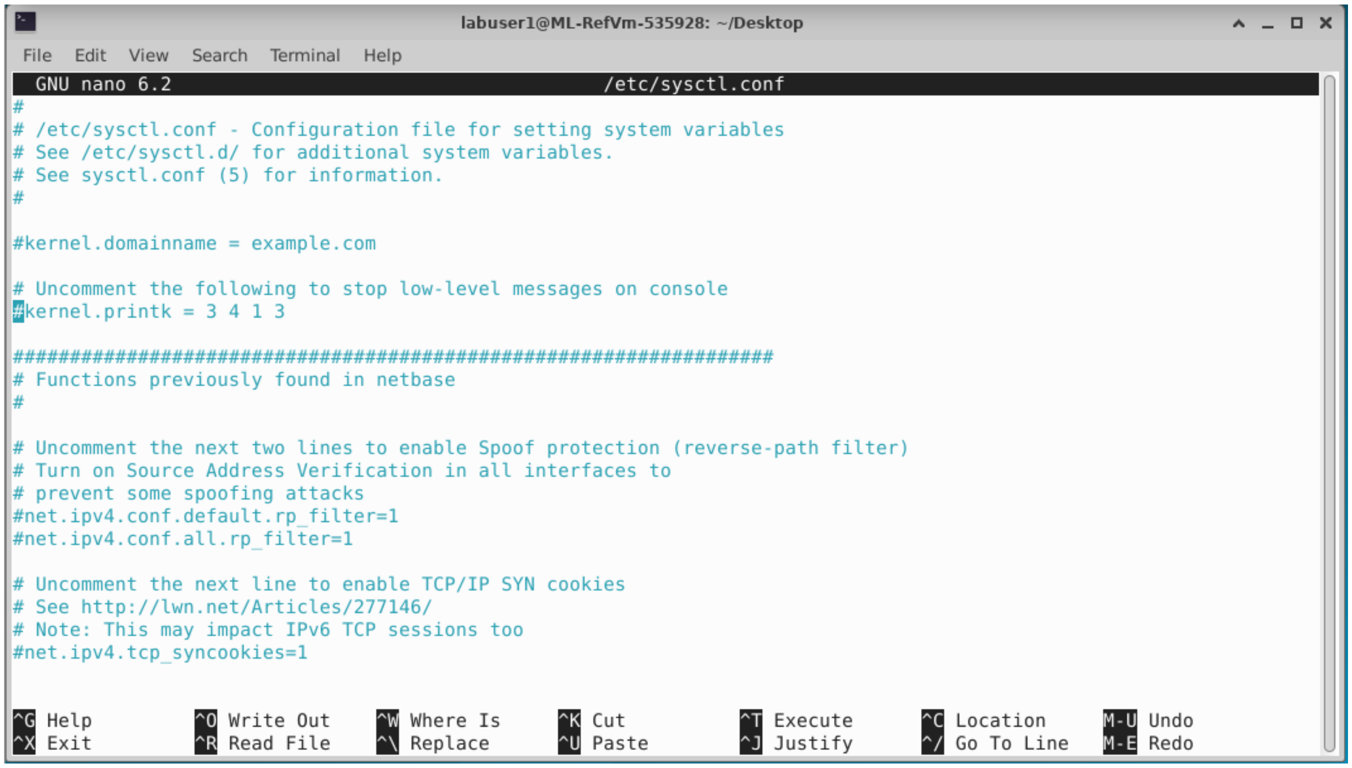
COSC 3364 – Principles of Cybersecurity
Lab 08

Provide screenshots where * is indicated.

Network Security Policy

You can use kernel parameters to modify the behavior of the kernel by adjusting key features. They can be used to change how the kernel manages devices, optimize memory usage, and enhance the security of the system.

- Navigate to the system control configuration file: **/etc/sysctl.conf**
- Determine which kernel parameters are enabled* (only a portion of the file is required for the screenshot)


labuser1@ML-RefVm-535928: ~/Desktop
File Edit View Search Terminal Help

```
  GNU nano 6.2                                 /etc/sysctl.conf
#
# /etc/sysctl.conf - Configuration file for setting system variables
# See /etc/sysctl.d/ for additional system variables.
# See sysctl.conf (5) for information.
#

#kernel.domainname = example.com

# Uncomment the following to stop low-level messages on console
#kernel.printk = 3 4 1 3

###################################################################
# Functions previously found in netbase
#

# Uncomment the next two lines to enable Spoof protection (reverse-path filter)
# Turn on Source Address Verification in all interfaces to
# prevent some spoofing attacks
#net.ipv4.conf.default.rp_filter=1
#net.ipv4.conf.all.rp_filter=1

# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lwn.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo
```

None of the parameters are currently enabled because they are all commented out.

Possible kernel parameters:

a. **Ignoring ping requests** – the ping command is often used to determine if a remote host is accessible through the network. An adversary can use ping to probe for active systems trying to find systems that they can break into. Responding to ping requests can leave a system vulnerable to denial of service attacks. To ignore ping requests, use the following setting in the **/etc/sysctl.conf:**

```
net.ipv4.icmp_echo_ignore_all = 1
```

b. **Ignoring Broadcast Requests** – Broadcast requests can be used for DoS and DDoS attacks. To ignore broadcast requests, use the following setting in the **/etc/sysctl.conf:**

```
net.ipv4.icmp_echo_ignore_broadcasts = 1
```

c. **Enabling TCP SYN Protection** – A SYN flood attack is another DoS attack where SYN requests are used to make a system unresponsive. To Ignore SYS requests, use the following setting in the **/etc/sysctl.conf:**

```
net.ipv4.tcp_syncookies = 1
```

d. **Disabling IP Source Routing** – a feature that enables the sender of a packet to specify the network route that should be taken. This feature bypasses routing tables and makes your system vulnerable to man-in-the-middle attacks. To disable this feature from a specific network device, use the following setting in the **/etc/sysctl.conf:**

```
net.ipv4.conf.eth0.accept_source_route = 0
```

