

# Booting Up and Shutting Down

Debian Linux: GRUB 2, systemd, service startup, and recovery

tsaimh (2022-2026, CC BY-SA)

lctseng (2019-2021, CC BY-SA)

? (1996-2018)

# What You Will Learn

- Explain how firmware, the boot loader, kernel, and PID 1 cooperate.
- Distinguish BIOS/MBR examples from UEFI/GPT installations.
- Manage Debian systemd services and inspect unit dependencies.
- Enter maintenance mode and diagnose a failed boot.
- Choose the correct command for halt, poweroff, or reboot.

## Course platform

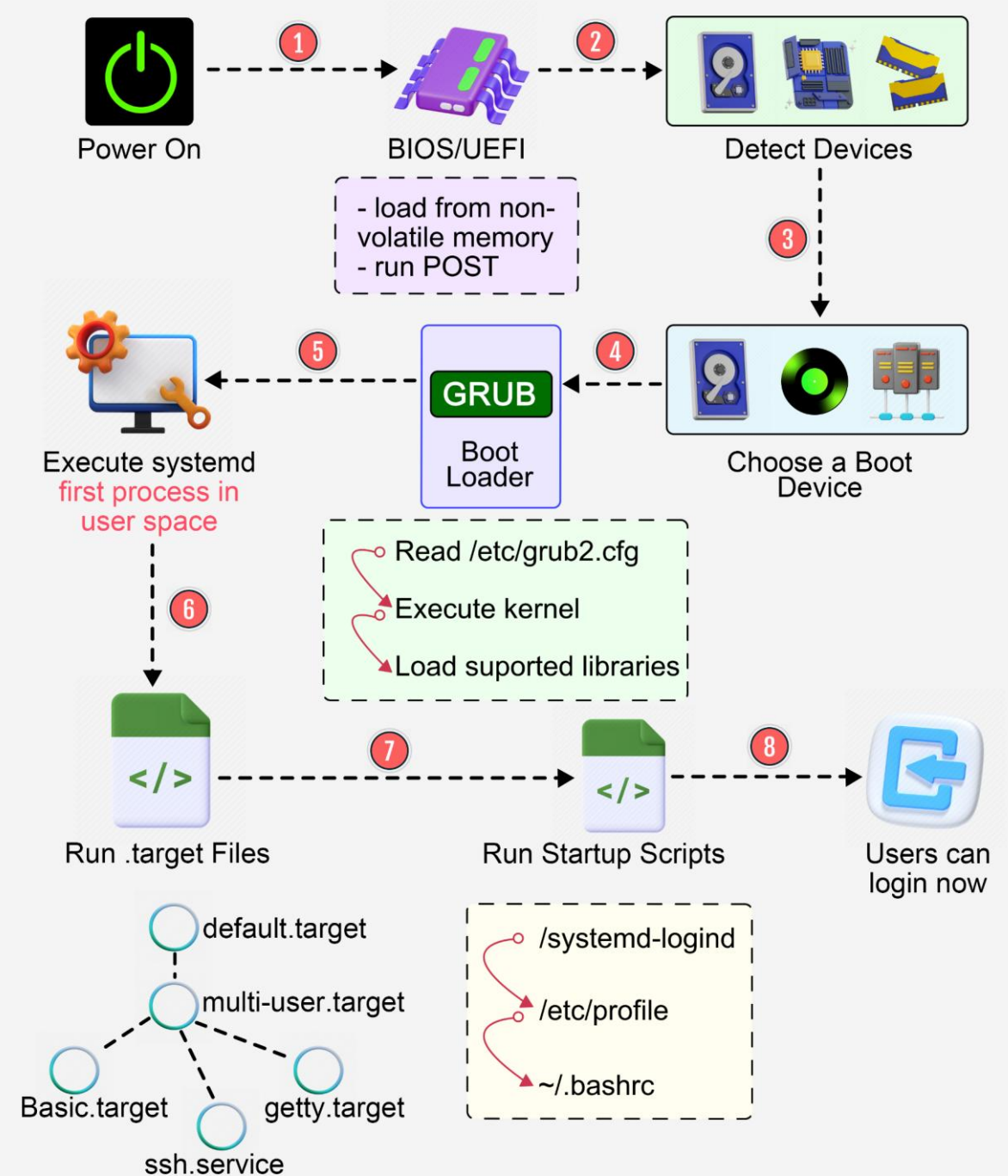
Debian 13 (trixie), amd64 VM, GRUB 2, and systemd. FreeBSD appears only in comparisons.

