



Controlling Boot Process



Unit objectives

After completing this unit, you should be able to:

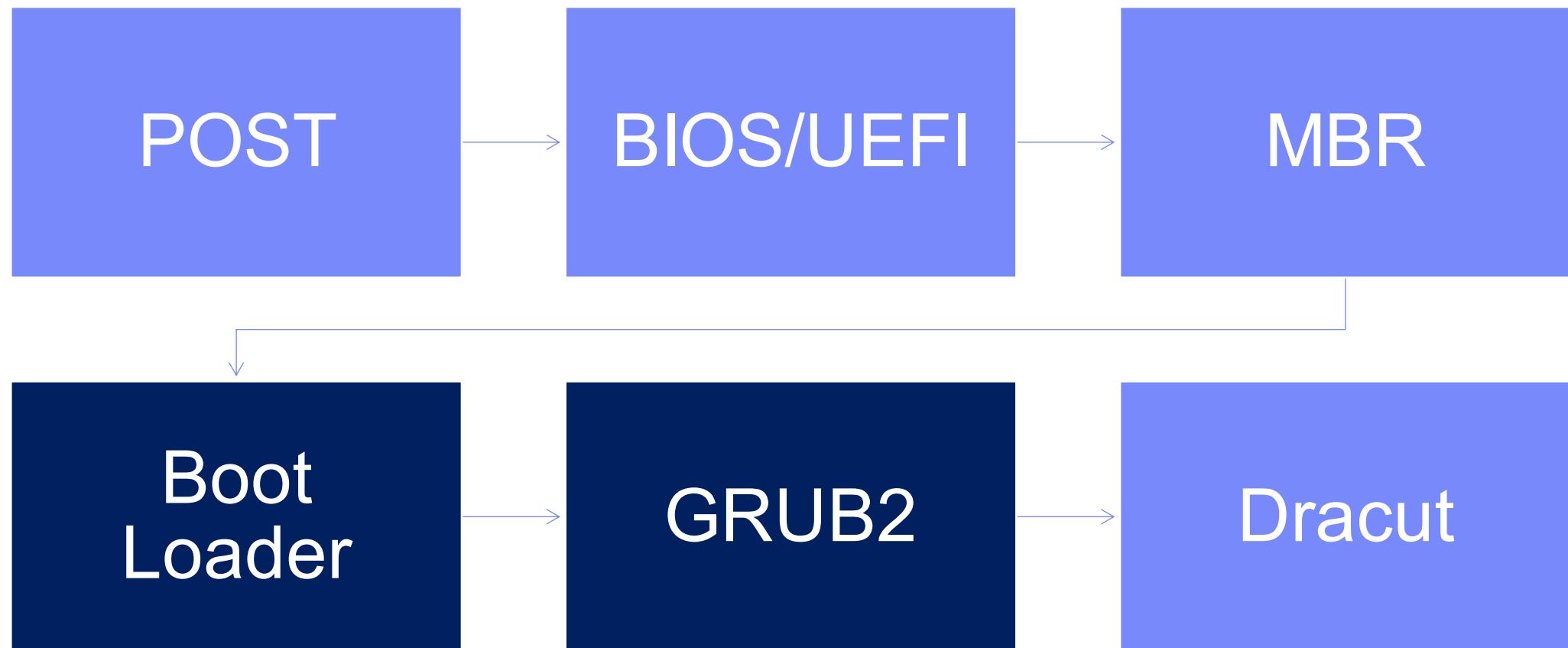
- Manage the Boot Loader and Kernel Command Line
- Explore the Boot Process and Selecting a Boot Target
- Repair Damaged File Systems at Boot Time



