



Super User / The Boot Process

CIS 68C1

UNIX System Administration

The Super User

- The Super User
 - ✘ AKA: root
 - ✘ Always UID 0
 - ✘ Has complete access to a UNIX system
 - ✘ Can perform operations that are restricted to normal users
 - ✘ Kernel specifically checks if $UID = 0$ for a given operation
 - ✘ Normal users are all non-root users

The Super User

□ Becoming Super User

- ✗ Login as user **root**
- ✗ Use the **su** utility to create a subshell with another UID
 - ✗ **su** is short for switch uers
 - ✗ Use “**su -**” to cause invocation of the users startup scripts
 - ✗ Without the **-**, **PATH** is not setup correctly
 - ✗ Many commands will not be found
 - ✗ Note: using **su** may give you unanticipated results
 - ✗ Many UNIX systems deny remote logins by root
 - ✗ This is the default security setting

The Super User

- Responsibilities of being Super User
 - ✗ Do not leave root shell running unattended
 - ✗ Do not give out root password
 - ✗ Change root password often
 - ✗ Do not create other users with UID 0
 - ✗ Do not use root account for non-admin work
 - ✗ Be extra careful!

The Boot Process

□ The UNIX Boot Process - Six Stages

1. Kernel loaded and initialized
2. Device detection and configuration
3. Kernel creates system process'
 - ✗ Init and other pseudo-process'
4. Operator Intervention
5. Execution of start-up scripts
6. Multi-user mode

The Boot Processes

□ Kernel Initialization

- ✗ The kernel is just a program
 - ✗ Some common names
 - ✗ **/unix, /vmunix, /boot/vmlinuz, /stand/vmunix**
- ✗ Boot loader is loaded from disk by ROM code
 - ✗ Common boot loader for Linux are LILO or Grub
- ✗ Boot loader loads UNIX into RAM, and transfers control to kernel code entry point
 - ✗ Boot loader is not running under UNIX

The Boot Process

□ Kernel Initialization

- ✘ The UNIX kernel is now running
 - ✘ The previous OS is no longer available
- ✘ The kernel then...
 - ✘ Performs RAM tests
 - ✘ Initializes its data structures
 - ✘ Reserves portions of RAM for itself
 - ✘ Sets up virtual memory
 - ✘ Prints initialization messages on the system console

The Boot Process

- **Hardware Configuration**
 - ✗ **Kernel checks out preconfigured hardware devices**
 - ✗ Many devices are specified when UNIX is installed or built
 - ✗ **Kernel probes bus(es) to discover hardware devices**
 - ✗ Allows UNIX to discover devices that were not pre-configured
 - ✗ **Kernel calls upon each device driver to initialize itself**
 - ✗ If successful, kernel registers devices as available to system
 - ✗ If unsuccessful, device will be unavailable for use
 - ✗ **On older UNIX systems, failed, or undiscovered devices could not be made available without reboot**

The Boot Process

□ System Processes

- ✘ Kernel creates primary process called **init**
 - ✘ The **init** process is always PID 1
- ✘ Kernel creates various pseudo-processes
 - ✘ Not really processes – just part of the kernel which appear to look like processes to programs such as **ps**
 - ✘ Linux: **kflushd**, **kupdate**, **kswapd**, **keventd**
 - ✘ BSD: **swapper**, **pagedaemon**
 - ✘ SYSV: **sched** and others
- ✘ Kernel's role in bootstrapping is now done

The Boot Processes

□ Manual Intervention

- ✗ Opportunity to start or work in single-user root shell
 - ✗ Enter root password for single user shell
 - ✗ Enter ^D to continue boot to multi-user
- ✗ Used for system maintenance and configuration
- ✗ In single-user mode
 - ✗ Only root filesystem is available on most UNIXes
 - ✗ Often mounted read-only
 - ✗ Red Hat Linux mounts additional filesystems and is more aggressive
 - ✗ Most daemons are not running
 - ✗ I.e.. no system services, networking, etc.
 - ✗ Filesystem checker **fsck** is run manually
 - ✗ Exiting single-user shell will continue boot to multi-user mode

