Assignment 2 for Cybersecurity CPS493
Professor Hoffman

This assignment is a very basic introduction to accessing and using a virtual machine running Ubuntu Server set up earlier in class, as well as a few simple user, group, and permissions commands.
I named this virtual machine "pal", as a sort of homage to HAL 9000.

1. When you first ssh and open your virtual machine, you should be told how many updates that can be applied. Enter the command to list the updates available.

Once I've SSH'd in, I run the command "sudo apt update" to fetch info on any upgradable packages.

```
Fetched 384 kB in 3s (126 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
15 packages can be upgraded. Run 'apt list --upgradable' to see them.
fisherb@pal:~$
```

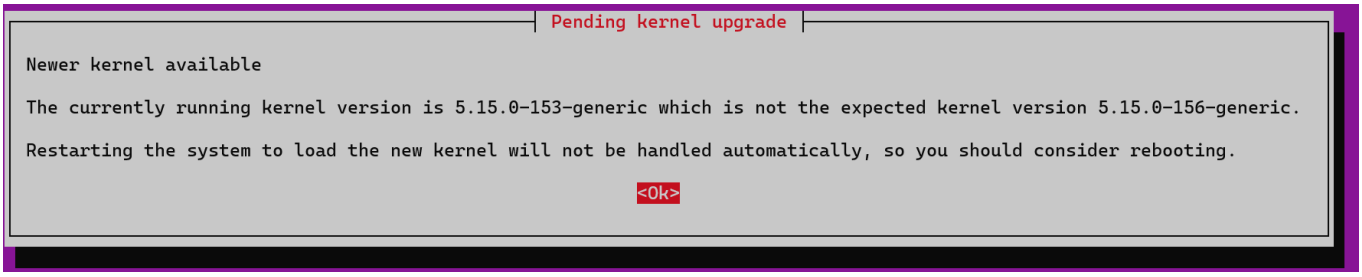
Here, I can see 15 packages can be upgraded.

2. Update and Upgrade your system.

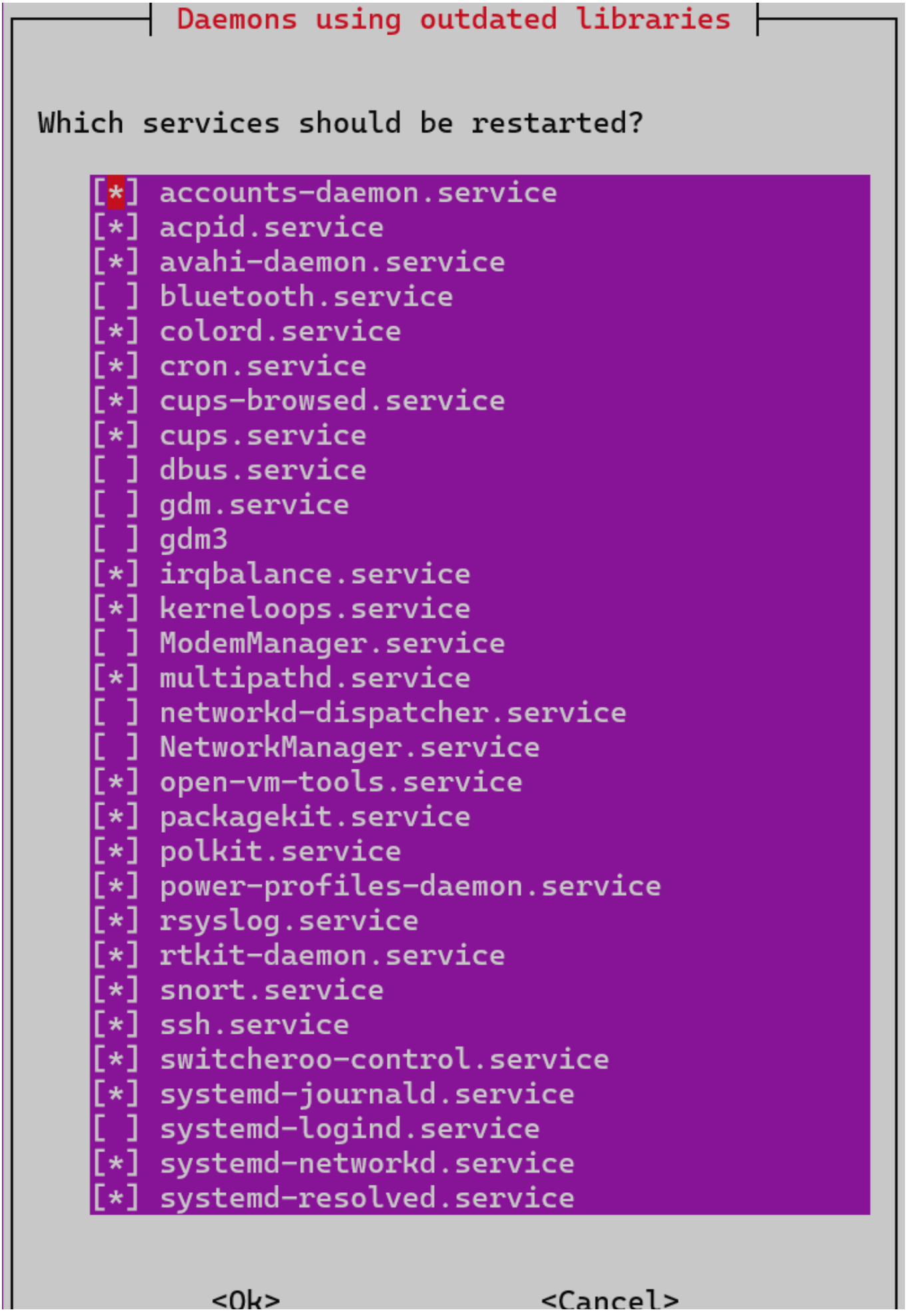
Let's upgrade those packages.

```
fisherb@pal:~$ sudo apt upgrade
```

