SYSTEM ADMINISTRATION
BOOTING AND SHUTTING DOWN

UNIX Programming 2014 Fall by Euiseong Seo
Bootstrapping

- Starting up a computer
- Boot time is especially vulnerable
  - Misconfiguration, missing or unreliable equipment and damaged filesystems result in boot failure
- Understanding booting procedure is essential for UNIX programmers and administrators
Basic Procedure

* From Inside the Linux boot process, IBM
Steps to Boot

1. Reading of the boot loader from the master boot record
2. Loading and initialization of the kernel
3. Device detection and configuration
4. Creation of kernel process
5. Administrator intervention (single-user mode only)
6. Execution of system startup scripts
When a machine boots, it begins by executing code stored in ROM

- BIOS (Basic Input/Output System)

- BIOS
  - Knows about some of devices on motherboard
  - Conducts POST (Power-on Self Test)
  - Provides services to OS later

- Boot device order is normally user-configurable
  - MBR of a boot device will be loaded by BIOS

- MBR (master boot record)
  - First sector of a boot device (Sec. 1 of Cyl. 0, Head 0)
  - 512 byte long
Boot Loader

- **First 446 bytes**
  - Primary boot loader
  - Executable text
  - Error message text

- **Next 64 bytes**
  - Partition table
  - For up to 4 partitions
  - 16 bytes each

- **Last 2 bytes**
  - Magic 0XAA55

Anatomy of the MBR

*From Inside the Linux boot process, IBM*
Boot Loader

- **Primary boot loader**
  - To find and load the secondary boot loader (stage 2)
  - Looking through partition table to find an active one
  - Secondary boot loader is located in the boot record of the active partition

- **Secondary boot loader (kernel loader)**
  - To load the kernel image and optional *initrd*, an initial RAM disk
  - This style is called chain-loading

- **Kernel image**
  - Compressed kernel + setup code
  - Context will be switched to the setup code after loading
GRUB

- De-facto standard boot loader set
- 3 stage boot loader (MBR->PBR->Filesystem)

- Grub configuration file is located in `/boot/grub`
- Grub loads the configuration file at every boot
  - To allow dynamic changes

- Example `grub.cfg`

```plaintext
default=0
timeout=10 splashimage=(hd0,0)/boot/grub/splash.xpm.gz
title Red Hat Enterprise Linux Server (2.6.18-92.1.10.el5)
root (hd0,0)
kernel /vmlinuz-2.6.18-92.1.10.el5 ro root=LABEL=/
title Windows XP
rootnoverify (hd0,0)
chainloader +1
```
Kernel Initialization

- HW initialization
  - Peripheral device initialization
  - Device drivers for peripherals are stored in initrd

- Kernel parameters
  - Command line arguments to the kernel
  - Boot loader passes them to the kernel during boot up
  - Example
    - /boot/vmlinux root=/dev/sda1 mem=256MB
  - See https://www.kernel.org/doc/Documentation/kernel-parameters.txt for more information
Init and Run Levels

- **Init**
  - The first process that a kernel creates
  - Root or parent process of all processes
  - Traditionally, `/sbin/init`
  - Executes shell scripts corresponding to the run level

- **Run level**
  - System operation mode
    - Safe/Safe with network/Command prompt/Normal in Windows systems
  - `/etc/rc*.d/` contains shell scripts to run at each run level
    - Eg) Shell scripts in `/etc/rc1.d/` will be executed during boot up at run level 1
Init and Run Levels

- Ubuntu run level definition
  - 0: Halt
  - 1: Single-user mode
  - 2: GUI multi-user mode with networking
  - 3-5: Undefined but configured the same as level 2
  - 6: Reboot

- Default run level is defined in `/etc/init/rc-sysinit.conf`

- Check runlevel
  - `runlevel`

- Change runlevel
  - `telinit N`
  - `init N`

- How can we change the run level during boot up?
Creation of Kernel Processes

- Some processes are not forked by init, but by kernel
- Kernel processes
  - In charge of important kernel-internal tasks
  - Identifiable by the brackets around their names in `ps`
  - You can’t kill these processes
- Some common kernel processes
  - kjournald
    - Commits filesystem journal updates to disk
  - kswapd
    - Swaps processes when physical memory is low
Startup Scripts

- Scripts in rc*.d

```
euiseong@MBA-VB:/etc$ ls rc1.d
K20kerneloops  K20speech-dispatcher  K70vboxadd-x11  S70dns-clean
K20rsync  K65vboxadd-service  README  S70pppd-dns
K20saned  K70vboxadd  S30killprocs  S90single
```

```
euiseong@MBA-VB:/etc$ ls rc2.d
README  S30vboxadd  S70dns-clean  S99rc.local
S20kerneloops  S30vboxadd-x11  S70pppd-dns
S20rsync  S35vboxadd-service  S99grub-common
S20speech-dispatcher  S50saned  S99ondemand
```

- Symbolic links to the scripts in /etc/init.d/
- Snn means the service should be started with the priority of nn (the smaller, the faster)
- Knn menas the service should be stopped with the priority of nn
Startup Scripts

- Another example

```
euiseong@accept:/etc/rc2.d$ ls
README      S75sudo   S99grub-common S99rc.local
S20unattended-upgrades S91apache2 S99ondemand
```

- When you want start a service defined in `/etc/init.d` at the current level, just create a symbolic link in the `rc` directory in the form of `SnnServiceName`

- Quiz
  - If you want to make your system back up all temporary files in `/tmp` to a backup file, of which name is `tmp-YYYY-MM-DD`, in `/backup/` after every boot up, how can you accomplish this?
Boot Process Completion

- System is fully functional after executing start up scripts in the `rc` directory
- init process stays and manages system run level
Rebooting and Shutting Down

- When your Windows system gets unresponsive or unstable, what is your first solution?
  - Blind rebooting is less effective in UNIX systems (Why?)

- Rebooting must be done gently not to corrupt filesystems and databases

- `shutdown [OPTION] TIME [message]`
  - The most gentle way to shut down or reboot
  - `-h: halt after TIME`
  - `-p: halt and power down after TIME`
  - `-r: reboot after TIME`
  - `-k: just print out message and disable following logins`