



Process Management on Linux

Learn about process management on Linux and how to do it with systemd.

What is process management?

Many processes must run for a computer to do its job.

Which processes to run and when to run them depends on the use case:

- On a desktop computer, you need processes to be launched to display the graphical user interface (GUI), for example.
- On a web-facing server, you probably need to run a database and a web server at least.

Some processes may have **dependencies**: i.e. they might need other processes to be already running in order to run properly themselves. For example, a PHP application might need a MySQL database to store some data.

Processes also need to be **managed**, i.e. it should be easy to start them, stop them, restart them, or to configure them to automatically restart if they crash.

Process managers

Process managers are programs that fulfill this role: i.e. make sure that the correct processes are run in the correct order and are managed correctly thereafter.

Most operating systems have a process manager built in:

Process manager	Operating system
<u>launchd</u>	<u>macOS</u>
<u>Service Control Manager (SCM)</u>	<u>Windows</u>
<u>systemd</u>	Many <u>Linux</u> distributions
<u>Unix System V</u> init system	<u>Unix</u>

More information

Some of these do more than just process management, like systemd which also manages the Linux init process to bootstrap the operating system.

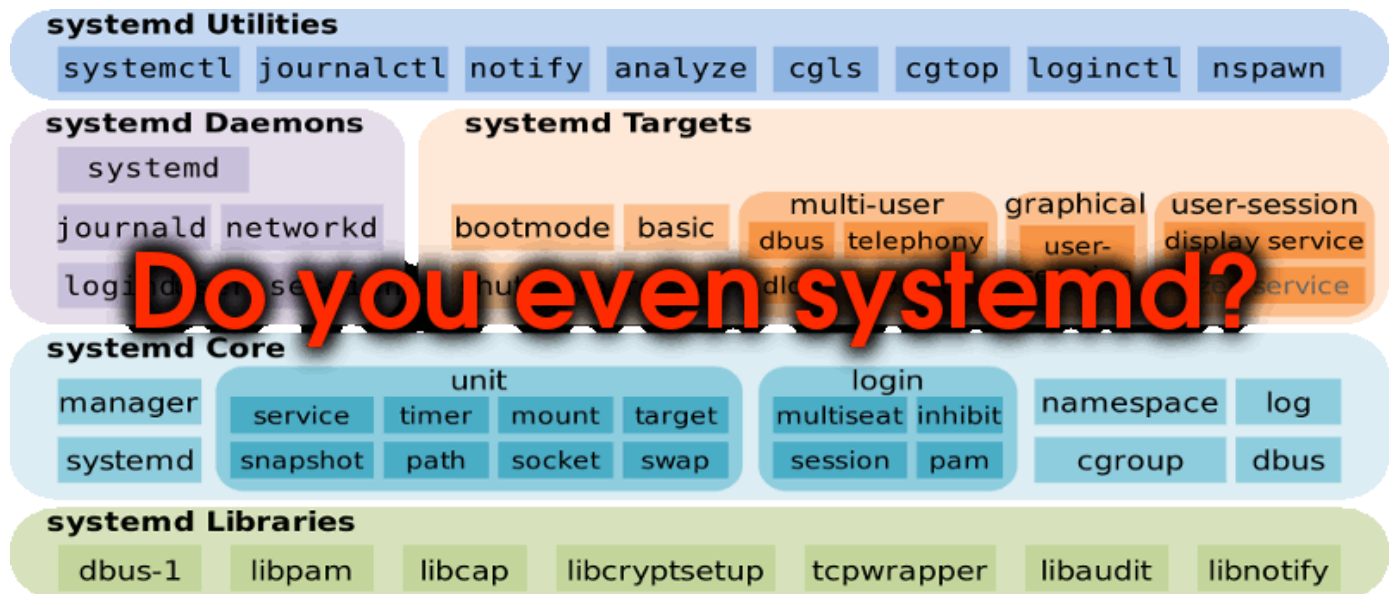
Lightweight process managers

Simpler process managers also exist, meant to be used to run and manage one application. Some of them are generic while others are specific to a programming language:

Process manager	Written in	Can run
<u>God</u>	<u>Ruby</u>	<i>Any executable</i>
<u>PHP FPM</u>	<u>PHP</u>	PHP programs
<u>PM2</u>	<u>Node.js</u>	Node.js programs
<u>Supervisor</u>	<u>Python</u>	<i>Any executable</i>

They are easier to use but have fewer features. For example, none of those can handle dependencies between processes.

