Introduction

When we run software on our system or execute a command, **processes** and **services** are created. To be an effective systems administrator we must understand how to identify what is running on our system, determine if they are running properly, and if they should be running at all.

Processes and services that are running improperly could cause issues with **system performance**. Processes that are running that are not needed increase the **attack surface** of the system, potentially creating vulnerabilities. Knowing what processes should be running can help you identify **malicious software** that might have infected the system.

Part One: Linux Processes

A **process** is a piece of code that has be created or executed by an application or command such as launching a web browser like Firefox. When a process first starts, it is known as a **Foreground Process** by default. When a process is in the foreground, it can receive input from the user (such as entering information on the keyboard) and display output to the user (such as displaying text or images on the screen).

If the user is interacting with the process, it stays in the foreground. When the interaction stops, the process then becomes a **Background Process** waiting for input to run again.

Processes have four different states that change according to its circumstances:

- **Running** – The process is either currently running in the CPU or is ready to run (waiting to be assigned CPU resources). This is usually the state of a Foreground process.
- **Waiting** – The process is waiting for an event or for a resource. This is usually the state of a Background Process.
- **Stopped** – The process has been stopped by receiving a signal from the user or the system. A stopped process can be used for debugging issues.
- **Zombie** – The process has halted for some unknown reason. The process is dead. This usually indicates an issue.

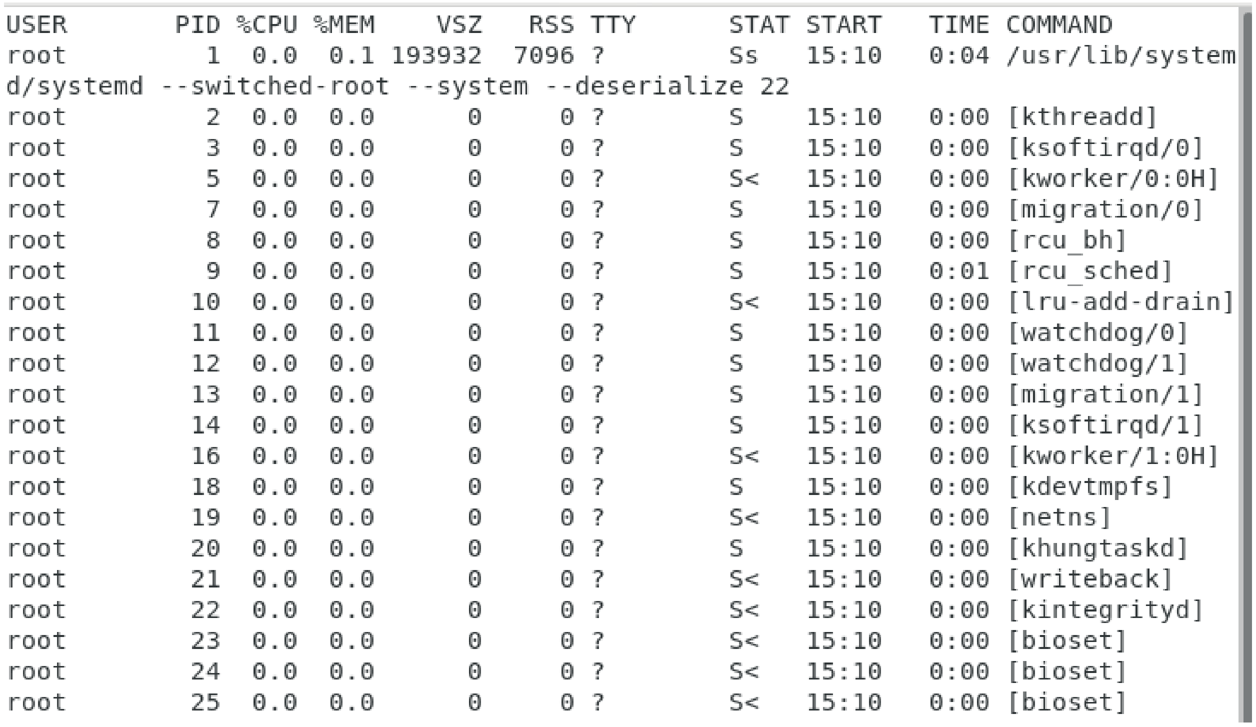
There are many tools that can be used to monitor processes. Two of the most popular that can be found on most Linux systems are **"ps"** and **"top".**

The **"ps" (Process Status)** command displays a **snapshot** of all the running processes at a moment in time. There are many different options that can be used with this command to give us different types of information.

Start by running the option to display all currently running processes by all users. This will output a lot of information, so we will **"pipe"** the output through **"less"** for a **page-by-page** view.

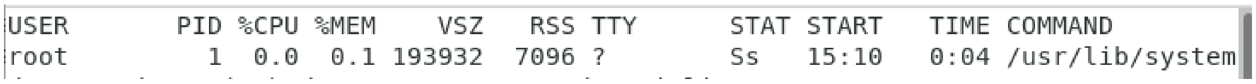
- Log into your GCP Linux VM and switch user (su -l user)
- Run the command:

```
ps aux | less
```



```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1 193932  7096 ?        Ss   15:10   0:04 /usr/lib/system
d/systemd --switched-root --system --deserialize 22
root         2  0.0  0.0      0     0 ?        S    15:10   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    15:10   0:00 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   15:10   0:00 [kworker/0:0H]
root         7  0.0  0.0      0     0 ?        S    15:10   0:00 [migration/0]
root         8  0.0  0.0      0     0 ?        S    15:10   0:00 [rcu_bh]
root         9  0.0  0.0      0     0 ?        S    15:10   0:01 [rcu_sched]
root        10  0.0  0.0      0     0 ?        S<   15:10   0:00 [lru-add-drain]
root        11  0.0  0.0      0     0 ?        S    15:10   0:00 [watchdog/0]
root        12  0.0  0.0      0     0 ?        S    15:10   0:00 [watchdog/1]
root        13  0.0  0.0      0     0 ?        S    15:10   0:00 [migration/1]
root        14  0.0  0.0      0     0 ?        S    15:10   0:00 [ksoftirqd/1]
root        16  0.0  0.0      0     0 ?        S<   15:10   0:00 [kworker/1:0H]
root        18  0.0  0.0      0     0 ?        S    15:10   0:00 [kdevtmpfs]
root        19  0.0  0.0      0     0 ?        S<   15:10   0:00 [netns]
root        20  0.0  0.0      0     0 ?        S    15:10   0:00 [khungtaskd]
root        21  0.0  0.0      0     0 ?        S<   15:10   0:00 [writeback]
root        22  0.0  0.0      0     0 ?        S<   15:10   0:00 [kintegrityd]
root        23  0.0  0.0      0     0 ?        S<   15:10   0:00 [bioset]
root        24  0.0  0.0      0     0 ?        S<   15:10   0:00 [bioset]
root        25  0.0  0.0      0     0 ?        S<   15:10   0:00 [bioset]
```

This output is a list of every process currently running at a moment the command was executed, sorted by the order in which they were started. At the top of the screen, you see the output is broken down into columns.



```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1 193932  7096 ?        Ss   15:10   0:04 /usr/lib/system
```

- **USER** – This column displays which user started the process. System processes will show the "root" user.
- **PID** – This is the **process ID** number. This number is assigned in sequence when the process is started.
- **%CPU** – This is the percentage of CPU time the process is currently using.
- **%MEM** – This is the percentage of system memory the process is currently using.
- **VSZ** – Virtual memory usage of the process (in KiB).
- **RSS** – Resident Set Size, the physical memory the task is using (in KiB).
- **TTY** – Controlling terminal.
- **START** – Staring time of the process.
- **TIME** – Cumulative CPU time.
- **COMMAND** – The command that started the process.

