# OPS245 Lab 4

## Contents

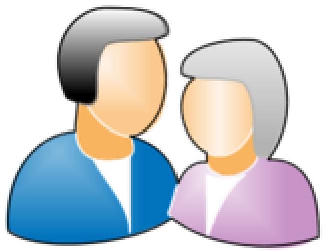
# LAB PREPARATION

## Purpose / Objectives of Lab 4

There are many other tasks that a Linux system administrator must perform other than installing Linux and installing software.

User account management is a very important operation that a Linux sysadmin does on an continual basis. The sysadmin not only needs to add or remove user accounts by issuing commands, but may need to automate user account creations a large number (batch) of potential employees. There are many features with the Linux command to create new users including: specification of a home directory, type of shell used, name, password and time-limit (referred to as "aging") for a new user account. Removing user accounts also have options such as removing the user account but keeping the home directory for reference or evidence of "wrong-doing"

Another important operation for a Linux sysadmin is to manage services (eg. starting, restarting, stopping, disabling, enabling system services). Many students may think that the following topic is small and "not a big deal". Those students may say, **"How hard is running and stopping services?"**


System administrators are required to add, remove and modify user accounts.

Linux system administrators need to know how to stop / disable Linux services.

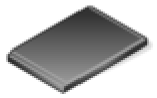
Linux system administrators need to know how to start / enable Linux services.

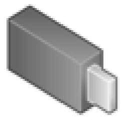
The process may not be hard, but knowing how to stop, start, restart and check the status of services is absolutely critical to a Linux server. **Aside from learning to trouble-shoot problems** by checking the status of running services, **understanding how to manage services is critical to help protect a Linux server from penetration** (this term is referred to as "**Hardening a system**"). Sometimes it is "what we don't know" that can harm us. One key element in hardening a computer system is to disable non essential networkng services to allow IDSs (**Intrusion Detection Systems**) to focus on a narrower range of policy violations. A Debian-based penetration testing distribution called **Kali** (formerly referred to as **"BackTrax"**) allows sysadmins and security professionals to identify vulnerabilities in their computer systems, and thus improve (harden) their systems against penetration. Learning to monitor the status, enable and disable networking services underlies the **Backtrax** motto: ***"The quieter you are, then more you will hear..."***

<u>Main Objectives</u>:

- Administer **(add, remove, modify) users** on a Linux system.
- Save time while adding new users using a template of **start-up files**.
- Create and manage **groups** on a Linux system.
- **Start, Restart and Stop services** on a Linux system.
- **Enabled and Disable services** on a Linux system.
- Display the **status of running services** on a Linux system.
- Create a Bash shell script to **generate multiple user accounts** from a text user database file

**Minimum Required Materials** **Linux Command Reference**


**Solid State Drive**


**USB key**
(for backups)

**Lab4 Log Book**

User Management

useradd (https://www.systutorials.com/docs/linux/man/8-useradd/)
userdel (https://www.systutorials.com/docs/linux/man/8-userdel/)
usermod (https://www.systutorials.com/docs/linux/man/8-usermod/)
groupadd (https://www.systutorials.com/docs/linux/man/8-groupadd/)
groupdel (https://www.systutorials.com/docs/linux/man/8-groupdel/)
chage (http://www.agr.unideb.hu/~agocs/informatics/11_e_unix/unixhelp/unixhelp.ed.ac.uk/CGI/man-cgied74.html?chage)

Managing Services
systemctl (http://www.dsm.fordham.edu/cgi-bin/man-cgi.pl?topic=systemctl)

Miscellaneous

/etc/passwd (http://man7.org/linux/man-pages/man5/passwd.5.html)
/etc/group (http://man7.org/linux/man-pages/man5/group.5.html)
/etc/shadow (http://man7.org/linux/man-pages/man5/shadow.5.html)
/etc/skel (http://archive.linuxfromscratch.org/blfs-museum/1.0/BLFS-1.0/postlfs/skel.html)
init vs systemd (http://zenit.senecac.on.ca/wiki/index.php/Init_vs_systemd)

