UNIX Administration

Chapter 5

Installing an Operating System

- Hardware Requirements
  - Before you attempt to install an OS on a machine, it would be wise to ensure that the OS in question runs on the hardware that you have available.
- Multi-platform OS Support
  - Most OS vendors that distribute software that runs on multiple hardware architectures publish a hardware compatibility list (HCL) for their OS.
- Single Platform OS Support
  - Vendors of single-architecture OS deal with many of the same problems faced by multi-platform vendors.

Installing an Operating System

- OS for Intel-based Systems
  - Recent releases of Linux are beginning to nip at the heels of Windows in regard to device driver availability, and the number of systems running the OS.
    - The open-source movement has the advantage of allowing anyone to submit a device driver for their favorite device.
  - Because there are so many PC systems available, many people know how to write “bad” software for the PC architecture.
    - This plethora of untested/unsupported software available for the system can lead to security and reliability problems.
    - Both Linux, and Windows suffer because of this

Installing an Operating System

- OS for Non Intel-based Systems
  - Proprietary hardware architectures may not allow you to run more than one OS.
  - Whereas some architectures may only work with one OS, others are blessed with multiple fully supported OS.
  - Other hardware vendors only support their proprietary OS on their hardware.

Types of Installations

- There are several types of installation models available to work with.
  - The installation could be a server or a workstation.
  - The goal of the installation could be to set up a machine usable as a desktop workstation, web server, database server, service machine, or interactive system.
  - The system could be a diskless host that boots from the network, or a diskful system that serves files to other hosts.

Types of Installations

- Installing a Solaris/Linux dual-boot Intel system often presents problems for the administrator.
  - The Solaris Intel installer limits the number of partitions allowed on a disk where Solaris will be installed.
    - **If the disk has more than three partitions, and/or the installation of Solaris would cause the disk to have more than three partitions, the installation will fail.**
  - Linux wants a boot partition, and a kernel partition (minimum).
  - Solaris also wants a boot partition and a kernel partition.
  - If you wanted to build a dual-boot Linux/Solaris system, you would need four partitions. However, Solaris will only allow you to install on a disk with three (or fewer) partitions.

Types of Installations

- Desktop Installations
  - Installing the OS on a desktop PC is often a very different problem than installing an OS on a corporate database server.
  - Networked Client Systems
    - Systems that rely on network-based servers for critical applications/services are typically referred to as networked client systems, or “thin clients.”

Types of Installations

- **Server Installations**
  - You have to install the OS and configure it to provide services to other computers.
  - The types of “clients” the server supports will usually complicate this configuration task.
  - The applications/services provided by the server may provide more complications.
- Homogenous Servers
  - This type of server only provides services to clients of the same architecture/kernel architecture.
  - Such systems may be used as a boot server, file server, name server, web server, database server, or many other purposes.
- Heterogeneous Servers
  - These systems may provide boot services, applications, and/or file services for a variety of systems of different kernel/hardware architectures.
    - **For example, a Linux system may be set up to provide file service to Linux, Solaris, and MacOS boxes via NFS, while providing file service to desktop PCs via Common Internet File Services (CIFS) by running the Samba application.**

Installation Methods

- **UNIX Installations**
  - Many flavors of UNIX allow (in fact, insist on) significant operator interaction during the installation process.
  - UNIX installers are often much more willing to allow custom installations than their Windows counterparts.
- CD/DVD Installations
- Network Installations

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 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 seen the basic series of events that occur during the system boot process, let's examine some of the internal

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.