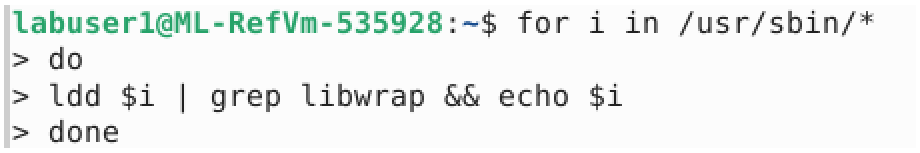
TCP wrappers are used when server programs that have been compiled with the **libwrap** library call that library when a system tries to access the service. An easy way to determine whether a service uses the **libwrap** library is to use the **ldd** command:

```
ldd <program>| grep libwrap
```

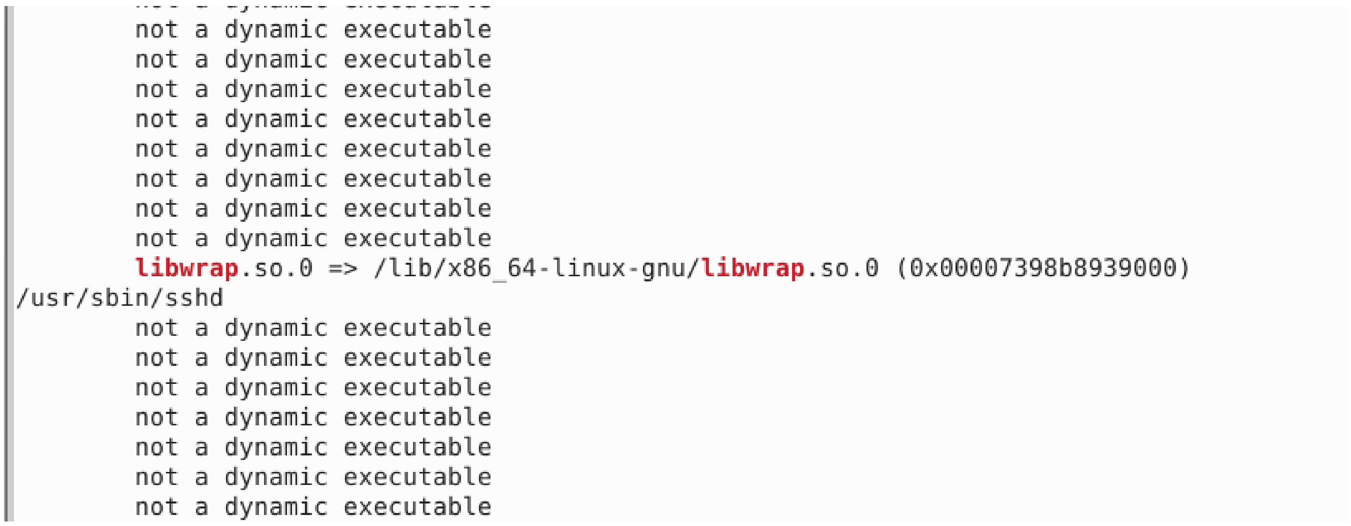
if the command returns **libwrap.so.0**, then the program uses TCP Wrappers.

- Execute to determine which program(s) use TCP Wrappers*:

```
for i in /usr/sbin/*
>do
>ldd $i | grep libwrap && echo $i
>done
```

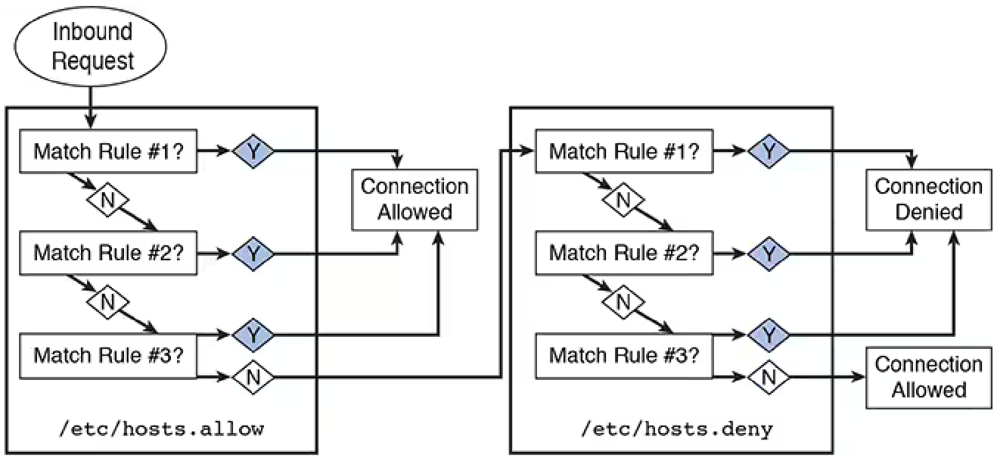


```
labuser1@ML-RefVm-535928:~$ for i in /usr/sbin/*
> do
> ldd $i | grep libwrap && echo $i
> done
```



```
        not a dynamic executable
        not a dynamic executable
        not a dynamic executable
        not a dynamic executable
        not a dynamic executable
        not a dynamic executable
        not a dynamic executable
        not a dynamic executable
        libwrap.so.0 => /lib/x86_64-linux-gnu/libwrap.so.0 (0x00007398b8939000)
/usr/sbin/sshd
        not a dynamic executable
        not a dynamic executable
        not a dynamic executable
        not a dynamic executable
        not a dynamic executable
        not a dynamic executable
        not a dynamic executable
```

The **libwrap** library uses configuration files to determine whether the SSH connection should be allowed, based on what machine is initiating the connection. The files used are the **/etc/hosts.allow** and **/etc/hosts.deny** files.



The syntax of the rules in the **/etc/hosts.allow** and **/etc/hosts.deny** files is

```
service_list: client_list [options]
```

The service is the name of the binary executable service program (for example, sshd or xinetd). The client list is what system(s) this rule should apply to.