The Boot Processes

- Startup Script Execution
 - ✗ The remainder of the boot process occurs via shell scripts
 - ✗ These configure the remaining UNIX process' and daemons
- Multi-user Operation
 - ✗ System is now fully operational
 - ✗ Users can login
 - ✗ Network interfaces are up
 - ✗ Daemons are waiting to respond to events
 - ✗ **init** continues to play its vital role

The Boot Process

□ PCs vs. Proprietary Hardware

✗ PCs

- ✗ Use a small OS to boot – the BIOS(es)
 - ✗ Initially controls the keyboard, IDE drives, serial & parallel ports
- ✗ Use Microsoft's IDE hard disk partitioning and boot scheme
 - ✗ Master Boot Record, Partition Boot Record, etc.
- ✗ Older BIOS's had many severe boot limitations and restrictions

✗ Proprietary Hardware

- ✗ Boots directly from boot firmware called PROM
- ✗ Boot firmware is typically fairly intelligent
- ✗ Can boot over a network

The Boot Processes

- LILO – Linux Boot Loader
 - ✗ Installed when Linux is installed in either the:
 - ✗ Master Boot Record
 - ✗ Partition Boot Record of the Linux boot partition
 - ✗ Configured with the **lilo** command
 - ✗ LILO's configuration file: **/etc/lilo.conf**
 - ✗ LILO must be re-run whenever the:
 - ✗ boot process is changed
 - ✗ boot partition changed
 - ✗ kernel is rebuilt

The Boot Process

- LILO – Linux Boot Loader
 - ✗ LILOs job is to load the kernel image and start it running
 - ✗ Kernel image: /boot/vmlinuz
 - ✗ The ‘z’ implies the kernel image is compressed
 - ✗ LILO has a prompt, and accepts commands
 - ✗ Used to specify which Linux image to load
 - ✗ Allows passing configuration parameters to the kernel
 - ✗ Single User Mode
 - ✗ At the LILO prompt, type: **linux single**
 - ✗ Loads the kernel named “linux” in single user mode

The Boot Processes

□ Startup Scripts

✗ The **init** process

- ✗ Is responsible for starting the next set of system processes
- ✗ **/etc/inittab** specifies the programs to start
- ✗ Startup scripts are indicated in **/etc/inittab**
- ✗ Has run-levels from 0 through 6
 - ✗ Used to designate how UNIX should be booted
 - ✗ Indicates which startup scripts in **/etc** should be run
 - ✗ Scripts are **/etc/rc#.d**, where **#** is the run-level
 - ✗ Eg. Scripts in **/etc/rc2.d** are run when going to run-level 2

The Boot Processes

□ Startup Scripts

- ✗ Scripts take a single functional argument:
 - ✗ **start**, **stop**, **restart**, etc.
 - ✗ Tells the script what it should do
- ✗ Scripts names dictate the order of script execution in the `/etc/rc#.d` directories
 - ✗ Scripts names look like:
 - ✗ **S**##*name* or **K**##*name*
 - ✗ **S** and **K** dictate when the script is called – during starting or killing
 - ✗ ## is two-digit number that relative run order of script
 - ✗ *name* is just a descriptive name for readability
- ✗ Scripts in `rc#.d` are symlinks to actual files in `/etc/init.d`

The Boot Processes

□ Startup Scripts

✗ **/etc/sysconfig**

- ✗ Additional scripts & configuration files provided by RedHat
- ✗ Most commonly changed configuration information is here
 - ✗ Networking
 - ✗ **/etc/sysconfig/network**
 - ✗ **/etc/sysconfig/network-scripts/ifcfg-ethX**

The Boot Processes

□ Shut down

✗ Shutdown the system with any of these commands:

✗ **shutdown**

✗ **halt**

✗ **reboot**

✗ **telinit**

✗ Incorrect Ways:

✗ Killing **init**

✗ Turning off power to system

✗ These methods can and eventually will corrupt filesystem(s) and cause data loss