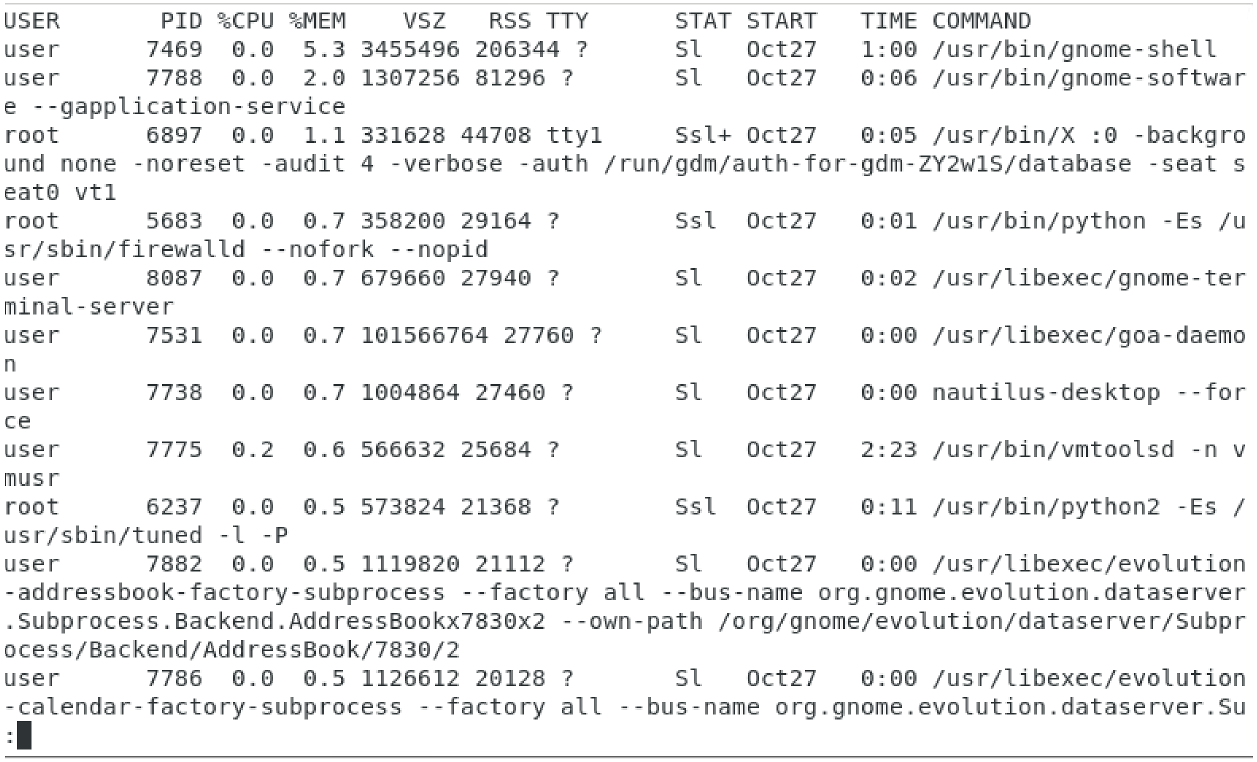
Since the default is to sort the processes by **PID** number and the list so long, it doesn't help us determine much information. Instead, we can sort the output by different metrics such as **CPU time** and **memory usage**.

Hit q to exit back into command prompt.

- Run the following command to sort by CPU percentage from highest to lowest:
  ```
  ps aux --sort -%cpu | less
  ```
- Run the following command to sort by memory percentage from highest to lowest:
  ```
  ps aux --sort -%mem | less
  ```

1. **Try sorting by the physical amount of system memory each process is using (Resident Set Size). Save a screen shot to upload for the assignment 6 assessment.**

Output will vary, but should be sorted by column "RSS"



```
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
user       7469  0.0  5.3 3455496 206344 ?      Sl   Oct27   1:00 /usr/bin/gnome-shell
user       7788  0.0  2.0 1307256 81296 ?       Sl   Oct27   0:06 /usr/bin/gnome-softwar
e --gapplication-service
root       6897  0.0  1.1 331628 44708 tty1     Ssl+ Oct27   0:05 /usr/bin/X :0 -backgro
und none -noreset -audit 4 -verbose -auth /run/gdm/auth-for-gdm-ZY2w1S/database -seat s
eat0 vt1
root       5683  0.0  0.7 358200 29164 ?        Ssl  Oct27   0:01 /usr/bin/python -Es /u
sr/sbin/firewalld --nofork --nopid
user       8087  0.0  0.7 679660 27940 ?        Sl   Oct27   0:02 /usr/libexec/gnome-ter
minal-server
user       7531  0.0  0.7 101566764 27760 ?     Sl   Oct27   0:00 /usr/libexec/goa-daemo
n
user       7738  0.0  0.7 1004864 27460 ?       Sl   Oct27   0:00 nautilus-desktop --for
ce
user       7775  0.2  0.6 566632 25684 ?        Sl   Oct27   2:23 /usr/bin/vmtoolsd -n v
musr
root       6237  0.0  0.5 573824 21368 ?        Ssl  Oct27   0:11 /usr/bin/python2 -Es /
usr/sbin/tuned -l -P
user       7882  0.0  0.5 1119820 21112 ?       Sl   Oct27   0:00 /usr/libexec/evolution
-addressbook-factory-subprocess --factory all --bus-name org.gnome.evolution.dataserver
.Subprocess.Backend.AddressBookx7830x2 --own-path /org/gnome/evolution/dataserver/Subpr
ocess/Backend/AddressBook/7830/2
user       7786  0.0  0.5 1126612 20128 ?       Sl   Oct27   0:00 /usr/libexec/evolution
-calendar-factory-subprocess --factory all --bus-name org.gnome.evolution.dataserver.Su
:█
```

The **"ps"** command can also be a useful tool for finding information about a process. If a process is misbehaving or not responding, we may need to find out the process ID # or see how many resources the process is currently using.

We will now install a text-based web browser called links and open it in order to start a process (If you have RDP set up correctly to log into the Linux GUI, you can go ahead and RDP into your VM, and start Firefox – the Firefox process can be monitored using the process-based statements below)

