

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 situation!

Installing an Operating System

- OS for Intel-based Systems

- Well behind Windows and Linux, BSD, and Solaris are also available for Intel architecture systems.
 - Solaris suffers from a lack of device drivers for the latest/greatest devices.
 - Solaris also often does not offer device drivers for very old devices.
 - Sun discontinued support for the Solaris Intel edition with the release of Solaris 9. They now offer a Linux “solution” in place of Solaris for Intel.
 - BSD variants are generally more secure than the competitors, and offer very robust/well tested code, but they have not caught on as general purpose offerings. BSD is often found in research labs, and under the hood of network “appliances” such as firewalls.

Installing an Operating System

- OS for Non Intel-based Systems
- Proprietary hardware architectures may not allow you to run more than one OS.
 - The “single offering” case greatly simplifies the decision regarding which OS you should load on the machine!
 - The single OS is also easier for the vendor to support, as the service personnel only have one system to learn.
 - Single-OS systems are also likely to be replaced by multi-OS systems over time.
 - **The world of computing is not a “one-size-fits-all” environment!**
 - **The UNIX vs. Windows war will be battled for many years.**

Installing an Operating System

- OS for Non Intel-based Systems
 - Whereas some architectures may only work with one OS, others are blessed with multiple fully supported OS.
 - Some vendors may provide full support for their proprietary OS but partial support for a second OS.
 - More often than not, the second OS is one of the Linux distributions.
 - Vendors such as Hewlett-Packard, IBM, Compaq and others now offer multiple OS for their hardware.
 - Other vendors (like Sun) offer a line of Intel based Linux systems, as well as the SPARC based systems.

Installing an Operating System

- OS for Non Intel-based Systems
 - Other hardware vendors only support their proprietary OS on their hardware.
 - This is true for Sun Microsystems, even though there are Linux distributions that support the SPARC architecture.
 - Although Linux is available for SPARC systems, many software vendors do not port their applications for the Linux/SPARC market.
 - Without applications available for Linux/SPARC, customers stay with the proprietary Solaris OS on Sparc.

Cataloging Hardware Requirements

- When you consider the OS hardware requirements, you need to pay close attention to any nonstandard hardware on your system.
- All hardware found on the system should be cataloged.
 - Need to determine if device drivers are available.
 - Need to determine if the hardware is mission critical.
- Looking over the inventory form before attempting to install the latest OS version may make the decision for you.

Cataloging Hardware Requirements

- PC systems typically force a few more restraints on the system hardware than other architectures.
 - For example, an Intel system requires a VGA display and keyboard in order to boot.
 - Unfortunately, there are hundreds of display adapters, thousands of keyboards, and tens of thousands of mice available for PCs of different heritage.
- OS designers also impose restraints on the system hardware.
 - Software kernel architectures may impose specific revision-level requirements on the hardware.
 - For example, Solaris 8 will not boot on the sun4c hardware architecture.

Installation Time Requirements

- When it comes to a discussion of installing an OS, the answer to “what hardware do I need?” is almost as nebulous as the answer to “how long will it take?”
- When everything goes right, a full OS install might take an hour. But when things go wrong, you might be looking at eight or more hours to get the OS installed and working.
 - How do you estimate the amount of time required for an OS installation?

Installation Time Requirements

- Never assume the OS installation will go smoothly.
- Do not forget that after the OS is installed you will need to install patches and service packs to bring the OS up to date.
- Once the OS is installed and patched, do not forget that you have to install all of the applications users need in order to get their work done.
- Allow time for the appropriate number of reboots during the OS and application installation phases.
- The “load” media for an install can also have a huge impact on the installation.

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.
- Each of these configurations may require that a particular set of software be installed such that the system performs its duties and users can get some work done.

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.
 - In this case, install Solaris first, and then install Linux on a separate partition later. Better yet, buy a second disk, and install one OS on each disk!

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.
 - Generally, desktop computers come in two flavors: systems that contain their own copies of everything and systems that rely on network-based servers for critical applications.
 - **Standalone Systems**
Self-contained systems are often referred to as standalone systems, or “thick clients.”
 - These machines typically contain the OS, and local copies of all applications required by users of the system.
 - The installation of a standalone system will require more time than some other systems, because you have to load the OS, and all of the applications on the local disk.
 - **Such installations can easily become multi-day tasks!**

Types of Installations

- Desktop Installations
 - 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.”
 - These machines usually contain a copy of the OS, but very little other software gets installed on these systems.
 - User files typically reside on a network-based file server.
 - Applications may reside on yet another network-based server.
 - These systems rely on the network to be operational for the system to be useful.
 - Such systems are typically very easy installations.
 - You load the OS, configure the network connection, and possibly configure a few other parameters to allow the system to locate the network-based servers and you are “done”.