# Linux Boot Process

Source: <https://bytebytego.com/guides/linux-boot-process-explained/>

## Linux Boot Process Explained

ByteByteGo



# 1. Firmware and Boot Loaders

Understand which component owns each step before changing boot configuration.

# Boot Process Overview

| Stage          | Responsibility                                                  |
|----------------|-----------------------------------------------------------------|
| Firmware       | Initialize the platform and select a boot device or EFI program |
| Boot loader    | Load the kernel, modules, and optional initramfs                |
| Kernel         | Initialize drivers and memory, then prepare the root filesystem |
| PID 1          | Start the user-space system and supervise services              |
| Login services | Provide console, SSH, or graphical sessions                     |

## Two separate layers

A boot loader loads the OS. A service manager starts and manages user-space services.

# BIOS/MBR and UEFI/GPT

| Feature            | Legacy BIOS / MBR example      | Typical UEFI / GPT installation         |
|--------------------|--------------------------------|-----------------------------------------|
| First executable   | Boot code in disk sector 0     | EFI program selected by firmware        |
| Partition metadata | MBR partition table            | GPT headers and partition entries       |
| Loader storage     | Boot sectors and OS filesystem | EFI System Partition (ESP), usually FAT |
| Debian example     | GRUB boot.img and core.img     | shim / GRUB EFI loader on the ESP       |

## Independent choices

Firmware mode and partition format are distinct. BIOS can boot GPT with suitable boot code.

# The Master Boot Record

- Classic MBR: 446 bytes for boot code, 64 bytes for four entries, and a 2-byte signature.
- Signature bytes at offsets 510 and 511 are 55 aa.
- GRUB BIOS boot code hands control to core.img; an EFI loader uses a different path.
- GPT usually retains a protective MBR. Its presence does not prove BIOS booting.

## Read the actual layout

Use lsblk and partition metadata. Never infer firmware mode from an MBR signature alone.

# Debian GRUB 2: BIOS Boot Path

| Component     | Responsibility                                    |
|---------------|---------------------------------------------------|
| BIOS          | Read the initial boot code from the selected disk |
| GRUB boot.img | Transfer control to core.img                      |
| GRUB core.img | Load modules and find the GRUB configuration      |
| GRUB menu     | Select a kernel and load kernel plus initramfs    |

## BIOS with GPT

GRUB normally needs a BIOS Boot Partition for core.img. This is separate from the ESP.

# Debian Boot Files

| Path                       | Purpose                           |
|----------------------------|-----------------------------------|
| /boot/vmlinuz-<version>    | Linux kernel image                |
| /boot/initrd.img-<version> | Early user-space image            |
| /boot/grub/grub.cfg        | Generated GRUB menu configuration |
| /etc/default/grub          | Administrator GRUB settings       |
| /etc/grub.d/               | Scripts used to generate the menu |

## Matching versions

Keep the kernel and initramfs pair consistent. Preserve a known working kernel for recovery.

# Debian UEFI Booting

```
# Inspect, do not modify
test -d /sys/firmware/efi && echo UEFI
lsblk -f
findmnt /boot/efi
sudo efibootmgr -v
```

- Firmware selects an EFI loader from the ESP.
- With Secure Boot, shim validates the next loader; GRUB loads the kernel and initramfs.
- On amd64, EFI/debian commonly holds shimx64.efi and grubx64.efi.
- Verify the actual EFI entry and mounted ESP.

## Installation detail

/boot/efi is the usual Debian ESP mount point. Secure Boot and architecture affect the installed files.

# GRUB Configuration on Debian

```
# Edit /etc/default/grub
GRUB_TIMEOUT=5
GRUB_CMDLINE_LINUX_DEFAULT="quiet"

# Regenerate the GRUB menu
sudo update-grub
```

- Edit /etc/default/grub and /etc/grub.d/.
- update-grub regenerates /boot/grub/grub.cfg.
- Do not edit the generated file for persistent changes.
- Kernel arguments are not systemd service settings.

## Two different operations

update-grub regenerates a menu. grub-install installs loader code; use it only for the identified boot path.