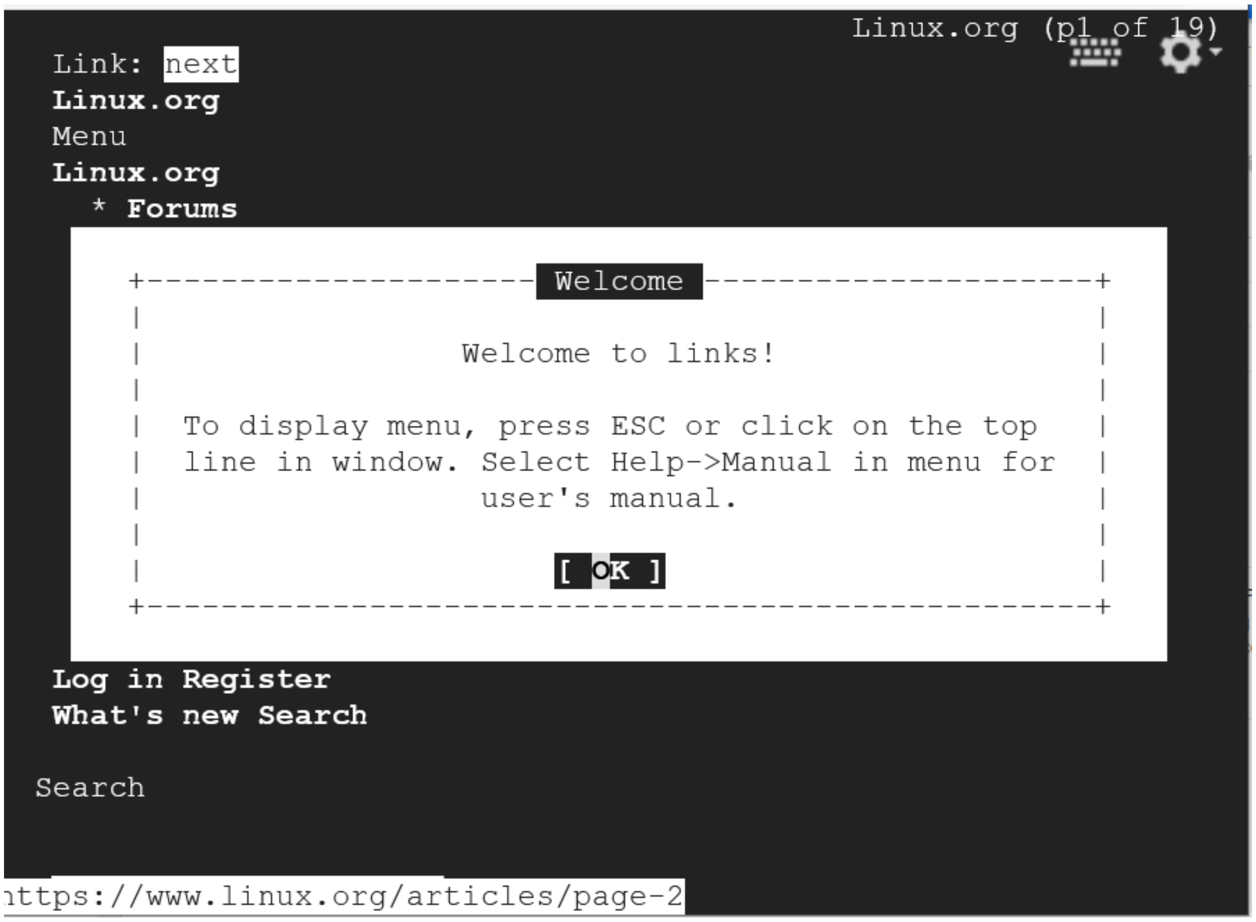
- Run the command

  sudo yum install links

- Once install is completed, run the command

  sudo links www.linux.org

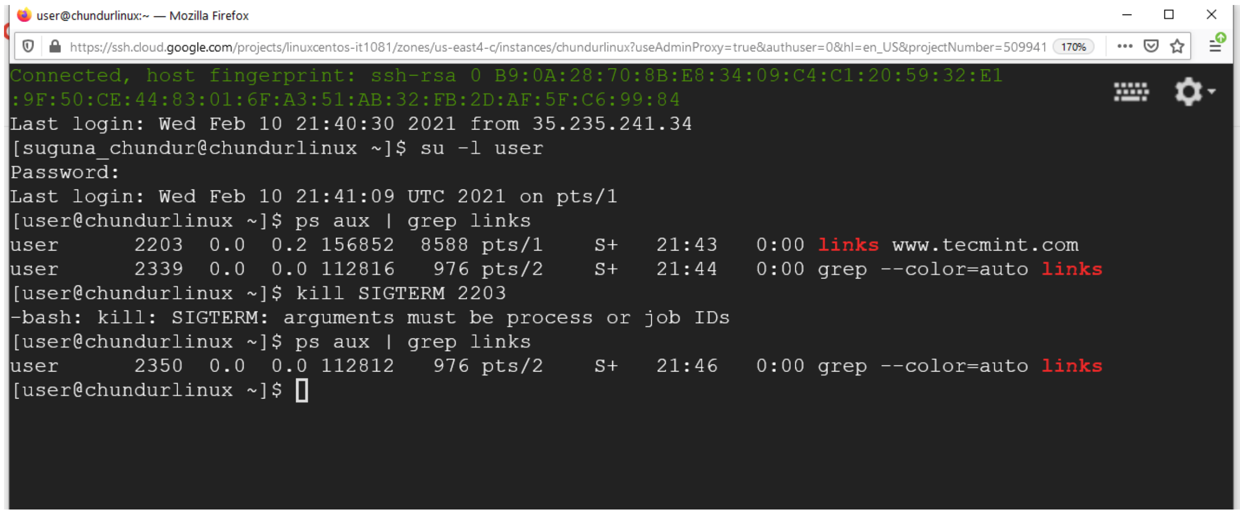
- The text version of the website opens up. Please remember that you can only use your keyboard (arrows) keys to navigate this site



```
                                                       Linux.org (p1 of 19)
  Link: next
  Linux.org
  Menu
  Linux.org
    * Forums
      +---------------------- Welcome ----------------------+
      |                                                     |
      |                  Welcome to links!                  |
      |                                                     |
      |  To display menu, press ESC or click on the top     |
      |  line in window. Select Help->Manual in menu for    |
      |                 user's manual.                      |
      |                                                     |
      |                      [ OK ]                         |
      +-----------------------------------------------------+
  Log in Register
  What's new Search

 Search

ıttps://www.linux.org/articles/page-2
```

- Keeping the current command window with the website open, open a new terminal window (top right gear symbol->new connection).
- Switch to user: su -l user
- Run the following command to determine information about the links process:

```
        ps aux | grep links
```


user@chundurlinux:~ — Mozilla Firefox
https://ssh.cloud.google.com/projects/linuxcentos-it1081/zones/us-east4-c/instances/chundurlinux?useAdminProxy=true&authuser=0&hl=en_US&projectNumber=509941
170%

```
Connected, host fingerprint: ssh-rsa 0 B9:0A:28:70:8B:E8:34:09:C4:C1:20:59:32:E1
:9F:50:CE:44:83:01:6F:A3:51:AB:32:FB:2D:AF:5F:C6:99:84
Last login: Wed Feb 10 21:40:30 2021 from 35.235.241.34
[suguna_chundur@chundurlinux ~]$ su -l user
Password:
Last login: Wed Feb 10 21:41:09 UTC 2021 on pts/1
[user@chundurlinux ~]$ ps aux | grep links
user      2203  0.0  0.2 156852  8588 pts/1    S+   21:43   0:00 links www.tecmint.com
user      2339  0.0  0.0 112816   976 pts/2    S+   21:44   0:00 grep --color=auto links
[user@chundurlinux ~]$ kill SIGTERM 2203
-bash: kill: SIGTERM: arguments must be process or job IDs
[user@chundurlinux ~]$ ps aux | grep links
user      2350  0.0  0.0 112812   976 pts/2    S+   21:46   0:00 grep --color=auto links
[user@chundurlinux ~]$ 
```

The output will show two entries (I have used a different website to access for this image but the principle is the same for any website you access with the links program). The second entry is for the **grep** process we ran for the search. The first in the list is the actual process for links.

In my example above, the **PID** for the process is 2203. Yours will be different.

This process is working just fine, but if it were acting up and not responding to input, we might have to **"kill"** the process. To do this we can use the **"kill"** command with one of two options:

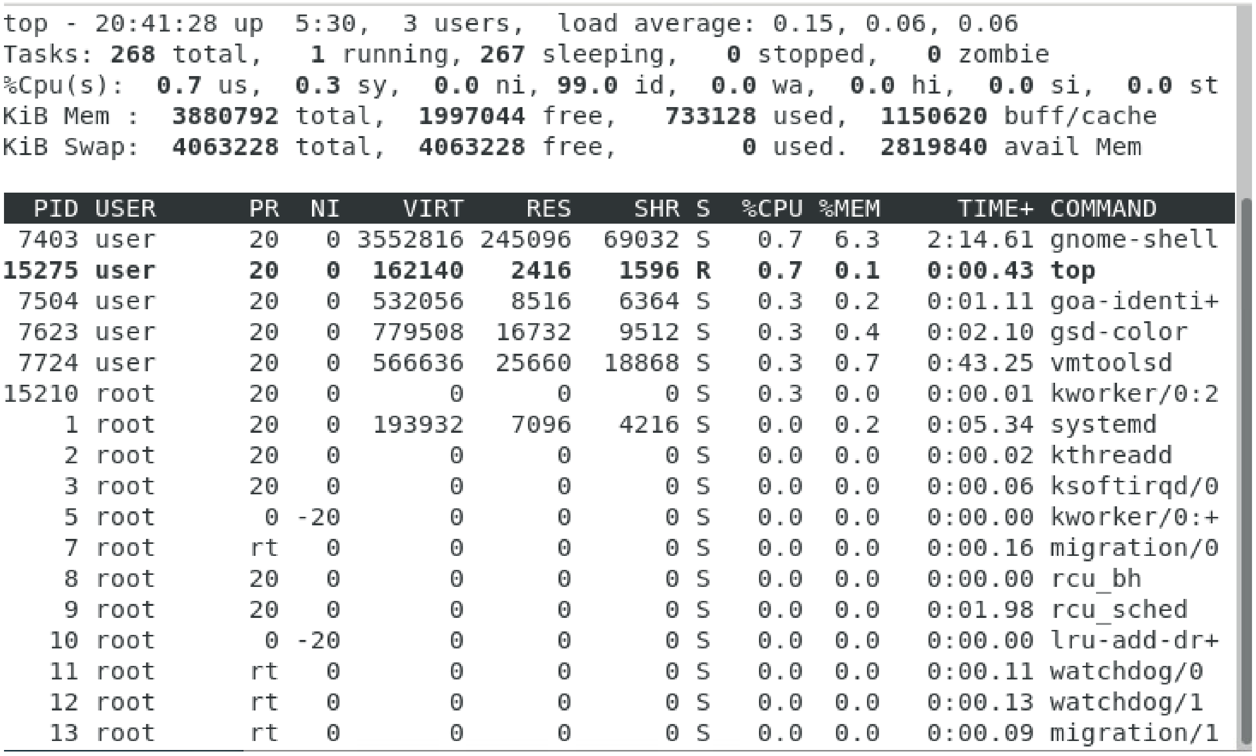
- **-9 or SIGTERM** – This option will ask the process to shut down on its own gracefully. This is what we always want to try first in case the process may have another process or file opened. If it shuts down on its own, it will also close any other process that may depend on it or close any file that is opened, avoiding potential corruption.
- **-15 or SIGKILL** – If the process won't go quietly, we may have to just pull the plug. This option will force the process to close, but it may have undesirable effects on a different process or file that is dependent on the misbehaving process.

When killing a process, we must specify the PID of the process instead of the process name.

- Run the following command to kill the links process. Replace the PID with the PID of your process:
  ```
  kill SIGTERM 2203
  ```
- Run `the ps aux | grep links` to make sure you have killed the process

Another useful tool for monitoring processes in real-time is the **"top"** command. The top command displays the **top 20 processes** running on the system and is updated every couple of seconds. By default, it sorts the processes by **CPU percent**.

- Run the command to open the top application:
  ```
  top
  ```



```
top - 20:41:28 up  5:30,  3 users,  load average: 0.15, 0.06, 0.06
Tasks: 268 total,   1 running, 267 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.7 us,  0.3 sy,  0.0 ni, 99.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  3880792 total,  1997044 free,   733128 used,  1150620 buff/cache
KiB Swap:  4063228 total,  4063228 free,        0 used.  2819840 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 7403 user      20   0 3552816 245096  69032 S   0.7  6.3   2:14.61 gnome-shell
15275 user      20   0  162140   2416   1596 R   0.7  0.1   0:00.43 top
 7504 user      20   0  532056   8516   6364 S   0.3  0.2   0:01.11 goa-identi+
 7623 user      20   0  779508  16732   9512 S   0.3  0.4   0:02.10 gsd-color
 7724 user      20   0  566636  25660  18868 S   0.3  0.7   0:43.25 vmtoolsd
15210 root      20   0       0      0      0 S   0.3  0.0   0:00.01 kworker/0:2
    1 root      20   0  193932   7096   4216 S   0.0  0.2   0:05.34 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.02 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   0:00.06 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:+
    7 root      rt   0       0      0      0 S   0.0  0.0   0:00.16 migration/0
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
    9 root      20   0       0      0      0 S   0.0  0.0   0:01.98 rcu_sched
   10 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 lru-add-dr+
   11 root      rt   0       0      0      0 S   0.0  0.0   0:00.11 watchdog/0
   12 root      rt   0       0      0      0 S   0.0  0.0   0:00.13 watchdog/1
   13 root      rt   0       0      0      0 S   0.0  0.0   0:00.09 migration/1
```

At the top of the tool, you will see the overall status for the system processes, including information such as **total process running**, **sleeping, stopped, or in a zombie state**. It also displays system metrics such as overall CPU usage and memory usage.