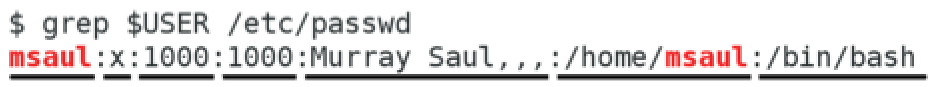
Python Reference
argparse (https://docs.python.org/3/howto/argparse.html)

## INVESTIGATION 1: User/Group Management

In your ULI101 course, you learned to change permissions for directories and files relating to user, same group members and other group members. In this course, since you are the sysadmin with root privileges, you can create or remove users and groups as well as change the ownership of directories and files! We will now learn to perform key user account management operations in this section.

## Part 1: Studying the /etc/passwd file

The /etc/passwd file is a database that stores user accounts (both system and regular users). Since we will be learning to create, modify and remove users on our Linux system, we should study this file in order to understand how those user account management commands will affect this file.



```
$ grep $USER /etc/passwd
msaul:x:1000:1000:Murray Saul,,,:/home/msaul:/bin/bash
```

The /etc/passwd file contains records for users (system accounts and regular-user accounts). There are several fields including: **Username**, **password link** ("x" indicates hashed password stored in /etc/shadow file only accessible by root), **user-id**, **primary group-id**, **user defined field** (eg. Full Name), **default home directory**, and **default shell**.

**Perform the following steps:**

1. Launch your **c7host** and **centos1** VMs.
2. Switch to your **centos1** VM.
3. Open a shell terminal.
4. Look at the `/etc/passwd` file.
5. Make note of the contents of that file.
6. Read about the file: http://man7.org/linux/man-pages/man5/passwd.5.html
7. Make sure you know what information each field contains.
8. Why do you think there are so many users?
9. Look at the names of the users. What do you think these user names represent? Are they people?
10. What is the numeric user ID (UID) of the root user?
11. The user IDs of real users (people) are different from the user IDs of system accounts. What is the pattern?

## Part 2: Adding, Removing, and Modifying Users

In this section, we will now learn how to properly add user accounts, remove user accounts, and modify existing user account information.

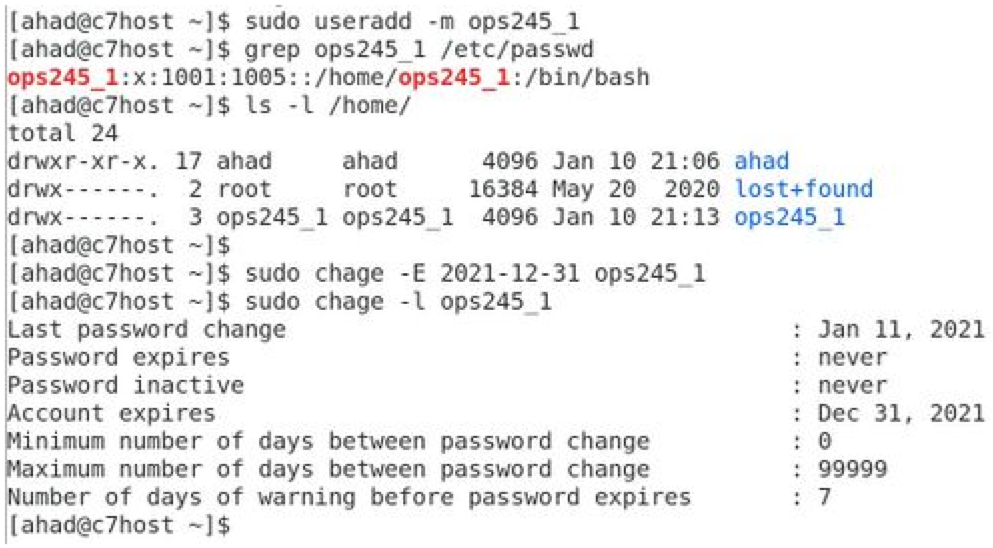
**Perform the following steps:**

1. Remain in your **centos1** VM for this section.
2. Read the man page for the `useradd` command.

3. Create a new user called **ops245_1** by issuing the command:
   `sudo useradd ops245_1`
4. Issue the command: `grep ops245_1 /etc/passwd` to see if that user account was created.
5. View the **/home** directory to view the contents. Is the user ops245_1's home directory there?