Packages upgrade, and then we are greeted by a lovely purple screen and this message:



The next step on the assignment is to reboot, so I hit enter.


Daemons using outdated libraries
Which services should be restarted?
[*] accounts-daemon.service
[*] acpid.service
[*] avahi-daemon.service
[ ] bluetooth.service
[*] colord.service
[*] cron.service
[*] cups-browsed.service
[*] cups.service
[ ] dbus.service
[ ] gdm.service
[ ] gdm3
[*] irqbalance.service
[*] kerneloops.service
[ ] ModemManager.service
[*] multipathd.service
[ ] networkd-dispatcher.service
[ ] NetworkManager.service
[*] open-vm-tools.service
[*] packagekit.service
[*] polkit.service
[*] power-profiles-daemon.service
[*] rsyslog.service
[*] rtkit-daemon.service
[*] snort.service
[*] ssh.service
[*] switcheroo-control.service
[*] systemd-journald.service
[ ] systemd-logind.service
[*] systemd-networkd.service
[*] systemd-resolved.service
<Ok>
<Cancel>

I hit enter on Ok to restart any services whose Daemons are using the old versions of the packages I just upgraded.

```
Service restarts being deferred:
 systemctl restart ModemManager.service
 systemctl restart NetworkManager.service
 systemctl restart bluetooth.service
 /etc/needrestart/restart.d/dbus.service
 systemctl restart gdm.service
 systemctl restart gdm3.service
 systemctl restart networkd-dispatcher.service
 systemctl restart systemd-logind.service
 systemctl restart unattended-upgrades.service
 systemctl restart user@1000.service
 systemctl restart user@132.service
 systemctl restart wpa_supplicant.service

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
```

Some restarts have been deferred. That isn't really anything to be concerned about, as I am about to reboot the system.

3. Reboot your system. (You may have to wait a few minutes to ssh again).

Here, I run the straightforward command to reboot the VM.

```
fisherb@pal:~$ sudo reboot
```

After rebooting, I SSH back in.

4. Change the current user to **root** using the command **`sudo su root`**. What does the prompt look like?

The next step is to change my user to root and show the prompt.

```
root@pal:/home/fisherb# cd ../..
root@pal:/#
```

I've successfully changed my user to root, and I back out of my previous directory.

5. Create a new user with the name **bobby** using the command **useradd**. Next, create another user with the name **sally** using the command **adduser**. What is the difference between the two?