Change the order in which the processes are sorted with a few simple keyboard commands:

- Shift+M to sort by memory usage
- Shift+P to sort by CPU usage
- Shift+N to sort by Process ID
- Shift+T to sort by running time

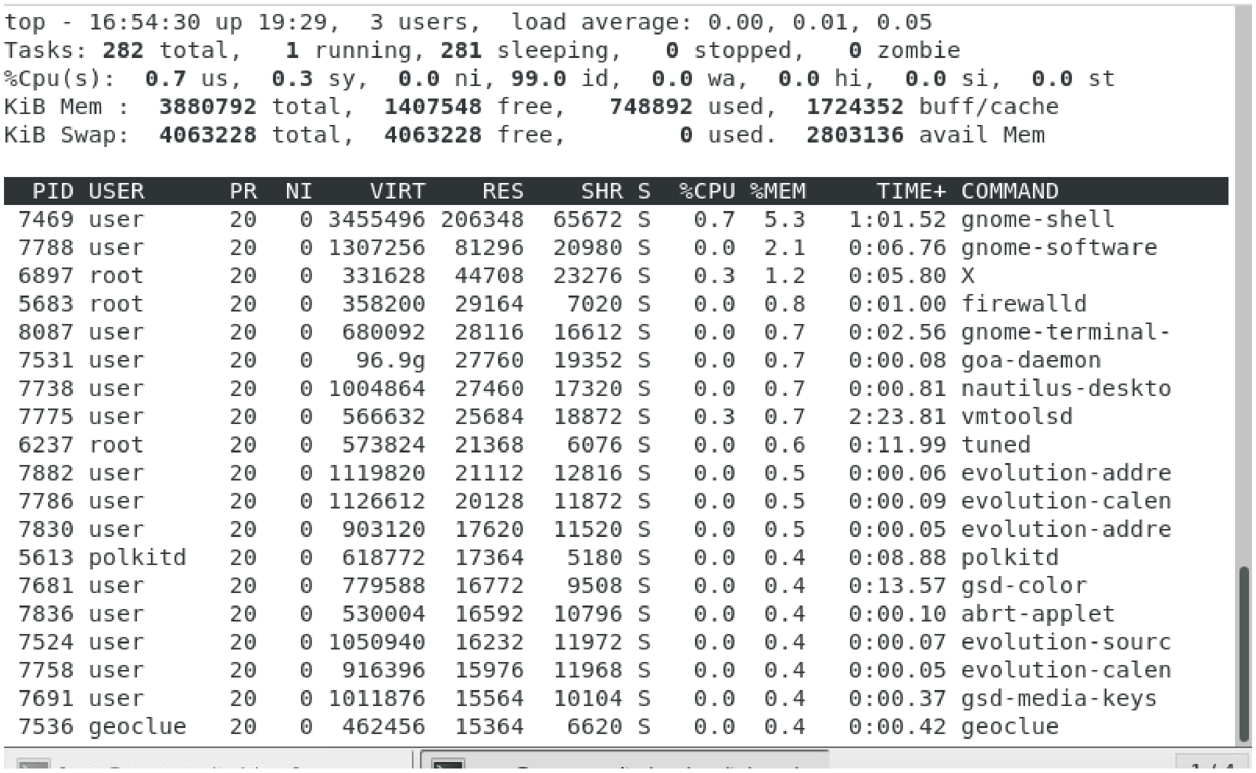
You can also kill processes from within **top** as well.

- Run the **links** application again in another terminal window (If you used sudo root account to run links, then the command to run the top process should also use sudo)
- In the first command line window, sort top by **Process ID** to help locate the process ID of links
- Hit **k** to bring up the kill prompt.
- Enter the **process ID** and hit **Enter**.
- Enter **15** to issue the SIGTERM option.

You can press q to go back to command prompt mode.

2. **Use the “top” application to sort processes by the highest memory usage and save a screenshot to upload for the assessment.**

Output will vary but should be sorted by %MEM column.



```
top - 16:54:30 up 19:29,  3 users,  load average: 0.00, 0.01, 0.05
Tasks: 282 total,   1 running, 281 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.7 us,  0.3 sy,  0.0 ni, 99.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  3880792 total,  1407548 free,   748892 used,  1724352 buff/cache
KiB Swap:  4063228 total,  4063228 free,        0 used.  2803136 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 7469 user      20   0 3455496 206348  65672 S   0.7  5.3   1:01.52 gnome-shell
 7788 user      20   0 1307256  81296  20980 S   0.0  2.1   0:06.76 gnome-software
 6897 root      20   0  331628  44708  23276 S   0.3  1.2   0:05.80 X
 5683 root      20   0  358200  29164   7020 S   0.0  0.8   0:01.00 firewalld
 8087 user      20   0  680092  28116  16612 S   0.0  0.7   0:02.56 gnome-terminal-
 7531 user      20   0   96.9g  27760  19352 S   0.0  0.7   0:00.08 goa-daemon
 7738 user      20   0 1004864  27460  17320 S   0.0  0.7   0:00.81 nautilus-deskto
 7775 user      20   0  566632  25684  18872 S   0.3  0.7   2:23.81 vmtoolsd
 6237 root      20   0  573824  21368   6076 S   0.0  0.6   0:11.99 tuned
 7882 user      20   0 1119820  21112  12816 S   0.0  0.5   0:00.06 evolution-addre
 7786 user      20   0 1126612  20128  11872 S   0.0  0.5   0:00.09 evolution-calen
 7830 user      20   0  903120  17620  11520 S   0.0  0.5   0:00.05 evolution-addre
 5613 polkitd   20   0  618772  17364   5180 S   0.0  0.4   0:08.88 polkitd
 7681 user      20   0  779588  16772   9508 S   0.0  0.4   0:13.57 gsd-color
 7836 user      20   0  530004  16592  10796 S   0.0  0.4   0:00.10 abrt-applet
 7524 user      20   0 1050940  16232  11972 S   0.0  0.4   0:00.07 evolution-sourc
 7758 user      20   0  916396  15976  11968 S   0.0  0.4   0:00.05 evolution-calen
 7691 user      20   0 1011876  15564  10104 S   0.0  0.4   0:00.37 gsd-media-keys
 7536 geoclue   20   0  462456  15364   6620 S   0.0  0.4   0:00.42 geoclue
```

Part Two: Linux Services

Services in Linux (sometimes referred to as **Daemons**) are system processes (or groups of processes) that work in the background waiting to be used. When in use, they perform essential tasks that provide a specific functionality to the system. In most cases services start when the system boots but can also be set to be manually started and stopped by the user.

Some examples of services that we will use include:

- **Network Service** – Provides network communication for the system.
- **sshd** – Provides secure access to the command shell from a remote computer.
- **firewalld** – Provides packet filtering for security.
- **samba** – Provides file sharing over a network connection.
- **Apache (httpd)** – Provides Web communication over http protocol.

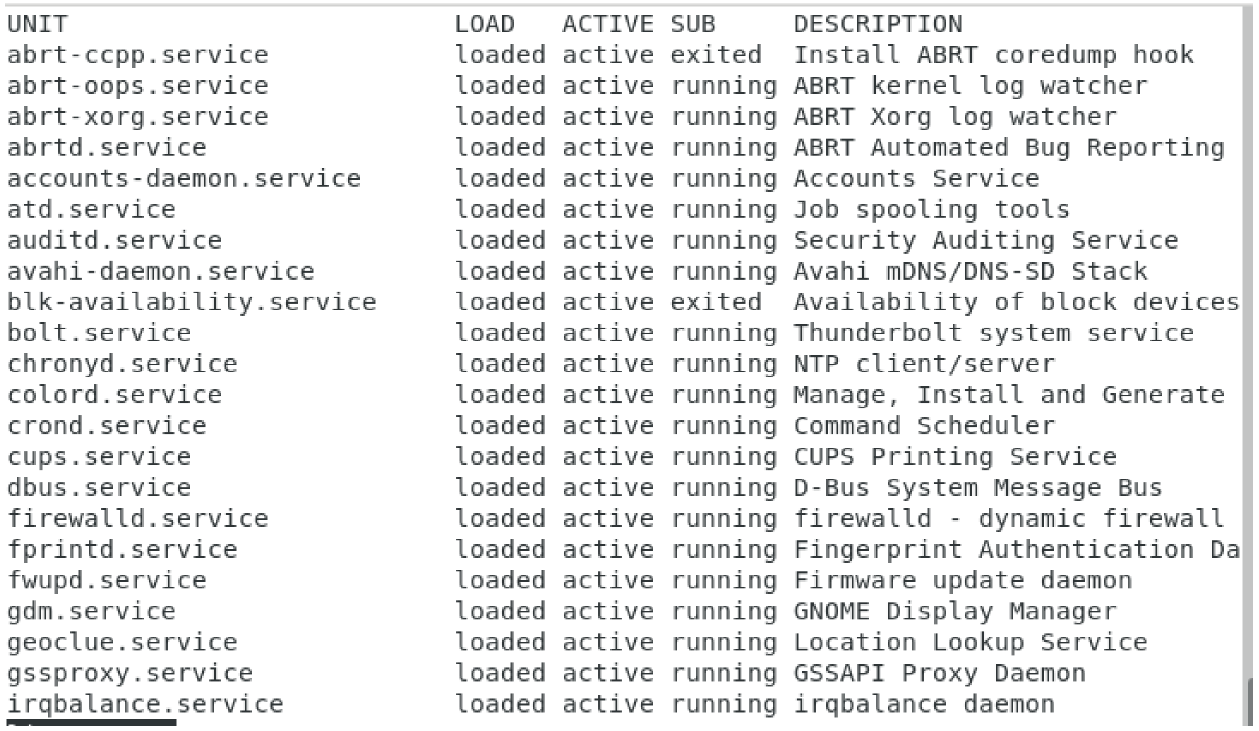
In modern Linux systems, services are controlled through a subsystem known as **systemd**. When a system boots, systemd calls what are known as **unit files** to start services. These unit files act as small applications that start the processes related to the services installed.

