1. Please explain the Linux boot process step by step, from powering on the machine until the kernel is running.

1. The workstation firmware starts, performing a quick check of the hardware, called a Power-On Self Test (POST), and then it looks for a bootloader program to run from a bootable device.
2. The bootloader runs and determines what Linux kernel program to load.
3. The kernel program loads into memory and starts the necessary background programs required for the system to operate (such as a graphical desktop manager for desktops or web and database servers for servers). (page 4*)

2. What is the difference between BIOS with MBR and UEFI with GPT in the boot process?

To get things started, the BIOS must know where to find the bootloader program on an installed storage device. Most BIOS setups allow you to load the bootloader program from several locations:

- An internal hard drive
- An external hard drive
- A CD/DVD drive
- A USB memory stick
- A network server

When booting from a hard drive, you must designate which hard drive, and which parti- tion on the hard drive, the BIOS should load the bootloader program from. This is done by defining a Master Boot Record (MBR).

The MBR is the first sector on the first hard drive partition on the system. There is only one MBR for the computer system. The BIOS looks for the MBR and reads the program stored there into memory. Since the bootloader program must fit in one sector, it must be very small—so it can’t do much. The bootloader program mainly points to the location of the actual operating system kernel file, which is stored in a boot sector of a separate parti- tion on the system. There are no size limitations on the kernel boot file.

In recent years this method has run into severe difficulties for various reasons:

- It uses 32 bit addressing, limiting it to disks no larger than 2TB.
- In fact it can use 2 different addressing methods, which can lead to inconsistencies.

- It's limited to a maximum of 4 partitions, although a somewhat messy kludge allows one of those "primary partitions" to be subdivided into an arbitrary number of "logical partitions".
- There is only a single copy of the partition table, so if it becomes unreadable or corrupt then all is lost.
- Partition types are defined by a single byte, which has resulted in collisions.
- There is no formal specification, resulting in different sometimes incompatible implementations between different vendors.

The UEFI specification includes a formal definition of new partition definition known as GPT (GUID Partition Table) which addresses all these problems. (page 7)

https://wiki.restarters.net/UEFI_and_GPT

3. What is GRUB (or GRUB2), and what role does it play in loading the operating system?

The GRUB Legacy bootloader was designed to simplify the process of creating boot menus and passing options to kernels. GRUB Legacy allows you to select multiple ker- nels and/or operating systems using a menu interface as well as an interactive shell. You configure the menu interface to provide options for each kernel or operating system you wish to boot. The interactive shell provides a way for you to customize boot commands on the fly.

Both the menu and the interactive shell utilize a set of commands that control features of the bootloader. (page 9)

4. How does the boot loader know where the kernel is located?

You can use many boot definition settings to customize how the bootloader finds the operating system kernel file. Fortunately, just a few commands are required to define the operating system. The ones to remember are as follows:

- title—The first line for each boot definition section, this is what appears in the boot menu.
- root—Defines the disk and partition where the GRUB /boot folder partition is locatedon the system.
- kernel—Defines the kernel image file stored in the /boot folder to load.
- initrd—Defines the initial RAM disk file, which contains drivers necessary for the kernel to interact with the system hardware.
- rootnoverify—Defines non-Linux boot partitions, such as Windows.

(page 10)

5. What is the purpose of the initramfs/initrd file in Linux booting?

initrd defines the initial RAM disk file, which contains drivers necessary for the kernel to interact with the system hardware.

(page 10)

6. How does UFI handle boot entries differently compared to BIOS?

The UEFI firmware utilizes a built-in mini bootloader (sometimes referred to as a boot manager), which allows you to configure just which bootloader program file to launch.

(page 8)

7. What are the limitations of MBR compared to GPT in terms of disk size and partitioning?

- It uses 32 bit addressing, limiting it to disks no larger than 2TB.
- In fact it can use 2 different addressing methods, which can lead to inconsistencies.
- It's limited to a maximum of 4 partitions, although a somewhat messy kludge allows one of those "primary partitions" to be subdivided into an arbitrary number of "logical partitions".
- There is only a single copy of the partition table, so if it becomes unreadable or corrupt then all is lost.
- Partition types are defined by a single byte, which has resulted in collisions.

https://wiki.restarters.net/UEFI_and_GPT

8. If a Linux system drops you into the GRUB rescue shell, how would you troubleshoot and boot the system manually?

In the case of kernel failure:

The GRUB menu allows you to start the system in single-user mode by adding the single command to the linux line in the boot menu commands. To get there, press the e key on the boot option in the GRUB boot menu.