The **client_list** is also flexible. The following list details the different values you can provide:

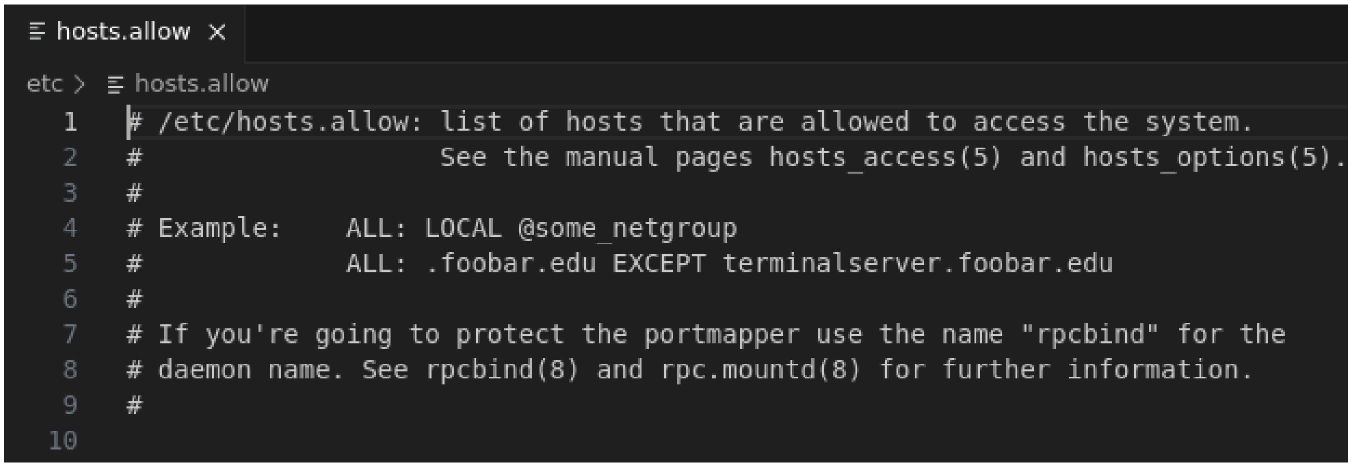
- **IP address**: Example: 192.168.0.100.
- **Network**: Example: 192.168.0.0/255.255.255.0 or 192.168.0.
- **Entire domain**: Example: **.example.com**.
- **ALL**: Matches every possible client.
- **LOCAL**: Matches clients without a dot in their hostname. Example: test1.
- **UNKNOWN**: Matches clients that can't be resolved via the hostname resolver (DNS, local hosts file, and so on).
- **KNOWN**: Matches clients that can be resolved via the hostname resolver (DNS, local hosts file, and so on).

Example: sshd: test.onecoursesource.com

Example: xinetd,sshd: test.onecoursesource.com

Example: ALL: test.onecoursesource.com

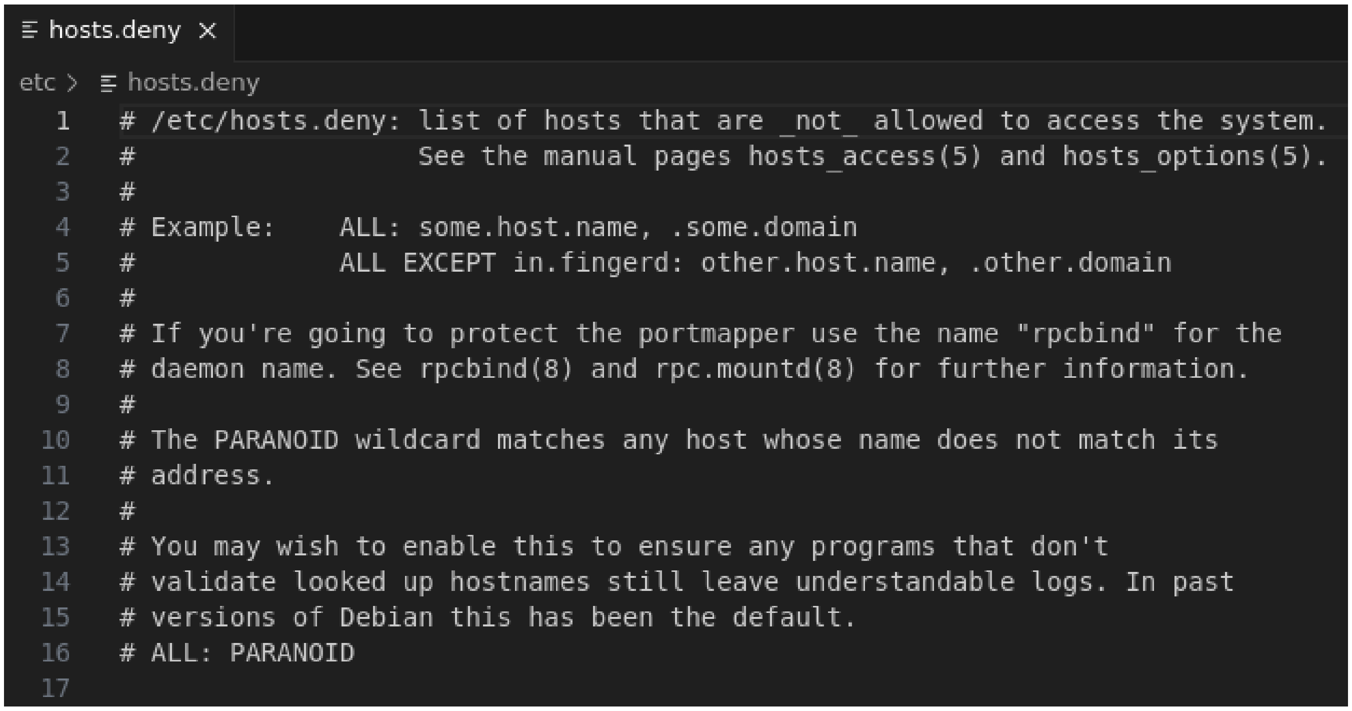
- Navigate to **/etc/hosts.allow** and determine what connections are allowed*



```
hosts.allow
etc > hosts.allow
# /etc/hosts.allow: list of hosts that are allowed to access the system.
#                   See the manual pages hosts_access(5) and hosts_options(5).
#
# Example:    ALL: LOCAL @some_netgroup
#             ALL: .foobar.edu EXCEPT terminalserver.foobar.edu
#
# If you're going to protect the portmapper use the name "rpcbind" for the
# daemon name. See rpcbind(8) and rpc.mountd(8) for further information.
#

```

No connections are currently allowed.

