

Learning Objectives

Learners will be able to...

Use the following `systemctl` services:

- stop
- start
- restart
- status
- enable
- disable
- mask

System Services

What are system services?

System services are processes that continuously run in the background, waiting for requests to come in.

In Unix, init is the first process that starts, and it starts up other processes..

For the most part, Linux is UNIX-like or UNIX-compatible. With only a few exceptions, the Linux and UNIX systems are very similar and it is easy to move between the two.

Linux's use of `systemd` instead of `init` is one of the few exceptions to this.

init

`init` starts the machine in one of the 7 **run levels** (from 0 to 6) which indicate machine state.

Example of standard Linux run levels:

- 0 – Shut down
- 1 – Single user mode
- 3 – Multiple user mode with command line interface
- 5 – Multiple user mode under GUI
- 6 – Reboot/Restart

Run level 5 is the standard run level for most of the LINUX based systems.

systemd

In Linux, `systemd`, the system and service manager for Linux systems, became widespread across distributions around 2015. The name **systemd** is short for system daemon. A **daemon** is a process that runs in the background. After the Linux kernel is booted **systemd** is activated to manage the user space components know as a **unit**. The **systemd** tools are used to start, stop, enable and disable services and retrieve status.

This move away from `init` was and is a highly debated move away from the Unix-based approach. Many disliked how bloated and inter-connected `systemd` was which is in direct opposition to the Unix philosophy of do one thing well. The uncharacteristically large and interconnected code base for

systemd causes concerns for both reliability and security. Others embraced it as a fix to the existing, Unix-inspired `init` solution since it addressed many long-standing issues.

Run the `top` command, you will see that **systemd** is the first process and it has a **PID** of 1.

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	30	10	224584	8640	6968	S	0.0	1.1	0:00.22	systemd
39	root	30	10	94672	10764	10220	S	0.0	1.4	0:00.08	systemd-journal
46	root	30	10	42112	3476	2952	S	0.0	0.4	0:00.01	systemd-udev
49	systemd+	30	10	79916	5128	4620	S	0.0	0.7	0:00.03	systemd-network
73	systemd+	30	10	70492	5216	4748	S	0.0	0.7	0:00.02	systemd-resolve
75	root	30	10	70464	5996	5372	S	0.0	0.8	0:00.02	systemd-logind
76	root	30	10	170380	17360	9640	S	0.0	2.2	0:00.18	networkd-dispatcher
77	root	30	10	31296	3100	2868	S	0.0	0.4	0:00.00	cron
78	message+	30	10	49932	4124	3664	S	0.0	0.5	0:00.03	dbus-daemon
79	syslog	30	10	123904	4224	3692	S	0.0	0.5	0:00.01	rsyslogd
82	root	30	10	15960	2396	2260	S	0.0	0.3	0:00.00	agetty
83	root	30	10	15960	2468	2324	S	0.0	0.3	0:00.00	agetty
85	root	30	10	72304	5920	5196	S	0.0	0.8	0:00.00	sshd
147	root	30	10	107988	7288	6272	S	0.0	0.9	0:00.01	sshd
149	codio	30	10	76400	7360	6560	S	0.0	0.9	0:00.00	systemd
150	codio	30	10	258560	2740	916	S	0.0	0.3	0:00.00	(sd-pam)
170	codio	30	10	107988	3712	2700	S	0.0	0.5	0:00.00	sshd
171	codio	30	10	21344	3704	3408	S	0.0	0.5	0:00.00	bash
176	codio	30	10	22248	2660	2488	S	0.0	0.3	0:00.00	script
177	codio	30	10	21608	4096	3512	S	0.0	0.5	0:00.00	bash
194	codio	30	10	39560	3636	3168	R	0.0	0.5	0:00.01	top

The `systemctl` program is the tool used to manage **systemd**. The `systemctl` allows you to manage services, check statuses and change system states.

You can find the configuration file for `systemd` in `/etc/systemd/system.conf`.

List it out the contents of `system.conf`:

```
cat /etc/systemd/system.conf
```

To view the complete `systemctl` manual:

```
man systemctl
```

Systemctl commands

Information about system services

- Use the arrow keys to scroll up and down.
- Type q to exit.