When you add the single command, the system will boot into runlevel 1, which creates a single login for the root user account. Once you log in as the root user account, you can modify the appropriate modules, init scripts, or GRUB boot menu options necessary to get your system started correctly.

(page 28)

9. Please explain the role of /boot and what critical files it contains.

Holds configuration files used during the system boot process.

The kernel binary file must be accessible by the bootloader program, which is what loads the kernel in memory at boot time. Because of that, the kernel binary files are normally stored under the /boot directory structure in the filesystem, although some Linux distributions keep the kernel binary files directly under the root directory (/).

(page 58, 104)

10. What is systemd (or init), and at what stage in the boot process does it start?

A Linux system comprises many programs running in background to provide services for the system. The init program starts all of those programs when the Linux system starts up. This is called the initialization process.

Currently three popular initialization process methods are used in Linux distributions:

- Unix System V (also called SysV)
- systemd
- Upstart

The systemd program was developed by the Red Hat Linux group to handle starting and stopping programs in dynamic Linux environments. Instead of runlevels, it uses targets and units to control what applications run at any time on the system. It uses separate configura- tion files that determine this behavior.

(page 16, 17)

11. How can you configure Linux to boot into a specific kernel version by default?

By creating multiple kernel entries in the GRUB boot menu, you can select which kernel version to boot. If the new kernel fails to boot properly, you can reboot and select the older kernel version.

(page 28)

12. How can you reduce GRUB timeout or skip the boot menu to boot directly into the OS?

we should edit grub config:

timeout: Specifies the amount of time to wait for a menu selection before using the default

If your Linux distribution uses the GRUB2 bootloader, after you install the new kernel binary into the boot menu, you just need to run the update-grub command

(page 10, 119)

13. What are the risks and benefits of disabling secure boot on a UEFI system?

benefit: removes additional layer of complexity to the Linux boot process
risk: can leave your Linux system vulnerable to attack, and not all systems allow you to disable secure boot

(page 15)

14. How would you recover a Linux machine that fails to boot because of a corrupted boot loader?

Many Linux distributions provide what's called a rescue disk to be used when fatal disk errors occur. The rescue disk usually boots either from the CD/DVD drive or as a USB stick, and it loads a small Linux system into memory. Since the Linux system runs entirely in memory, it can leave all of the workstation hard drives free for examination and repair.

(page 29)

15. Please explain how chainloading works in GRUB when you want to boot another OS.

The bootloader program isn't required to point directly to an operating system kernel file—it can point to any type of program, including another bootloader program. You can create a primary bootloader program that points to a secondary bootloader program, which provides options to load multiple operating systems. This process is called chainloading.

(page 7)

16. Please explain what runlevels were in traditional SysVinit and how they map to systemd targets in modern Linux systems.

The original Linux init program was based on the Unix System V init program, and it became commonly called SysV (or sometimes SysV-init). The SysV init program uses a series of shell scripts divided into separate runlevels in order to determine which programs run at what times.

To make the transition from SysV to systemd smoother, there are targets that mimic the standard 0 through 6 SysV runlevels, called runlevel0.target through runlevel6.target.

(page 16, 21)

17. What is the default target in system, and how can you check and change it?

The default target used when the Linux system boots is defined in the /etc/systemd/ system folder as the file default.target. This is the file the systemd program looks for when it starts up. This file is normally set as a link to a standard target file in the /lib/ systemd/system folder:

```
# ls -al default.target
lrwxrwxrwx. 1 root root 36 Oct 1 09:14 default.target ->

/lib/systemd/system/graphical.target
```

On this CentOS system, the default target is set to the graphical.target unit.

In the systemd method, you use the systemctl program to control services and targets.

`systemctl default ...`

(page 24)

18. Please compare graphical.target vs multi-user.target - what do they represent, and when would you use each?

- multi-user for text mode (use for desktop)
- graphical for graphical mode (use for server)

(page 17)

19. How can you boot a Linux system temporarily into a different target (for example, rescue mode or emergency mode?

To change the target that is currently running, you must use the isolate command. For example, to enter single-user mode, you'd use the following:

```
systemctl isolate rescue.target
```

(page 25)

20. Please explain what a system unit is. What are the main types of units (service, socket, device, mount, target, etc.)?