- Navigate to **/etc/hosts.deny** and determine what connections are denied*



```
hosts.deny
etc > hosts.deny
# /etc/hosts.deny: list of hosts that are _not_ allowed to access the system.
#                  See the manual pages hosts_access(5) and hosts_options(5).
#
# Example:    ALL: some.host.name, .some.domain
#             ALL EXCEPT in.fingerd: other.host.name, .other.domain
#
# If you're going to protect the portmapper use the name "rpcbind" for the
# daemon name. See rpcbind(8) and rpc.mountd(8) for further information.
#
# The PARANOID wildcard matches any host whose name does not match its
# address.
#
# You may wish to enable this to ensure any programs that don't
# validate looked up hostnames still leave understandable logs. In past
# versions of Debian this has been the default.
# ALL: PARANOID

```

No connections are currently denied.

Process Control

The **ps** command is used to list processes that are running on the system. With no arguments, the command will list any child process of the current shell as well as the BASH shell itself.

Each line describes one process. By default, the **ps** command displays the following information:

- **PID**: Process ID number; each process has a unique ID that can be used to control the process.

- **TTY**: This is the terminal window that started the process. A terminal window is essentially a place where a user is able to issue commands from the command line.
- **TIME**: CPU time; how much time the process has used to execute code on the CPU. Although this number can grow over time, it is typically a very small number (a handful of seconds or maybe a few minutes, but rarely more), unless something is wrong with the process.
- **CMD**: The command.

To list all processes running on the system, add the **-e** option. The command **wc -l** can be used with the **ps** command to display the number of lines of data produced by the command to determine the number of processes running on a system. Each process is displayed on a separate line, so to be able to display a specific process you will want to use a **grep** command to filter the output.

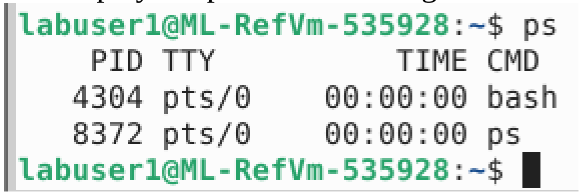
The **top** command displays process information that is updated on a regular basis (by default, every two seconds). The first half of the output of the top command contains overall information, whereas the second half displays a select list of processes (by default, the processes with the most CPU utilization).

The **free** command displays memory statistics.

To execute a process in the background, add an ampersand (**&**) character to the end of the command. Running a process in the background allows you to continue to work in the BASH shell and execute additional commands. Each BASH shell keeps track of the processes that are running from that BASH shell. These processes are referred to as jobs. To list the currently running jobs, execute the **jobs** command from the BASH shell.

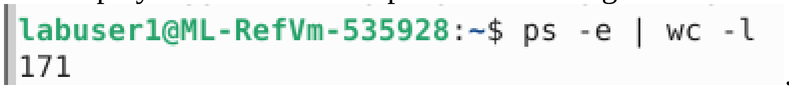
The phrase "kill a process" is used to describe when you completely stop a process. Several methods are available: the **kill** command, the **pkill** command, the **killall** command and the **xkill** command. The **kill** command can be used to change the state of a process, including to stop (kill) a process. To stop a process, first determine its process ID or (%) job number and then provide that number as an argument to the **kill** command. You can kill a process by running the xkill command and then just clicking the process that you want to stop.

- Display the processes running*



```
labuser1@ML-RefVm-535928:~$ ps
    PID TTY          TIME CMD
   4304 pts/0    00:00:00 bash
   8372 pts/0    00:00:00 ps
labuser1@ML-RefVm-535928:~$ 
```

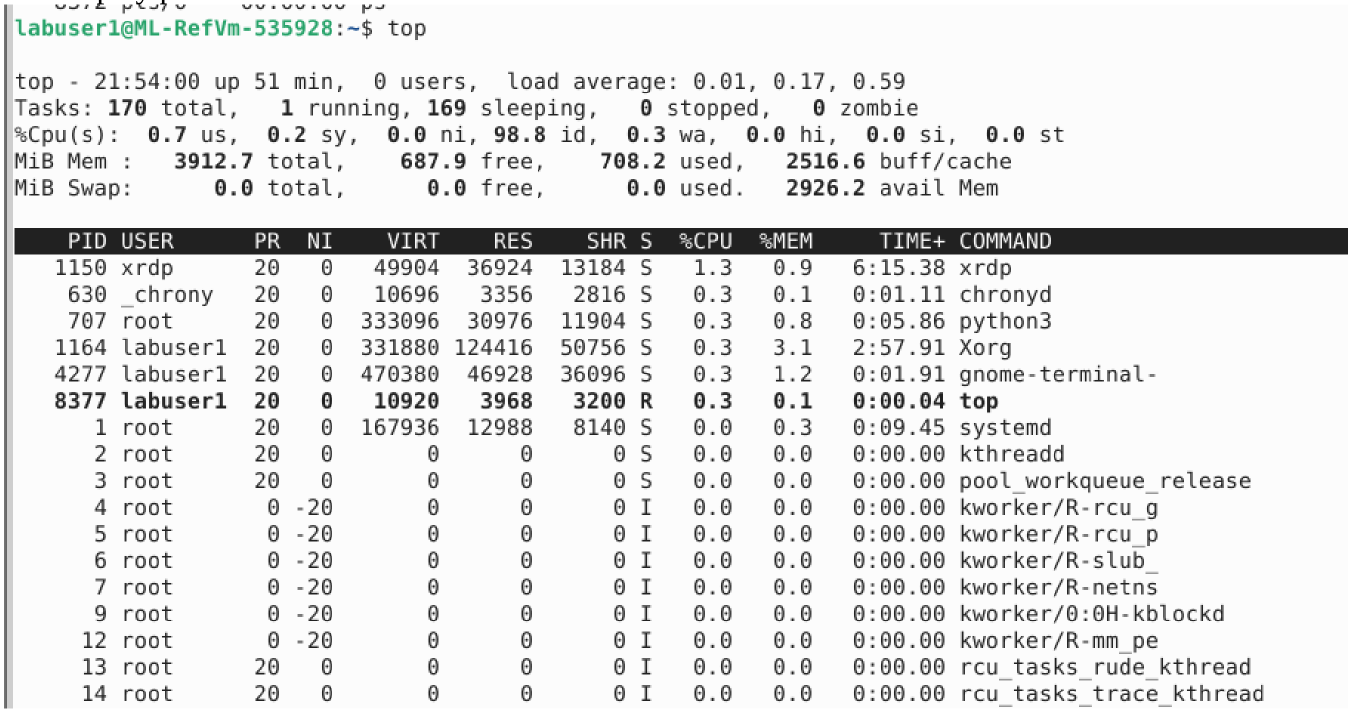
- Display the number of all processes running*



```
labuser1@ML-RefVm-535928:~$ ps -e | wc -l
171
```