   **NOTE:** In some versions of Linux, you may have to issue the **-m** option with the useradd command in order to create a home directory for that user.

6. Issue the following command to create the user called ops245_2:
   `sudo useradd -m ops245_2`
7. View the **/home** directory to verify that the home directory for **ops245_2** has been created. What does the -m option do for the useradd command?
8. Issue the following command to remove the user called ops245_2: `sudo userdel ops245_2`
9. Issue the grep command with the /etc/passwd file to verify that the username ops245_2 was removed.



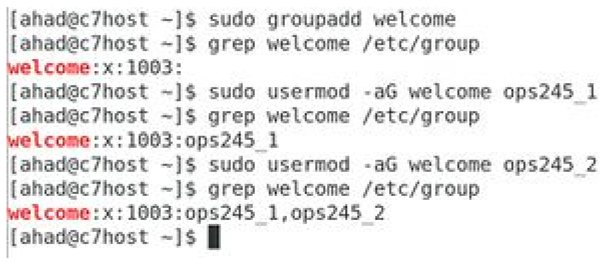
```
[ahad@c7host ~]$ sudo useradd -m ops245_1
[ahad@c7host ~]$ grep ops245_1 /etc/passwd
ops245_1:x:1001:1005::/home/ops245_1:/bin/bash
[ahad@c7host ~]$ ls -l /home/
total 24
drwxr-xr-x. 17 ahad     ahad      4096 Jan 10 21:06 ahad
drwx------.  2 root     root     16384 May 20  2020 lost+found
drwx------.  3 ops245_1 ops245_1  4096 Jan 10 21:13 ops245_1
[ahad@c7host ~]$
[ahad@c7host ~]$ sudo chage -E 2021-12-31 ops245_1
[ahad@c7host ~]$ sudo chage -l ops245_1
Last password change                                    : Jan 11, 2021
Password expires                                        : never
Password inactive                                       : never
Account expires                                         : Dec 31, 2021
Minimum number of days between password change          : 0
Maximum number of days between password change          : 99999
Number of days of warning before password expires       : 7
[ahad@c7host ~]$
```

Another essential responsibility for Linux system administrators is to **add**, **modify**, and **remove** user accounts.

10. View the contents of the **/home** directory. Was the home directory for user **ops245_2** removed?
11. Issue the following command to remove ops245_2's home directory: `sudo rm -rf /home/ops245_2`
12. Issue the **userdel** command to remove the **ops245_1** account, but this time include the **-r option** to also remove the home directory.
13. Issue the useradd -m command to recreate the user called: **ops245_1**.
14. Use the **passwd** command to set the password for the user **ops245_1**.
15. View the contents for **ops245_1's** home directory and note the files.
16. Create a new file in the **/etc/skel** directory with the following command: `sudo touch /etc/skel/foo`
17. Recreate the new user (with home directories automatically created) for **ops245_2**.
18. Set the password for the user **ops245_2**.
19. View the contents for **ops245_2's home directory** and note the files. What do you notice that is different. What do you think is the purpose of the **/etc/skel** directory?
20. Be sure to record your observations in your lab notes.
21. Look in the man pages for the **useradd** command. Explain the purpose of using the **-e** option for the *useradd* command.
22. Issue the following command: `sudo chage -E 2021-12-31 ops245_1`
23. Issue the following command: `sudo usermod -c "New Name" ops245_2`.
24. View ops245_2's account information in the **/etc/passwd** file. What do you notice is different?
25. Issue the following command to obtain information regarding the user called ops245_1: `sudo chage -l ops245_1`. What do you think is the purpose of the chage command and the useradd command with the **-e** option?

