the default execution policy in Server 2012 R2.
    d. *Unrestricted.* The Unrestricted policy allows all scripts to be run, regardless of whether they're signed and where they come from.

Note: Execution policy does not offer complete security. There are ways around it

In the PowerShell CLI you opened first (you can do it in the ISE, but I didn't feel like having to take a

bunch of new screen shots) Enter

```
Get-ExecutionPolicy
```

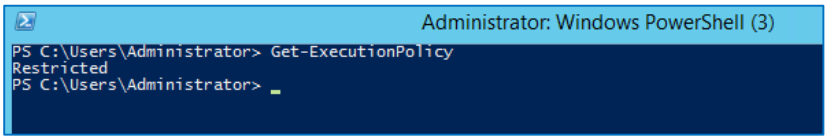

*Figure 6: * In spite of what the above explanation says, on AWS Windows Server 2012 R2 machines, at least, the default execution policy is actually Restricted*

So, enter

```
Set-ExecutionPolicy RemoteSigned
```

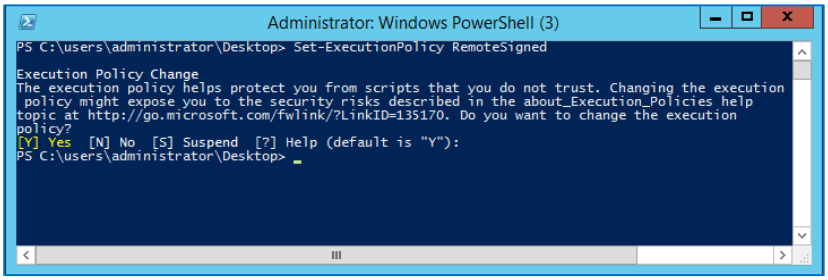

Figure 7: Setting the execution policy

13. Return to the PowerShell ISE and click the Run Script () button
14. Open the Server Manger
15. Click on Tools -> Active Directory Users and Computers
16. Click on 'Type' to arrange the users and security groups by type

17. Scroll down to see your newly added users

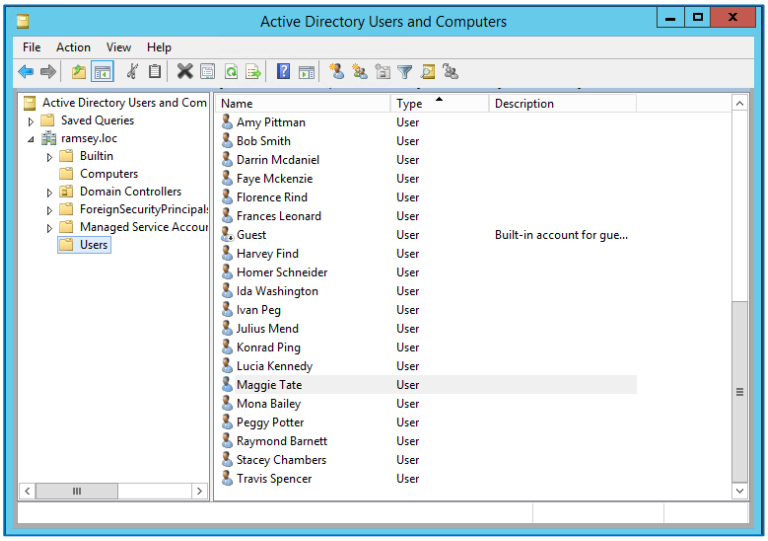

Figure 8: AD DS Users and Computers - User List

18. While we're in AD DS Users and Computers, let's take a look at Security Groups. Scroll up until you find Domain Admins and double-click on it. What are some of the property (groupings) that are available. Without clicking on 'Members,' who might you expect to be members of the Domain Admins group? (Hint: at this point, there are two).