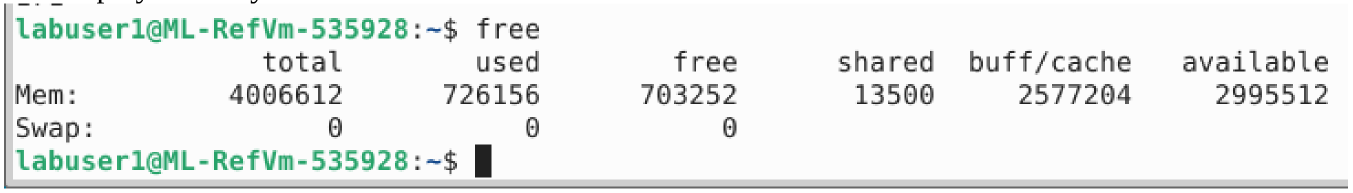
.

- Display process information*



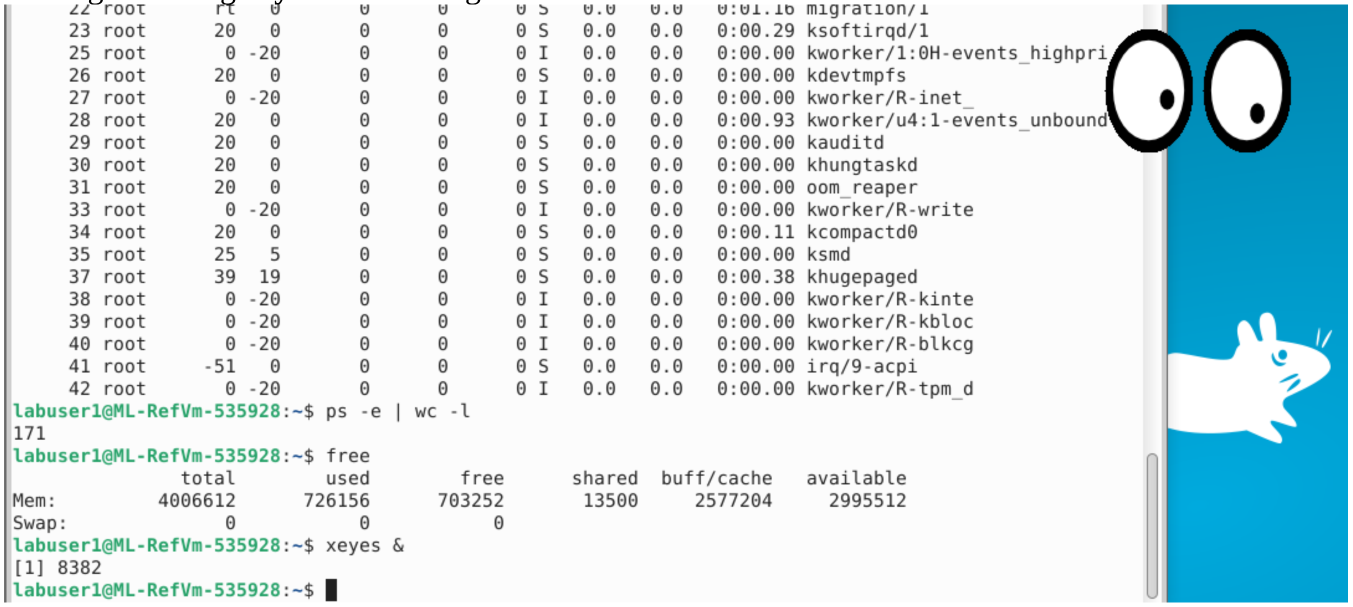
```
labuser1@ML-RefVm-535928:~$ top

top - 21:54:00 up 51 min,  0 users,  load average: 0.01, 0.17, 0.59
Tasks: 170 total,   1 running, 169 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.7 us,  0.2 sy,  0.0 ni, 98.8 id,  0.3 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   3912.7 total,    687.9 free,    708.2 used,   2516.6 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used.   2926.2 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   1150 xrdp      20   0   49904  36924  13184 S   1.3   0.9   6:15.38 xrdp
    630 _chrony   20   0   10696   3356   2816 S   0.3   0.1   0:01.11 chronyd
    707 root      20   0  333096  30976  11904 S   0.3   0.8   0:05.86 python3
   1164 labuser1  20   0  331880 124416  50756 S   0.3   3.1   2:57.91 Xorg
   4277 labuser1  20   0  470380  46928  36096 S   0.3   1.2   0:01.91 gnome-terminal-
   8377 labuser1  20   0   10920   3968   3200 R   0.3   0.1   0:00.04 top
      1 root      20   0  167936  12988   8140 S   0.0   0.3   0:09.45 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kthreadd
      3 root      20   0       0      0      0 S   0.0   0.0   0:00.00 pool_workqueue_release
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-rcu_g
      5 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-rcu_p
      6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-slub_
      7 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-netns
      9 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/0:0H-kblockd
     12 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-mm_pe
     13 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tasks_rude_kthread
     14 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tasks_trace_kthread
```