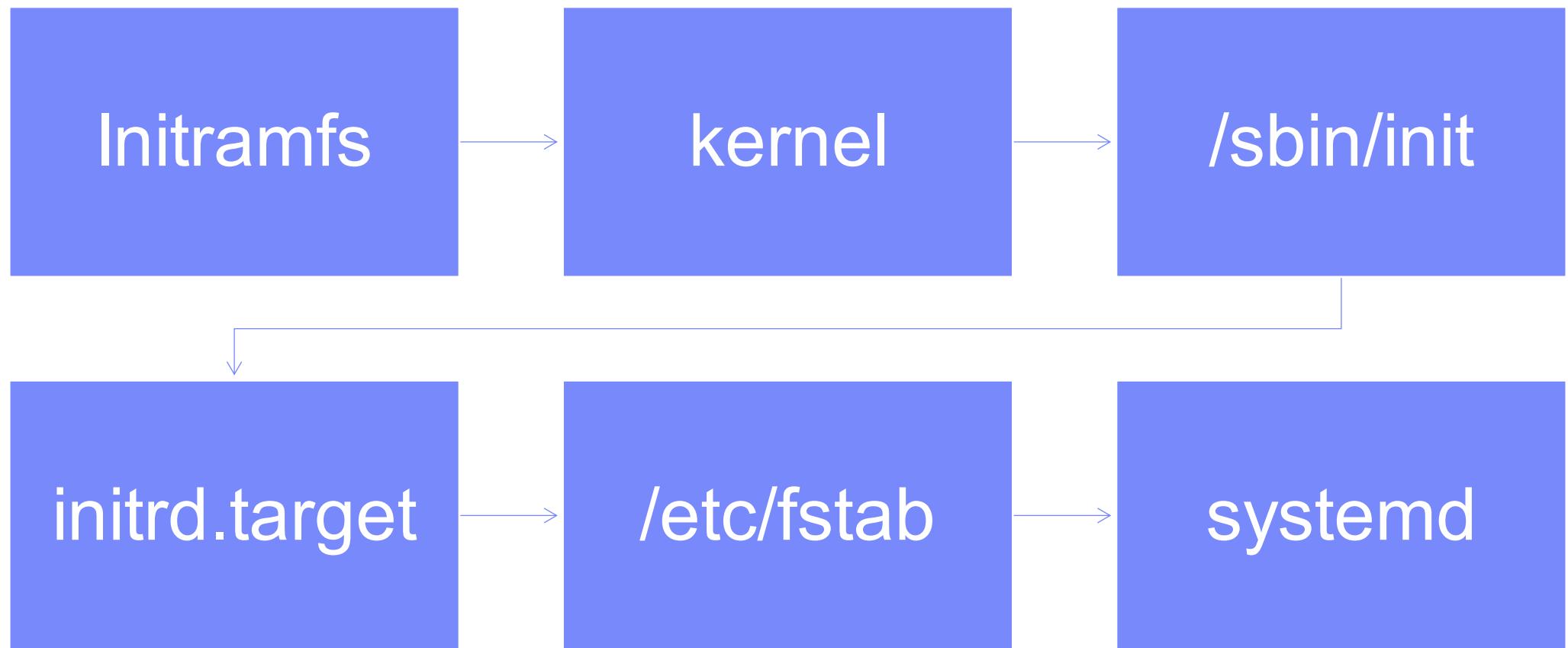
Managing the Boot Loader and Kernel Command Line