Show system status

```
bash systemctl status
```

List all running services

```
systemctl
```

Show all unit files

```
bash systemctl list-unit-files
```

Show the status of a particular service

```
bash systemctl status cron.service
```

Install a service

For this portion we will install a database server so that we can start and stop the service. The `sudo apt update` command updates a list with all the latest versions of packages. Once this list is updated we will install the Mariadb server and can use that service to try things out with the `systemctl` command. You will be prompted to confirm the install during the process.

Note - Starting the following commands with `sudo` allows you to run them as a “super user” and provides you with the permissions you need.

```
sudo apt update  
sudo apt install mariadb-server
```

info

View status after running each command below

```
systemctl status mariadb.service
```

Manage services

Activate a service

```
sudo systemctl start mariadb.service
```

Deactivate a service immediately

```
sudo systemctl stop mariadb.service
```

Restart a service

```
sudo systemctl restart mariadb.service
```

Enables a service to be started on bootup:

```
sudo systemctl enable mariadb.service
```

Disables a service from starting on bootup:

```
sudo systemctl disable mariadb.service
```

Mask a service so it can't be started:

```
sudo systemctl mask mariadb.service
```

Systemd Units

A **unit** is an object that **systemd** operates on. It is a standardized representation of a resource.

Types of units

service

- A service or an application on the system, including instructions for starting, restarting, and stopping the service. On the previous page we looked at **Mariadb** which is a service.

List all the service units in this Linux installation:

```
find / -type f -name "*.service" 2>/dev/null
```

info

Reminder - the 2>/dev/null suppresses the permission denied error messages.

socket

- Used by **systemd** for socket-based activation.

List all the socket units in this Linux installation:

```
find / -type f -name "*.socket" 2>/dev/null
```

device

- A device specifically managed with **systemd**. May be used for ordering, mounting and accessing devices

List all the device units in this Linux installation:

```
find / -type f -name "*.device" 2>/dev/null
```

mount

- A mountpoint managed with **systemd**.

List all the mount units in this Linux installation:

```
find / -type f -name "*.mount" 2>/dev/null
```

automount

- A mountpoint automatically mounted on boot.

List all the automount units in this Linux installation:

```
find / -type f -name "*.automount" 2>/dev/null
```

swap

- These describe swap space on the system.

List all the swap units in this Linux installation:

```
find / -type f -name "*.swap" 2>/dev/null
```

target

- Used as a synchronization point for other units.

List all the target units in this Linux installation:

```
find / -type f -name "*.target" 2>/dev/null
```

path

- Specifies a path for path-based activation.

List all the path units in this Linux installation:

```
find / -type f -name "*.path" 2>/dev/null
```

timer

- A timer that is used to schedule activation of another unit.

List all the timer units in this Linux installation:

```
find / -type f -name "*.timer" 2>/dev/null
```

snapshot

- A snapshot of the current **systemd** state. Can be used to rollback after making temporary changes to systemd.

List all the snapshot units in this Linux installation:

```
find / -type f -name "*.snapshot" 2>/dev/null
```

slice

- Provides restriction of resources through Linux Control Group nodes

List all the slice units in this Linux installation:

```
find / -type f -name "*.slice" 2>/dev/null
```

scope

- Mostly used to manage external system processes.

List all the scope units in this Linux installation:

```
find / -type f -name "*.scope" 2>/dev/null
```


Scheduling Services

The cron command

- The cron daemon is a Linux command that can be used to schedule tasks on your computer.
- The crontab (cron table) contains information about the date and time **cron** should run something.

More information about cron

```
man cron
```

More information about crontab

```
man crontab
```

You can create a cron table using the command below, you will be presented with options for editors.

```
crontab -e
```

The at command

The at utility can be used for scheduling one off tasks. It does not output to the console when it is run, it is meant for tasks you might run when you are not logged in.

The time and date syntax is very flexible but the time must come before the day. The current day is the default if day is not specified. The following words are recognized: now, midnight, noon, teatime (4 PM), AM, PM. You can use 24 hour time as well.

You can use the following words to designate a time relative to now (using the + sign): minutes, hours, days, weeks, months, years.

Before you run the at command, list out the contents of `timed.sh` so you know what it will do.

```
cat timed.sh
```

Try the at command:

```
at now +1 minute -f timed.sh
```

Once it has run, list out the contents of `output.txt` so you know it worked.

```
cat output.txt
```