- Display memory statistics*



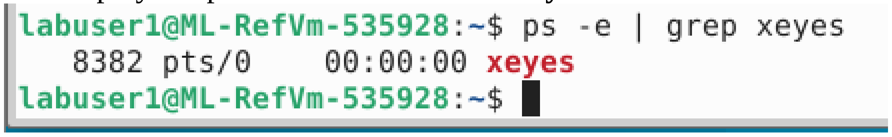
```
labuser1@ML-RefVm-535928:~$ free
               total        used        free      shared  buff/cache   available
Mem:         4006612      726156      703252       13500     2577204     2995512
Swap:              0           0           0
labuser1@ML-RefVm-535928:~$ 
```

- Begin running **xeyes** in the background*



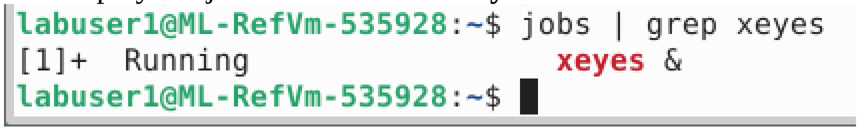
```
     23 root      20   0       0      0      0 S   0.0   0.0   0:00.29 ksoftirqd/1
     25 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/1:0H-events_highpri
     26 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kdevtmpfs
     27 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-inet_
     28 root      20   0       0      0      0 I   0.0   0.0   0:00.93 kworker/u4:1-events_unbound
     29 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kauditd
     30 root      20   0       0      0      0 S   0.0   0.0   0:00.00 khungtaskd
     31 root      20   0       0      0      0 S   0.0   0.0   0:00.00 oom_reaper
     33 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-write
     34 root      20   0       0      0      0 S   0.0   0.0   0:00.11 kcompactd0
     35 root      25   5       0      0      0 S   0.0   0.0   0:00.00 ksmd
     37 root      39  19       0      0      0 S   0.0   0.0   0:00.38 khugepaged
     38 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-kinte
     39 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-kbloc
     40 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-blkcg
     41 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 irq/9-acpi
     42 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-tpm_d
labuser1@ML-RefVm-535928:~$ ps -e | wc -l
171
labuser1@ML-RefVm-535928:~$ free
               total        used        free      shared  buff/cache   available
Mem:         4006612      726156      703252       13500     2577204     2995512
Swap:              0           0           0
labuser1@ML-RefVm-535928:~$ xeyes &
[1] 8382
labuser1@ML-RefVm-535928:~$ 
```

- Display the process information for **xeyes***



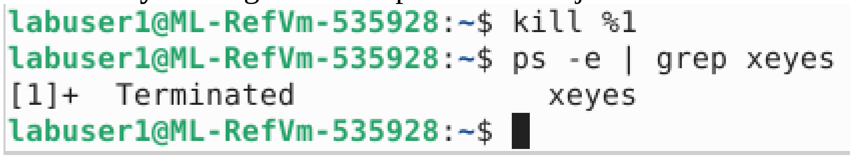
```
labuser1@ML-RefVm-535928:~$ ps -e | grep xeyes
   8382 pts/0    00:00:00 xeyes
labuser1@ML-RefVm-535928:~$ 
```

- Display the job information for **xeyes***



```
labuser1@ML-RefVm-535928:~$ jobs | grep xeyes
[1]+  Running                 xeyes &
labuser1@ML-RefVm-535928:~$ 
```

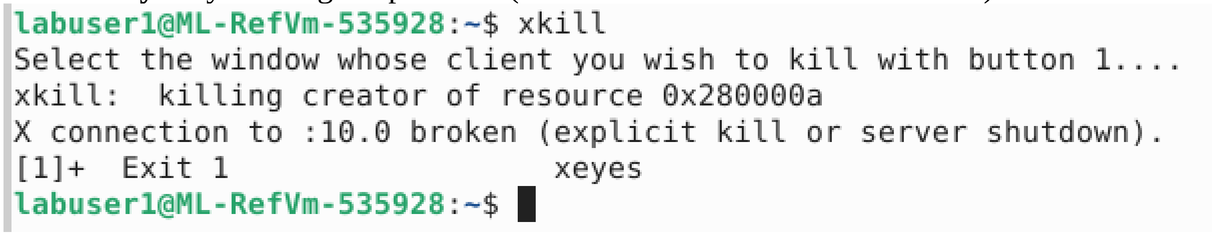
- Kill **xeyes** using either the process ID or job number*



```
labuser1@ML-RefVm-535928:~$ kill %1
labuser1@ML-RefVm-535928:~$ ps -e | grep xeyes
[1]+  Terminated              xeyes
labuser1@ML-RefVm-535928:~$ 
```

- Begin running **xeyes**

- Kill **xeyes** by clicking the process* (the screenshot of the command used)