12.1

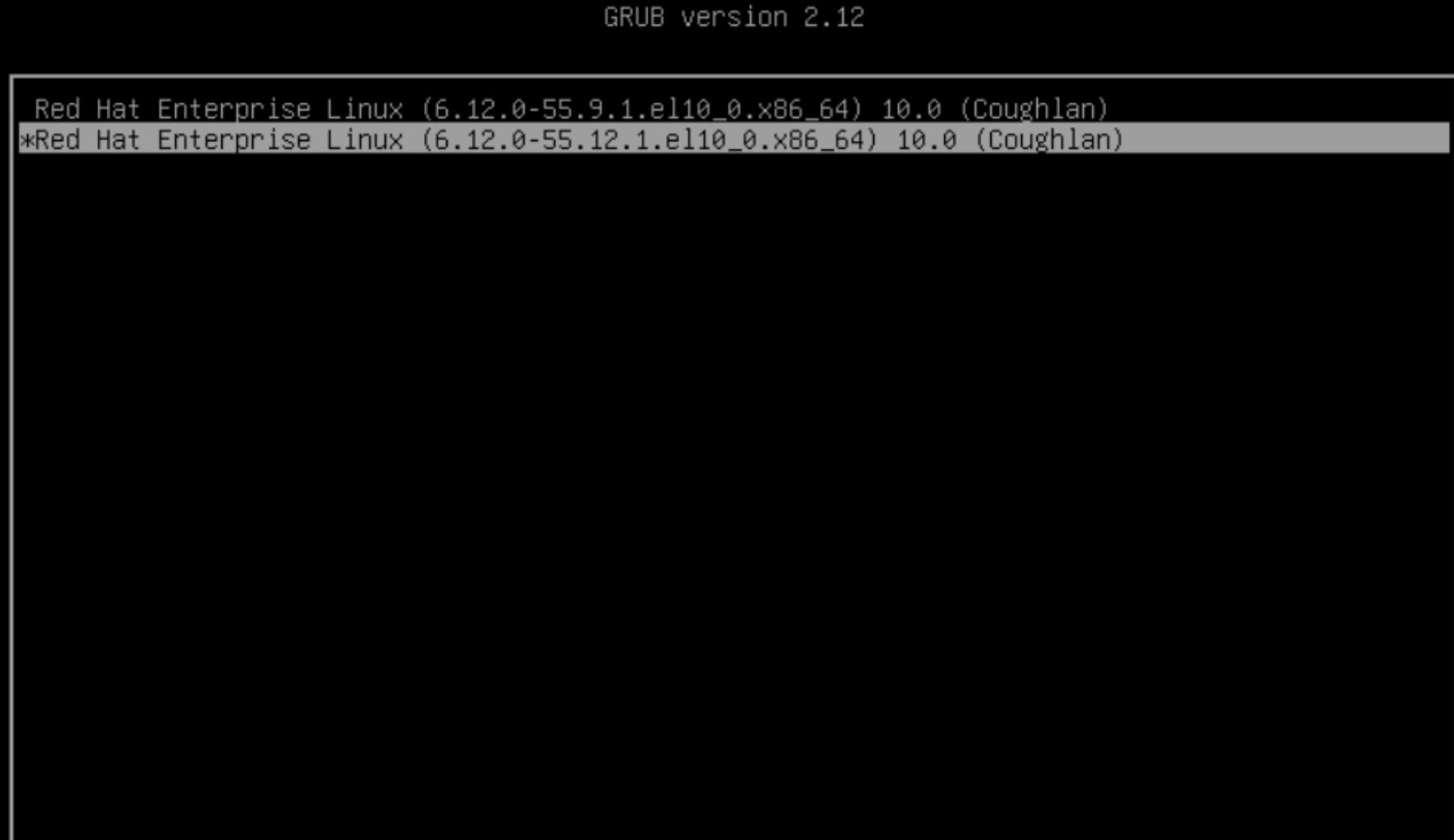
Boot Sequence 1/2



Boot Sequence 2/2



Further Boot Stages



Use the ↑ and ↓ keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the commands before booting or 'c' for a command-line.

GRUB2 editor

GRUB version 2.12

```
load_video
set gfxpayload=keep
insmod gzio
linux ($root)/boot/vmlinuz-6.12.0-55.12.1.el10_0.x86_64 root=UUID=15507695-22bb-4c65-94e6-a4\38e095983f console=tty0 console=ttyS0,115200n8 no_timer_check crashkernel=2G-64G:256M,64G-:5\12M
initrd ($root)/boot/initramfs-6.12.0-55.12.1.el10_0.x86_64.img $tuned_initrd
```

Minimum Emacs-like screen editing is supported. TAB lists completions. Press Ctrl-x or F10 to boot, Ctrl-c or F2 for a command-line or ESC to discard edits and return to the GRUB menu.