This step doesn't mention anything about specific flags for the useradd command, so I'm assuming there will be no flags set.

```
root@pal:/# useradd Bobby
root@pal:/#
```

Looks like the command ran successfully.

```
root@pal:/home# dir
fisherb
root@pal:/home#
```

Because I didn't specify a home directory, useradd didn't create one for bobby.

Next, I will use the adduser command to create a user for sally.

```
root@pal:/home# adduser sally
Adding user `sally' ...
Adding new group `sally' (1002) ...
Adding new user `sally' (1002) with group `sally' ...
Creating home directory `/home/sally' ...
Copying files from `/etc/skel' ...
```

It's clear that using adduser automates some steps of user creation that useradd would not.

```
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for sally
Enter the new value, or press ENTER for the default
        Full Name []: sally smith
        Room Number []: 223
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
```

I enter a password for sally, and I am prompted to enter some basic information about the new user. I enter some made up values, then just hit enter until the prompt ends.

6. Change the current user to sally. What does the prompt look like now?

Pretty simple, I use the su command to switch users to sally.

```
root@pal:/home# su sally
sally@pal:/home$
```

Now, the prompt shows I am sally, and the hashtag from root has changed back to the dollar sign, also indicating I am no longer root.

7. While you're logged in as sally still, try to create a new user with the name earl. What happens? Why?

Sally does not belong to sudo, so this action should be blocked.

```
sally@pal:/home$ sudo adduser earl
[sudo] password for sally:
sally is not in the sudoers file.  This incident will be reported.
```

As expected, sally is blocked from creating a user.

8. Enter **exit** until you are the original user, ubuntu, again. Delete the user earl. I didn't show you the command, but Google it! "Googling" skills are a great skill in CS; It's impossible to know everything.

I'm assuming this step has a typo. Because sally couldn't create the user earl, I believe this is meant to say "Delete the user bobby". So, I will delete bobby.

The command to delete a user is "deluser". I exit back to the original user, named "fisherb" in my case.

```
fisherb@pal:~$ sudo deluser Bobby
Removing user `Bobby' ...
Warning: group `Bobby' has no more members.
Done.
```

Goodbye bobby!

9. Change the password of sally to something you can remember using **sudo passwd sally**

```
fisherb@pal:~$ sudo passwd sally
[sudo] password for fisherb:
New password:
Retype new password:
passwd: password updated successfully
```

This is a pretty simply command, just entering a new password for sally.

10. For the rest of the tasks, use the ubuntu user. Even though it's easier to complete tasks/commands, why is it bad practice to stay logged in as root?

Oops! I didn't realize I was still supposed to be signed in as root. It's bad practice to stay logged in as root for a number of reasons. If I type a command that would break something within my machine, root might not give me any warning or sign that said command is bad. Additionally, staying logged in as root creates a chance for somebody else to come up to my keyboard and do whatever they please on my machine.

11. Enter the command to see what your user id is.

I'm a big fan of how intuitive simple linux commands are.

```
fisherb@pal:~$ id
uid=1000(fisherb) gid=1000(fisherb) groups=1000(fisherb),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),110(lxd)
```

My user id is 1000.

Group Tasks:

12. What groups does ubuntu belong to?

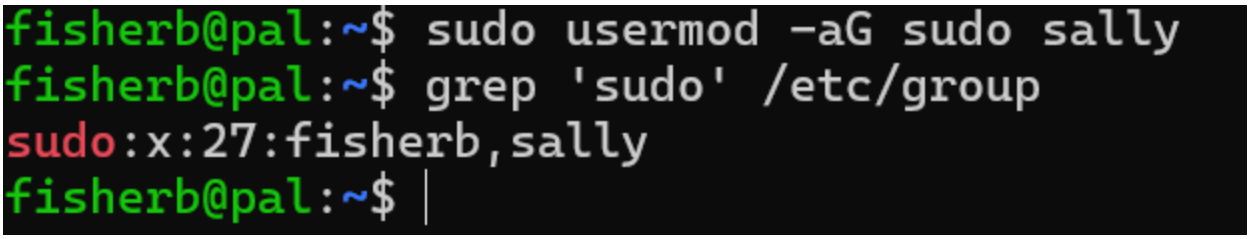
Looks like fisherb belongs to a few groups:

- fisherb, a group just for my user
- adm, the system monitoring group

- cdrom, the group to allow other users to access CD/DVD drives
- sudo, the group to use sudo to temporarily elevate privileges
- dip, the group able to use dial-up tools
- plugdev, the group able to use and mount external drives
- lxd, the group of users who can manage the LXD container hypervisor

13. Give sally the ability to execute sudo commands. Next, try to create a new user while logged in as sally.

Here, I am running usermod with the flag -aG, which stands for "append group", which will add sally to the end of the group file being edited (sudo in this case). Next, I search the group file for the word "sudo" using grep, which outputs the line of the file showing sudo users.



My command succeeded, and sally has been added to the sudo group. I use su to switch to sally. Now, I'll verify that she can create a new user.

```
fisherb@pal:~$ sudo su sally
[sudo] password for fisherb:
sally@pal:/home/fisherb$ sudo adduser mikey
[sudo] password for sally:
Adding user `mikey' ...
Adding new group `mikey' (1001) ...
Adding new user `mikey' (1001) with group `mikey' ...
Creating home directory `/home/mikey' ...
Copying files from `/etc/skel' ...
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
Sorry, passwords do not match.
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for mikey
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
sally@pal:/home/fisherb$
```

Looks good! Sally successfully created a new user "mikey" (after being yelled at for a weak password).

14. Create a new group called cybersec

Since I just gave sally sudo access, I can create this group while logged in as her.

```
sally@pal:/home/fisherb$ sudo groupadd cybersec
sally@pal:/home/fisherb$ grep cybersec /etc/group
cybersec:x:1003:
```

The group was successfully created! However, there are no members of the group currently. Let's fix that.

15. Add sally to the group, cybersec

Using the same command we used to add sally to sudo, just changing the group to cybersec:

```
sally@pal:~$ sudo usermod -aG cybersec sally
```

Checking that sally was added properly:

```
sally@pal:~$ grep 'cybersec' /etc/group
cybersec:x:1003:sally
```

Looks good! Now sally is a member of cybersec.

16. Check to see which groups sally belongs.

I'm going to use the "groups" command to see which groups sally belongs to.

```
sally@pal:~$ groups
sally sudo
sally@pal:~$ sudo su fisherb
fisherb@pal:/home/sally$ sudo su sally
[sudo] password for fisherb:
sally@pal:~$ groups
sally sudo cybersec
```

Initially, the command didn't display the updated result, so I quickly switched to my user, then back to sally. After that, the "groups" command displayed the correct result.

Permission and Access Control Lists:

17. Create a new directory called **lab1**. Enter the command to find the permissions of the directory. Who is the owner and group owner of this directory? What permissions does the owner, group and other have?

I'm going to switch back to fisherb to do this step. Then, I use "mkdir" to create the lab1 directory, and check that it executed properly using "dir".

```
fisherb@pal:~$ mkdir lab1
fisherb@pal:~$ dir
Desktop  Documents  Downloads  lab1  Music  Pictures  Public  Templates  Videos
fisherb@pal:~$
```

Next, I check the permissions of the directory using ls -ld. This will list directories in a long format, as well as just the directories and not any files. I provide the location "./lab1/", which navigates from the working directory straight into the lab1 directory, rather than providing the entire path to lab1.

```
fisherb@pal:~$ ls -ld ./lab1/
drwxrwxr-x 2 fisherb fisherb 4096 Sep 28 21:00 ./lab1/
```

Looks like fisherb is both the owner and group owner here. The permissions are as follows:

- Owner: Read, Write, Execute
- Group: Read, Write, Execute
- Other: Read, Execute

18. Change your directory to lab1. Create a new bash file called, ***helloWorld***. When ran, your program should just print “Hello World!”. (Don’t forget to make your bash file executable).

First, I'll navigate into lab1 using cd.

```
fisherb@pal:~$ cd lab1
fisherb@pal:~/lab1$
```

Now I am in lab1.

Now for the bash file. I open a new instance of nano on a file called "hello".

```
fisherb@pal:~/lab1$ nano hello
```

First, I add the shebang to indicate that this is a script, and specify that it should be run using a bash shell.

