CHAPTER-9 CONTROL SERVICES AND DAEMONS

Identify Automatically Started System Processes

Introduction to the systemd Daemon

The **systemd daemon** manages the **startup process for Linux**, including service startup and service management in general. The **systemd** daemon **activates system resources**, server daemons, and other processes both **at boot time and on a running system**.
Daemons are processes that either **wait or run in the background**, to perform various tasks. Generally, **daemons start automatically at boot time** and continue to run until shutdown or until you manually stop them. It is a convention to end the daemon names with the letter d. A service in the systemd sense often refers to one or more daemons.

In Red Hat Enterprise Linux, the **first process that starts (PID 1) is the systemd daemon**, **which provides these features:**

- Parallelization capabilities (starting multiple services simultaneously), which increase the boot speed of a system.
- On-demand starting of daemons without requiring a separate service.
- Automatic service dependency management, which can prevent long timeouts. For example, a network-dependent service does not attempt to start until the network is available.
- A method of tracking related processes together by using Linux control groups.

Service Units Description

A **systemd unit** is an abstract concept that is used to define **objects that the system knows** how to manage.

Units are represented by configuration files called unit files, which encapsulate information about system services, listening sockets, and other objects that are relevant to the systemd init system.

A unit has a name and a unit type. The name provides a unique identity to the unit. The unit type enables units to be grouped together with other similar unit types.

The systemd daemon uses units to manage different types of objects:

- **Service units** have a **.service extension and represent system services**. You can use service units to start frequently accessed daemons, such as a web server.
- **Socket units** have a **.socket extension and represent inter-process communication** (IPC) sockets that systemd should monitor. If a client connects to the socket, then the system manager starts a daemon and passes the connection to it. You can use socket units to delay the start of a service at boot time and to start less frequently used services on demand.
- **Path units** have a **.path extension and delay the activation of a service until a specific filesystem change occurs**. You can use path units for services that use spool directories, such as a printing system.

DESCRIPTIN	COMMANDS / OPTIONS
To manage units, use the systemctl command ----------	**Syntax:** systemctl [command] [service] **To lists and paginates all currently loaded service units** **Example:** [root@host ~]# systemctl **list-units** --type=service-l↵ ---------- The --type=service option limits the type of systemd units to service units. The output has the following columns: **UNIT** **:** The service unit name. **LOAD** **:**Whether the systemd daemon properly parsed the unit's configuration and loaded the unit into memory. **ACTIVE** **:** The high-level activation state of the unit. This information indicates whether the unit started successfully. **SUB** **:** The low-level activation state of the unit. This information indicates more detailed information about the unit. The information varies based on unit type, state, and how the unit is executed. **DESCRIPTION:** The short description of the unit. ----------

	To lists all service units regardless of the activation states **Example:** [root@host ~]# systemctl **list-units** --type=service --all↵ **To View a unit's status of service** **Example:** [root@host ~]# systemctl **status** sshd.service
---	---
Service Unit Information	**Loaded** Whether the service unit is loaded into memory. **Active** Whether the service unit is running and if so, for how long. **Docs** Where to find more information about the service. **Main PID** The main process ID of the service, including the command name. **Status** More information about the service. **Process** More information about related processes. **CGroup** More information about related control groups.
Service States in the Output of systemctl	**loaded** The unit configuration file is processed. **active** (running) The service is running with continuing processes. **active (exited)** The service successfully completed a one-time configuration. **active (waiting)** The service is running but waiting for an event. **inactive** The service is not running. **enabled** The service starts at boot time. **disabled** The service is not set to start at boot time. **static** The service cannot be enabled, but an enabled unit might start it automatically.
---	---
	To verifies the specific states of a service **Example:** [root@host ~]# systemctl **is-active** sshd.service **To verify whether a service unit is enabled to start automatically during system boot** **Example:** [root@host ~]# systemctl **is-enabled** sshd.service

Note : for more systemctl command check **Summary of systemctl Commands**	**To verify whether the unit failed during startup** **Example:** [root@host ~]# systemctl **is-failed** sshd.service

Control System Services

You can manually start, stop, or reload services to update the service, update the configuration file, uninstall the service, or manually manage an infrequently used service. Use the systemctl status command to verify the status of a service, if the service is running or stopped.

Summary of systemctl Commands

Command	Task
`systemctl status UNIT`	View detailed information about a unit's state.
`systemctl stop UNIT`	Stop a service on a running system.
`systemctl start UNIT`	Start a service on a running system.
`systemctl restart UNIT`	Restart a service on a running system.
`systemctl reload UNIT`	Reload the configuration file of a running service.
`systemctl mask UNIT`	Disable a service from being started, both manually and at boot.
`systemctl unmask UNIT`	Make a masked service available.
`systemctl enable UNIT`	Configure a service to start at boot time. Use the `--now` option to also start the service.
`systemctl disable UNIT`	Disable a service from starting at boot time. Use the `--now` option to also stop the service.