Types of Installations

- Server Installations
 - Installing an OS on a server is often a long, arduous task.
 - 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
 - The homogenous server is probably the simplest server to install.
 - This type of server only provides services to clients of the same architecture/kernel architecture.
 - **This means that only one version of the OS and all applications need be installed on the system.**
 - Such systems may be used as a boot server, file server, name server, web server, database server, or many other purposes.

Types of Installations

- Heterogeneous Servers
 - Heterogeneous servers are probably the most difficult system you will ever have to install.
 - 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.**
 - **Such servers are typically very complicated beasts to install and configure.**
 - **You will have to install copies of multiple OS for the system to function as a boot server.**
 - **Similarly, you will have to install application binaries for multiple architectures in order to support application services for client machines.**

Types of Installations

- Planning for an Installation

- The “footprint” or size of the OS should be considered to ensure that the system contains enough disk space.
- How that disk space is parceled might play a role in the OS installation.
- Similarly the size of the main memory might need to be taken into consideration.

- **Disk Space Requirements**

One of the most important decisions you will need to make before you install an OS is how much space to allocate to the system software.

- If you allocate too much space to the system software, users may not have enough space. If you allocate too much space to users, the system may run out of space.
 - Calculate how much space the OS and application binaries will occupy.
 - Once you have a number in mind, double it. In a few weeks or months you will be glad you did.

Types of Installations

- Every successive release of an OS is larger than its predecessor.
 - This is an important point to remember, because you may have to upgrade the OS on a machine two or three times over the lifetime of the hardware.

Installation Methods

- Current OS are typically distributed on CD or DVD media.
- Older releases were distributed on tape cartridges or floppy diskettes.
- More often than not, the distribution media is bootable, and therefore all you have to do is place the media in the appropriate device and turn on the power.
- The magic of the boot process boots the installation media, and an installation program guides you through the installation process.

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.
 - This generally infers that the operator needs to be more knowledgeable about the specifics of the system to successfully complete the installation process.
 - It also means that unattended installations are not feasible without plenty of advance configuration and planning.

Installation Methods

- CD/DVD Installations
 - As with Windows distribution media based installations, the installers used by UNIX OS are somewhat automated.
 - A difference between UNIX and Windows installers is that MOST UNIX installers ask all of the questions up-front, then use the answers to drive the remainder of the install.
 - **A typical Solaris 8 installation requires about 20 minutes of operator interaction, then for the next hour (or more) no interaction is required.**
 - **RedHat Linux installations are similar to Solaris in regards to operator interaction.**
 - While MOST UNIX installations often take care of the interactive portion up-front, a few of the installers “hold the user’s hand” throughout the installation process (much like Windows).

Installation Methods

- Network Installations
 - Most versions of UNIX support a network-based installation system of one form or another.
 - Like Windows, these installers require a network-based boot server, rules files that dictate how the installation is performed, and a boot daemon that runs on the server to manage the process.
 - **The Solaris JumpStart package is one such network-based installer.**
 - **Sun's WebStart and the Linux KickStart service are other examples of the automated network-based installer.**
 - **Because there is no official standard for these network-based tools, and each vendor has one (or more) of these installers, describing all of the current offerings is difficult, if not impossible.**

Installation Methods

- Solaris Network Boot Service Daemon
 - To provide a Solaris installation server, you must build and configure a system that will listen for install requests, and know how to deliver the proper files to the host being installed.
 - This requires the sysadmin to complete the following two major tasks.
 - Build a server, install the boot server software, and configuration files.
 - Install the binaries to be served on the boot server.

Installation Methods

- Saving Critical Data

In the event your installation involves the upgrade from an old version of an OS to a new version, there is another critical point to consider.

- What do you do with all of the old files on the system when you get ready to install the new software?
 - If the system is a standalone desktop, and you have a new OS (as well as new applications to install), you may not need to worry about saving anything from the old system.
 - More often than not, however, before an old OS is shut down for the last time there are several files you may wish to save.
- **You are strongly advised to make a full backup of every file on the system.**

Installation Methods

- Servers often present an even bigger challenge when you are upgrading an OS.
 - For instance, because you may not have to change the partitioning to load a new OS, it would help to know how the disks on the system are currently partitioned.
 - Printing out information regarding current disk partitions may be very helpful information during the upgrade.
 - Saving the password file, NIS name, server maps, user files, and other critical information before you begin the installation procedure is always strongly recommended.
 - Again, err on the side of caution, and perform a full file system backup before you begin the upgrade. Nine out of ten times you will be glad you spent the extra time to do so.

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 seen 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=vmlinux-stable

vga = normal

root = /dev/hda1

image = vmlinux-2.5.99

label = net test kernel

restricted

image = vmlinux-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 $1$ÕpîÁÜdpî$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.
 - 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 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.

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.

Step 4: The OS Kernel

- Typically, BSD variants include a **run control** (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.

The Superuser

- 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

The Superuser, cont

- 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

Multuser 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 its 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$bKJL9wAns39v1xQ1BZ8wd/: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]  
\br/>[-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`