Services are only one type of unit file that is controlled by systemd. Other types of units include sockets, mounts, targets, devices, etc. These functions are outside of the scope of this course.

A systems administrator can manually control services through a systemd command "**systemctl**". With this command we can **start, stop, restart, enable, and disable** services as needed. We can also use this command to find information about the services installed on our system.

- Run the following command to view services currently installed on the system:

```
systemctl list-units --type service
```

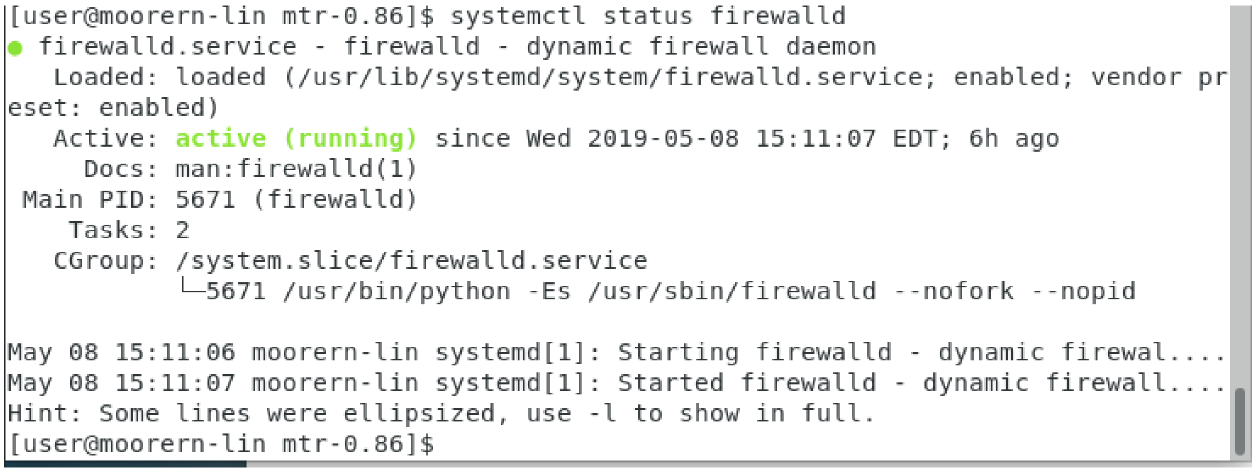


```
UNIT                         LOAD   ACTIVE SUB     DESCRIPTION
abrt-ccpp.service            loaded active exited  Install ABRT coredump hook
abrt-oops.service            loaded active running ABRT kernel log watcher
abrt-xorg.service            loaded active running ABRT Xorg log watcher
abrtd.service                loaded active running ABRT Automated Bug Reporting
accounts-daemon.service      loaded active running Accounts Service
atd.service                  loaded active running Job spooling tools
auditd.service               loaded active running Security Auditing Service
avahi-daemon.service         loaded active running Avahi mDNS/DNS-SD Stack
blk-availability.service     loaded active exited  Availability of block devices
bolt.service                 loaded active running Thunderbolt system service
chronyd.service              loaded active running NTP client/server
colord.service               loaded active running Manage, Install and Generate
crond.service                loaded active running Command Scheduler
cups.service                 loaded active running CUPS Printing Service
dbus.service                 loaded active running D-Bus System Message Bus
firewalld.service            loaded active running firewalld - dynamic firewall
fprintd.service              loaded active running Fingerprint Authentication Da
fwupd.service                loaded active running Firmware update daemon
gdm.service                  loaded active running GNOME Display Manager
geoclue.service              loaded active running Location Lookup Service
gssproxy.service             loaded active running GSSAPI Proxy Daemon
irqbalance.service           loaded active running irqbalance daemon
```

This list shows the names of the services, a description of what the service does, and whether it's **active** and **running** or not.

We can find out the status and more information from a single service using the "**status**" option.

- Run the following command to check the status of the "firewalld" service:
  `systemctl status firewalld`



```
[user@moorern-lin mtr-0.86]$ systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor pr
eset: enabled)
   Active: active (running) since Wed 2019-05-08 15:11:07 EDT; 6h ago
     Docs: man:firewalld(1)
 Main PID: 5671 (firewalld)
    Tasks: 2
   CGroup: /system.slice/firewalld.service
           └─5671 /usr/bin/python -Es /usr/sbin/firewalld --nofork --nopid

May 08 15:11:06 moorern-lin systemd[1]: Starting firewalld - dynamic firewal....
May 08 15:11:07 moorern-lin systemd[1]: Started firewalld - dynamic firewall....
Hint: Some lines were ellipsized, use -l to show in full.
[user@moorern-lin mtr-0.86]$
```

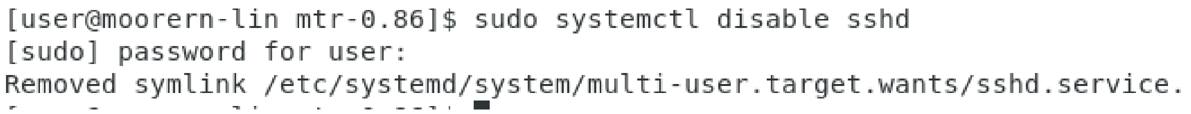
From this output, we can tell if the service is enabled, **active (running),** and how long it's been running for. It will also display the last few lines written to the system log file related to this service. This can be helpful when troubling shooting an issue.

When a service is **enabled**, that means the service is set to start at boot. Sometimes we may want to manually start services only when needed. For example, sshd, a service that provides secure shell access from remote computers, could be disabled when not needed to prevent potential attacks on that service.

To disable a service, we can use the disable option. The example below disables and enables sshd. Since we are connected to the VM through ssh, **I would suggest that students do not actually execute the disable command for ssh**

- the following command can be run to disable sshd:

  sudo systemctl disable sshd



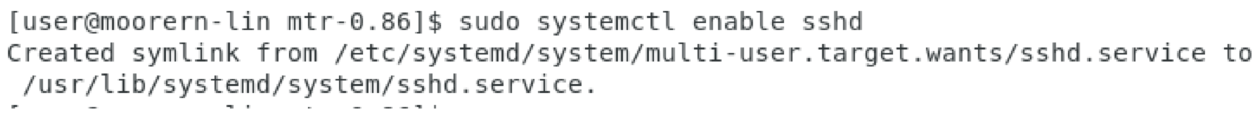
```
[user@moorern-lin mtr-0.86]$ sudo systemctl disable sshd
[sudo] password for user:
Removed symlink /etc/systemd/system/multi-user.target.wants/sshd.service.
```

What is happening when we disable a service is a symbolic link is removed from the directory systemd uses during boot. Without this link to the service unit, systemd does not start that service during the boot process.

If we wanted a service to start at boot, we must **enable** it to do so. Often, when we install a new service, we must manually enable the service.