Controlling GRUB2 from a Booted System

- Use grubby command to:
 - View information about existing boot-menu entries

```
user@host:~$ grubby --info 1
index=1 1
kernel="/boot/vmlinuz-6.12.0-55.12.1.el10_0.x86_64" 2
args="console=tty0 ... crashkernel=2G-64G:256M,64G-:512M $tuned_params" 3
root="UUID=15507695-22bb-4c65-94e6-a438e095983f" 4
initrd="/boot/initramfs-6.12.0-55.12.1.el10_0.x86_64.img $tuned_initrd" 5
title="Red Hat Enterprise Linux (6.12.0-55.12.1.el10_0.x86_64) 10.0 (Coughlan)" 6
id="b70d796f931842cb9776f658cd068435-6.12.0-55.12.1.el10_0.x86_64" 7
```

Controlling GRUB2 from a Booted System

- Use grubby command to:
 - Changing default boot-menu entry

```
root@host:~# grubby --set-default-index 0
The default is /boot/loader/entries/ffffffffffffffffff-6.12.0-
55.9.1.el10_0.x86_64.conf with index 0 and kernel /boot/vmlinuz-6.12.0-
55.9.1.el10_0.x86_64
```

Controlling GRUB2 from a Booted System

- Use grubby command to:
 - Add/Remove kernel arguments

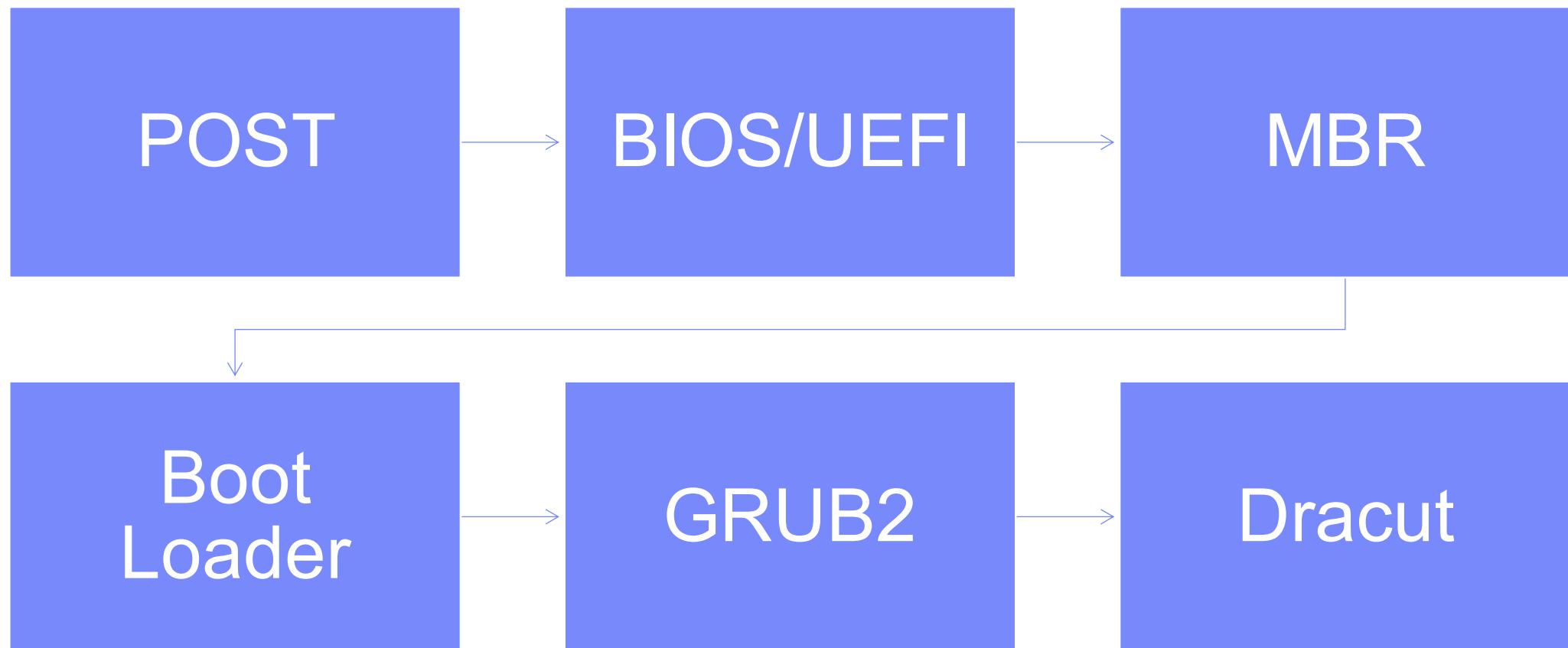
```
root@host:~# grubpy --update-kernel /boot/vmlinuz-6.12.0-55.9.1.el10_0.x86_64 \
--args="rhgb quiet"
```

```
root@host:~# grubpy --update-kernel /boot/vmlinuz-6.12.0-55.9.1.el10_0.x86_64 \
--remove-args="rhgb quiet"
```

