# SESSION 2 – BOOTING AND SYSTEMS MANAGEMENT

**Course Writer: Dr. Ed. Danso Ansong**, Dept of Computer Sc.
Contact Information: edansong@ug.edu.gh


UNIVERSITY OF GHANA
College of Education
**School of Continuing and Distance Education**

- This section introduces students to the whole process of Booting and its associated Systems Management till shutdown.

The key topics to be covered in the session are as follows:

- Self Diagnostic Tests (Power On Self-Test, or POST)
- Role of PROM in
- Bootloader
- Hard disk boot sequence
- kernels and init process
- Basic Input Output Software (BIOS)
- Startup Scripts, System Processes and
- Booting Configuration Management.

- Refer to the following reading material which is available on Sakai

**RECOMMENDED TEXT**

- Unix and Linux Systems Administration Handbook 5th Edition [Pages 30-62], Essential Linux Administration: A Comprehensive Guide for Beginners (Cengage Press) [Pages 127-173].

# Chapter Objectives

At the end of the session, the student will be able to:

- Understand the sequence of events that occur when a system is booted.
- Demonstrate the methods used to modify the boot sequence.
- Configure run-level configuration scripts and daemons.
- Explain the workings of the boot manager and boot loader.
- Express the technical processes of a system from shutdown to power on.

# An Overview Of The Boot Sequence

- When power is applied, the system resets all of the logic gates (chips) to a known state.
- A series of diagnostic tests (Power-On Self- Test, or POST) are run from the PROM Monitor to ensure that some minimal set of hardware components are available for system operation.
- Once the POST operation completes, the system loads a small “boot loader” program from the system PROM.
- The boot-loader loads the system boot block.

# An Overview Of The Boot Sequence

- The system executes the code from the boot block.
  - This program loads the secondary boot loader that will eventually load the operating system kernel and start it.
    - Most secondary boot loaders allow the operator to select the device to boot from, as well as the program that will be loaded into memory and started.

# An Overview Of The Boot Sequence

- The kernel creates a few necessary processes:
  - the init process,
  - a process scheduler,
  - and usually a few memory management processes.
- The init process runs a series of start-up scripts that configure the system for multi-user operation.
- Now that we've see the basic series of events that occur during the system boot process, let's examine some of the internal events that occur during each major step of the boot process.

# Step 1: PROM Monitor

- The PROM monitor includes a small program (Power On Self Test, or POST) that initializes the system hardware, and runs a miniature diagnostic program.
  - The diagnostics ensure a minimally operational base for the OS to run on.
  - POST diagnostic routines do not guarantee that the system hardware is fully functional.
- Once the diagnostic completes, the PROM monitor probes the system busses to determine what hardware is connected to the system.
- Once the hardware has been probed, the PROM monitor loads and starts the second phase of the boot process; the boot block.

# PC PROM Monitors

- PC BIOS
  - Limited alternate boot device selection
    - Typically: Disk, CDROM, Floppy, and net.
    - Sometimes alternate disk boot is supported.
  - Configure/modify hardware settings
    - Enable/disable device, change IRQ/memory ranges.
  - Allows for device BIOS interaction

# Sun PROM Monitor

- Sun PROM Monitor
  - Actually a 2 level monitor which includes “new” and “old” modes. The SUN monitor includes a FORTH language interpreter.
  - Allows the user to select alternate boot devices, enable/disable hardware features, probe devices, test devices.
- Apple Monitor
- RS6000 Monitor
- SGI and HP monitors

# Step 2: Boot Block

- The boot block's job is to initialize some of the system's peripherals and memory, and to load yet another program, the **secondary boot loader**, (sometimes known as the **boot manager**) that will load the OS kernel.
- A boot block is typically placed on the disk as part of the OS installation process. The sysadmin should be familiar with the similarities and differences between the following boot blocks:
  - Windows Boot Block
  - Linux Boot Block
  - Solaris Boot Block
  - IRIX Boot Block
  - MacOS and AIX Boot Block

# Step 3: Secondary Boot Loader

- The secondary boot loader loads the kernel and starts it.
- The secondary boot loader often allows operator intervention:
  - Select alternate kernel to load
  - Modify boot parameters (single user, extensions, …)
    - USE PASSWORD SECURITY HERE!!!
      - Securing the boot process such that users cannot boot alternate media, or an alternate kernel, is a very wise step to take!