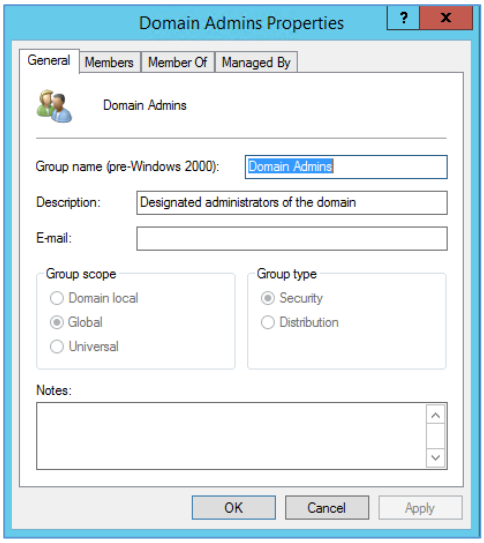

Figure 9: Domain Admins Security Group

19. So let's make a new Security Group. Right-click on Users in the left pane. Hover over New. Select Group

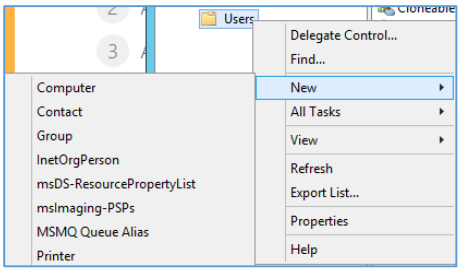

Figure 10: Creating a new security group

20. Give your new group the name 'IT Ops'. Notice the scopes that are available, but leave 'Global' selected. Remember the difference between the three? What is the difference between a Security Group and a Distribution Group?
21. Right-click on your new IT Ops group. Notice how groups can be nested ('Add to a group…'). Click on 'All Tasks'. Notice that we can (if we have a mail server set up) send an email to the members of the group.
22. Lets add good ole' Bob to the IT Ops group. Click on Properties, then the Members tab. Click on Add
23. Click on Advanced and then, Find Now. Notice that the search results include all users and groups. Click Cancel
24. In the search window, type Bob, then click OK
25. Select Bob and click Add
26. Confirm that our IT department membership status now stands at one by again right-clicking IT Ops, selecting Properties, and clicking on the Members tab
27. Click OK

Another way to create PowerShell Scripts

28. Let's try another way to create PowerShell scripts
29. Open Notepad. Enter the following:

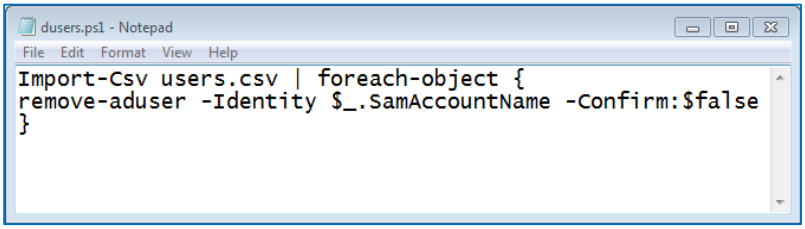

Figure 11: Text for dusers.ps1

30. Again, we're reading in the 'users.csv' file and piping its contents to the 'foreach-object' loop, which invokes the 'remove-aduser' command to delete each AD user, based on his or her account name. The '-Confirm:$false' switch eliminates the need to confirm each deletion.
31. Save the file to your Desktop, naming it 'dusers.ps1'
32. In the PowerShell CLI (not the ISE) window, enter '.\dusers.ps1'

33. If your present working directory (pwd) isn't C:\Users\Administrator\Desktop, cd to it
34. Enter 'Get-ADUser -Filter * > currentusers.txt'.
35. Open 'currentusers.txt' (it should have been added to the Desktop following the last step) in Notepad and confirm that your list of users has been deleted.
36. Notice that there is a user account named 'krbtgt'. What is this used for? Do you think it would be safe to delete it, assuming AD DS will allow you to? (Hint: here's one of the many places where Google is your best friend)
37. Close all open windows
38. Stop the instance. Return to the EC2 console and stop it there as well

Ubuntu

1. Log in to your Ubuntu instance via PuTTY
2. Use the `adduser` command to add a new user named `alice`. Make the password `Passw0rd1!` and her full name 'Alice Liddel' (You can enter office #, phone, etc. information if you want. I usually just hit Enter to click through it, leaving them blank)
3. Create a new user named bob (`sudo adduser bob`). Use the `Passw0rd1!` password when prompted. Use 'Bob Thomas' for the account's full name
4. Add bob to the sudoers group (`sudo adduser bob sudo` or `sudo usermod -G sudo` bob). I know I didn't mention the first option in lecture last week. I forgot. But these two options are a lot less complex than visudo
5. Switch user to `alice` (`sudo su alice`). Attempt to switch user alice to root (`sudo su`). What happens?
6. Return to your **ubuntu** user (`exit`) and switch user to **bob**. Again attempt to switch user **bob** to root. What happens? (return to the **ubuntu** account (`exit exit`)
7. Enter `cat /etc/passwd | egrep "1[0-9]{3}"` (The regular expression matches all numbers from 1000 to 1999. Ubuntu starts user ids at 1000 by default). Note that this displays the user accounts on the system, which should be you, alice, and bob
8. Type **`clear`** and press **Enter**.
9. Since we don't have the luxury of a GUI with our Ubuntu servers, we're going to have to use a new tool to copy 'museradd.sh' and 'users.txt' to it.
10. First, on PuTTY, create a 'scripts' folder under your ~/ (user ubuntu's home) directory