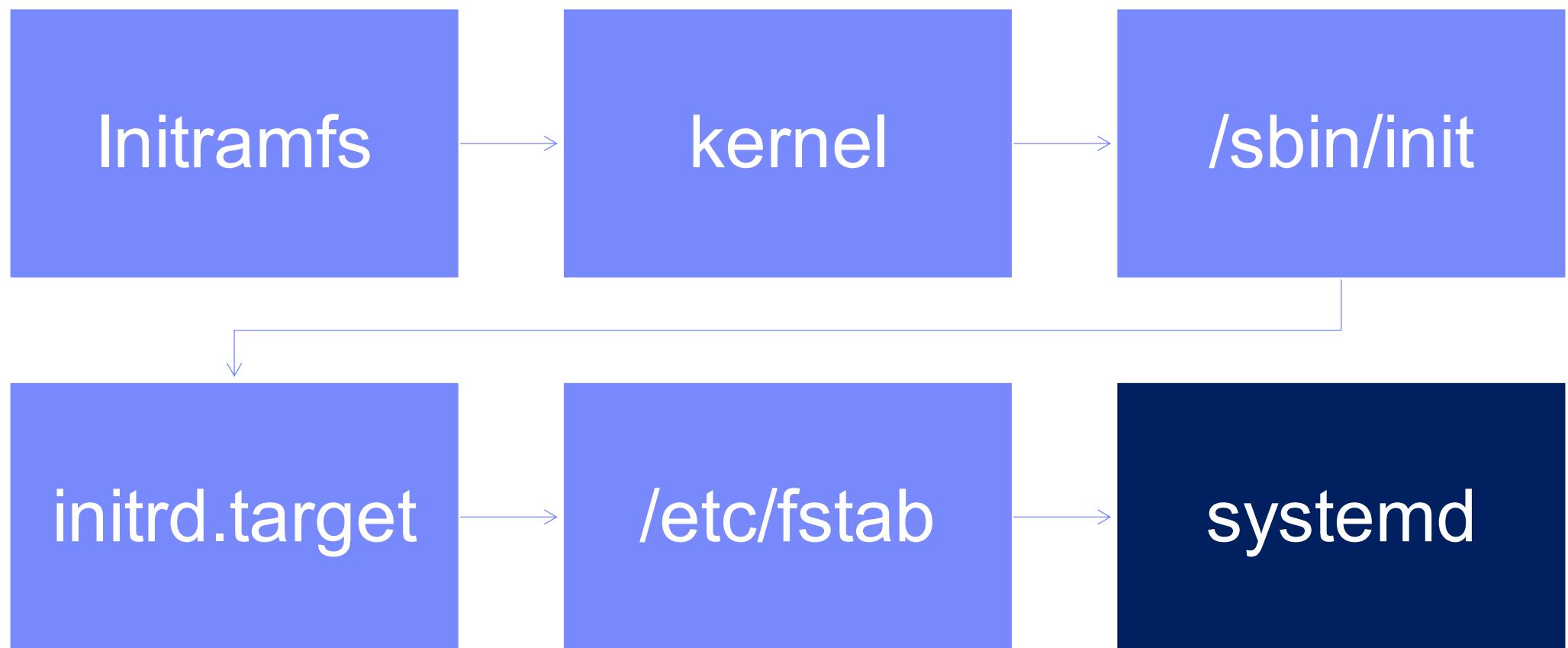
Explore the Boot Process and Selecting a Boot Target