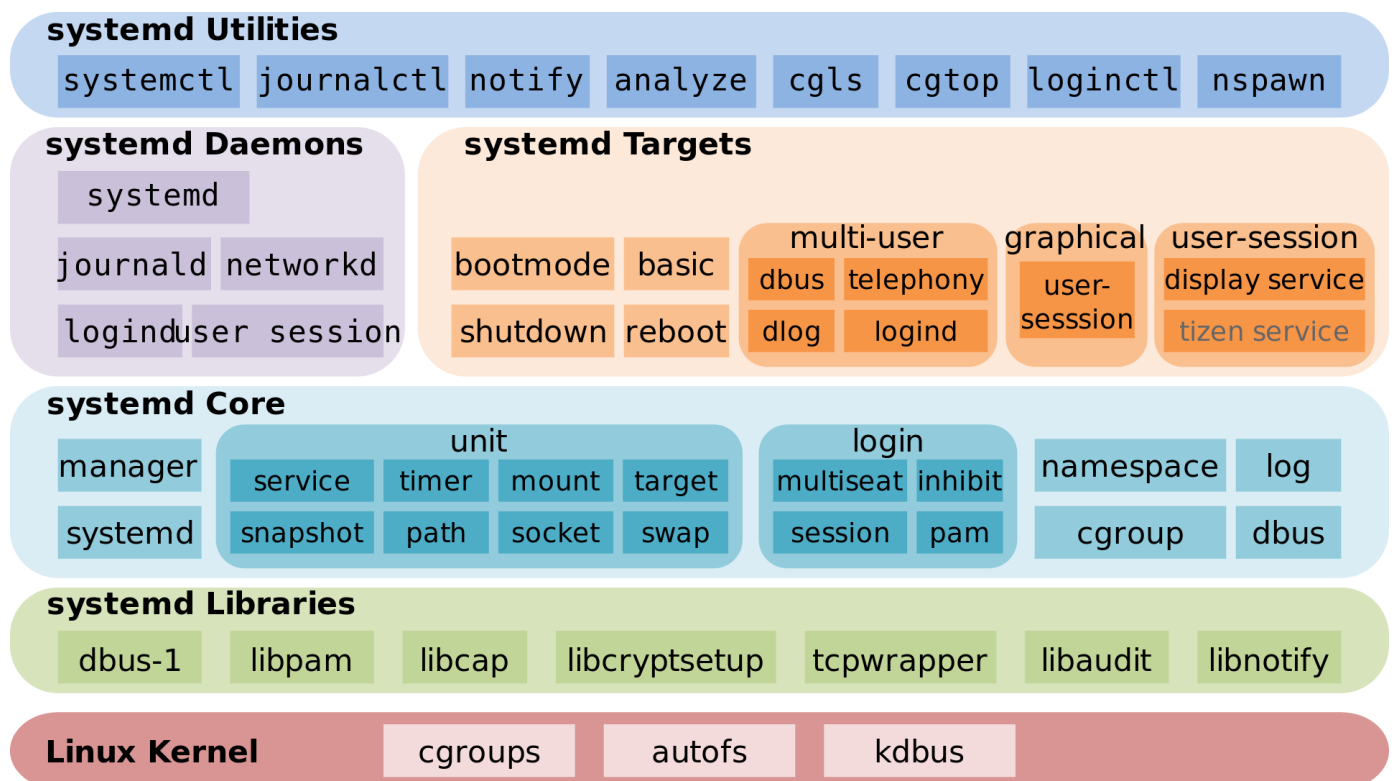
Systemd



Do you even systemd?

What is systemd?

Systemd provides not only process management, but also the fundamental building blocks of a Linux operating system, such as an init system.



Systemd is a replacement for the Unix System V init system and has been the de facto standard for most Linux distributions since 2015.

Unit files

Systemd records the instructions on how to launch and manage a process in a configuration file referred to as a **unit file**. It supports many unit file types such as `service`, `socket`, `timer`, etc.

The start of a unit file usually looks like this:

```
[Unit]
Description=Something to run

[Install]
WantedBy=something-else.target

...
```

`[Unit]` is one section in the configuration file. There will be other sections depending on the type of unit file.

`[Install]` is an optional section that can be used to express dependencies between units. Here, the `WantedBy` option specifies that this unit will be started after the `something-else` unit has been started.

Service unit file

The unit file type that interests us is `service`, which will configure, launch and manage a long-running process such as a database or application.

A service unit file will have a `[Service]` section. Common options are:

Option	Description
<code>ExecStart</code>	Command to run to start the service
<code>WorkingDirectory</code>	Directory in which to run the command
<code>User</code>	User to run the process as (can be used to limit access)

Tip

And many more.

Location of unit files

Unit files are named after their type. For example, a `service` unit file will be named `something.service`.

Systemd will look in several directories for unit files:

Location	Description
<code>/lib/systemd/system</code>	Unit files installed by the operating system or by packages (e.g. a database)
<code>/etc/systemd/system</code>	Unit files installed by the system administrator

The `systemctl` command

The `systemctl` or **system control** command can be used to enable/disable and start/stop units once their configuration file is in place:

Command	Description
<code>sudo systemctl status <unit></code>	Display the current status of a unit.

Command	Description
<code>sudo systemctl enable <unit></code>	Enable a new unit file. This will enable it to start on boot if it has the correct dependencies.
<code>sudo systemctl start <unit></code>	Start a unit.
<code>sudo systemctl stop <unit></code>	Stop a unit.
<code>sudo systemctl restart <unit></code>	Restart (stop and start) a unit.
<code>sudo systemctl reenabale <unit></code>	Re-enable an existing unit file after its dependencies have been modified.
<code>sudo systemctl daemon-reload</code>	Reload unit files after they have been modified.

The `journalctl` command

Services started with systemd have their standard output and standard error streams collected and logged by [journald](#), one of the default services provided by systemd itself. The `systemctl status` command sometimes shows you an excerpt of these logs.

The `journalctl` command allows you to read the full logs:

Command	Description
<code>sudo journalctl -u <unit></code>	Display a unit's logs.
<code>sudo journalctl -f -u <unit></code>	Display and follow a unit's logs in real time.

References

- [Systemd Unit Configuration](#)
- [Systemd Service Configuration](#)
- [Systemd Execution Environment Configuration](#)
- [Red Hat Enterprise Linux - System Administrator's Guide - Managing Services With Systemd](#)
- [Systemd journald Service](#)

Table of contents

- [What is process management?](#)
 - [Process managers](#)
 - [Lightweight process managers](#)
- [Systemd](#)
 - [What is systemd?](#)
 - [Unit files](#)
 - [Service unit file](#)
 - [Location of unit files](#)
 - [The systemctl command](#)
 - [The journalctl command](#)
- [References](#)