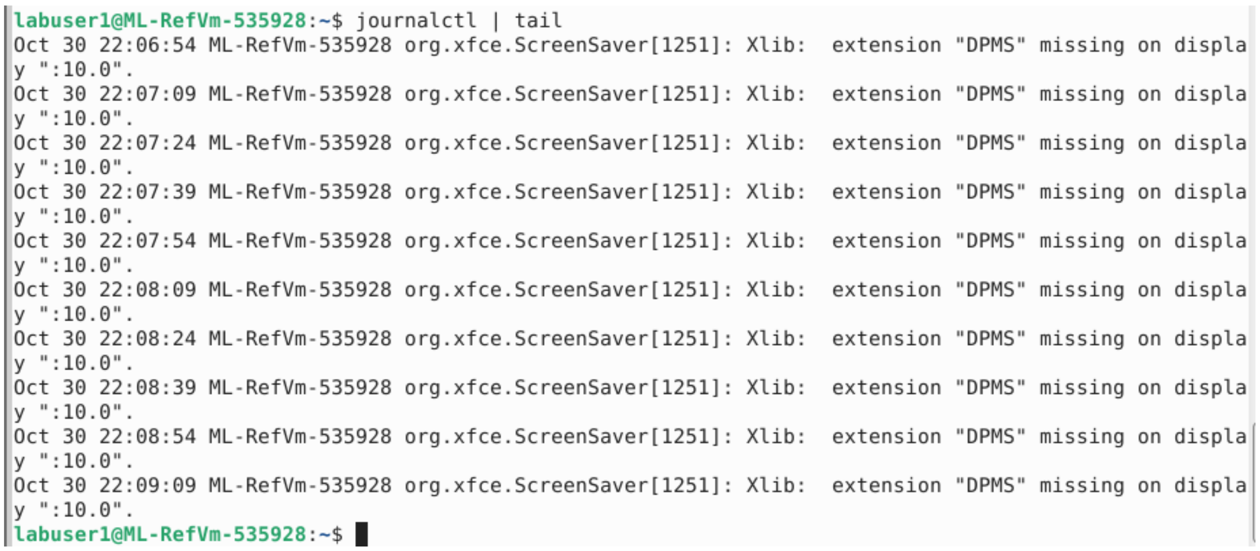


```
labuser1@ML-RefVm-535928:~$ xkill
Select the window whose client you wish to kill with button 1....
xkill:  killing creator of resource 0x280000a
X connection to :10.0 broken (explicit kill or server shutdown).
[1]+  Exit 1                  xeyes
labuser1@ML-RefVm-535928:~$ 
```

System Logging

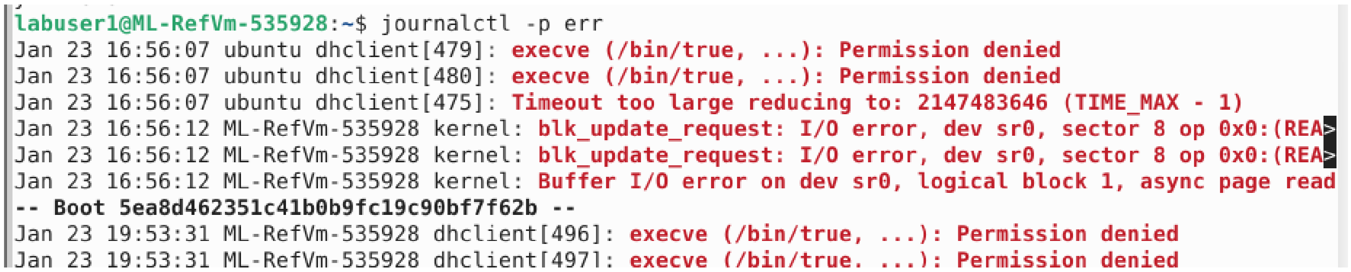
System logs are critical for several reasons: These logs provide administrators with useful information to aid in troubleshooting problems. They are also useful in identifying potential hacking attempts. Additionally, logs can be used to provide general information about services, such as which web pages have been provided by a web server. On modern Linux systems, the logging process is handled by the **systemd-journald** service. To query **systemd** log entries, use the **journalctl** command. The log entries can be filtered in a variety of forms such as by priority (**-p**), unit/process (**-u**), and boot logs(**-b**).

- View the tail of the log entries*



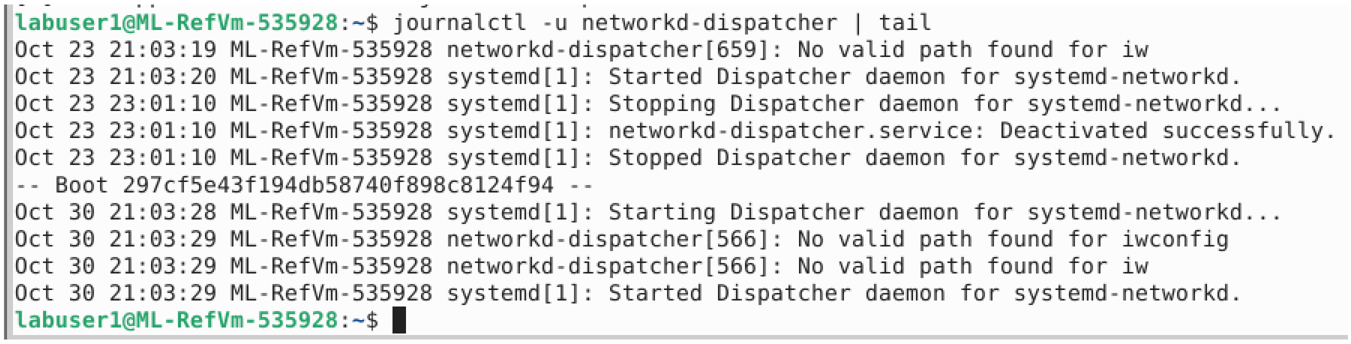
```
labuser1@ML-RefVm-535928:~$ journalctl | tail
Oct 30 22:06:54 ML-RefVm-535928 org.xfce.ScreenSaver[1251]: Xlib:  extension "DPMS" missing on displa
y ":10.0".
Oct 30 22:07:09 ML-RefVm-535928 org.xfce.ScreenSaver[1251]: Xlib:  extension "DPMS" missing on displa
y ":10.0".
Oct 30 22:07:24 ML-RefVm-535928 org.xfce.ScreenSaver[1251]: Xlib:  extension "DPMS" missing on displa
y ":10.0".
Oct 30 22:07:39 ML-RefVm-535928 org.xfce.ScreenSaver[1251]: Xlib:  extension "DPMS" missing on displa
y ":10.0".
Oct 30 22:07:54 ML-RefVm-535928 org.xfce.ScreenSaver[1251]: Xlib:  extension "DPMS" missing on displa
y ":10.0".
Oct 30 22:08:09 ML-RefVm-535928 org.xfce.ScreenSaver[1251]: Xlib:  extension "DPMS" missing on displa
y ":10.0".
Oct 30 22:08:24 ML-RefVm-535928 org.xfce.ScreenSaver[1251]: Xlib:  extension "DPMS" missing on displa
y ":10.0".
Oct 30 22:08:39 ML-RefVm-535928 org.xfce.ScreenSaver[1251]: Xlib:  extension "DPMS" missing on displa
y ":10.0".
Oct 30 22:08:54 ML-RefVm-535928 org.xfce.ScreenSaver[1251]: Xlib:  extension "DPMS" missing on displa
y ":10.0".
Oct 30 22:09:09 ML-RefVm-535928 org.xfce.ScreenSaver[1251]: Xlib:  extension "DPMS" missing on displa
y ":10.0".
labuser1@ML-RefVm-535928:~$
```

