**312-F02**

**Systemctl commands to manage systemd service**

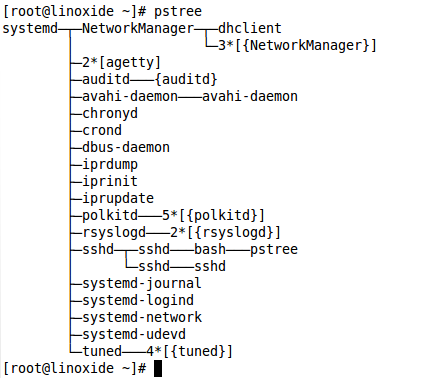
Updated August 20, 2020

By [Raghu](https://linoxide.com/author/raghu/)[LINUX COMMANDS](https://linoxide.com/category/linux-command/), [LINUX HOWTO](https://linoxide.com/category/linux-how-to/)

Systemd is a service manager suitable for any Linux Distribution like Ubuntu/Centos. It is a replacement for init system and can manage system startup and services. It starts up and supervises the entire system. In this tutorial, I will show you how to use systemctl commands to manage systemd service in Linux.

PID 1 is occupied by "systemd" and can be seen from pstree command as well:

# pstree



Let's explore what systemd is capable of and what possibilities we have with the new replacement for sysVinit.

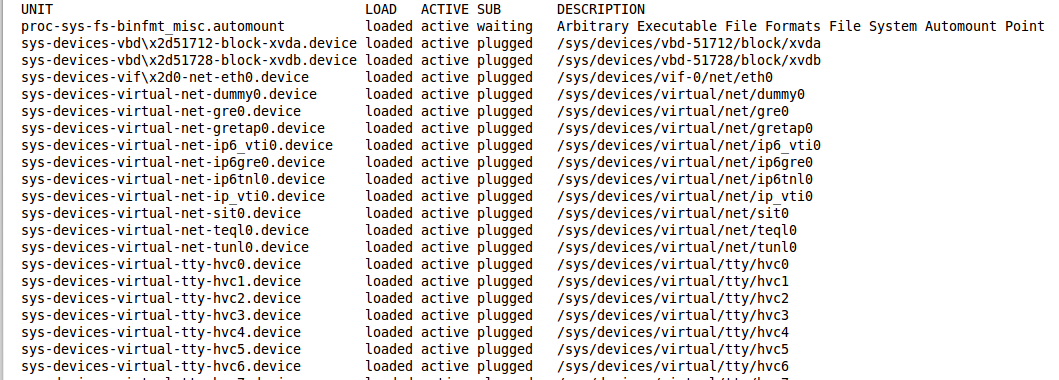
**1)** **List Units**

Systemctl command without any option lists all the running units. The list-units switch also does the same.

# systemctl

or

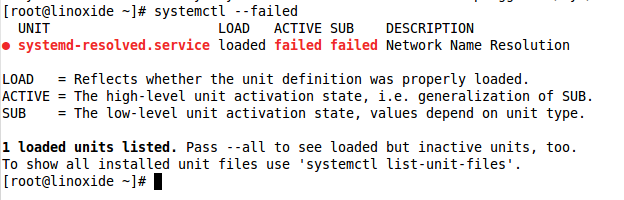
# systemctl list-units



**Listing failed units**

The failed units can be listed with --failed switch.

# systemctl --failed

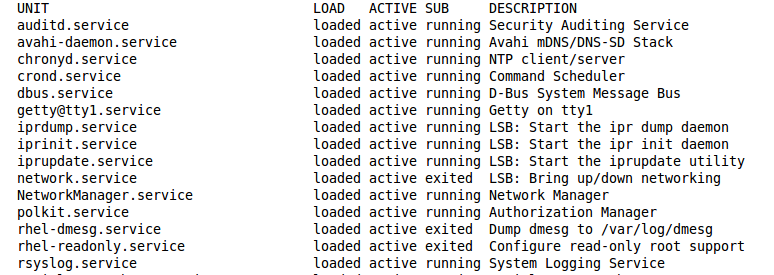


You will see the use of systemctl command at many places in this article.