- the following command can be run to enable sshd:

```
sudo systemctl enable sshd
```

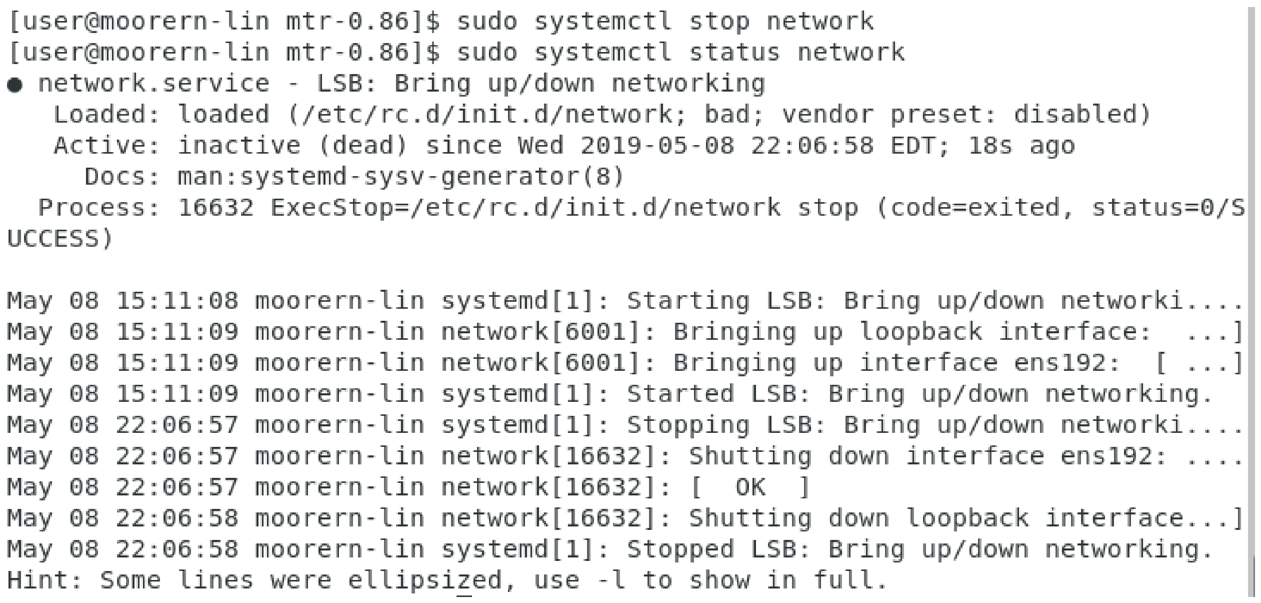


```
[user@moorern-lin mtr-0.86]$ sudo systemctl enable sshd
Created symlink from /etc/systemd/system/multi-user.target.wants/sshd.service to
 /usr/lib/systemd/system/sshd.service.
```

In this case, systemd created a symbolic link to the sshd unit file to start at boot.

Disabling and enabling does not change the current state of the service. To do this, we must use the options **stop and start**. With Linux on GCP, when you stop a network, then the CLI seems to stop working. You need to restart the system and open a new command line. **For this assignment it is sufficient if you learn how to stop/restart the network. You do not need to execute these commands in your GCP**

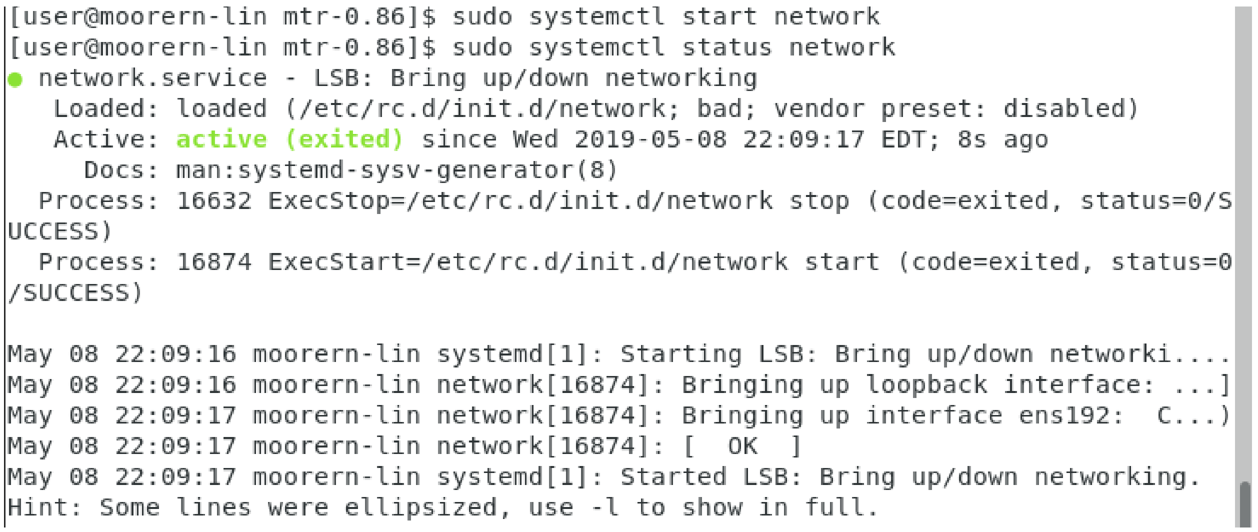
- Stop the network service using the following command:
  ```
  systemctl stop network
  ```
- You will not get any output from this command, but you can check the state of the service:
  ```
  systemctl status network
  ```



```
[user@moorern-lin mtr-0.86]$ sudo systemctl stop network
[user@moorern-lin mtr-0.86]$ sudo systemctl status network
● network.service - LSB: Bring up/down networking
   Loaded: loaded (/etc/rc.d/init.d/network; bad; vendor preset: disabled)
   Active: inactive (dead) since Wed 2019-05-08 22:06:58 EDT; 18s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 16632 ExecStop=/etc/rc.d/init.d/network stop (code=exited, status=0/S
UCCESS)

May 08 15:11:08 moorern-lin systemd[1]: Starting LSB: Bring up/down networki....
May 08 15:11:09 moorern-lin network[6001]: Bringing up loopback interface:  ...]
May 08 15:11:09 moorern-lin network[6001]: Bringing up interface ens192:  [ ...]
May 08 15:11:09 moorern-lin systemd[1]: Started LSB: Bring up/down networking.
May 08 22:06:57 moorern-lin systemd[1]: Stopping LSB: Bring up/down networki....
May 08 22:06:57 moorern-lin network[16632]: Shutting down interface ens192: ....
May 08 22:06:57 moorern-lin network[16632]: [  OK  ]
May 08 22:06:58 moorern-lin network[16632]: Shutting down loopback interface...]
May 08 22:06:58 moorern-lin systemd[1]: Stopped LSB: Bring up/down networking.
Hint: Some lines were ellipsized, use -l to show in full.
```

You can now see the active state is **inactive (dead)**. You may have also noticed you lost network connectivity. With this service stopped, network communications no longer work.

- Start the network service again with the following command:
  ```
  sudo systemctl start network
  ```
- Check the status of the service again. It should show as active.



```
[user@moorern-lin mtr-0.86]$ sudo systemctl start network
[user@moorern-lin mtr-0.86]$ sudo systemctl status network
● network.service - LSB: Bring up/down networking
   Loaded: loaded (/etc/rc.d/init.d/network; bad; vendor preset: disabled)
   Active: active (exited) since Wed 2019-05-08 22:09:17 EDT; 8s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 16632 ExecStop=/etc/rc.d/init.d/network stop (code=exited, status=0/S
UCCESS)
  Process: 16874 ExecStart=/etc/rc.d/init.d/network start (code=exited, status=0
/SUCCESS)

May 08 22:09:16 moorern-lin systemd[1]: Starting LSB: Bring up/down networki....
May 08 22:09:16 moorern-lin network[16874]: Bringing up loopback interface: ...]
May 08 22:09:17 moorern-lin network[16874]: Bringing up interface ens192:  C...)
May 08 22:09:17 moorern-lin network[16874]: [  OK  ]
May 08 22:09:17 moorern-lin systemd[1]: Started LSB: Bring up/down networking.
Hint: Some lines were ellipsized, use -l to show in full.
```

When we are installing and configuring new services, we often must make changes to configuration files. These configuration files are only read by the services when they start. Since most services start at boot, if we made a change, it would not take effect until we restarted the system. The good news is that this is not necessary as we can use the **"restart"** option to quickly reload a service with the new configuration.

- Restart the network service using the following command:

```
systemctl restart network
```

Part Three: Scheduling Tasks

Sometimes we may need to run a process repeatedly or on a regular schedule. Perhaps we want to run backups every hour, trouble shoot a network issue, or perhaps just to create our own log files to monitor something on our system.

Manually doing these repetitive tasks can take away valuable time from other work, lead to mistakes, and just be incredibly boring. A good systems administrator will find ways to automate these types of tasks. There are many different tools that can help with automation, but the two most basic tools that are free and easy to use are **scripts** and **cronjobs**.

Scripting

Writing out a series of commands repeatedly is very tedious. Scripting is a way for us to create a simple "program" using the command line that allows us to save that same series of commands. At the most basic level, a script is nothing more than a text file that contains a list of commands. It is possible to create complex scripts with variables, user input, formatted output, and many other features. For now, let us just focus on the basics.

For an example, we will create a simple script that will monitor established connections to our server and create a log file with time stamps. To do this, we use a couple new commands, **netstat** and **date**.

The 'netstat' command displays network connections to our system.

The 'date' command simply displays that date and time on the system. This is useful for creating timestamps in a script.