# Multibooting and Boot Repair

- Identify BIOS or UEFI mode, disk layout, root filesystem, and encryption first.
- Inspect EFI entries and ESP files, or BIOS GRUB embedding areas.
- Try a known working kernel from the GRUB menu before reinstalling the loader.
- Use Debian installer rescue mode or suitable live media for repairs.

## Scope matters

Mount the actual root, /boot, and ESP as needed. A menu update does not repair every loader failure.

## 2. Kernel and Service Startup

Follow the handoff from the kernel to PID 1 and the service framework.

# Kernel and Initramfs Initialization

| Debian file             | Role                                             |
|-------------------------|--------------------------------------------------|
| /boot/vmlinuz-<version> | Initialize the Linux kernel and built-in drivers |

/boot/initrd.img-<version> Prepare storage and mount the real root filesystem

- Early user space may load drivers, unlock encrypted storage, and activate LVM.
- After the root handoff, systemd runs as PID 1.
- initramfs-tools and dracut have different configuration and recovery details.

## Identify the generator

The examples here use initramfs-tools. Inspect the installed setup before rebuilding an image.

# Devices and Kernel Modules

```
# Read-only Debian inspection
uname -r
lsmod
modinfo virtio_blk
sudo journalctl -k -b
```

- Modules normally live under `/lib/modules/<kernel-version>`.
- `modprobe` resolves module dependencies.
- Root-storage drivers must be available in the kernel or `initramfs`.
- Do not unload a driver in active use.

## Persistent configuration

`modprobe.d` controls module options. `modules-load.d` requests static loading; most devices use automatic loading.

# PID 1 and Kernel Tasks

```
ps -p 1 -o pid,comm  
ps ax | head
```

| Platform              | PID 1   | Kernel tasks     |
|-----------------------|---------|------------------|
| Debian<br>(systemd)   | systemd | kthreadd, others |
| FreeBSD<br>comparison | init    | Kernel threads   |

- PID 1 starts or supervises the user-space system and handles orphaned processes.
- Kernel tasks originate inside the kernel. Their names and PID assignments can change.

## Stable concept

Teach the role of PID 1 rather than memorizing a version-specific list of kernel PIDs.

# systemd Unit Files on Debian

| Location or command                     | Purpose                                 |
|-----------------------------------------|-----------------------------------------|
| <code>/usr/lib/systemd/system/</code>   | Package-provided units                  |
| <code>/etc/systemd/system/</code>       | Local units and administrator overrides |
| <code>/run/systemd/system/</code>       | Runtime unit configuration              |
| <code>systemctl cat ssh.service</code>  | Show the unit and drop-in files         |
| <code>systemctl edit ssh.service</code> | Create a local drop-in override         |

## After an edit

Run `daemon-reload` for manual unit edits, then restart/reload as appropriate. Do not edit vendor units in place.

# Debian SSH Service Management

```
# Requires openssh-server
systemctl status ssh.service
systemctl is-enabled ssh.service

# Start now and enable for boot
sudo systemctl enable --now ssh.service

# Check SSH configuration syntax
sudo /usr/sbin/sshd -t
```

- Debian openssh-server provides ssh.service.
- start changes runtime; enable changes activation policy.
- Validate SSH configuration before reload or restart.

## Connection test

Keep a console or existing session until a second SSH login succeeds. Inspect ssh.socket if socket activation is configured.

# Other Startup Frameworks

| Framework          | Configuration and tool                        |
|--------------------|-----------------------------------------------|
| Debian systemd     | Units, drop-ins, systemctl                    |
| Legacy SysV        | /etc/init.d and /etc/rcN.d; ordered S/K links |
| FreeBSD comparison | /etc/rc.conf, rcorder, service, sysrc         |

- Recognize legacy interfaces when reading older material.
- Use native systemd units for Debian course examples.

## Different platforms

FreeBSD rc settings and boot0/boot1/boot2 are not Debian configuration or boot stages.

# systemd Units and Dependencies

| Unit type | Purpose                                | Example               |
|-----------|----------------------------------------|-----------------------|
| service   | Run or supervise a service             | ssh.service           |
| socket    | Activate a service from a socket       | Example: app.socket   |
| mount     | Represent a filesystem mount           | home.mount            |
| timer     | Schedule activation                    | Example: backup.timer |
| target    | Group units and synchronization points | multi-user.target     |

## Ordering vs requirement

After= controls ordering. Requires= and Wants= express dependency relationships.