12.3

Boot Sequence 1/2



Boot Sequence 2/2



Understanding Systemd Units and Unit Type

- Query each unit with

```
# systemctl -at <unit_type>
```

Unit Type	Description
Service	Process / Daemon
Target	Group of systemd services / units
Automount	Auto-mount file system
Device	Device file recognized by kernel
Mount	File system mount point
Path	File or directory in a file system
Scope	Externally created process
Slice	Group of hierarchilly organized units that manage system processes
Socket	Inter-process communication
Swap	Swap device or swap file
Timer	Systemd timer

Understanding Systemd Targets

- Is runlevel in System V
- Boot info a specific state
- Start group of services at each state
- Find default target in /etc/systemd/system
- Boot with all necessary processes / services
- Can be managed via
 - [systemctl command](#)
 - [Cockpit web-based](#)
 - [System Management Tool](#)

systemd Target Descriptions

Target name	Description
poweroff.target	Shutdown all services. Don't set as default target ;)
rescue.target	Only root can login for troubleshooting, diagnosing or any repairing work . Networking, GUI, multi-user not started
emergency.target	Starts the most minimal system for repairing your system when rescue.target fails to start.
graphical.target	Start all services with X Window system Default target Common state for desktop or workstation use
reboot.target	Reboot the system. Don't set as default target
network.target	use as part of dependency. Cannot not be isolated
local-fs.target	mount local file system. use as part of dependency. Cannot not be isolated
multi-user.target	Almost all services except services that start X Window system (GUI) Text based console login capability

View dependencies of a target

```
user@host:~$ systemctl list-dependencies graphical.target | grep target
graphical.target
* └─multi-user.target
*   ├─basic.target
*   ├─paths.target
*   ├─slices.target
*   ├─sockets.target
*   ├─sysinit.target
*   ├─cryptsetup.target
*   ├─integritysetup.target
*   ├─local-fs.target
...output omitted...
```

List all available targets

```
user@host:~$ systemctl list-units --type=target --all
UNIT                      LOAD   ACTIVE   SUB     DESCRIPTION
-----
basic.target                loaded  active   active  Basic System
...output omitted...
cloud-config.target          loaded  active   active  Cloud-config availability
cloud-init.target            loaded  active   active  Cloud-init target
cryptsetup-pre.target        loaded  inactive dead    Local Encrypted Volumes
(Pre)
cryptsetup.target             loaded  active   active  Local Encrypted Volumes
...output omitted...
```

Set default target

List all targets

```
# systemctl -at target
```

List all services

```
# systemctl -at service
```

Select a Target at Runtime

```
# systemctl isolate multi-user.target
```

Set target

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

Verify default target

```
# systemctl get-default
```

```
# ls -l /etc/systemd/system/default.target
```

Not all targets can be isolated

You can isolate graphical.target:

```
user@host:~$ systemctl cat graphical.target
# /usr/lib/systemd/system/graphical.target
...output omitted...
[Unit]
Description=Graphical Interface
Documentation=man:systemd.special(7)
Requires=multi-user.target
Wants=display-manager.service
Conflicts=rescue.service rescue.target
After=multi-user.target rescue.service rescue.target display-manager.service
AllowIsolate=yes
```

But you cannot isolate the cryptsetup target:

```
user@host:~$ systemctl cat cryptsetup.target
# /usr/lib/systemd/system/cryptsetup.target
...output omitted...
[Unit]
Description=Local Encrypted Volumes
Documentation=man:systemd.special(7)
```

Using Emergency and Rescue Targets

- Rescue : All important file systems is mounted. Services in `sysinit.target` are started
- Emergency : Only root file system mounted with read-only; immediately stops all others processes

Enter into rescue target immediately

```
# systemctl isolate rescue.target
```

```
Or # systemctl rescue
```

If rescue target failed to repair then use emergency target

```
# systemctl emergency
```

Quickly back to default target

```
# systemctl default
```

Enter into rescue mode on next and every reboot

```
# systemctl set-default rescue
```

```
# reboot
```

Select a Different Target at Boot Time

1. Boot or reboot system.
2. In GRUB2 menu, interrupt countdown by pressing any key.
3. Move cursor to desired kernel version **entry**.
4. Press **E** to edit.
5. Move cursor to line starts with **linux**.
6. Append **systemd.unit=emergency.target**.
7. Press **Ctrl + X** to boot with changes.

