CSE 265: System and Network Administration

- System startup and shutdown
  - Bootstrapping
  - Booting PCs
  - Boot loaders
  - Booting into single user mode
  - Startup scripts
  - Rebooting and shutting down
Bootstrapping

i.e., starting the computer

- System is particularly vulnerable to errors
- Steps in boot process
  - Execution of boot code in ROM
  - Loading and initialization of kernel
  - Device detection and configuration
  - Creation of spontaneous system processes
  - Operator intervention (manual boot only)
  - Execution of system startup scripts
  - Multiuser operation
Kernel initialization

- Kernel is a program
- First real task is to load and run
  - `/boot/vmlinuz`
  - Other examples: `/unix`, `/vmunix`
- Two steps
  - Load small booting program (bootloader)
  - Booting program loads kernel
- Kernel then begins to run
Hardware configuration

• Kernel examines system environment
• Tries to locate and initialize every device that it is supposed to have
• Hardware configuration info in kernel is often underspecified
  – Probes buses for devices and asks drivers for info (i.e., which interrupt, which PCI address, etc.)
• Drivers can sometimes be added later
“Spontaneous” system processes

- Not created via usual fork mechanism
- init is always process 1
- Plus special memory and kernel processes
  - kflushd, kupdate, kpiod, kswapd
  - Not really processes (portions of kernel)
- Everything else (other processes) are started via init

Optionally view such processes on edgar with “ps -aux | more”
Operator intervention (manual booting)

- Kernel tells `init` if single-user mode is desired
- Typically requests root password
- Single-user shell is similar to normal shell
  - Often fewer disk partitions mounted (such as root partition only)
    - Other partitions must be mounted by hand if needed
  - Daemons typically not running
  - Can run `fsck` if needed to repair filesystems
- When you exit, system attempts to boot into multi-user mode
Multiuser operation

• After initialization scripts run, system is fully operational, except that no one can log in

• init spawns
  − getty processes that listen on terminals (including console)
  − graphical login such as xdm or gdm if configured

• init later responsible for moving from one runlevel to the next
Booting PCs

- PC starts by executing code in ROM (the BIOS)
  - Usually BIOS has a configuration mode with special keypress during boot
  - Tries to load first 512B of the boot disk – the Master Boot Record
  - MBR contains program to specify which partition from which to load the secondary boot program (the “boot loader”)
Boot loaders

- Boot loaders load and start the kernel
  - Could be one of many kernels or OSes!
  - Each disk partition can have its own second stage loader
  - MBR set to load the master boot loader
- LILO is the older, traditional Linux boot loader
- GRUB is the modern Linux boot loader
  - Actually, supports most OSes, not just Linux
GNU GRUB version 0.95 (639K lower / 1047296K upper memory)

Solaris Next Build 14
Windows XP

Use the ↑ and ↓ keys to select which entry is highlighted. Press enter to boot the selected OS, 'e' to edit the commands before booting, or 'c' for a command-line.

The highlighted entry will be booted automatically in 7 seconds.
Example multi-boot laptop GRUB

# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes
to this file
# NOTICE: You have a /boot partition. This means that
# all kernel and initrd paths are relative to /boot/, eg.
# root (hd0,2)
# kernel /vmlinuz-version ro root=/dev/hda6
# initrd /initrd-version.img

#boot=/dev/hda
default=1
timeout=10
splashimage=(hd0,2)/grub/splash.xpm.gz
title Red Hat Linux (2.4.20-8)
  root (hd0,2)
  kernel /vmlinuz-2.4.20-8 ro root=LABEL=/
  initrd /initrd-2.4.20-8.img
title Microsoft XP
  rootnoverify (hd0,1)
  chainloader +1
Booting into single user mode

- GRUB is more difficult (graphical interface) than LILO to add options at boot time
- Need to modify a GRUB entry to include the keyword single, e.g.

  ```
  title Red Hat Linux (2.4.9-21) single user mode
  root (hd0,0)
  kernel /vmlinuz-2.4.9-21 ro root=/dev/hda6 single
  initrd /initrd-2.4.9-21.img
  ```

- Can be done at run-time
- Better is to set up a single-user mode entry ahead
Startup scripts

At the end of single user mode, init executes system startup scripts

Typical tasks:

- Setting name of computer
- Setting the time zone
- Checking the disks with fsck
- Mounting the system disks
- Removing old files from /tmp
- Configuring network interfaces
- Starting daemons and network services
init and run levels

- Seven run levels defined
  - 0 is for system shut down
  - 1 or S is single user mode
  - 2-5 are multi-user levels
    - In RH/Fedora, 3 is networked multi-user, 5 is X-windows
  - 6 is reboot level
- /etc/inittab specifies what init has to do in each level
- During booting, system goes from 0 to default run level (in /etc/inittab), and calls /etc/rc.d/rc for each change

Optionally view /etc/inittab on edgar
init and rc scripts

- Startup scripts live in `/etc/init.d/` (linked to `/etc/rc.d/init.d/`)
- Each script starts, stops, restarts some service
- `/etc/rc.d/rc` knows to look in `/etc/rc.d/` where there is a subdirectory for each runlevel
  - Symbolic links are made to the actual script in `/etc/init.d/` within each subdirectory for the services appropriate for that level, e.g.
    - `ln -s /etc/init.d/sshd /etc/rc3.d/S99sshd`
- Script names indicate order of Start or Kill
- Red Hat/Fedora has optional 'interactive' mode
  - Requires confirmation of each service startup
/etc/sysconfig

- Additional scripts and configuration for Red Hat
- Stores networking configuration
  - /etc/sysconfig/network-scripts/ifcfg-ethX
- Scripts can be used to individually start or stop network interfaces
  - /etc/sysconfig/network-scripts/ifdown eth0
  - /etc/sysconfig/network-scripts/ifup eth0
Rebooting and shutting down

- Not needed as often as in consumer OSes
  - Needed for
    - Adding or removing hardware
    - Change to boot configuration
      - Including new kernel
      - System really wedged
  - Ways to reboot or shutdown
    - Use the `shutdown` command
    - Use the `halt` and `reboot` commands
    - Use `telinit` to change init's run level
    - Use `poweroff` to tell system to turn off power
    - Use hardware reset switch or turn off power (last resort!)
Turning off power

- Turning off power can cause data loss and leave filesystem in an inconsistent state
  - Linux (and other modern OS) filesystems buffer changes in memory, and only sporadically write them back to disk
    - Makes disk I/O faster, but more sensitive to loss
- Uninterrupted power is important
- Sometimes is necessary
  - Flood, fire, etc.
shutdown

- shutdown command is safest, considerate, and most thorough to halt, reboot, or change to single user mode
- shutdown can wait before bringing down system
- shutdown sends warning messages (like `wall`) to logged-in users
  - should explain why, and when it is coming back
- Can specify whether to halt, or reboot:
  - `shutdown -h 09:30 'Going down for scheduled maintenance; back by 10am.'`
  - `shutdown -r +15 'Rebooting to unwedge NFS'`
Halt; reboot

- halt
  - called by `shutdown -h`
  - logs the shutdown
  - kills non-essential processes
  - executes `sync`
    - waits for filesystem to finish writes
    - puts IDE drives in standby mode (flushing write caches)
  - halts the kernel

- reboot
  - called by `shutdown -r`
  - similar to halt, but tells kernel to reboot system
**telinit; poweroff**

- **telinit**
  - Directs init to go to a specific run level
    - telinit 1 - takes system to single-user mode

- **poweroff**
  - Identical to halt, but adds request to power management system to turn off system's power