## Part 3: Managing Groups

In this section, we will learn how to create, remove and modify groups in our Linux VM. You learned in ULI101 how to change permissions with the **chmod** command, but you didn't have admin privileges to **create groups** to allow directory and regular file sharing. Since you now have admin privileges with your VM, you can now create groups, and add users to this group to allow file-sharing among users.



```
[ahad@c7host ~]$ sudo groupadd welcome
[ahad@c7host ~]$ grep welcome /etc/group
welcome:x:1003:
[ahad@c7host ~]$ sudo usermod -aG welcome ops245_1
[ahad@c7host ~]$ grep welcome /etc/group
welcome:x:1003:ops245_1
[ahad@c7host ~]$ sudo usermod -aG welcome ops245_2
[ahad@c7host ~]$ grep welcome /etc/group
welcome:x:1003:ops245_1,ops245_2
[ahad@c7host ~]$ █
```

When a file is created it is assigned an owner and a default (primary) group. The Linux system administrator can not only change a file's ownership, but also change the default group that file belongs to. In addition, the sysadmin can also add other users to a supplementary group that they have created via the **usermod** command. This is useful in setting "same group" permissions via the **chmod** command.

**Perform the following steps:**

1. Make certain that you are still in your **centos1** VM.
2. Close all application windows, and switch user accounts (within your centos1 VM) by clicking on the top right-hand side of the screen (power icon), click **your regular username**, click **switch user**, and login as **"New Name"** (i.e. ops245_2).
3. Open a shell terminal.
4. Create a file called **information.txt** in home directory of that user.
5. Issue the following command: `ls -l information.txt`. Who owns that file? What primary group does that file belong to?
6. Issue the following command to create a group called **welcome**: `sudo groupadd welcome`
7. You'll notice that sudo does not work for this user, because we haven't given them permissions to run any commands with elevated privileges.
8. Log out, and log back in as your normal user.
9. Now re-issue the command to create **welcome** group. This time, sudo should work.
10. Issue the command: `grep welcome /etc/group` file to confirm that the group **welcome** was created.
11. Read the man page for the `usermod` and `groupmod` commands. Which command (and which option) will allow you to set the Group ID number (**GID**) when you create a new group? Which command (and options) allow you to both append and assign users to an existing group name?
12. Issue two separate **usermod** commands to add both **ops245_1** and **ops245_2** to the newly-created **welcome** group.
13. Verify that both ops245_1 and ops245_2 now belong to the **welcome** group.

## Practical Example

Management has sent you (the Linux systadmin) that a "new" employee has been hired and will be on on probation for 3 months. As the Linux system administrator, they want you to perform the following steps:

1. Remain in your **centos1** VM for this section.
2. Use the **useradd** command to create a user account called: **noobie** to expire in 3 months from this date as part of the security policy of this organization (issue man useradd to determine correct option to set expiry date).
3. Set an appropriate password for this user account.

4. Add this newly-created user to the newly-created **welcome** group.
5. Examine `/etc/group` to verify that you made the correct operations.
6. Use the **usermod** command to set the full name of the user account **noobie** to **"Really Green"**. Examine the result of running that command in the `/etc/passwd` file. What has changed?

   Unfortunately, you were later informed that this "**noobie**" employee was caught stealing from the company. They want you to perform the following operations:

7. Remove this account, but keep "noobie's" home directory for police investigation purposes.
8. Verify that you correctly issued the correct commands.

**Answer INVESTIGATION 1 observations / questions in your lab log book.**

# INVESTIGATION 2: Controlling Sudo Elevated Privileges

When you created your first user on your host, centos1 and centos2, you made them an administrator. This allowed them to (when they request it) run commands with root priveleges. But you won't always know in advance if a user is going to be an administrator (what if someone gets promoted, or changes jobs?), so you can't always do that in advance. Checking that box also allows them to run *any* command with root priveleges by using sudo. In many cases, administrators won't be allowed to do *everything*, but instead be restricted to certain tasks (e.g. managing user accounts, managing software, managing services, etc.). Sudo will allow us this detailed control, so we can pick and choose who gets to run which commands as root.

## Part 1: Finding out why Your First User can do Anything

You've already observed that your first user can use sudo to execute any command, but what about their account actually makes that possible?