**2) List Active services**

Let us now see how services can be managed with systemd. All the active [services can be listed](https://linoxide.com/linux-command/how-to-list-services-in-linux/) with the following command:

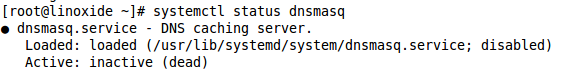
# systemctl list-units -t service



**3) Service status**

In the sysvinit, we could use the  service command to manage the services, but with systemd, the systemctl command is used to manage services. In order to see whether a service is running or not, we can use the systemctl command like this:

# systemctl status dnsmasq

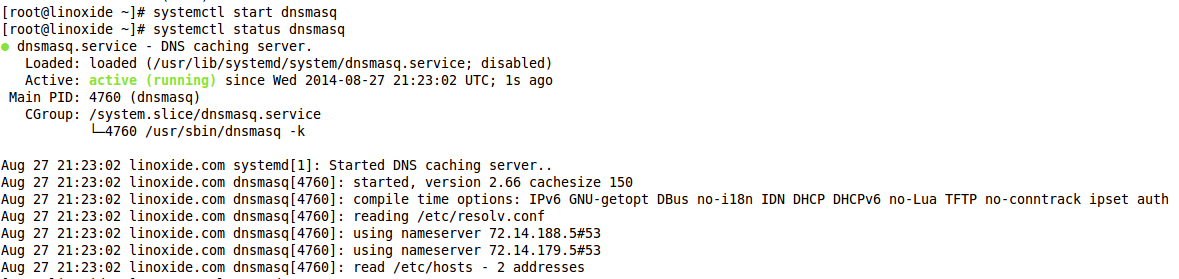


**4) Start Service**

To start a service, again we use the systemctl command as:

# systemctl start dnsmasq

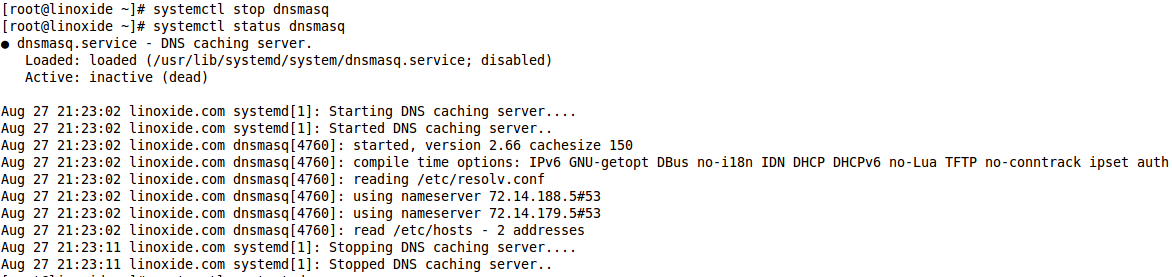
As opposed to service command, this command does not give any output. But of course, we can check the status of the service once again to confirm that its started successfully:



**5) Stopping service**

Now you are smart enough and already know the command to stop a service with systemd:

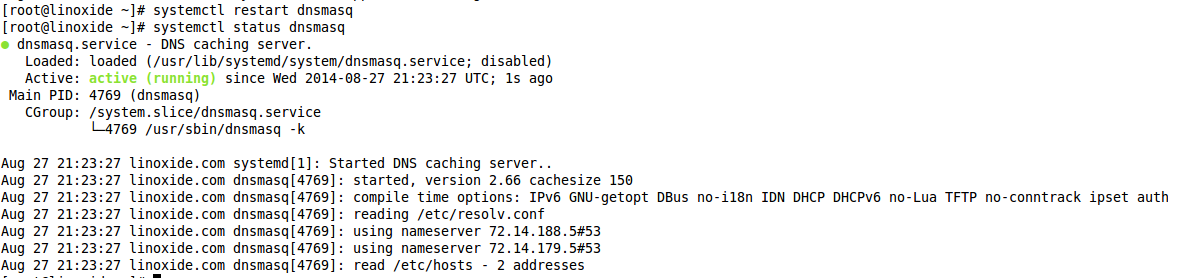
# systemctl stop dnsmasq



**6) Restart service**

Similarly, restarting a service is managed using 'systemctl restart ':

