

Service Management

LinuXpert Systems

No.5, Dr. Ambedkar St, Rathinapuram,

Nanganallur, Chennai – 600 061.

Phone: 98841 65649, 90808 01382

E-mail: baskar@linuxpert.in

TABLE OF CONTENTS

1. systemctl 1

2. service 11

3. journalctl 15

4. loginctl 23

1. systemctl

`systemctl` is the primary command-line utility used to **control and manage the `systemd` system and service manager** on modern Linux distributions.

It allows you to:

- Start and stop services
- Enable or disable services at boot
- Restart services
- Check service status
- Manage system targets (runlevels)
- Shut down or reboot the system
- View logs and system information

It is available on most modern Linux distributions such as:

- Ubuntu
- Debian
- Fedora
- RHEL
- CentOS
- Rocky Linux
- AlmaLinux
- openSUSE
- Arch Linux

Syntax

```
systemctl [OPTIONS] COMMAND [UNIT]
```

- **OPTIONS** – Additional flags (optional)
- **COMMAND** – Action to perform
- **UNIT** – Service, socket, timer, mount, target, etc.

What is a Unit?

A **unit** is any resource managed by systemd.

Examples:

Unit Type	Extension	Example
Service	`.service`	ssh.service
Socket	`.socket`	cups.socket
Timer	`.timer`	logrotate.timer
Mount	`.mount`	home.mount
Target	`.target`	multi-user.target

Normally you can omit the extension:

```
systemctl start ssh
```

instead of

```
systemctl start ssh.service
```

Common Commands

1. Check service status

```
systemctl status ssh
```

Example output:

```
● ssh.service - OpenSSH Server
   Loaded: loaded
   Active: active (running)
```

Shows:

- Running state
- PID
- Logs
- Startup information

---

2. Start a service

```
sudo systemctl start apache2
```

Starts the Apache web server.

---

3. Stop a service

```
sudo systemctl stop apache2
```

Stops the service.

---

4. Restart a service

```
sudo systemctl restart apache2
```

Stops and starts the service again.

Useful after changing configuration files.

5. Reload a service

```
sudo systemctl reload apache2
```

Reloads configuration without stopping the service (if supported).

---

6. Reload or Restart

```
sudo systemctl reload-or-restart apache2
```

Reloads if possible.

Otherwise restarts.

---

7. Enable a service

```
sudo systemctl enable ssh
```

Makes the service start automatically during boot.

---

8. Disable a service

```
sudo systemctl disable ssh
```

Prevents automatic startup at boot.

---

9. Start immediately and enable

```
sudo systemctl enable --now ssh
```

Equivalent to:

```
sudo systemctl enable ssh
sudo systemctl start ssh
```

10. Disable and stop

```
sudo systemctl disable --now ssh
```

Stops it now and disables future automatic startup.

---

11. Check if enabled

```
systemctl is-enabled ssh
```

Output:

```
enabled
```

or

```
disabled
```

---

12. Check if active

```
systemctl is-active ssh
```

Possible outputs:

```
active
inactive
failed
activating
deactivating
```

---

13. List running services

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

---

14. List all services

```
systemctl list-unit-files --type=service
```

Shows both enabled and disabled services.

15. View failed services

```
systemctl --failed
```

Displays services that failed to start or crashed.

Managing the System

Reboot

```
sudo systemctl reboot
```

Shutdown

```
sudo systemctl poweroff
```

Suspend

```
sudo systemctl suspend
```

Hibernate

```
sudo systemctl hibernate
```

Change System Target

View current target:

```
systemctl get-default
```

Example:

```
graphical.target
```

Set command-line mode:

```
sudo systemctl set-default multi-user.target
```

Set GUI mode:

```
sudo systemctl set-default graphical.target
```

View Boot Time

```
systemctl show --property=FirmwareTimestamp
```

or

```
systemd-analyze
```

Show Service Dependencies

```
systemctl list-dependencies ssh
```

Shows services required by SSH.