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
```

Step 4: The OS Kernel

- Once the kernel is loaded, and started, it probes devices.
- 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 alilkes 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.
- 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.

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.

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.

The Superuser, cont

- By default, one account has elevated privileges to issue any command, access any file, and perform every function
- Superuser, a.k.a. root
  - Technically, can change to anything – but don't
- User and group number 0
- Must limit use of root
  - Inexperienced users can cause serious harm
  - Use of root for non-privileged tasks unnecessary and can be open to attack
  - Security and privacy violations – root can look at anyone's files
- Limit what root can do remotely
- Ensure a strong password

Superuser Privileges

- What usually works best is short periods of superuser privilege, only when necessary
- Obtain privileges, complete task, relenquish privileges
- Most common ways are su and sudo
- Can also use the setuid/setgid method (Ch. 4), but not recommended

su

- Short for *substitute* or *switch user*
- Syntax: `su [options] [username]`
  - If `username` is omitted, `root` is assumed
- After issuing command, prompted for that user's password
- A new shell opened with the privileges of that user
- Once done issuing commands, must type exit

sudo

- Allows you to issue a single command as another user
- Syntax:
  `sudo [options] [-u user] command`
- Again, if no user specified, root assumed
- New shell opened with user’s privileges
- Specified command executed
- Shell exited

sudoers

- Must configure a user to run commands as another user when using `sudo`
- Permissions stored in `/etc/sudoers`
- Use utility `visudo` to edit this file (run as root)
- Permissions granted to users or groups, to certain commands or all, and with or without password being required

Runlevels

- A set of defined system states that init can bring the system into (varies on distro)
- 0: Halt/shutdown
- 1: Single user mode
- 2: Multiuser mode
- 3: Multiuser mode with networking
- 4: Not used
- 5: Multiuser mode with networking and GUI
- 6: Reboot

Runlevels, cont

- On boot, init checks /etc/inittab to see what runlevel to bring system to
- To change runlevel after boot
  - telinit *runlevel*
  - shutdown/halt/reboot
- Any time the runlevel changes, init consults a set of scripts to determine what to stop/start

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

Single User Mode

- Runlevel 1
- Console only – no terminals
- Very minimal environment
- Some filesystems might not be mounted
- Maintenance of filesystems
- Fixing configuration errors
- Disaster recovery

Multiuser Mode

- Runlevels 2-5
- Runlevel 2 allows terminal logins
- Runlevel 3 allows remote terminal logins
- Runlevel 5 enable X11 graphical environment
- Runlevels 3 and 5 are the most common levels for day-to-day operations

User Administration

- User configuration stored in `/etc/passwd`
- File got it’s name because it originally contained passwords as well
  - Security problem – too many processes need to read `passwd`
  - A shadow file used now instead (more in a sec)
- Each line contains info for one user

passwd

```
jsmith:x:1001:1001:Joe Smith,Rm27,(234)555-8910,
  (234)555-0044,email:/home/jsmith:/bin/bash
```

- First field is username
- Second was password – now a dummy char
- Third is userid (uid)
- Fourth is groupid (gid)
- Fifth is GECOS field
  - Full name, contact info
  - Gen. Elec. Comprehensive OS
- Sixth is user’s home directory
- Seventh is user’s default shell

passwd, cont

- Originally `passwd` contained a user's password information
- How it works
  - User picks a password
  - A random number is generated (called the salt)
  - The salt and the password is passed into a hash function (a one-way cryptographic algorithm)
  - The salt and result are stored in ASCII

passwd, cont

- Problem – user-level programs need to read `passwd`
  - Get user name, location
  - Home directory, shell
- So `passwd` was world readable
- So anyone on system could see a user's salted hash
- It's encrypted – what's the big deal???

passwd, cont

- Original salt was 12-bit ... 4096 possibilities
- Many early users used bad passwords
  - Dictionary words
- Even with 1970’s computing, it wouldn’t take very long to try all combinations of salts and passwords through the hash function
- Just wait for a match
- Brute force crack

shadow

- Wasn’t acceptable to have `passwd` world readable if it contained hashes
- So salted hashes moved to a new file
- `/etc/shadow`
- Format similar to `passwd`, one user per line
- Readable only by root

shadow, cont

```
jsmith:
  $1$CzzxUSse$bKJL9wAns39vlxQlBZ8wd/:13744:0:99999
  :7:::
```

- First field is username
- Second is the salted hash or account status
  - `NP` or `!` or null for blank password
  - LK or `*` for locked/disabled account
  - !! for account with expired password
- Third is days since last password change
  - Measured from epoch (midnight UTC 1/1/1970)

shadow, cont

- Fourth is days until password is eligible to be changed
- Fifth is days before change is required
- Sixth is days before expiration to warn
- Seventh is days before account expires
- Eighth is days since epoch when account expires
- Ninth is unused/reserved

Adding Users

- If you really wanted to, edit `/etc/passwd` by hand
- Some distributions have graphical or simplified ways to add users
- Most widely available however is command line utility `useradd`

Adding Users, cont

- Syntax:
  `useradd [options] [-g group] [-d home] \`
  `[-s shell] username`
- -g to define user's initial group
- -d to define user's home directory
- -s to define user's default shell
- Other options for expiration, using defaults, etc

Deleting Users

- Again, could just hack /etc/passwd
- More elegant:
- Syntax: `userdel [-r]` *`username`*
- -r to delete home directory and it’s contents

Modifying Users

- Syntax: `usermod [options]` *`username`*
- Options are pretty much identical to those of useradd
- Also, -l to change the user’s login name
- And –G to list additional groups to add user to

Group Management

- Group info housed in `/etc/group`
- Similar to user management
  - groupadd
  - groupdel
  - groupmod