1. View (but do not edit) the contents of **/etc/sudoers**. Search for your user account. You won't find them.
2. Check the contents of **/etc/passwd** and **/etc/group** for entries with your user account. Is there anything different between your account and **ops245_1**?
3. You should find that your user is part of a secondary group. What group is it? Are they part of that group on **centos3**?
4. The **wheel** group represents administrators with complete sudo privileges. Go back to **/etc/sudoers** and read the entry for **wheel**. It should look something like this:
   ```
   %wheel ALL=(ALL) ALL
   ```

   This means that anyone who is part of that group can run *any* command, as *any* user. Effectively, they can use sudo to be root. So why not just give those admins the root password?

5. During the lecture, you should have learned some reasons to limit access to the actual root account, and why using sudo is a better practice. Record your observations.
6. On centos3, add your user to **wheel** as a secondary group so you can use sudo the same way there that you can on your other machines.

## Part 2: Adding Limited Sudo Capabilities to Other Users

The wheel group is very useful for senior admins who should be able to run any command, but what about admins who haven't demonstrated the responsibility necessary to wield that power yet? We can use the sudoer files to give them priveleges to run some commands, but not all. Note: While this could be done in the main **/etc/sudoers** file, the better practice is to create files for each admin user in the **/etc/sudoers.d** directory.

1. Login as your **ops245_1** account. Try to run the command `systemctl restart sshd`

   If successful, that command would restart the sshd service on that machine, but that user does not have permission to do that.

2. Try running that command again, this time with sudo.
3. It still won't work, because this user does not have permission to use sudo for anything.
4. Log out from **ops245_1** and log back in as your normal user.
5. Create a file called **ops245_1** in **/etc/sudoers.d**. Add the following line to it: `ops245_1 ALL=(ALL) /usr/bin/systemctl`

   This indicates this user can use sudo to run systemctl commands as if they were any account (root is the important one).

6. Log out from your normal user and log back in as **ops245_1**.
7. Try restarting sshd again. This time it should work.
8. Change to your **ops245_2** account, and try restarting sshd (with and without sudo).

   That account still can't. Sudo entries only affect the users and groups listed.

9. We don't want **ops245_2** to manage services, that's a job for **ops245_1**, but we do want them to manage user accounts. So log back in as your regular user and create a sudeors file for **ops245_2** and set it so that they can run the useradd, usermod, userdel, groupadd, groupmod, and groupdel commands through sudo.
10. Test to make sure it works.

# INVESTIGATION 3: Managing System Services and Run-levels

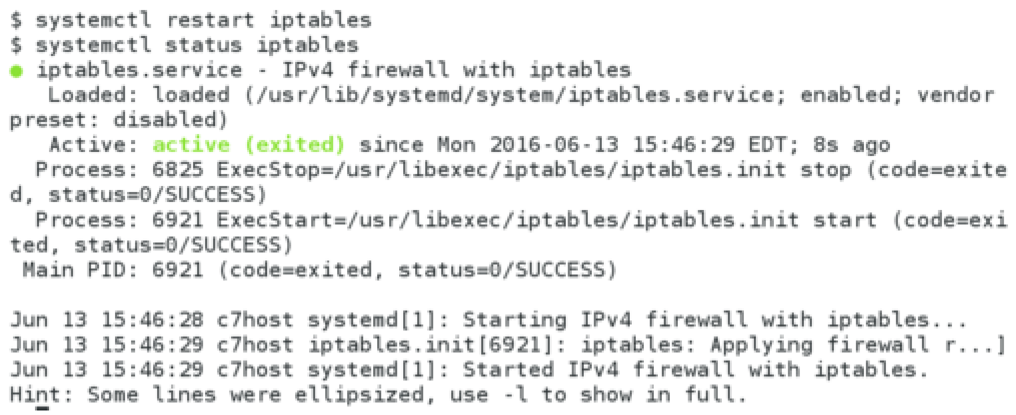
## Part 1: How do we Manage System Services?

