Skills Practice - Processes, Pipes, and Grep

Goal:

This goal of these practice exercises is to familiarize you with viewing processes, using pipes, and limiting output to just that which you are interested in.

Instructions:

Start a Virtual Machine

For this practice let's use a virtual machine that we created in a previous project. First, start a command line session. Change into your `linuxclass` folder and then change into the `testbox01` directory.

```
cd linuxclass
cd testbox01
```

Next, start the virtual machine using the `vagrant up` command. If the virtual machine is already running, vagrant will let you know that it's ready to use. If it's stopped or paused, vagrant will start the virtual machine.

```
vagrant up
```

Connect to the virtual machine.

```
vagrant ssh
```

Practice

Let's start out by viewing the processes associated with our command line session. Running the `ps` command without any arguments will list those processes.

```
ps
```

You'll likely only see two processes. One process is `bash` and the other is the `ps` command itself. The `bash` process is the shell. It's the program that is started when you log into a system. It

accepts your commands and executes them. Executing the `ps` command is a prime example of how the shell works.

Here is example output from the `ps` command.

```
  PID TTY          TIME CMD
 3724 pts/0    00:00:00 bash
 3764 pts/0    00:00:00 ps
```

It lists the PID, or process ID, the terminal associated with the process (TTY), the cumulated CPU time (TIME) used by the process, and the actual command or process name itself.

Let's use the "-f" option to `ps` to display a full-format listing.

```
ps -f
```

Here is the output generated by the command.

```
UID        PID  PPID  C STIME TTY          TIME CMD
vagrant   3724  3723  0 12:52 pts/0    00:00:00 -bash
vagrant   4085  3724  0 12:57 pts/0    00:00:00 ps -f
```

You'll notice that the output includes more information. Now, it shows what user owns the process in the UID (user ID) column. You'll also notice a PPID column. This lists the parent process ID (PPID) of the process. The parent process is the process that spawned, or started, the process. The C column represents the CPU utilization for the process. It is the CPU time used divided by the time the process has been running. The STIME column is the start time for the process.

Let's single out one of the processes to examine. Let's look at the parent (PPID) of the bash process. Be sure to use the PPID displayed in your output as it will probably be different from the example given here.

```
ps -fp 3723
```

Output:

```
UID        PID  PPID  C STIME TTY          TIME CMD
vagrant   3723  3719  0 12:52 ?        00:00:00 sshd: vagrant@pts/0
```

This output shows that an `sshd` process running as the vagrant user started the `bash` process. This makes sense since we connected to the system using SSH. You could keep working your way

up the process tree. Eventually you would end up looking at the process with a PID of 1. Let's look at that process now.

```
ps -fp 1
```

Output:

```
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 12:51 ?        00:00:00 /usr/lib/systemd/systemd
```

The process with a PID of 1 is the very first process that was started when the Linux system booted. The `systemd` process is initially responsible for starting all the other process when a system boots. When you run the "systemctl enable SERVICE_NAME" command it tells systemd to start that service on boot.

Let's look at all the process running on the system.

```
ps -ef
```

If you want to page through the process listing, pipe the output of the `ps` command to `less`.

```
ps -ef | less
```

Let's use the `ps` command in conjunction with `grep`. Let's use grep to look for any processes that match the string "syslogd".

```
ps -ef | grep syslogd
```

Output:

```
root       567     1  0 12:51 ?        00:00:00 /usr/sbin/rsyslogd -n
vagrant   5757  3724  0 13:23 pts/0    00:00:00 grep --color=auto syslog
```

You'll see a process named "rsyslogd" which matches the string "syslogd," so that line of output is displayed to your screen. You'll also notice that the grep command itself is displayed because the grep command itself contains the string we are looking for. To exclude the grep command from the output, you can perform an inverted match.

```
ps -ef | grep syslogd | grep -v grep
```

Output:

```
root       567     1  0 12:51 ?        00:00:00 /usr/sbin/rsyslogd -n
```

Now we are left with just the `rsyslogd` process. You'll notice that the process is owned, or is running as, the root user. Let's find out what other processes are running as root.

```
ps -fu root | less
```

Most of the processes on this system are running as root. If you have different services and users on a system you'll find some processes that are not running as root.

Alternatives to `ps`

Let's use the top command to display the processes running on our system.

```
top
```

The `top` command will refresh the process listing every few seconds. You can leave top running for several seconds or for a few minutes to get an idea of what processes are active on a system. Also, notice a summary of system performance is available at the top of the output.

To change how often top updates, what it sorts by, and so forth consult the built-in help by typing "h" or "?". When you're done with help type "q" to quit and return to top. Also, typing "q" will cause top to exit.

Now, let's use the `htop` command.

```
htop
```

If you get a "command not found" error message, you'll need to install it. Let's use the normal process of looking for a program we want to install and then installing it once we find it.

```
dnf search htop
```

If no results are returned, then you'll need to enable the EPEL repository.

```
sudo dnf install -y epel-release
```

Now try again.

```
dnf search htop
dnf info htop
sudo dnf install -y htop
```

Now you should be able to use the htop utility.

```
htop
```

Like `top`, `htop` displays system usage information at the top of the screen followed by a list of processes. The htop command uses visualizations that can help you quickly determine the overall system usage. The first line shows the cpu utilization on cpu number 1. If you have multiple CPUs, then they will be represented the same way. The next line is a graphical representation of the amount of memory being used by the system. The following line displays the amount of swap usage.

Also like `top`, you can type "h" or "?" to get help. For example, you'll learn that by typing "M" you can make `htop` sort the processes by memory utilization. You can also filter the list of processes displayed, sort them in different ways, kill processes, and more. To exit out of `htop`, type "q".

Finish

When you're done practicing your piping, process, and grepping skills, exit the virtual machine and power it off.

```
exit
vagrant halt
```