Mask a Service

Masking prevents a service from being started, even manually.

```
sudo systemctl mask bluetooth
```

Unmask:

```
sudo systemctl unmask bluetooth
```

Show Service Configuration

```
systemctl cat ssh
```

Displays the unit file.

Edit a Service

```
sudo systemctl edit ssh
```

Creates an override configuration.

Reload systemd Configuration

After modifying unit files:

```
sudo systemctl daemon-reload
```

View Service Properties

```
systemctl show ssh
```

Shows detailed properties such as:

- Main PID
- Memory usage
- Dependencies
- Restart policy
- Environment

Service States

State	Meaning
active	Running normally
inactive	Not running
failed	Service encountered an error
activating	Starting up
deactivating	Stopping
reloading	Reloading configuration

Frequently Used Examples

Start MySQL:

```
sudo systemctl start mysql
```

Check MySQL status:

```
systemctl status mysql
```

Restart NetworkManager:

```
sudo systemctl restart NetworkManager
```

Enable SSH at boot:

```
sudo systemctl enable ssh
```

Check Docker:

```
systemctl status docker
```

Reload Apache configuration:

```
sudo systemctl reload apache2
```

List failed services:

```
systemctl --failed
```

Reboot:

```
sudo systemctl reboot
```

Shutdown:

```
sudo systemctl poweroff
```

Advantages of `systemctl`

- Centralised management of system services.
- Supports parallel service startup, improving boot speed.
- Automatically manages service dependencies.
- Provides detailed service status and diagnostics.
- Integrates with timers, sockets, mounts, and targets.
- Allows services to be enabled or disabled at boot.
- Offers consistent service management across most modern Linux distributions.

Summary

`systemctl` is the standard tool for managing **systemd** on Linux. It lets administrators control services, inspect their status, configure automatic startup, manage system power states, and troubleshoot failures. It is one of the most essential commands for Linux system administration.

2. service

The **`service`** command is used to **start, stop, restart, reload, or check the status of system services**. It provides a simple interface for managing services without directly interacting with the underlying init system.

On modern Linux distributions that use **systemd**, the `service` command acts as a compatibility wrapper around `systemctl`. On older systems, it interacts with **SysVinit** scripts.

Syntax

```
service <service_name> <action>
```

General form:

```
service [OPTIONS] SERVICE COMMAND
```

- **SERVICE** – Name of the service (e.g., `ssh`, `apache2`, `nginx`)
- **COMMAND** – Action to perform (`start`, `stop`, `restart`, `status`, etc.)

Common Actions

Action	Description
`start`	Starts the service
`stop`	Stops the service
`restart`	Stops and starts the service
`reload`	Reloads configuration without stopping the service (if supported)
`status`	Shows the current status of the service
`force-reload`	Forces configuration reload or restarts if reload isn't supported

Examples

1. Start a service

```
sudo service apache2 start
```

Starts the Apache web server.

---

2. Stop a service

```
sudo service apache2 stop
```

Stops the Apache service.

---

3. Restart a service

```
sudo service apache2 restart
```

Useful after changing configuration files.

---

4. Reload a service

```
sudo service nginx reload
```

Reloads Nginx configuration without interrupting existing connections.

---

5. Check service status

```
service ssh status
```

Example output:

```
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded
     Active: active (running)
```

6. List all services (SysV-style)

```
service --status-all
```

Example output:

```
[ + ]  cron
[ + ]  ssh
[ - ]  apache2
```

- `[ + ]` = Running
- `[ - ]` = Stopped
- `[ ? ]` = Status unknown

Common Services

Service	Purpose
`ssh`	Secure Shell server
`apache2`	Apache web server
`nginx`	Nginx web server
`mysql`	MySQL database
`cron`	Scheduled jobs
`networking`	Network configuration
`cups`	Printing service

Relationship with `systemctl`

On **systemd** systems:

`service` Command	Equivalent `systemctl` Command
`service ssh start`	`systemctl start ssh`
`service ssh stop`	`systemctl stop ssh`
`service ssh restart`	`systemctl restart ssh`
`service ssh status`	`systemctl status ssh`

`systemctl` is the recommended command on modern Linux distributions because it provides more features, such as enabling/disabling services at boot, viewing logs, and managing dependencies.