Instead of using shell scripts and runlevels, the systemd method uses units and targets. A unit defines a service or action on the system. It consists of a name, a type, and a configura- tion file. There are currently eight different types of systemd units:

- automount
- device
- mount
- path
- service
- snapshot
- socket
- target

The systemd program identifies units by their name and type using the format name.type.

(page 21, 22)

21. What is the difference between a system service unit and a target unit?

The systemd method uses service type units to manage the daemons on the Linux system. The target type units are important in that they group multiple units together so that they can be started at the same time. For example, the network.target unit groups all of the units required to start the network interfaces for the system.

(page 22)

22. How can you enable a service to start automatically on boot using systemd?

```
systemctl enable <service-name>
```

(page 25)

23. Please explain how dependencies between units are defined. What is the difference between Requires=, Wants=, and After= in a system service file?

The target configuration defines what targets should be loaded first (the After line), what targets are required for this target to start (the Requires line), what targets conflict with this target (the Conflicts line), and what targets or services the target requires to be running (the Wants line).

(page 24)

24. Where are systemd unit configuration files stored by default? How can you override a unit's default configuration without modifying the original file?

Packages ship unit files typically in `/lib/systemd/system/`. These are *not* to be edited. Instead, `systemd` allows you to override these files by creating appropriate files in `/etc/systemd/system/`.

For a given service `foo`, the package would provide `/lib/systemd/system/foo.service`. You can check its status using `systemctl status foo`, or view its logs using `journalctl -u foo`. To override something in the definition of `foo`, do:

```
1   sudo systemctl edit foo
```

This creates a directory in `/etc/systemd/system` named after the unit, and an `override.conf` file in that directory (`/etc/systemd/system/foo.service.d/override.conf`). You can add or override settings using this file (or other `.conf` files in `/etc/systemd/system/foo.service.d/`). This is also applicable to non-service units - you could do `systemctl edit foo.mount`, `systemctl edit foo.timer`, etc. It assumes `.service` as the default type if you didn't specify one.

https://askubuntu.com/a/659268

25. How would you troubleshoot a service that fails to start during boot? Which commands and logs would you use?

`systemctl status <service-name>`
or
`journalctl`

(page 28)

26. Please explain the difference between rescue.target and emergency.target in systemd.

Rescue mode provides a convenient single-user environment and allows you to repair your system in situations when it is unable to complete a regular booting process. In rescue mode, the system attempts to mount all local file systems and start some important system services, but it does not activate network interfaces or allow more users to be logged into the system at the same time.

https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/7/html/system_administrators_guide/chap-managing_services_with_systemd#sect-Managing_Services_with_systemd-

Targets-Change_Current

Emergency mode provides the most minimal environment possible and allows you to repair your system even in situations when the system is unable to enter rescue mode. In emergency mode, the system mounts the root file system only for reading, does not attempt to mount any other local file systems, does not activate network interfaces, and only starts a few essential services.

https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/7/html/system_administrators_guide/chap-managing_services_with_systemd#sect-Managing_Services_with_systemd-Targets-Emergency

27. How do you isolate a target so the system only runs that target and stops others?

`systemctl isolate target`

(page 24)

28. What is the role of default.target symlink in system, and how does it replace the old /etc /inittab runlevel configuration? If you need to create a custom service (for example, start a script at boot), what is the structure of the . service file, and how would you configure it?

The default target used when the Linux system boots is defined in the /etc/systemd/ system folder as the file default.target. This is the file the systemd program looks for when it starts up.

(page 24)

1. `cd /etc/systemd/system`
2. Create a file named your-service.service and include the following
3.

```
[Unit]
Description=<description about this service>

[Service]
User=<user e.g. root>
WorkingDirectory=<directory_of_script e.g. /root>
ExecStart=<script which needs to be executed>
# optional items below
Restart=always
RestartSec=3
```

```
11
12  [Install]
13  WantedBy=multi-user.target
```

4. Start your service: `sudo systemctl start your-service.service`

https://www.shubhamdipt.com/blog/how-to-create-a-systemd-service-in-linux/

`*`: LPIC2 study guide book