At the beginning of this lab we mentioned that running unneeded **packages can be a security risk** due to the unnecessary increase in the complexity of your system. Similarly, it is also unnecessarily hazardous, and even more so, to leave unneeded services running. In this investigation, we will learn how to **control services, and turn off those services that we think are not necessary to help reduce security risks**.

Although there is a command called: **service** that may appear to manage services on your Linux system, it is considered **deprecated** (i.e. "obsolete"). It has been replaced by using the systemctl (http://zenit.senecac.on.ca/wiki/index.php/Init_vs_systemd#systemd_Command_Usage) command.

**Perform the following steps:**

1. Remain in your **centos1** VM for this section.
2. To verify the status of your iptables service, issue the following command: `systemctl status iptables`
3. Use the commands you used in Lab2 to **stop** and **disable** the iptables service.
4. Issue a command to verify you **disabled** and **stopped** the iptables service.



```
$ systemctl restart iptables
$ systemctl status iptables
● iptables.service - IPv4 firewall with iptables
   Loaded: loaded (/usr/lib/systemd/system/iptables.service; enabled; vendor
preset: disabled)
   Active: active (exited) since Mon 2016-06-13 15:46:29 EDT; 8s ago
  Process: 6825 ExecStop=/usr/libexec/iptables/iptables.init stop (code=exite
d, status=0/SUCCESS)
  Process: 6921 ExecStart=/usr/libexec/iptables/iptables.init start (code=exi
ted, status=0/SUCCESS)
 Main PID: 6921 (code=exited, status=0/SUCCESS)

Jun 13 15:46:28 c7host systemd[1]: Starting IPv4 firewall with iptables...
Jun 13 15:46:29 c7host iptables.init[6921]: iptables: Applying firewall r...]
Jun 13 15:46:29 c7host systemd[1]: Started IPv4 firewall with iptables.
Hint: Some lines were ellipsized, use -l to show in full.
```

It is important for a Linux system administrator to be able to start/stop, enable/disable and check the status of services on their Linux server. Students will be commonly performing these operations in their OPS335 course when configuring and troubleshooting network services.

**Note:** There is a major difference between stopping a service and disabling a service: If a service is stopped but enabled, the service will start upon reboot. Therefore to prevent it being started upon boot-up, the service will need to be disabled as well!

5. Issue the commands to **start** and **enable** the iptables service, and **verify** that it is started and enabled.

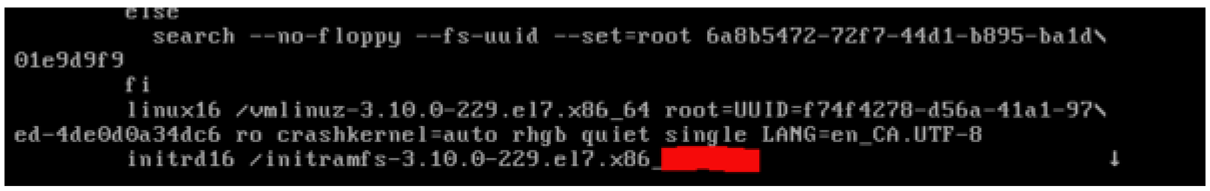
**Note:** If you performed the commands correctly, the iptables service should be running, and will automatically run upon your Linux machine start-up.

## Part 2: How do we Manage Runlevels?

Running Linux servers in graphical mode can make the server vulnerable to penetration (i.e. a potential break-in to the server from unauthorized intruders). The X-windows framework can be vulnerable to attacks when these servers are connected to the Internet. This is why when you install **server versions** of Linux, they work in text-based mode only. Desktop versions of Linux are then installed on workstations (working in graphical mode) that connect to the **Linux server** (for security reasons since those servers are closest to the router and the Internet).

The Linux sysadmin can also change the target (or state) of a graphical Linux server to run in text-based mode and run the graphical mode by issuing a command when graphic mode is required. You may also encounter this capability described as run-levels, but that term is now deprecated in Fedora/RHEL/CentOS.