- View the log entries of the **err** priority*



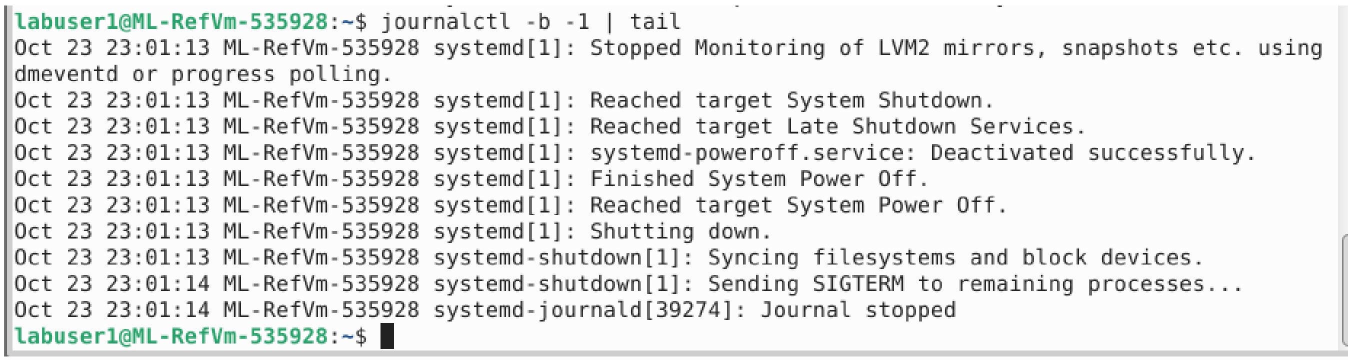
```
labuser1@ML-RefVm-535928:~$ journalctl -p err
Jan 23 16:56:07 ubuntu dhclient[479]: execve (/bin/true, ...): Permission denied
Jan 23 16:56:07 ubuntu dhclient[480]: execve (/bin/true, ...): Permission denied
Jan 23 16:56:07 ubuntu dhclient[475]: Timeout too large reducing to: 2147483646 (TIME_MAX - 1)
Jan 23 16:56:12 ML-RefVm-535928 kernel: blk_update_request: I/O error, dev sr0, sector 8 op 0x0:(REA>
Jan 23 16:56:12 ML-RefVm-535928 kernel: blk_update_request: I/O error, dev sr0, sector 8 op 0x0:(REA>
Jan 23 16:56:12 ML-RefVm-535928 kernel: Buffer I/O error on dev sr0, logical block 1, async page read
-- Boot 5ea8d462351c41b0b9fc19c90bf7f62b --
Jan 23 19:53:31 ML-RefVm-535928 dhclient[496]: execve (/bin/true, ...): Permission denied
Jan 23 19:53:31 ML-RefVm-535928 dhclient[497]: execve (/bin/true, ...): Permission denied
```

- View the tail of the log entries of the **networkd-dispatcher** process*



```
labuser1@ML-RefVm-535928:~$ journalctl -u networkd-dispatcher | tail
Oct 23 21:03:19 ML-RefVm-535928 networkd-dispatcher[659]: No valid path found for iw
Oct 23 21:03:20 ML-RefVm-535928 systemd[1]: Started Dispatcher daemon for systemd-networkd.
Oct 23 23:01:10 ML-RefVm-535928 systemd[1]: Stopping Dispatcher daemon for systemd-networkd...
Oct 23 23:01:10 ML-RefVm-535928 systemd[1]: networkd-dispatcher.service: Deactivated successfully.
Oct 23 23:01:10 ML-RefVm-535928 systemd[1]: Stopped Dispatcher daemon for systemd-networkd.
-- Boot 297cf5e43f194db58740f898c8124f94 --
Oct 30 21:03:28 ML-RefVm-535928 systemd[1]: Starting Dispatcher daemon for systemd-networkd...
Oct 30 21:03:29 ML-RefVm-535928 networkd-dispatcher[566]: No valid path found for iwconfig
Oct 30 21:03:29 ML-RefVm-535928 networkd-dispatcher[566]: No valid path found for iw
Oct 30 21:03:29 ML-RefVm-535928 systemd[1]: Started Dispatcher daemon for systemd-networkd.
labuser1@ML-RefVm-535928:~$
```

- View the tail of the log entries for the previous system boot (**-1**)*



```
labuser1@ML-RefVm-535928:~$ journalctl -b -1 | tail
Oct 23 23:01:13 ML-RefVm-535928 systemd[1]: Stopped Monitoring of LVM2 mirrors, snapshots etc. using
dmeventd or progress polling.
Oct 23 23:01:13 ML-RefVm-535928 systemd[1]: Reached target System Shutdown.
Oct 23 23:01:13 ML-RefVm-535928 systemd[1]: Reached target Late Shutdown Services.
Oct 23 23:01:13 ML-RefVm-535928 systemd[1]: systemd-poweroff.service: Deactivated successfully.
Oct 23 23:01:13 ML-RefVm-535928 systemd[1]: Finished System Power Off.
Oct 23 23:01:13 ML-RefVm-535928 systemd[1]: Reached target System Power Off.
Oct 23 23:01:13 ML-RefVm-535928 systemd[1]: Shutting down.
Oct 23 23:01:13 ML-RefVm-535928 systemd-shutdown[1]: Syncing filesystems and block devices.
Oct 23 23:01:14 ML-RefVm-535928 systemd-shutdown[1]: Sending SIGTERM to remaining processes...
Oct 23 23:01:14 ML-RefVm-535928 systemd-journald[39274]: Journal stopped
labuser1@ML-RefVm-535928:~$
```