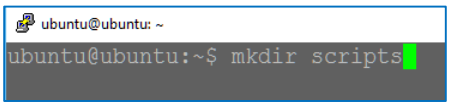

Figure 12: Creating a 'scripts' folder

11. We're going to use a CLI application from the PuTTY suite to upload our scripts and users file to the Ubuntu instance. We'll use our 4417key.ppk key to gain access, just like we do with PuTTY
12. On your local computer, copy museradd.sh, 4417key.ppk (if it isn't already there), and users.txt to your external drive's System Administration folder
13. Now, on your local computer, open up a command prompt (cmd.exe)

14. Change directory to C:\"Program Files <x86>"\Putty (using the Tab key here will help fill in the correct folder names)
15. Enter the following at the command line:

    `psftp -i D:\CSCI4417\4417key.ppk` *`ip-address`*

    Where
    - psftp: PuTTY Secure File Transfer Protocol
    - -i: Authenticate with a private key
    - D:\CSCI4417\4417key.ppk: Path to the key
    - ip-address: The public IP address of *<u>your</u>* Ubuntu instance (not the one in the screen shot below)

```
c:\Program Files (x86)\PuTTY>psftp -i D:\CSCI4417\4417key.ppk 54.84.12.157
login as: ubuntu
Remote working directory is /home/ubuntu
```

Figure 13: Log in to Ubuntu instance using private key (be sure to use the public IP address for your instance)

16. Log in as user 'ubuntu'
17. Type 'help'

```
psftp> help
!       run a local command
bye     finish your SFTP session
cd      change your remote working directory
chmod   change file permissions and modes
close   finish your SFTP session but do not quit PSFTP
del     delete files on the remote server
dir     list remote files
exit    finish your SFTP session
get     download a file from the server to your local machine
help    give help
lcd     change local working directory
lpwd    print local working directory
ls      list remote files
mget    download multiple files at once
mkdir   create directories on the remote server
mput    upload multiple files at once
mv      move or rename file(s) on the remote server
open    connect to a host
put     upload a file from your local machine to the server
pwd     print your remote working directory
quit    finish your SFTP session
reget   continue downloading files
ren     move or rename file(s) on the remote server
reput   continue uploading files
rm      delete files on the remote server
rmdir   remove directories on the remote server
psftp>
```

Figure 14: Available commands in psftp

18. Notice that commands like **`cd`** are executed on the remote instance, while **`lcd`** will be executed on the local machine. You can execute local CLI commands, such as **`cls`** (clear screen) by prefacing them with an exclamation mark, i.e., **`!cls`**
19. Enter `cd scripts`
20. Enter

    ```
    put D:\CSCI4417\useradd.sh
    put D:\CSCI4417\users.txt
    ```

    Where:
    - put uploads a file to the current (remote) directory

- D:\CSCI4417 is the path to your files
- museradd.sh is your shell file
- users.txt is the text file containing new user information

21. Type **ls**
22. Is the shell file executable? How can you tell?
23. Interestingly, psftp doesn't support alphabetical arguments for **chmod** (e.g., +x). So we'll have to use octal representation to change the shell files' modes

    Enter

    ```
    chmod 775 museradd.sh
    ```

```
psftp> chmod 775 museradd.sh
/home/ubuntu/scripts/museradd.sh: 0664 -> 0775
psftp> ls
Listing directory /home/ubuntu/scripts
drwxrwxr-x    2 ubuntu   ubuntu       4096 Feb 23 12:42 .
drwxr-xr-x    7 ubuntu   ubuntu       4096 Feb 16 17:01 ..
-rwxrwxr-x    1 ubuntu   ubuntu         54 Feb 13 20:50 bashTwoH
-rw-rw-r--    1 ubuntu   ubuntu       1804 Feb 23 12:41 dusers.sh
-rwxrwxr-x    1 ubuntu   ubuntu       2662 Feb 23 12:41 museradd.sh
-rw-rw-r--    1 ubuntu   ubuntu        752 Feb 23 12:42 users.txt
psftp>
```