**Perform the following steps:**



```
        else
          search --no-floppy --fs-uuid --set=root 6a8b5472-72f7-44d1-b895-ba1d\
01e9d9f9
        fi
        linux16 /vmlinuz-3.10.0-229.el7.x86_64 root=UUID=f74f4278-d56a-41a1-97\
ed-4de0d0a34dc6 ro crashkernel=auto rhgb quiet single LANG=en_CA.UTF-8
        initrd16 /initramfs-3.10.0-229.el7.x86_[redacted]                        ↓
```

The purpose of **Linux servers** are to run network-based services (i.e. they **"serve"** the users that operating in that Linux/Unix system). It is common that these Linux servers are separated (for security purposes) and they are **run in Command-Line mode only**. Running these Linux/Unix servers in **Graphics Mode will make them more vulnerable to penetration from hackers, etc.** Therefore, it is common that the Linux servers are CLI only, but the Workstations that connect to them within the network are GUI. Therefore, it is important that a Linux/Unix system administrator understand to switch to these different "run-levels".

1. Remain in your **centos1** VM for this section.
2. Issue the following Linux command:
   `systemctl get-default`

   **Note:** The output should read **graphical.target**
3. Try the same command on your **centos3** VM and observe how the output differs. Go back to your **centos1** VM.
4. You can use the **systemctl isolate** command to change the current target. See a list of targets here (https://www.centos.org/docs/5/html/5.2/Installation_Guide/s2-init-boot-shutdown-rl.html).
5. Change the current target in **centos1** to **multi-user.target** by issuing the following command:
   `sudo systemctl isolate multi-user.target`
6. What did you notice?
7. Reboot your **centos1** VM. It should return to the graphical login screen. You should notice at this point that the command **systemctl isolate** did not change the default target the system will boot to.
8. Issue the `sudo systemctl set-default multi-user.target` command (with elevated permissions) to change the current default target in **centos1** to **multi-user.target**, then reboot your machine. What do you notice?
9. Change the current run-level in **centos1** to **graphical.target** by issuing the following command:
   `sudo systemctl isolate graphical.target`
10. Try to do the same thing to your **centos3** VM. Did it work? Why or why not?
11. Set the default target on your **centos1** VM back to graphical.target before continuing.

**Answer INVESTIGATION 3 observations / questions in your lab log book.**

# INVESTIGATION 4: USING ARGUMENTS IN SHELL SCRIPTS

## Using argparse to Obtain Positional Arguments from the Command Line

In this investigation we will use python's argparse module to make our scripts more automation-capable by reducing (or eliminating) how much interactivity we need from the user.

**Perform the following steps:**

1. You will be using your **c7host** machine for this section.
2. Change to your **bin** directory.
3. Use your **tarchiver.py** (from lab 3) as a command to make a tar archive of /tmp called mytmp.tar.
   You'll notice that even after hitting enter to run the command, you still needed to give more data to your script (to tell it which directory you wanted to archive, what to call it, and what compression to use).
   Requiring this much interaction from the user means that this script is not very good for automation. We can't schedule this script to automatically run, because we (or another admin) need to be present to type answers to the prompts.
4. Make a copy of your tarchiver.py script and call it **tarchiver2.py**. We will work with tarchiver2.py for the rest of this investigation.
5. Import the argparse module into tarchiver2.py.
6. Add the following lines to your script, after the import, but before you prompt the user for anything:
   ```
   parser = argparse.ArgumentParser()
   args = parser.parse_args()
   ```
   This creates an argument parser and makes it read all the command line arguments the user entered. However, we haven't defined any that we expect yet, so all this will do is display a default help message if the user runs our script with -h.
7. Try that now:
   ```
   tarchiver2.py -h
   ```
8. For argparse to be really useful, we need to tell it to expect some command line arguments (and then do something with them).
   Modify your script so the argparse portion of it looks like this:
   ```
   parser = argparse.ArgumentParser()
   parser.add_argument("dest",help="The name you would like to give the archive.")
   args = parser.parse_args()
   ```
   And replace the line where you prompt the user for the destination archive name with:
   ```
   destination = args.dest
   ```
   Note: Instead of **destination**, use the variable name were already using to store the value you were getting from the user. That way you won't have to change it in the rest of your script.
9. Try using your script to make another archived copy of /tmp, this time calling it **secondtmp.tar**.
   If you didn't provide secondtmp.tar on the command line when you ran the command, you'll notice that your script complained. Try running:
   ```
   tarchiver2.py secondtmp.tar
   ```
10. You should still be getting prompted about the directory you want to archive, and whether or not you want compression, but you are now telling the script that the created archive should be called secondtmp.tar.
11. Run the script again, but this time give the archive a different name of your own choice. Your script is part way to being automatable: the user can set the name of the created archive before the script runs. We just need to make this possible for the rest of the required data.
12. Add a second parser.add_argument line to your script so that you can also obtain the name of the directory to archive from the command line. You can choose if it should go before or after the name of the archive. Just remember to use a different argument name, and an appropriate help message.
13. Replace the line in your script that prompts the user for the name of the directory with code that will retrieve the value the user entered on the command line.
14. Run you script to make sure it works.
    You should now be able to enter both the directory to archive, and the name of the resulting archive on the command line, and should only be prompted about compression.