Poweroff, Reboot and Halt

Stop all running services, unmount all file systems and then powers down the system.

`systemctl poweroff`
Or # `poweroff`

Stop all running services, unmount all file systems and then restarts the system

`systemctl reboot`
Or # `reboot`

Similar to poweroff except do not power off system. (in some hardware, it bring down the VM to firmware prompt)

`systemctl halt`
Or # `halt`

Repair Damaged File Systems at Boot Time

12.5

Repairing File System Issues at Boot

- Diagnosing and Fixing File System Issues
 - errors in `/etc/fstab` will prevent system from booting
 - Systemd drops to emergency repair shell (require root password)

Common File System Issues

Problem	Result
Corrupt file system	systemd attempts to repair the file system. If the problem is too severe for an automatic fix, the system drops the user to an emergency shell.
Nonexistent device or UUID referenced in <code>/etc/fstab</code>	systemd waits for a set amount of time, waiting for the device to become available. If the device does not become available, the system drops the user to an emergency shell after the timeout.
Nonexistent mount point in <code>/etc/fstab</code>	The system drops the user to an emergency shell.
Incorrect mount option specified in <code>/etc/fstab</code>	The system drops the user to an emergency shell.

- After edit `/etc/fstab`, run `systemctl daemon-reload`

Example: systemd failed to mount /dev/sda2 and timed out

```
...output omitted...
[*      ] A start job is running for /dev/sda2 (27s / 1min 30s)
[ TIME ] Timed out waiting for device /dev/sda2.
[DEPEND] Dependency failed for /mnt/mountfolder
[DEPEND] Dependency failed for Local File Systems.
[DEPEND] Dependency failed for Mark need to relabel after reboot.
...output omitted...
[ OK    ] Started Emergency Shell.
[ OK    ] Reached target Emergency Mode.
...output omitted...
Give root password for maintenance
(or press Control-D to continue):
```

Checking and Repairing File-system Corruption

- In case of minor issues, RHEL 10 automatically repair file system. Example:
 - XFS or EXT4 file system uses journal to replay changes or undo corruption.
- Alternatively, manually
 - Unmount and remount file system to replays log files or undo corruptions.
- In case of major issues, RHEL enter emergency / rescue target automatically.
- Use respective file system type tools to scan and repair issues

Scan file system for any physical issues.

To check and repair issues with XFS file systems

```
# xfs_repair /dev/sda1
```

To check and repair issues with EXT4 file systems

```
# fsck.ext4 -p /dev/sdb2
```

Although there is fsck.xfs binary.

- Only purpose is to be used during boot times

Other procedure

In rescue/emergency mode root typically mounted read only mode.

```
# mount -o remount,rw /
```

Mount all file system listed in /etc/fstab

```
# mount --all or mount -a
```

In event mount point not found:

```
root@host:~# mount --all
mount: /mnt/mountfolder: mount point does not exist.
```

- Fix issue by either create /mnt/mountfolder or modify /etc/fstab with correct mountpoint

Using nofail option in /etc/fstab

```
/dev/sda1 /mnt/mountfolder xfs rw,nofail 0 0
```

Guided Exercise

Topic	Section Number	Time (min)
Manage the Boot Loader and Kernel Command Line	12.2	10
Explore the Boot Process and Select a Boot Target	12.4	10
Repair a Damaged File System at Boot Time	12.6	10
Lab: Control and Troubleshoot the Boot Process	12.7	25



Unit summary

Having completed this unit, you should be able to:

- control which services start at boot, and how to diagnose and repair file-system issues during boot time.
- Understand RHEL boot process
- Select a Boot Target
- Troubleshoot file system related issues.