Common Options

Option	Description
`--status-all`	Display the status of all available services
`--full-restart`	Perform a complete restart (if supported by the service)

Exit Status

Exit Code	Meaning
`0`	Command completed successfully
Non-zero	An error occurred (such as an unknown service or insufficient permissions)

When to Use `service`

Use **`service`** when:

- Working on older Linux systems that use **SysVinit**.
- Running legacy scripts that expect the `service` command.
- You want a simple interface to manage services.

Use **`systemctl`** when:

- Managing services on modern Linux distributions (Ubuntu 16.04+, Debian 8+, RHEL/CentOS 7+, Fedora, etc.).
- You need advanced features like enabling services at boot, checking dependencies, or viewing detailed status.

Summary

- **Purpose:** Manage system services.
- **Common actions:** `start`, `stop`, `restart`, `reload`, `status`.
- **Requires:** Usually `sudo` for administrative actions.
- **Modern equivalent:** `systemctl`.
- **Best for:** Legacy compatibility and simple service management.

3. journalctl

`journalctl` is a Linux command used to **view, search, and manage system logs** collected by **systemd-journald**. It provides a centralized way to examine logs from the kernel, services, applications, and the boot process.

Unlike traditional log files stored in `/var/log`, `journalctl` reads from the **systemd journal**, which can contain both persistent and in-memory logs.

Syntax

```
journalctl [OPTIONS]
```

What `journalctl` Logs

- Kernel messages
- System boot logs
- Service logs
- Application logs
- User session logs
- Authentication events
- Hardware events
- Systemd service status

Basic Examples

1. View all logs

```
journalctl
```

Displays every log entry stored in the journal.

Example:

```
Jul 27 10:15:01 server systemd[1]: Started Network Manager.
Jul 27 10:15:03 server sshd[854]: Accepted password for user
```

2. View recent logs

```
journalctl -e
```

Jumps directly to the end (most recent entries).

---

3. Follow logs in real time

```
journalctl -f
```

Similar to:

```
tail -f /var/log/messages
```

Useful for monitoring a service while it is running.

---

4. Show logs from the current boot

```
journalctl -b
```

Displays only messages generated since the last boot.

---

5. Show logs from the previous boot

```
journalctl -b -1
```

Useful after rebooting following a crash.

---

6. List previous boots

```
journalctl --list-boots
```

Example:

```
-2  5b2c...
-1  8f71...
 0  a132...
```

7. Show kernel messages

```
journalctl -k
```

Equivalent to viewing the kernel ring buffer.

Example:

```
Linux version 6.8
USB device detected
EXT4 filesystem mounted
```

---

8. Monitor kernel messages

```
journalctl -kf
```

Shows new kernel messages as they are generated.

View Logs for a Service

Example:

```
journalctl -u ssh
```

or

```
journalctl -u sshd
```

Shows only SSH service logs.

Another example:

```
journalctl -u NetworkManager
```

Show Recent Lines

Last 20 lines:

```
journalctl -n 20
```

Last 100 lines:

```
journalctl -n 100
```

Follow Recent Lines

```
journalctl -n 50 -f
```

Displays the last 50 entries, then continues showing new ones.