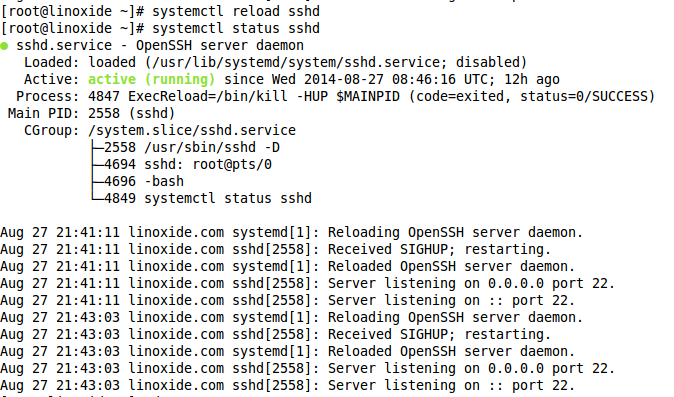
# systemctl restart dnsmasq



**7) Reload service**

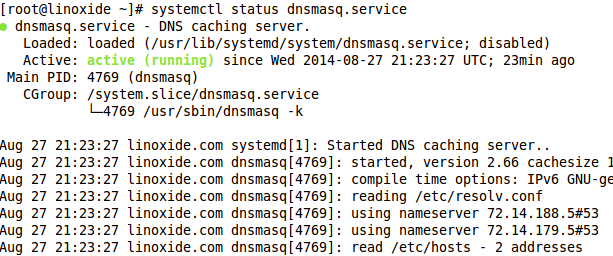
In case we need to reload the configuration of service (say ssh), without restarting it, we can use the command:

# systemctl reload sshd



Although all of the above syntax is working, the official documentation suggests that these command be run with the following syntax:

# systemctl status dnsmasq.service



**8) Checking services status at boot**

The chkconfig command was used to manage services at boot. The same command systemd is used with systemd to manage services at boot.

In order to check if a service is enabled on boot or not:

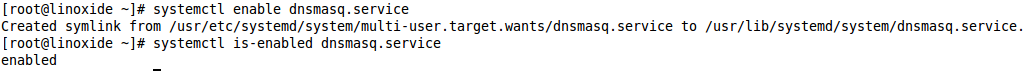
# systemctl is-enabled dnsmasq.service

Service at system boot

**9) Enable service at boot**

Systemctl command can be used like this to enable a service at boot (this corresponds to sysvinit 'chkconfig on')

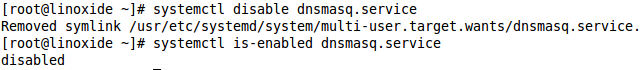
# systemctl enable dnsmasq.service



**10) Disable service at boot**

Similarly, the services can be disabled at boot with systemctl command:

# systemctl disable dnsmasq.service

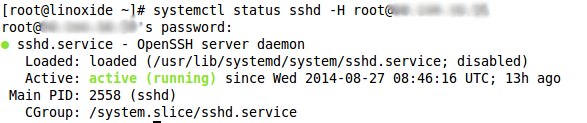


Note : sudo systemctl enable nginx --now  - This will enable and startup the service (here nginx) immediately in a single command.

**11) Managing Remote systems**

Typically, all of the above systemctl commands can be used to manage a remote host with systemctl command itself. This will use ssh for communication with the remote host. All you need to do is add the user and host to systemctl command like this:

# systemctl status sshd -H root@1.2.3.4



**12) Managing targets**

Systemd has a concept of targets having a similar purpose to runlevels in sysVinit.

The runlevels in sysVinit were mostly numeric (0,1,2,...). Here are the runlevels in sysVinit with their systemd counterparts:

0 runlevel0.target, poweroff.target

1, s, single runlevel1.target, rescue.target

2, 4 runlevel2.target, runlevel4.target, multi-user.target

3 runlevel3.target, multi-user.target

5 runlevel5.target, graphical.target

6 runlevel6.target, reboot.target

emergency emergency.target

**Changing current target**

The current target(runlevel) can be changed with the command:

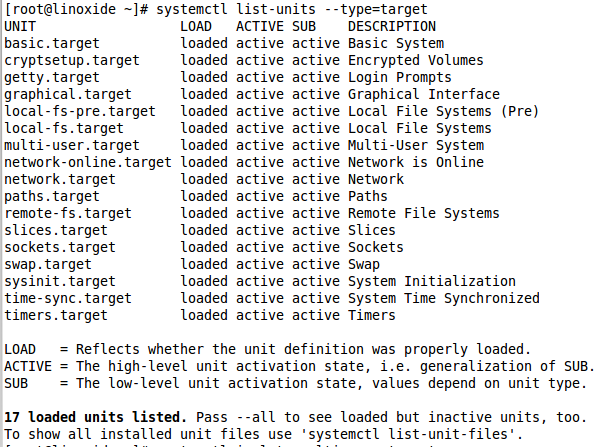
# systemctl isolate graphical.target

Change target

**List current target**

If you want to see what target you are in, you need to list all the corresponding units. It might not feel at home with this new way, but its the way systemd works.