# Linux Loader (LILO)

```
# sample /etc/lilo.conf
boot = /dev/hda
        delay = 40
        password=SOME_PASSWORD_HERE
        default=vmlinuz-stable
        vga = normal
        root = /dev/hda1
        image = vmlinuz-2.5.99
              label = net test kernel
        restricted
    image = vmlinuz-stable
            label = stable kernel
      restricted
          other = /dev/hda3
            label = Windows 2000 Professional
            restricted
            table = /dev/hda
```

# GRUB

```
# /etc/grub.conf generated by anaconda
timeout=10
splashimage=(hd0,1)/grub/splash.xpm.gz
password --md5 $1$ÕpîÁÜdþï$J08sMAcfyWW.C3soZpHkh.
title Red Hat Linux (2.4.18-3custom)
        root (hd0,1)
        kernel /vmlinuz-2.4.18-3custom ro root=/dev/hda5
        initrd /initrd-2.4.18-3.img
title Red Hat Linux (2.4.18-3) Emergency kernel (no afs)
        root (hd0,1)
        kernel /vmlinuz-2.4.18-3 ro root=/dev/hda5
        initrd /initrd-2.4.18-3.img
title Windows 2000 Professional
        rootnoverify (hd0,0)
        chainloader +1
```

# The Windows Multi-Boot Loader

```
timeout=10
default=multi(0)disk(0)rdisk(0)partition(2)\WINNT
[operating systems]
multi(0)disk(0)rdisk(0)partition(2)\WINNT="Microsoft Windows 2000
    Professional" /fastdetect
multi(0)disk(0)rdisk(0)partition(1)\WINNT="Windows NT Workstation Version
    4.00"
multi(0)disk(0)rdisk(0)partition(1)\WINNT="Windows NT Workstation Version 4.00
    [VGA mode]" /basevideo /sos
```

# Step 4: The OS Kernel

- Once the kernel is loaded, and started, it probes devices.
  - Some architectures use the PROM monitor to build a device tree that is handed to the kernel during the boot process.
  - Other architectures require the kernel to probe the system busses for devices to build the device tree.
- Once the device tree is available, the kernel parses it, and each device is probed to determine if it is operational, (and if so, the driver module is loaded into the kernel).
  - **This search for memory and devices is sometimes referred to as auto-configuration.**

# Step 4: The OS Kernel

- UNIX Run Levels
  - All flavors of UNIX, and UNIX- work-alikes use similar foundations for the system run modes.
  - As far as UNIX and its ilk are concerned, there are basically two run modes:
    - single user (sometimes called maintenance mode), and
    - multi-user.
      - There may be several forms of the multi-user mode (with services, without services, and so on) on any given UNIX OS.

# Step 4: The OS Kernel

- MacOS Run Levels
  - MacOS also provides for “multiple” run levels.
    - On the older versions of MacOS, different run levels meant simply a choice between “boot normally”, or “boot without extensions”.”
    - Now that MacOS X is a Mach/UNIX based kernel, there are more run levels available.
      - The MacOS X kernel enjoys the same set of run levels, as do the other UNIX variants.

# Windows Run Levels

- Windows has a limited set of run levels
  - Multi-user
  - Safe Mode
  - Safe mode with networking
    - Typically used to repair damaged system.

# Step 4: The OS Kernel

- Start-up Scripts
  - (Un)fortunately, within the OS world there are several “camps” of believers when it comes to how the system should start system services.
  - The following sections provide an overview of the details of the Windows, MacOS, System V UNIX, and BSD start-up sequences, pointing out OS- specific oddities as applicable.