15. All that is left to finish the script is to replace the prompts for compression with command line options. You could do this by adding a third argument and requiring it to include a compression type, or by creating a mutually exclusive group with three arguments in it (one for each compression type). Neither of these is more **correct** than the other. Pick which one you would like to try and finish the script with it.
16. When you are finished, you should be able to specify the directory to archive, the name of the archive to create, and the compression type (if any) from the command line. The user should no longer be prompted for anything after hitting `<enter>`

You have completed lab4. Proceed to Completing The Lab, and follow the instructions for "lab sign-off"

**Answer INVESTIGATION 4 observations / questions in your lab log book.**

## LAB 4 SIGN-OFF (SHOW INSTRUCTOR)

Follow the submission instructions for lab 4 on Blackboard.

**Time for a new backup!**
If you have successfully completed this lab, make a new backup of your virtual machines as well as your host machine.

**Perform the Following Steps:**

1. Make certain that your **c7host**, **centos1** and **centos2** VMs are running.
2. Switch to your **c7host** VM.
3. Open a shell terminal, and change to your **bin** directory.
4. Issue the Linux command: `wget https://raw.githubusercontent.com/OPS245/labs/main/lab4-check.bash`
5. Give the **lab4-check.bash** file execute permissions (for the file owner).
6. Run the shell script and if any warnings, make fixes and re-run shell script until you receive "congratulations" message.
7. Arrange proof of the following on the screen:

   ✓ **centos1** VM:

   - Demonstrate that this VM 's current run-level is set to **5**.

   ✓ **c7host** machine

   - Run the **lab4-check.bash** script (must have all `OK` messages)

   ✓ **Lab4** log-book filled out.
8. Take a screenshot of the proof in the previous step, and upload it, your tarchiver2.py script, your log book, and the file generated by **lab4-check.bash** to blackboard.

## Practice For Quizzes, Tests, Midterm & Final Exam

1. Describe all of the field in `/etc/passwd`
2. What is the command to create a user? What option to create a home directory for that user?
3. What is the command to change the full name of an already-created user?
4. What is the command to delete a user account? What option allows for the user's home directory to be removed as well?
5. What is the command to create a group? What is the command (or steps) to include a user in a newly-created group?
6. What is the purpose of `/etc/shadow`?
7. What is the purpose of `/etc/skel`?
8. What does the term run-level mean?
9. How to set the run-level of a Linux system to text-based only? How to set to graphical mode?
10. What is the command to view the status of running services?
11. What is the command to start a service (like httpd, or sshd)?
12. What is the command to stop a service (like httpd, or sshd)?
13. What is the difference between **starting** a service and **enabling** a service?
14. Can a service be stopped and started by issuing just one command?

Retrieved from "https://wiki.cdot.senecacollege.ca/w/index.php?title=OPS245_Lab_4&oldid=158472"

- This page was last edited on 8 May 2022, at 23:24.