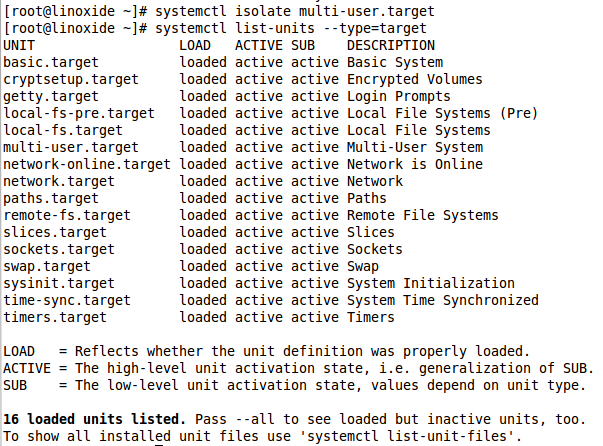
# systemctl list-units --type=target



You can see "graphical.target" listed here. This is what we changed our target into. Now let's change the runlevel again to multi-user.target and then analyze this output:

# systemctl isolate multi-user.target

# systemctl list-units --type=target



**List default target**

To list the default target, we use systemctl command like this:

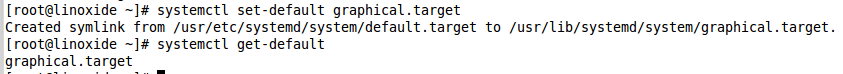
# systemctl get-default

Get default target

**Change default target**

The default target can be set with set-default command with systemctl:

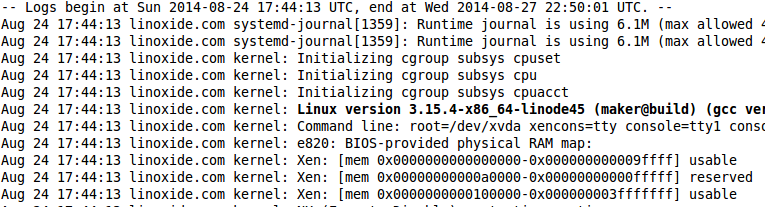
# systemctl set-default graphical.target



**13) Logging in systemd**

The systemd has its own logging system called journald. It replaces the syslog daemon from sysVinit. The command [journalctl is used to read the logs](https://linoxide.com/linux-how-to/systemd-journalctl-command-logs/).

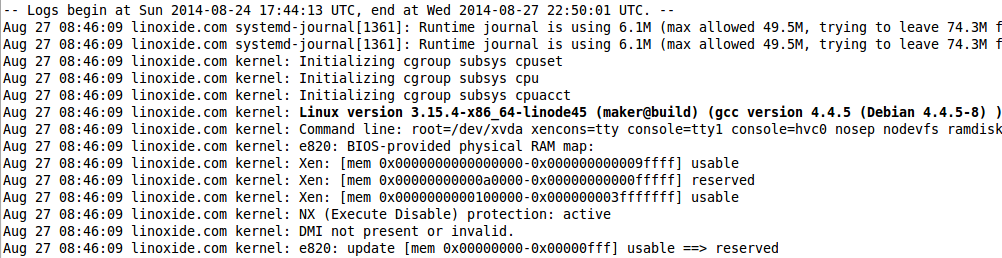
# journalctl



**Boot messages**

To see all boot messages, run the command "journalctl -b".