# Step 4: The OS Kernel

- Typically, BSD variants include a **r**un **c**ontrol (also known as an rc) script in the /etc directory.
  - This allows the administrator to edit a single /etc/rc script to make it start a new service at boot time.
- Other BSD variants have simply increased the number of rc files in the /etc directory.
  - For example, the FreeBSD start-up directory contains scripts with names such as /etc/rc.atm, /etc/rc.firewall, and /etc/rc.network.
    - These individual scripts, respectively, configure the ATM network cards, cause the system to become a firewall, and configure the network links respectively.
    - These scripts are called by the master script, /etc/rc

# rc script problems

- Using a small number of rc scripts can cause problems:
  - Startup becomes more difficult to test, as each script starts multiple services.
    - Need to reboot system to test scripts.
  - Not easy to “shut down” individual services using rc scripts.

# System V Scripts

- System V employs a directory full of init scripts. Each System V variant seems to put these scripts in a different directory:
  - Solaris /etc/init.d
  - Linux /etc/rc.d/init.d
  - HPUX /sbin/init.d
- Each script starts/stops one service.
- Symbolic, or hard links connect the /etc/init.d scripts into the correct run level directories (/etc/rcN.d/ where “N” is the run level between 0 and 6).
- As system enters a run level, the scripts in the corresponding run level directory are executed:
  - First the K (Kill) scripts are run in numeric, then alphabetic order.
  - Next, the S (Start) scripts are run in numeric, then alphabetic order.

# System V Scripts

```
grumpy% ls -lsa /etc/rc*.d/*afs*
  2 lrwxrwxrwx   1 root     other   15 May 31  2001 /etc/rc0.d/K66afs -> /etc/init.d/afs
  2 lrwxrwxrwx   1 root     other   15 May 31  2001 /etc/rc2.d/S70afs -> /etc/init.d/afs
```

- Typical init script format:

```
#!/bin/sh
Setup code (set environment variables, subroutine definitions, …)
case $1 in
    start)
      Code to start service
    stop)
      Code to stop service
    *)
      Error handling code and/or usage message
```

# Benefits of init Scripts

- One of the benefits of using init directory scripts is that they are easily tested.
  - The scripts may be manually invoked with the stop and start arguments as a check to determine whether they function correctly before creating the links to the rc directories, and trying them under actual system boot conditions.
  - This procedure is recommended because it can help you catch mistakes that might interrupt the boot process and leave the system unusable.

# Changing Run Levels

- The following commands are typically reserved for system maintenance activities.
  - UNIX shutdown Command
  - The System V init 0 Command
  - The telinit Command
    - NOTE: The shutdown, telinit, and init 0 commands can only be executed with root access.
  - Windows Shutdown Command
  - Legacy MacOS Shutdown Command

# Emergency Measures

```
# STOP-A pressed
Abort at PC OxFOO31870.
>sync
panic: zero
syncing file systems ...2 2 2 2 2 2 2 done.
 586 static and sysmap kernel pages
  34 dynamic kernel data pages
 179 kernet-pageable pages
  0 segkmap kernel pages
  0 segvn kernel pages
 799 total pages (799 chunks)
dumping to vp ff1b9004, offset 181288
117 pages left    109 pages left  101 pages left
5 pages left
799 total pages, dump succeeded
rebooting...
```

# Summary

- The administrator should understand the sequence of events that occur when a system is booted.
- The administrator should understand the methods used to modify the boot sequence.
- The administrator should know how to use the PROM monitor to select alternate boot devices.
- The administrator should understand the workings of the boot manager and boot loader.
- The administrator should understand how to shut a system down properly.

- Unix And Linux System Administration Handbook, 5th Edition By Evi Nemeth, Garth Snyder, Trent R. Hein, Ben Whaley, Dan Mackin. Released September 2017. Publisher(s): Addison-Wesley Professional. ISBN: 9780134278308
- Practice Of System And Network Administration, The: Devops And Other Best Practices For Enterprise IT, Volume 1, By Thomas A. Limoncelli, Strata R. Chalup, Christina J. Hogan. Released November 2016. Publisher(s): Addison-Wesley Professional. ISBN: 9780133415087
- Essential System Administration, Third Edition by Æleen Frisch, Published by O'Reilly Media, Inc. (20( 0-596-00343-9