Figure 15: Changing museradd.sh's mode to executable

*** We'll talk about permissions next week, if we don't get snowed out, but 775 gives read/write/execute permission to the file owner; read/write/execute permission to the file's group; and read/execute permission to everybody else*

24. Return to your Ubuntu instance. By now, PuTTY is probably dead. If so, right-click on the title bar and select 'Duplicate Session.' Then log in again as 'ubuntu' and change directory to ~/scripts
25. Enter **ll** (that's ell-ell, not eleven) to confirm that museradd.sh is present and executable
26. Enter **echo $PATH**. This will display the path that the kernel follows when you enter a command to try to find that command. Note that /home/ubuntu/scripts is not on the path
27. Remember that to add users, you have to have root privileges, so enter

    ```
    sudo ./museradd.sh users.txt
    ```

(If a script is in a folder that isn't on the machine's path, you have to prepend its path to its name before you can run it. './' means 'this directory')

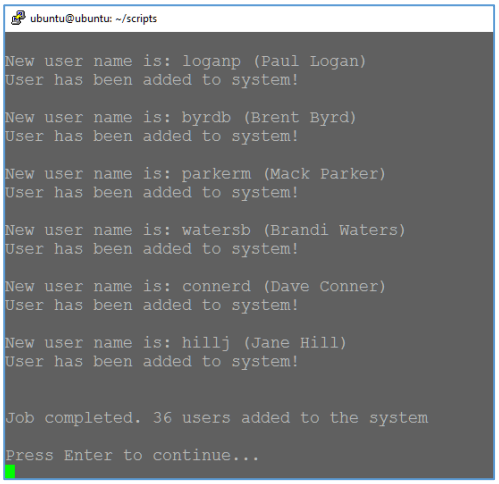

Figure 16: Output from museradd.sh

28. The script should add all of the user names in 'users.txt' to the system. Verify with:

```
cat /etc/passwd | egrep -o "[A-Za-z]+ [A-Za-z]+"
```

Where

- cat will display the contents of a file
- /etc/passwd contains information about all user accounts
- | pipes the output to egrep (extended grep)
- -o selects only the matched pattern
- "[A-Za-z]+ [A-Za-z]+" is a regular expression that matches a pattern of any number (the

'+') of upper- or lower-case letters followed by a space followed by another set of upper- or lower-case letters. Remember, each field in /etc/passwd is delimited by a colon (:)

Figure 17: Output from the above command

29. Your assignment this week, in addition to your report, is to modify 'museradd.sh' so that it will delete users from a system based on an input text file (You can use users.txt to delete all of the users you added, or edit it and delete a few of the entries to delete some users and leave others). Name the file dusers.sh and execute it. It should include feedback about each account that is deleted (see below). If you're unsure about how to proceed, Chapter 2 of your text and Google are good resources. Make sure to document (comment) your code adequately. Include a screen shot with your report, showing successful output from the script. Remember to upload your script to D2L along with your lab report

```
ubuntu@ubuntu: ~/scripts
User frazierm deleted
Deleting account mcbrides (Shelia Mcbride)
User mcbrides deleted
Deleting account colonnw (Willie Colonnade)
User colonnw deleted
Deleting account clarkeb (Bertha Clarke)
User clarkeb deleted
Deleting account delgadon (Nettie Delgado)
User delgadon deleted
Deleting account tuckerw (William Tucker)
User tuckerw deleted
Deleting account loganp (Paul Logan)
User loganp deleted
Deleting account byrdb (Brent Byrd)
User byrdb deleted
Deleting account parkerm (Mack Parker)
User parkerm deleted
Deleting account watersb (Brandi Waters)
User watersb deleted
Deleting account connerd (Dave Conner)
User connerd deleted
Deleting account hillj (Jane Hill)
User hillj deleted
ubuntu@ubuntu:~/scripts$
```

Figure 18: Output from dusers.sh

REMEMBER TO SHUT DOWN YOUR INSTANCES WHEN YOU ARE DONE!