# Targets and Historical Runlevels

| SysV convention | systemd target    | Meaning                                |
|-----------------|-------------------|----------------------------------------|
| 0               | poweroff.target   | Power off                              |
| 1               | rescue.target     | Maintenance with a rescue shell        |
| 2, 3, 4         | multi-user.target | Non-graphical multi-user system        |
| 5               | graphical.target  | Multi-user system with graphical login |
| 6               | reboot.target     | Reboot                                 |

## Compatibility mapping

Runlevel meanings vary among SysV distributions. systemd targets are dependency groups, not numeric boot steps.

# systemctl: Runtime and Boot Policy

```
# Read-only inspection
systemctl get-default
systemctl status ssh.service
systemctl is-enabled ssh.service

# Start now and enable at boot
sudo systemctl enable --now ssh.service
```

- start changes runtime state.
- enable configures activation at boot or through dependencies.
- daemon-reload rereads unit definitions after you edit a unit file.

## Debian unit names

Use ssh.service for the packaged OpenSSH server. Inspect the installed unit and any socket activation.

# Login Sessions and fork/exec

| Step                | What happens                                             |
|---------------------|----------------------------------------------------------|
| 1. Console listener | init or a service manager starts a getty process         |
| 2. Authentication   | login checks the account and authentication policy       |
| 3. User session     | login executes the configured login shell                |
| 4. External command | The shell typically forks a child that execs the program |

## Process identity

fork creates a new process/PID. exec replaces a process image while keeping its PID. SSH follows its own session path.

# 3. Maintenance and Recovery

Use a console and the correct recovery path for the installed system.

# Rescue and Emergency Modes

| Mode             | Typical environment                              | Limitation                          |
|------------------|--------------------------------------------------|-------------------------------------|
| rescue.target    | Maintenance shell with basic services and mounts | Network is not guaranteed           |
| emergency.target | Minimal shell; fewer units and mounts            | Root may be read-only               |
| Installer rescue | Repair the installed system from external media  | Mount/unlock the correct root first |

## Authentication

Rescue authentication depends on root-account state and sulogin configuration. Test your VM recovery path in advance.

# Entering a Maintenance Mode

```
# In GRUB: press e on a menu entry
# Append to the linux line
systemd.unit=rescue.target
# Or use emergency.target

# Running Debian, console only
sudo systemctl rescue
```

- GRUB Ctrl+X or F10 boots the temporary edited entry.
- Temporary edits affect this boot, not /etc/default/grub.
- Entering rescue on a running system interrupts services and sessions.

## Console required

Use a tested VM console. Do not switch to rescue over SSH and assume the connection will survive.

# Rescue Authentication on Debian

```
# Read-only policy inspection  
sudo passwd -S root  
systemctl cat rescue.service  
systemctl cat emergency.service
```

- Rescue/emergency shells commonly use sulogin and root-account authentication.
- Some Debian installations lock root and give the first user sudo access.
- A locked root account may prevent the usual rescue login; behavior depends on the configured rescue service.
- Prepare installer rescue media and verify access before a real failure.

## Different protection layers

Console access, GRUB access control, root authentication, and disk encryption protect different parts of the path.

# Filesystems in Maintenance Mode

```
# Inspect the current root  
findmnt /  
lsblk -f  
  
# Only when safe to make root writable  
sudo mount -o remount,rw /
```

- Read the error and inspect mounts before changing files.
- Repair ext4 with suitable tools on an unmounted filesystem, usually from rescue media.
- XFS and Btrfs use their own recovery tools; identify the filesystem first.
- After repair, reboot or return to the normal target as appropriate.

## No universal fsck command

Do not run a repair operation against the mounted root filesystem. Remounting is not filesystem repair.

# Boot Failure: Locate the Layer

| Observed symptom          | First checks                                           |
|---------------------------|--------------------------------------------------------|
| No firmware / loader menu | Boot order, detected storage, EFI entries or boot code |
| Loader cannot find kernel | Loader paths, installed kernel, filesystem access      |
| Kernel cannot mount root  | Root device, drivers, encryption, and pool/filesystem  |
| Service startup fails     | systemd units, drop-ins, dependencies, and journal     |
| Login fails               | Account, authentication, shell, SSH policy             |

## Efficient diagnosis

Record the first meaningful error and the last successful stage before changing configuration.