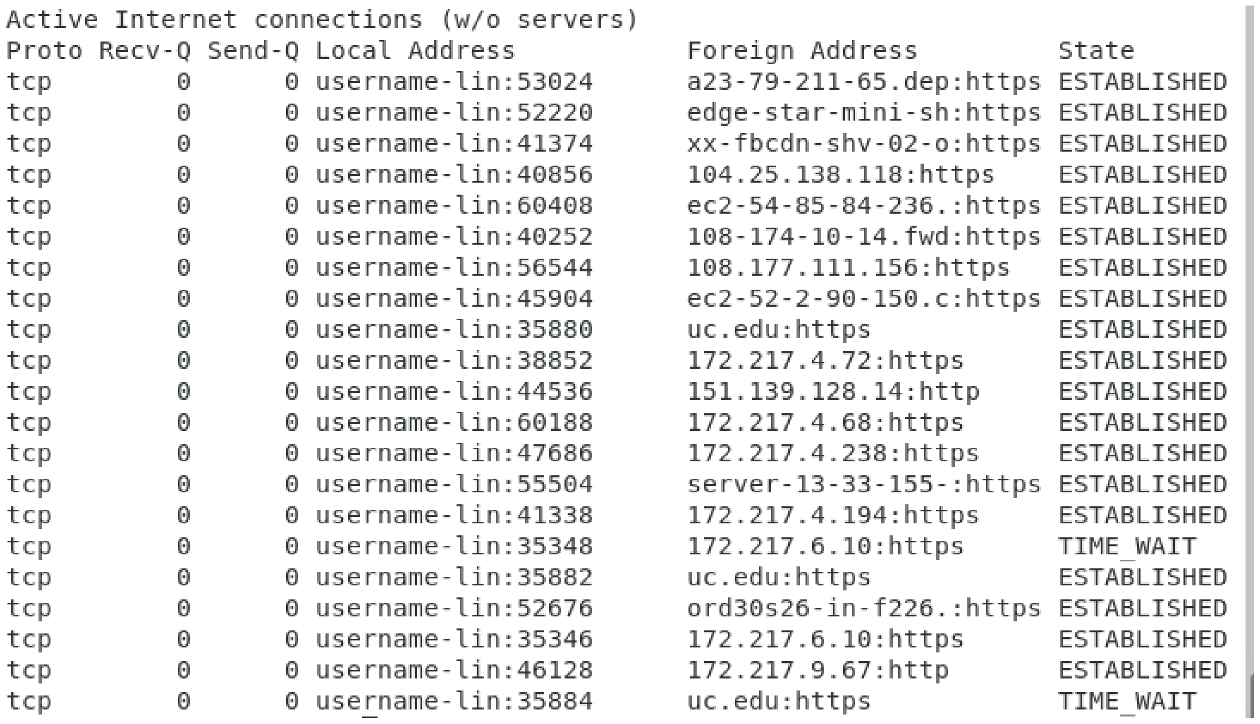
Before we begin, if you have RDP for GUI set up correctly, log into your VM using RDP, open a **Firefox** web browser to **google.com** to establish some connections. You can open two or three websites using Firefox tabs. **If you don't have RDP set up to connect to the GUI for the VM, please go ahead and use links to access a website on command line:**

- sudo links www.google.com (You can open one more terminal window and access another website as well as this will provide us with more information when we run netstat.)
- You will have to use your keyboard to navigate this text-based version of the website.
- Keep this Command Line window open.
- Open a new command line (gear symbol on the top right ->new connection), switch to user account : su -l user

Let us start by figuring out the exact commands we want to use. The netstat command by itself displays a lot of information. Using the **-t** option, we can limit the results to only TCP connections. This will display the web site connections made to our system.

- Run the command

```
netstat -t
```



```
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 username-lin:53024      a23-79-211-65.dep:https ESTABLISHED
tcp        0      0 username-lin:52220      edge-star-mini-sh:https ESTABLISHED
tcp        0      0 username-lin:41374      xx-fbcdn-shv-02-o:https ESTABLISHED
tcp        0      0 username-lin:40856      104.25.138.118:https    ESTABLISHED
tcp        0      0 username-lin:60408      ec2-54-85-84-236.:https ESTABLISHED
tcp        0      0 username-lin:40252      108-174-10-14.fwd:https ESTABLISHED
tcp        0      0 username-lin:56544      108.177.111.156:https   ESTABLISHED
tcp        0      0 username-lin:45904      ec2-52-2-90-150.c:https ESTABLISHED
tcp        0      0 username-lin:35880      uc.edu:https            ESTABLISHED
tcp        0      0 username-lin:38852      172.217.4.72:https      ESTABLISHED
tcp        0      0 username-lin:44536      151.139.128.14:http     ESTABLISHED
tcp        0      0 username-lin:60188      172.217.4.68:https      ESTABLISHED
tcp        0      0 username-lin:47686      172.217.4.238:https     ESTABLISHED
tcp        0      0 username-lin:55504      server-13-33-155-:https ESTABLISHED
tcp        0      0 username-lin:41338      172.217.4.194:https     ESTABLISHED
tcp        0      0 username-lin:35348      172.217.6.10:https      TIME_WAIT
tcp        0      0 username-lin:35882      uc.edu:https            ESTABLISHED
tcp        0      0 username-lin:52676      ord30s26-in-f226.:https ESTABLISHED
tcp        0      0 username-lin:35346      172.217.6.10:https      ESTABLISHED
tcp        0      0 username-lin:46128      172.217.9.67:http       ESTABLISHED
tcp        0      0 username-lin:35884      uc.edu:https            TIME_WAIT
```

The above screen shot shows the results you should see. If you browse to a different web site, you will see different connections made.

Now we need to enter this command into a text file and change the permissions to be **executable**. (Please refer to Assignment 3 – Part 5 – Working with Text Files to refresh your memory on how to create/edit/save text files)

- Open a new text file with **vi** called "**connections.sh**"

```
vi connections.sh
```

- Add the text as follows and save the file:

```
#!/bin/bash

netstat -t
```

The first line here is known as a "**shebang**" or a "**hashbang**". This tells the system which type of shell to run the script in. Since we are using the BASH shell, we specify the **/bin/bash** file path, which is knowns as the "**interpreter directive**". Save and close the file.

Next, we must change the permissions so the text file is executable.

- Run the command

```
chmod 755 connections.sh
```

Now, let's test out our new script. In order to execute a script, you must enter the full path to the file or specify "**./**" for the relative path.

- Enter the following into the command prompt

```
./connections.sh
```

You should see similar results as when you ran the netstat command itself. But running just a single command in a script doesn't make much sense. Let's add a little more to make this more useful. You may have noticed that some connections had a different state, some were **ESTABLISHED** and others had the state of **TIME_WAIT**. The latter are not currently active, so let's edit our script to display two results, one for each state.

- Edit the connections.sh file to match the following:

```
#!/bin/bash

netstat -t | grep ESTABLISHED

netstat -t | grep TIME_WAIT
```

- Make sure you have at least one text-based website open by using links in another command line window (If you can RDP to your GUI interface for Linux, open the browser to a website or refresh your browser to make sure we have current connections), then run the script again.

Finally, let's add a time stamp and place the output into a log file.

- Edit the connection file again to match the following:

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

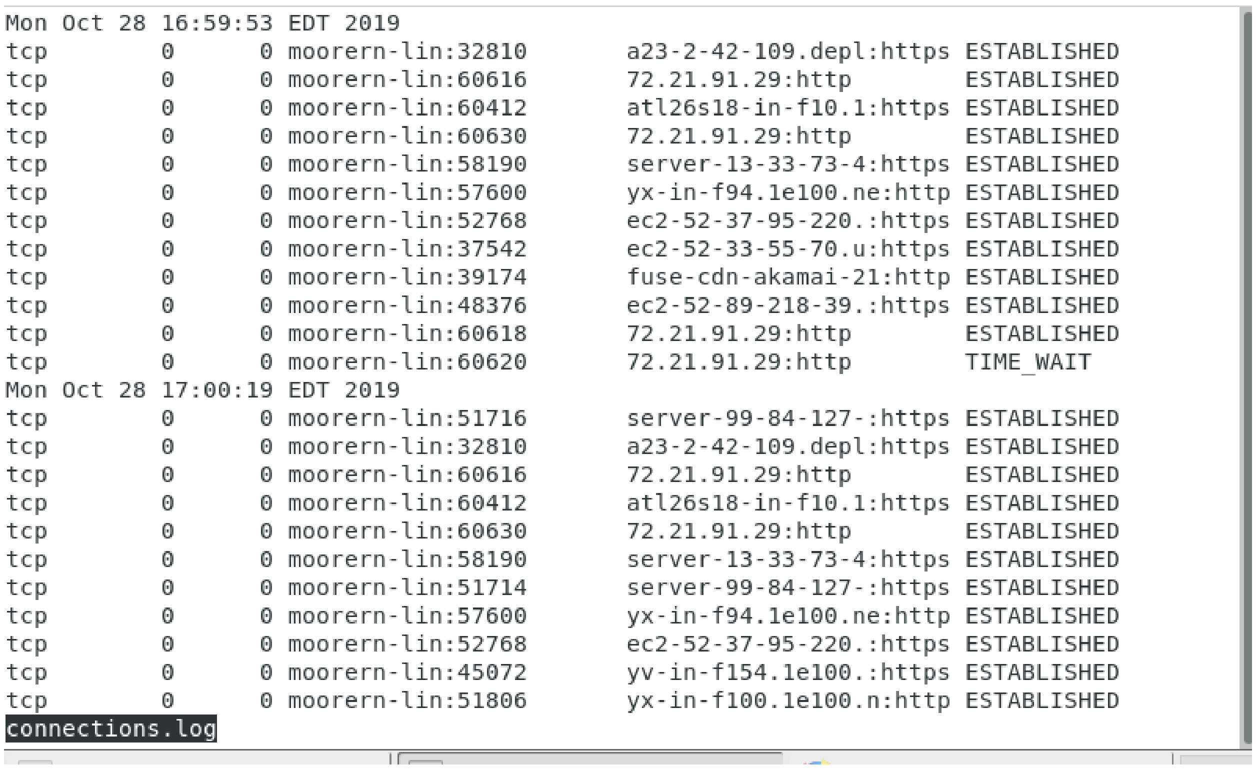
```
date >> connections.log

netstat -t | grep ESTABLISHED >> connections.log

netstat -t | grep TIME_WAIT >> connections.log
```

Reminder: The ">>" takes the output from a command and appends it to a text file.