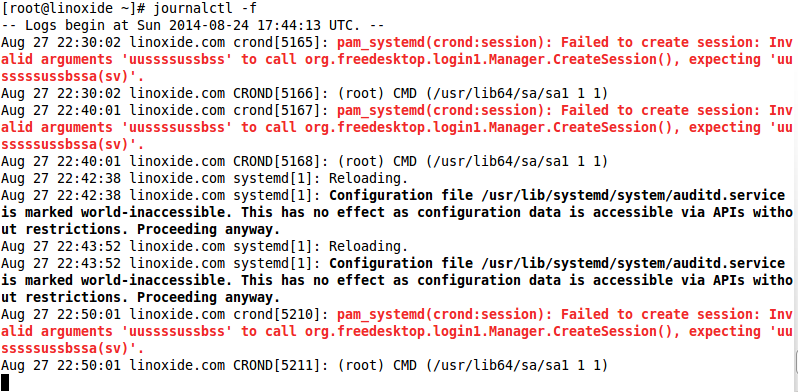
# journalctl -b



**Follow logs**

The following command follows the system logs in real time (similar to tail -f).

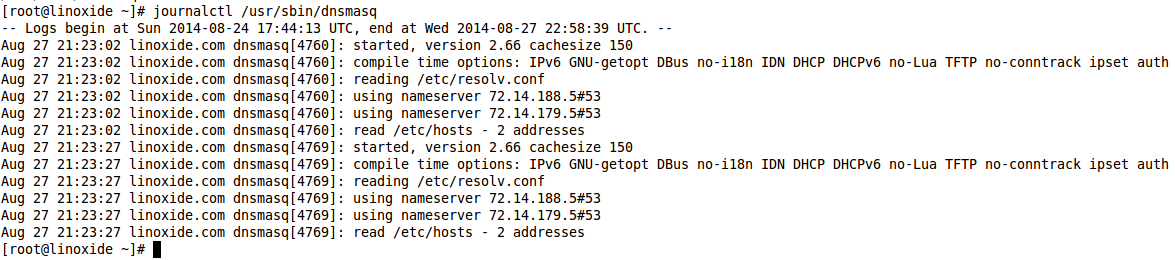
# journalctl -f



**Service specific logs**

To check logs specific to a particular service or executable, use journalctl like this:

# journalctl /usr/sbin/dnsmasq



**14) Power management**

The systemctl command can be used to put the system down, or reboot or hibernate.

To poweroff, reboot, suspend and hibernate, use the following commands respectively:

# systemctl poweroff

# systemctl reboot

# systemctl suspend

# systemctl reboot

**15) Systemd Faster startup**

The sysvinit starts the processes serially, one at a time. Systemd starts services in parallel and starts only those services which are actually required, reducing the boot time significantly.  
You can get the boot process duration with the following command:

# systemd-analyze

systemd-analyze

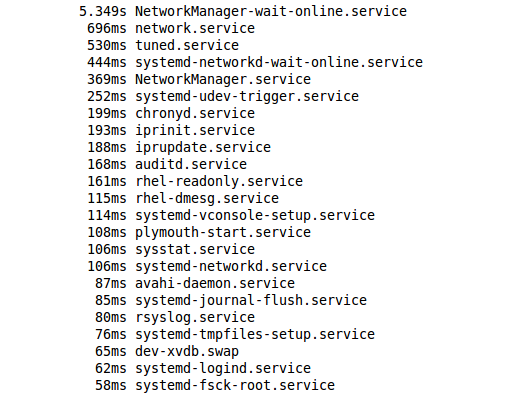
The command systemd-analyze time also shows the same information.

# systemd-analyze time

systemd-analyze time

If you want to print a list of all running units, the blame option to systemd-analyze command can provide you with that, ordered by the time taken to initialize.

# systemd-analyze blame

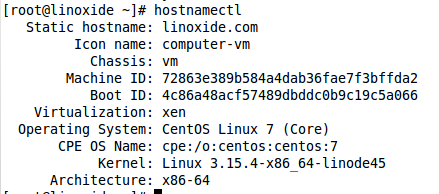


The above screen shows only a small number of processes, you can scroll through the list with arrows just like in less pager.

**16) Systemd Hostnamectl command**

The systemd brings out the whole new approach to interacting with your operating system. The systemd is so full of features. For example, you can get the **hostname and other useful features** about your Linux machine, you can use hostnamectl command

# hostnamectl



**Conclusion**

In this tutorial we learned some of the commonly used systemctl commands to manage service in new linux distribtions that use systemd. I hope you enjoyed reading and please leave your suggestions in the below comment section.