2 : Installing an Operating System

IT5406 - Systems and Network Administration

Level III - Semester 5

Overview

At the end of this lesson, you will be able to;

- Define boot process and boot loaders
- Describe system management daemons
- Compare boot loaders

Overview

2.1 The Boot Process and Boot Loaders

2.2 The Grand Unified Boot Loader

2.3 System Management Daemons

2.4 Reboot and Shutdown Procedure

2.5 Stratagems for a non booting System

2.6 Drivers and the Kernel

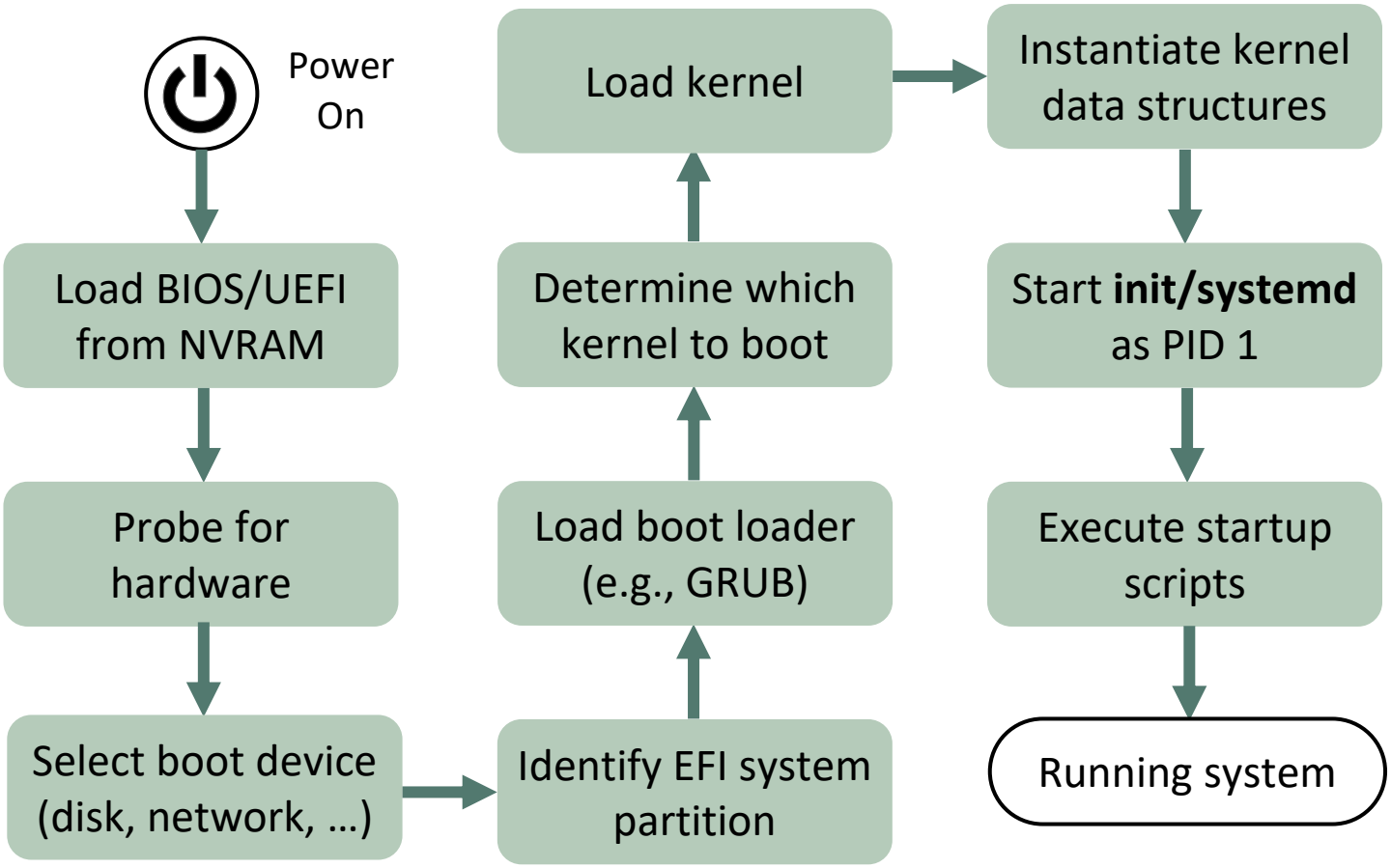
2.1 The Boot Process and Boot Loaders

- Linux distributions use a system manager daemon called *systemd* to streamlines the boot process by adding dependency management, support for concurrent startup processes, and a comprehensive approach to logging, among other features.
- During bootstrapping, the kernel is loaded into memory and begins to execute.
- Administrators can modify bootstrap configurations by editing config files for the system startup scripts or by changing the arguments the boot loader passes to the kernel.