# Debian Boot and Service Logs

```
systemctl --failed  
sudo journalctl -b -p warning  
sudo journalctl -k -b  
sudo journalctl -b -u ssh.service
```

```
# Previous boot, if retained  
sudo journalctl -b -1
```

## Log retention

Previous-boot logs require retained journal data. `/var/log/syslog` depends on whether a `syslog` daemon is installed.

## 4. Shutdown and Reboot

Stop services cleanly and select the intended final machine state.

# Orderly Shutdown

- Notify users and check for active work or maintenance tasks.
- The shutdown framework stops services and terminates remaining processes.
- The OS flushes pending writes and unmounts filesystems when possible.
- The final action halts, powers off, or restarts the system.

## Data integrity

sync alone does not replace an orderly shutdown. Sudden power loss can interrupt application transactions.

# Halt, Poweroff, and Reboot

| Action   | Final state                         | Debian example     |
|----------|-------------------------------------|--------------------|
| Halt     | Stop execution; power may remain on | systemctl halt     |
| Poweroff | Stop and request power removal      | systemctl poweroff |
| Reboot   | Stop and restart                    | systemctl reboot   |

## Command semantics

halt and reboot wrappers differ across systems. Prefer explicit high-level commands and read the local manual.

# Debian Shutdown Commands

```
# Schedule a reboot
sudo shutdown -r +10 "Maintenance"

# Cancel the pending schedule
sudo shutdown -c

# Immediate explicit actions
# sudo systemctl poweroff
# sudo systemctl reboot
```

- +10 means ten minutes from now; now means immediately.
- Confirm the maintenance window and a way to reconnect.
- Use explicit poweroff/reboot commands for the intended final state.

## Class demonstration

Schedule and cancel on the disposable Debian VM. Verify that no pending shutdown remains.

# Shutdown Syntax Across Platforms

| Action        | Debian with systemd             | FreeBSD comparison           |
|---------------|---------------------------------|------------------------------|
| Power off now | <code>systemctl poweroff</code> | <code>shutdown -p now</code> |
| Reboot now    | <code>systemctl reboot</code>   | <code>shutdown -r now</code> |
| Halt          | <code>systemctl halt</code>     | <code>shutdown -h now</code> |

- On Debian systemd, `shutdown -h` normally requests `poweroff`.
- On FreeBSD, `shutdown -h` requests `halt`; `-p` requests `poweroff`.

## Do not copy flags blindly

Read the manual on the actual platform. Avoid `--force` during ordinary administration.

# Poweroff Troubleshooting

- Orderly shutdown may succeed even if firmware fails to remove power.
- Check the final console message, ACPI support, firmware settings, and system logs.
- In a virtual machine, verify the hypervisor receives and handles the poweroff request.
- Legacy APM kernel examples are historical and are not a general fix for modern systems.

## Failure distinction

A service that hangs during shutdown and a machine that halts without powering off require different investigations.

# Lab: Inspect the Boot Path

- In a disposable VM, record firmware mode, partition scheme, root filesystem, and PID 1.
- Locate the boot-loader configuration and the service-startup configuration.
- Inspect one service and explain runtime state versus enablement.
- Use the console to enter maintenance mode, inspect mounts, and return to normal boot.
- Schedule and cancel a reboot on the Debian VM; verify no pending shutdown remains.

## **Deliverable**

Submit the commands, relevant output, and a short explanation of the boot stages.

# Debian Boot and Shutdown Cheat Sheet

| Task                        | Debian command or file                    |
|-----------------------------|-------------------------------------------|
| GRUB settings               | /etc/default/grub; update-grub            |
| Service state / boot policy | systemctl status / is-enabled ssh.service |
| Enable and start            | systemctl enable --now ssh.service        |
| Boot messages               | journalctl -b; journalctl -k -b           |
| Maintenance boot            | systemd.unit=rescue.target                |
| Power off / reboot          | systemctl poweroff / reboot               |

## Verify before changing

Identify the layer, inspect the evidence, make one relevant change, and confirm the result.

# References

- Debian Reference: system initialization and system management
- GNU GRUB Manual; Debian update-grub(8)
- Debian Kernel Handbook; initramfs-tools(7), update-initramfs(8)
- Debian manuals: systemctl(1), systemd.unit(5), systemd.special(7), journalctl(1)
- Debian manuals: sulogin(8), shutdown(8), mount(8)

## Course scope

Debian 13 with systemd and GRUB 2. FreeBSD/SysV pages are comparisons; verify the installed configuration.