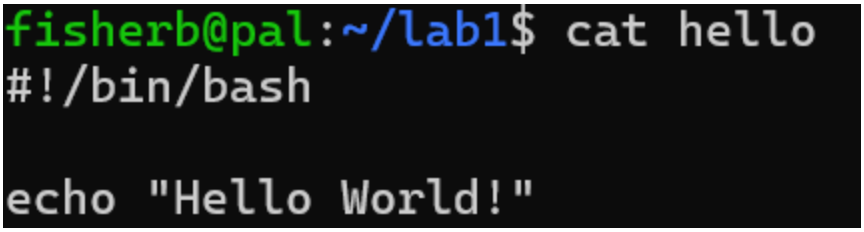
```
#!/bin/bash
```

Next, I'll use a simple echo command to print "Hello World!".

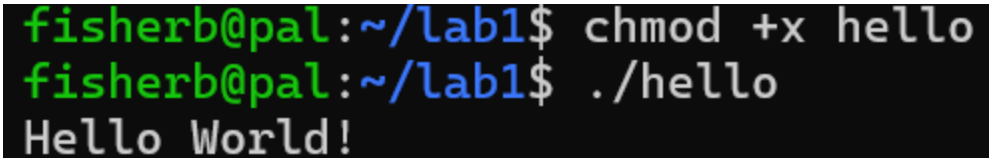
```
#!/bin/bash

echo "Hello World!"
```

All set! Now I save and exit nano. Next, I use cat to verify the contents of the file.

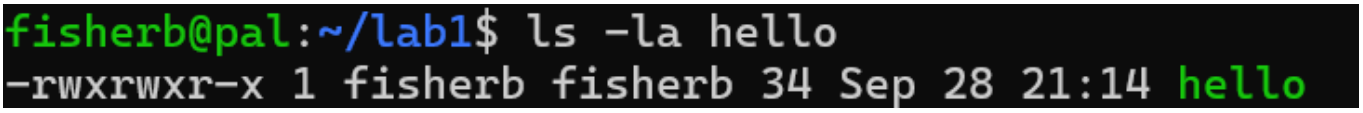


Looks good! Time to make it executable. I'll use chmod to make the file executable using the flag +x (add executable). Then, I'll run it using ./hello



It worked!

19. Enter the command **ls -la helloWorld**. What are the reading, writing, and executing permissions for the owner, group and other?
    a. Change the permissions so the group also has w and x permissions.



Again, running ls with the -l flag shows a long listing (including permissions), and the -a flag shows all files. Specifying "hello" will show files/directories whose name contains "hello".

The permissions are:

- Owner: Read, Write, Execute
- Group: Read, Write, Execute
- Other: Read, Execute

Now, to change the permissions so the group has write and execute permissions:

This handy cheat sheet from GeeksForGeeks shows octal values of permissions for files.

Octal	Binary	File Mode
0	000	---
1	001	--x
2	010	-w-
3	011	-wx
4	100	r--
5	101	r-x
6	110	rw-
7	111	rwx

Currently, the permissions for group are set to rwx, or 7 in octal. So, the file already has the permissions asked for in the assignment.

20. Use the **getfacl** command to view the ACL of the file.

```
fisherb@pal:~/lab1$ getfacl hello
# file: hello
# owner: fisherb
# group: fisherb
user::rwx
group::rwx
other::r-x
```

Running getfacl shows the access control list for the file, similar to the output of "ls -la ", just a little more detailed and organized.

21. Using the **setfacl** command, allow the user, sally, the ability to read and write to the file.

Here, I'm running setfacl with the -m flag (Modify) for the user sally, giving her read write permissions on the file hello.

```
fisherb@pal:~/lab1$ setfacl -m u:sally:rw hello
fisherb@pal:~/lab1$ getfacl hello
# file: hello
# owner: fisherb
# group: fisherb
user::rwx
user:sally:rw-
group::rwx
mask::rwx
other::r-x
```

After running setfacl, I fetch the ACL for hello using getfacl, showing that sally was successfully given read write permissions!

That's the end of this assignment. Overall, it was a good review of basic commands for switching between users, manipulating groups, and editing file permissions!