Refresh your browser and run the script again. If you are using the text-based links, make sure the link to the website is still valid or open a link to another website in a new terminal window. This time you will not see any output to the screen. This is because we sent all the information to a new file called "**connectons.log**". Open this file to see the results. Notice that a time stamp was placed at the top. This will be useful to separate results as they are run at different times.

3. **Open another website (Command Line should use links in a new command window and use links to open the website, RDP GUI can use Firefox) and run the script one more time. Take a screen shot of the log file showing the new timestamp and some of the new entries. Save this to upload in the assignment 6 assessment.**

Details may vary but should show time stamp and connection information.

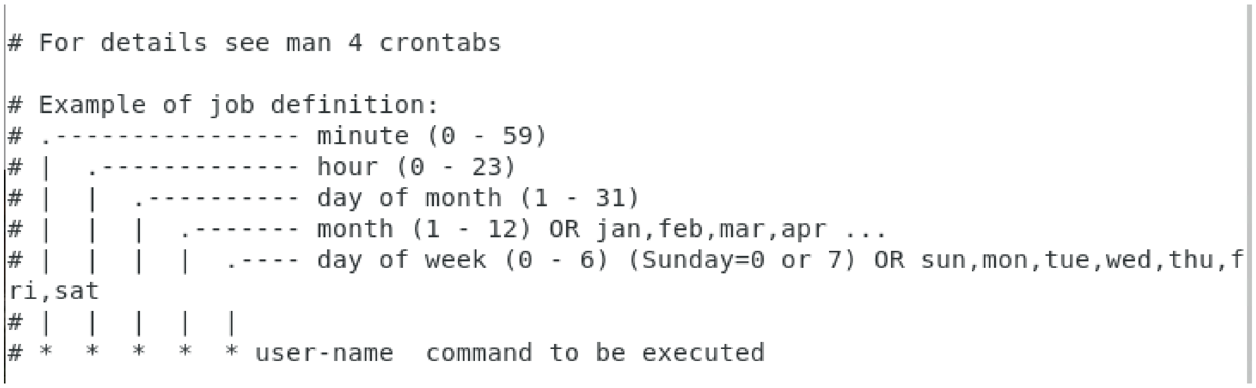


```
Mon Oct 28 16:59:53 EDT 2019
tcp        0      0 moorern-lin:32810       a23-2-42-109.depl:https ESTABLISHED
tcp        0      0 moorern-lin:60616       72.21.91.29:http        ESTABLISHED
tcp        0      0 moorern-lin:60412       atl26s18-in-f10.1:https ESTABLISHED
tcp        0      0 moorern-lin:60630       72.21.91.29:http        ESTABLISHED
tcp        0      0 moorern-lin:58190       server-13-33-73-4:https ESTABLISHED
tcp        0      0 moorern-lin:57600       yx-in-f94.1e100.ne:http ESTABLISHED
tcp        0      0 moorern-lin:52768       ec2-52-37-95-220.:https ESTABLISHED
tcp        0      0 moorern-lin:37542       ec2-52-33-55-70.u:https ESTABLISHED
tcp        0      0 moorern-lin:39174       fuse-cdn-akamai-21:http ESTABLISHED
tcp        0      0 moorern-lin:48376       ec2-52-89-218-39.:https ESTABLISHED
tcp        0      0 moorern-lin:60618       72.21.91.29:http        ESTABLISHED
tcp        0      0 moorern-lin:60620       72.21.91.29:http        TIME_WAIT
Mon Oct 28 17:00:19 EDT 2019
tcp        0      0 moorern-lin:51716       server-99-84-127-:https ESTABLISHED
tcp        0      0 moorern-lin:32810       a23-2-42-109.depl:https ESTABLISHED
tcp        0      0 moorern-lin:60616       72.21.91.29:http        ESTABLISHED
tcp        0      0 moorern-lin:60412       atl26s18-in-f10.1:https ESTABLISHED
tcp        0      0 moorern-lin:60630       72.21.91.29:http        ESTABLISHED
tcp        0      0 moorern-lin:58190       server-13-33-73-4:https ESTABLISHED
tcp        0      0 moorern-lin:51714       server-99-84-127-:https ESTABLISHED
tcp        0      0 moorern-lin:57600       yx-in-f94.1e100.ne:http ESTABLISHED
tcp        0      0 moorern-lin:52768       ec2-52-37-95-220.:https ESTABLISHED
tcp        0      0 moorern-lin:45072       yv-in-f154.1e100.:https ESTABLISHED
tcp        0      0 moorern-lin:51806       yx-in-f100.1e100.n:http ESTABLISHED
connections.log
```

Cron

Cron is a utility built into the Linux system that allows you to schedule jobs (commands or scripts) to run on a periodic schedule. These tasks are commonly referred to as "**cron jobs**". Cron jobs are scheduled in the **crontab** (cron table), which is a text configuration file used to store multiple instructions.

Before setting up cron jobs, we must first understand the format of the crontab file. The crontab file is laid out like a table, with 7 different columns each separated by a space and a row for each instruction. The following shows an example of the job definition format:



```
# For details see man 4 crontabs

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,f
ri,sat
# |  |  |  |  |
# *  *  *  *  * user-name  command to be executed
```

- Column 1 – This defines the minute of the hour to execute the job, from 0-59
- Column 2 – This defines the hour of the day to execute the job, from 0-23
- Column 3 – This defines the day of the month to execute the job, from 1-31
- Column 4 – This defines the month in which to execute the job, from 1-12
- Column 5 – This defines the day of the week to execute the job, from 0-7 (Sundays can be 0 or 7)
- Column 6 – This defines the user in which to run the job as. If left blank, it will be the user that created the job.
- Column 7 – This is the command or path to the script to be executed.

When setting the schedule, if a column is not needed, simply enter an **Asterix (*)** for the column. If you need to set multiple times for a column, you can use the following operators:

- ***/#** - This will repeat the job every interval. For example, */15 will repeat every 15 minutes
- **#,#,#** - Separating by commas allows for multiple entries that are not in sequence. For example, 1,3,5 in the day of the week column would run on Monday, Wednesday, and Friday
- **#-#** - Placing a dash will run the job for multiple entries in sequence. For example, 6,7,8 in the month column will run the job during the months of June, July, and August.

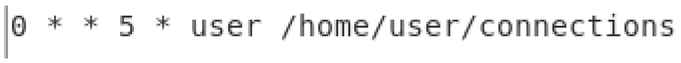
To edit the crontab file, we will use the command **crontab**. This will open the file in the **vi** text editor. Specifying the **-e** option will open the current crontab for the logged in user.

Let's go ahead and create a cronjob that will run our script every hour during the month of May.

- Open the crontab with the following command:

  ```
  crontab -e
  ```

- Edit the file with the following schedule on the first line:

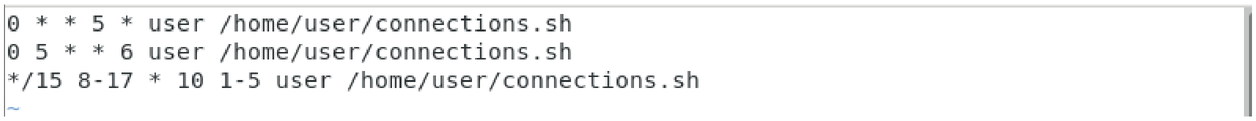
  ```
  0 * * 5 * user /home/user/connections
  ```

In this example, the first 0 means the job will run at the top of every hour. Since it will run every hour, we used the * for the second column. Since it will run every day, we used the * for the third column. In the fourth column, since we specified the number 5, it will only run in the 5th month of May. The fifth column once again has an * since it will run every day, regardless of which day it is. We specified the user name but could have left this blank. Sometimes you may need to run a job as a root user. Finally, the last column is the absolute path of the script. If you do not specify the full path, the job will not run.

- Create two new jobs that will run the script according to the following schedules:
    - Every Saturday morning at 5am
    - Every 15 minutes, between 8am-5pm, Monday-Friday, during this month.

4. **Take a screen shot of all 3 schedules in the crontab file and save them to upload for the assignment 6 assessment.**

Month column (4th column) will vary depending on the month this assignment is given.



```
0 * * 5 * user /home/user/connections.sh
0 5 * * 6 user /home/user/connections.sh
*/15 8-17 * 10 1-5 user /home/user/connections.sh
~
```