Filter by Time

Since today

```
journalctl --since today
```

Since yesterday

```
journalctl --since yesterday
```

Last hour

```
journalctl --since "1 hour ago"
```

Specific date

```
journalctl --since "2026-07-25"
```

Between two dates

```
journalctl --since "2026-07-25" --until "2026-07-26"
```

Between two times

```
journalctl --since "09:00" --until "10:00"
```

Filter by Priority (Log Level)

Priority	Meaning
0	Emergency
1	Alert
2	Critical
3	Error
4	Warning
5	Notice
6	Information
7	Debug

Example:

Errors only:

```
journalctl -p err
```

Warnings and above:

```
journalctl -p warning
```

Critical only:

```
journalctl -p crit
```

View Logs for a Process

By PID:

```
journalctl _PID=1234
```

View Logs for a User

```
journalctl _UID=1000
```

View Logs by Executable

```
journalctl /usr/bin/sshd
```

Search Logs

```
journalctl | grep error
```

Or search directly:

```
journalctl | grep "failed"
```

Output Without Paging

Normally `journalctl` opens in the `less` pager.

Disable paging:

```
journalctl --no-pager
```

Display Newest Entries First

```
journalctl -r
```

Display in Short Format

```
journalctl -o short
```

Display as JSON

```
journalctl -o json
```

Useful for scripting and automation.

Check Disk Usage

```
journalctl --disk-usage
```

Example:

```
Archived and active journals take up 420.0M on disk.
```

Remove Old Logs

Keep only the last 7 days:

```
sudo journalctl --vacuum-time=7d
```

Keep only 500 MB:

```
sudo journalctl --vacuum-size=500M
```

Keep only 5 journal files:

```
sudo journalctl --vacuum-files=5
```

Common Troubleshooting Examples

Why didn't a service start?

```
journalctl -u nginx
```

Why did the system reboot?

```
journalctl -b -1
```

Check boot errors

```
journalctl -b -p err
```

Watch authentication logs

```
journalctl -u sshd -f
```

Monitor USB events

```
journalctl -k -f
```

Common Options

Option	Description
`-b`	Current boot logs
`-b -1`	Previous boot
`-f`	Follow logs in real time
`-k`	Kernel logs only
`-u service`	Logs for a specific service
`-n N`	Show last *N* lines
`-e`	Jump to the end
`-p level`	Filter by priority
`-r`	Reverse order (newest first)
`--since`	Show logs after a specific time
`--until`	Show logs before a specific time
`--no-pager`	Print directly to the terminal
`--disk-usage`	Show journal storage usage
`--vacuum-time`	Remove logs older than a specified age
`--vacuum-size`	Limit total journal size
`-o json`	JSON output

Advantages of `journalctl`

- Centralized logging for the entire system.
- Combines kernel, service, and application logs in one place.
- Powerful filtering by service, boot, time, priority, user, or process.
- Supports real-time monitoring with `-f`.
- Can store logs persistently across reboots (if configured).
- Structured log format makes automation and scripting easier.

Summary

`journalctl` is the primary log-viewing utility on **systemd-based Linux distributions** (such as Ubuntu, Debian, Fedora, RHEL, CentOS Stream, and openSUSE). It allows administrators to inspect system activity, troubleshoot service failures, analyse boot problems, and monitor logs in real time from a single, unified journal.

4. loginctl

`loginctl` is a **systemd utility** used to **manage and inspect user login sessions, seats, and users** on a Linux system. It communicates with the **systemd-logind** service, which tracks user logins, graphical sessions, SSH sessions, virtual terminals, and power management events.

It is commonly used on modern Linux distributions such as Ubuntu, Debian, Fedora, RHEL, CentOS Stream, openSUSE, and Arch Linux.

Syntax

```
loginctl [OPTIONS] COMMAND [NAME...]
```

What `loginctl` Can Do

- View active login sessions
- View logged-in users
- Show session details
- Terminate user sessions
- Lock or unlock sessions
- Enable/disable user lingering
- Inspect system seats (physical workstations)

Common Commands

1. List all active sessions

```
loginctl
```

or

```
loginctl list-sessions
```

Example:

```
SESSION UID USER    SEAT  TTY
      2 1000 john   seat0 tty2
      5    0 root         pts/0
```

Columns:

Column	Description
SESSION	Session ID
UID	User ID
USER	Username
SEAT	Physical seat (desktop)
TTY	Terminal associated with the session

---

2. List logged-in users

```
loginctl list-users
```

Example:

```
UID USER
0   root
1000 john
1001 alice
```

3. View session details

```
loginctl show-session 2
```

Example output:

```
Id=2
User=1000
Name=john
TTY=tty2
Remote=no
Service=login
Type=tty
State=active
```

Useful information includes:

- Session state
- Login service
- TTY
- Remote/local login
- Idle time

---

4. Show user information

```
loginctl show-user john
```

Displays:

- UID
- Runtime directory
- Active sessions
- Linger status

5. Show seat information

```
loginctl list-seats
```

Example:

```
SEAT
seat0
```

To inspect a seat:

```
loginctl show-seat seat0
```

---

6. Lock a session

```
sudo loginctl lock-session 2
```

Locks session ID 2.

---

7. Unlock a session

```
sudo loginctl unlock-session 2
```

---

8. Lock all sessions

```
sudo loginctl lock-sessions
```

---

9. Terminate a session

```
sudo loginctl terminate-session 2
```

Ends that session immediately.

10. Kill all sessions of a user

```
sudo loginctl terminate-user john
```

This logs out all of the user's sessions.

---

11. Enable user lingering

```
sudo loginctl enable-linger john
```

This allows the user's systemd user services to continue running even after they log out.

Verify:

```
loginctl show-user john
```

Look for:

```
Linger=yes
```

---

12. Disable lingering

```
sudo loginctl disable-linger john
```

---

13. Show all session properties

```
loginctl show-session 2
```

Only one property:

```
loginctl show-session 2 -p State
```

Example:

```
State=active
```

Multiple properties:

```
loginctl show-session 2 -p Name -p State -p IdleHint
```

14. Show current session

```
loginctl session-status
```

Or specify a session:

```
loginctl session-status 2
```

This displays:

- Processes
- Login time
- TTY
- Resource usage

---

15. Display user status

```
loginctl user-status john
```

Shows:

- Sessions
- Running processes
- User services

Common Options

Option	Description
`--no-pager`	Disable paging of output
`--no-legend`	Hide table headers
`--help`	Show help
`--version`	Show version
`-H host`	Connect to a remote host
`-M machine`	Operate on a local container or VM

Example:

```
loginctl --no-pager list-sessions
```

Practical Examples

Check who is logged in

```
loginctl list-users
```

View all active sessions

```
loginctl list-sessions
```

View details of session 4

```
loginctl show-session 4
```

Lock your desktop session

```
sudo loginctl lock-session
```

End a user's session

```
sudo loginctl terminate-user alice
```

Check whether lingering is enabled

```
loginctl show-user alice -p Linger
```

Enable lingering

```
sudo loginctl enable-linger alice
```

Relationship with `systemd-logind`

`loginctl` is a client for the **systemd-logind** daemon.

`systemd-logind` is responsible for:

- Tracking user logins
- Managing login sessions
- Handling seats and virtual terminals
- Managing idle detection
- Coordinating suspend, hibernate, and shutdown requests
- Supporting desktop environments (GNOME, KDE, etc.)

Without `systemd-logind`, `loginctl` cannot manage sessions.

When to Use `loginctl`

- Troubleshoot login or desktop session issues.
- Check which users are currently logged in.
- View SSH, console, and graphical sessions.
- Log out or terminate stuck user sessions.
- Lock or unlock desktop sessions.
- Enable lingering so a user's services continue after logout.
- Inspect session, seat, and user information on systems using `systemd`.

Summary

Feature	`loginctl`
Purpose	Manage and inspect login sessions, users, and seats
Managed by	`systemd-logind`
Requires root?	Viewing information generally does not; terminating sessions, locking/unlocking, and changing lingering typically do
Common use cases	List sessions, view session details, terminate sessions, lock screens, manage lingering
Typical systems	Linux distributions that use `systemd`