**Linux Boot Process**

1. The POST (Power On Self Test) initializes and performs hardware checks.

2. When the POST finishes, the system control is passed to the first stage boot loader, which is stored on either the boot sector of one of the hard disks (for older systems using BIOS and MBR), or a dedicated (U)EFI partition.

3. The first stage boot loader then loads the second stage boot loader, most usually GRUB (GRand Unified Boot Loader), which resides inside /boot, which in turn loads the kernel and the initial RAM–based file system (also known as initramfs, which contains programs and binary files that perform the necessary actions needed to ultimately mount the actual root filesystem).

4. We are presented with a splash screen that allows us to choose an operating system and kernel to boot.

5. The kernel sets up the hardware attached to the system and once the root filesystem has been mounted, launches process with PID 1, which in turn will initialize other processes and present us with a login prompt.

**An Introduction to SystemD**

1. Systemd allows more processing to be done in parallel during system startup (as opposed to older SysVinit, which always tends to be slower because it starts processes one by one, checks if one depends on another, and then waits for daemons to launch so more services can start), and

2. It works as a dynamic resource management in a running system. Thus, services are started when needed (to avoid consuming system resources if they are not being used) instead of being launched without a valid reason during boot.

3. Backwards compatibility with SysVinit scripts.

Systemd is controlled by the systemctl utility. If you come from a SysVinit background, chances are you will be familiar with:

the service tool, which -in those older systems- was used to manage SysVinit scripts, and

the chkconfig utility, which served the purpose of updating and querying runlevel information for system services.

shutdown, which you must have used several times to either restart or halt a running system.

Systemd also introduced the concepts of units (which can be either a service, a mount point, a device, or a network socket) and targets (which is how systemd manages to start several related process at the same time, and can be considered -though not equal- as the equivalent of runlevels in SysVinit-based systems.

**1. Adjust the execution priority as far as the use of system resources is concerned of a process:**

**Renice =** alters the scheduling priority of one or more running processes. In simple terms, the scheduling priority is a feature that allows the kernel (present in versions => 2.6) to allocate system resources as per the assigned execution priority (aka niceness, in a range from -20 through 19) of a given process.

2. Kill (or interrupt the normal execution) of a process as needed:

In more precise terms, killing a process entitles sending it a signal to either finish its execution gracefully (SIGTERM=15) or immediately (SIGKILL=9) through the kill or pkill commands.

The difference between these two tools is that the former is used to terminate a specific process or a process group altogether, while the latter allows you to do the same based on name and other attributes.

In addition, pkill comes bundled with pgrep, which shows you the PIDs that will be affected should pkill be used.