Ref 1: Pg. (31)

2.1 The Boot Process and Boot Loaders...(2)

- Linux boot process



Ref 1: Pg. (31)

2.1 The Boot Process and Boot Loaders…(3)

- BIOS vs. UEFI
  - Basic Input/Output System (BIOS)
  - Unified Extensible Firmware Interface (UEFI)
  - Traditional BIOS assumes that the boot device starts with a record called the MBR (Master Boot Record).
  - The UEFI specification includes a modern disk partitioning scheme known as GPT (GUID (Globally Unique Identifier) Partition Table)

Ref 1: Pg. (33-34)

2.1 The Boot Process and Boot Loaders…(4)

- Boot loaders
  - Identify and load an operating system kernel.
  - Marshalling of configuration arguments for the kernel.
  - Eg:
    - GRUB
    - Windows Boot Manager (BOOTMGR)
    - LILO (Linux Loader)

Ref 1: Pg. (35)

2.2 The GRand Unified Boot loader

- Developed by GNU project
- Default boot loader of most of the Linux operating systems.
- Config file is called *grub.cfg*, and it's kept in */boot/grub* (or */boot/grub2*)
- Configuration is specified in */etc/default/grub*
- After editing */etc/default/grub*, run *update-grub* or *grub2-mkconfig* to translate your configuration into a proper *grub.cfg* file.

Read more: https://www.gnu.org/software/grub/manual/grub/grub.html

Ref 1: Pg. (35)

2.2 The GRand Unified Boot loader...(2)

- Common GRUB configuration options

Shell variable name	Contents or function
GRUB_BACKGROUND	Background image
GRUB_CMBLINE_LINUX	Kernel parameters to add to menu entries for Linux
GRUB_DEFAULT	Number or title of the default menu entry
GRUB_DISABLE_RECOVERY	Prevents the generation of recovery mode entries
GRUB_PRELOAD_MODULES	List of GRUB modules to be loaded as early as possible
GRUB_TIMEOUT	Seconds to display the boot menu before autoboot

Ref 1: Pg. (35)

2.2 The GRand Unified Boot loader...(3)

- GRUB Command Line
  - GRUB supports a command-line interface for editing config file entries on the fly at boot time.
  - GRUB Commands

Command	Function
boot	Boots the system from the specified kernel image
help	Gets interactive help for a command
linux	Loads a Linux kernel
reboot	Reboots the system
search	Searches devices by file, filesystem label, or UUID
usb	Tests USB support

Ref 1: Pg. (38)

2.3 System Management Daemons

- Once the kernel is loaded and completes it initialisation process, it starts some processes autonomously in the user space.
- Most of these processes are part of the kernel implementation.
- They are not configurable, and they don't require administrative attention.
- They have low process ids (PID) [run ***ps*** command in the terminal to see the PID]
- These daemons are background processes.
- Init is the system process which has PID 1 and it make sure the system runs the right complement of services and daemons at any given time.

Ref 1: Pg. (38)

2.3 System Management Daemons…(2)

- Responsibilities of init
  - Setting the name of the computer
  - Setting the time zone
  - Checking disks with fsck
  - Mounting filesystems
  - Removing old files from the /tmp directory
  - Configuring network interfaces
  - Configuring packet filters
  - Starting up other daemons and network services

- Init just run the scripts and commands that have been designed for execution in particular context.

Ref 1: Pg. (42)

2.3 System Management Daemons…(3)

- Migration of tradition *init* to *systemd*
  - *systemd* takes all the *init* features implemented and formalised them.
  - *systemd* manages processes in parallel, network connections (*networkd*), kernel log entries (*journald*), and logins (*logind*).
  - *systemd* is a collection of programs, daemons, libraries, technologies, and kernel components.
  - ***systemctl*** is an all-purpose command for investigating the status of *systemd* and making changes to its configuration.

Ref 1: Pg. (42-46)

2.4 Reboot and Shutdown Procedure

- Shutting down physical systems
  - ***halt*** command performs the essential duties required for shutting down the system. Halt,
    - logs the shutdown
    - kills nonessential processes
    - flushes cached filesystem blocks to disk
    - halts the kernel
  - ***reboot*** is essentially identical to halt, but it causes the machine to reboot instead of halting.
  - The shutdown command is a layer over halt and reboot that provides for scheduled shutdowns and warnings to logged-in users.

Ref 1: Pg. (59)

2.5 Stratagems for a non booting System

- Number of problems can prevent a system from booting,
- Following are three basic approaches to overcome the problem,
  - Don’t debug; just restore the system to a known-good state.
  - Bring the system up just enough to run a shell, and debug interactively.
  - Boot a separate system image, mount the sick system’s filesystems, and investigate from there.

Ref 1: Pg. (60)

2.6 Drivers and the Kernel

- The kernel hides the complexity of the system's hardware underneath.
- It provide an API for application programmers.
- This well defined interface provide useful functionalities such as,
  - Management and abstraction of hardware devices
  - Processes and threads (and ways to communicate among them)
  - Management of memory(virtual memory and memory space protection)
  - I/O facilities (filesystems, network interfaces, serial interfaces, etc.)
  - Housekeeping functions (startup, shutdown, timers, multitasking, etc.)

Ref 1: Pg. (325)

2.6 Drivers and the Kernel...(2)

- Drivers
  - A device driver is an abstraction layer that manages the system's interaction with a particular type of hardware so that the rest of the kernel doesn't need to know its specifics.
  - The driver translates between the hardware commands understood by the device and a programming interface defined (and used) by the kernel.

Ref 1: Pg. (328)