Linux System Administration

If you're just getting started with administrative commands, tread carefully! There's a *lot* you can screw up accidentally. If you're reading this to fix the problem of not being able to access a shared folder, go ahead and skip to the very end, but then come back and read the rest later.

Root

This is the normal administrative account; it has privileges which allow it to do nearly anything on the host. As such, when running as the root user, you need to be very careful about what you do. It is also called the "supervisor" or "superuser" account.

On older systems, there was typically a password for the root account, and you would open a shell as the root user (or login to the system) using this password. The `su` command is what you would use if you were already logged into the system as a normal user, and wanted to "become root". We still use `su` occasionally, but we use it slightly differently, as we'll see later.

Now, it's more common for root not to have a password, so we have to have another way to become root. For this, we have the `sudo` command. Here's how it works:

```
vmuser@f18marsh:~$ wc -c /var/log/tallylog
wc: /var/log/tallylog: Permission denied
vmuser@f18marsh:~$ sudo wc -c /var/log/tallylog
64128 /var/log/tallylog
```

If you haven't run sudo recently, it will prompt you for *your* password. This is because you've been granted special permission to call sudo in the file `/etc/sudoers`

```
vmuser@f18marsh:~$ groups
vmuser adm cdrom sudo dip plugdev lpadmin sambashare wireshark docker vboxsf
vmuser@f18marsh:~$ grep %sudo /etc/sudoers
/etc/sudoers: Permission denied
vmuser@f18marsh:~$ sudo grep %sudo /etc/sudoers
%sudo  ALL=(ALL:ALL) ALL
```

Let's unpack this. First we list all of the permissions groups to which we belong. One of these is named `sudo`. If we look for this group (which is what prepending `%` denotes) in the file `/etc/sudoers`, we see that there's a matching line. However, along the way we discover that `/etc/sudoers` is itself only readable by root, so we have to use `sudo` to run `grep` over the file!

What does the line in `/etc/sudoers` mean? Here it is again:

```
%sudo  ALL=(ALL:ALL) ALL
```

If we run `man sudoers`, we can see all of the documentation, but we'll cut to the chase. First, `%sudo` means this is a rule for the permission group `sudo`, of which we happen to be a member. If you leave off the `%`, then this would match on username instead.

Next, we have the hosts on which this is valid, since this file might be shared between a number of similarly configured hosts. In this case, we use the wildcard `ALL`. So far we have

```
%sudo  ALL
```

to indicate that on all hosts, the group `sudo` will have the specified permissions. After the `=`, we see `(ALL:ALL)`. This says that we're allowed to run as any user or group. We could have restricted this to `(man:tape)` if we wanted to grant this user or group permission to run as the `man` user (which can install or rebuild the database of manual pages) or the `tape` group (for access to a tape drive).

Finally, the last `ALL` says that when running as the provided user or group, we're allowed to run any command.

Running as Another User

The easiest way to run as another user is to use the `-u` option to `sudo`:

```
vmuser@f18marsh:~$ whoami
vmuser
vmuser@f18marsh:~$ sudo -u man whoami
man
```

Since you can run any command you like, you could also use this to start a shell:

```
vmuser@f18marsh:~$ sudo -u man /bin/bash
man@f18marsh:~$ whoami
man
man@f18marsh:~$ exit
exit
vmuser@f18marsh:~$
```

One thing that's important to note, however, is that each shell has an *environment*, which defines things like the directory path to search for executables, special options to pass certain programs, etc. Sometimes you don't want to carry the environment over to the new shell, but rather have the shell initialized as if the target user had just *logged in*. For this, we can use the `-i` flag. Compare the output of the following on your VM:

```
sudo /usr/bin/env
sudo -i /usr/bin/env
```

This can be very important in some circumstances, and often when starting a root shell, you'll want to include the `-i` flag. The following two commands end up being equivalent:

```
sudo -i /bin/bash
sudo su -
```

Here, the `-` option to `su` says to treat this as a login shell, just like `sudo -i`.

Managing Users and Groups

We're going to consider just a couple of things here: changing passwords and assigning users to groups. These should be the bulk of what you need to do.

To change your own password, run:

```
vmuser@f18marsh:~$ passwd
Changing password for vmuser.
(current) UNIX password:
```

You are prompted for your current password, then for the new password, and finally for the new password *again*, just to make sure you typed it correctly.

When run as root, you can change *another* user's password:

```
vmuser@f18marsh:~$ sudo passwd man
Enter new UNIX password:
```

Now you're only prompted for the new password, not the current one. This is because you're running it as the superuser.

We saw the `groups` command earlier, which lists your current groups. You can switch your currently active group (which on our VM defaults to `vmuser`) by running `newgrp`:

```
vmuser@f18marsh:~$ echo $GROUPS
1000
```

```
vmuser@f18marsh:~$ newgrp docker
vmuser@f18marsh:~$ echo $GROUPS
124
```

It is important to note that when you run `newgrp`, you open a new shell. That means your shell history will be gone, until you `exit` from that shell and return to your previous shell (and group).

Most of the time, you will not need to change your active group, but you might need to change the list of groups to which you belong. For this, you run the `vigr` command (as root):

```
sudo vigr
```

In particular, consider the following line at the bottom of the file:

```
vboxsf:x:999:
```

If you see this line, it means the `vmuser` account will not be able to access shared folders. We can fix this by changing the line to:

```
vboxsf:x:999:vmuser
```

Now, when we save and quit, we'll see a message telling us to run `vigr -s`, which edits the *shadow* copy of the file. This is a security feature that hides some of the group details from the `/etc/group` file. Run:

```
sudo vigr -s
```

and make the same change:

```
vboxsf:!::vmuser
```

At this point, you will have to log out of your VM and back in (you don't need to restart it, though that will also work), and you'll now have access to shared folders!