# RH133 - Red Hat Linux System Administration

## Introduction - RH133 Red Hat Enterprise Linux Administration



## Unit 1 - System Initialization

# Unit 2 - Package Management



# Unit 3 - Kernel Services

# Unit 4 - System Services

# Unit 5 - User Administration



# Unit 6 - Filesystem Management

# Unit 7 - Advanced Filesystem Management

# Unit 8 - Network Configuration



# Unit 9 - Installation

# Unit 10 - Virtualization with Xen

# Unit 11 - Troubleshooting

# Introduction

# RH133 Red Hat Enterprise Linux Administration

RH133-RH133-RHEL5-en-1-20070321



# Copyright







# Welcome

Please let us know if you have any special needs while at our training facility.





# Participant Introductions

Please introduce yourself to the rest of the class!




# Red Hat Enterprise Linux

- Enterprise-targeted operating system
- Focused on mature open source technology
- 18-24 month release cycle
  - Certified with leading OEM and ISV products
- Purchased with one year Red Hat Network subscription and support contract
  - Support available for seven years after release
  - Up to 24x7 coverage plans available




# Red Hat Enterprise Linux Variants

- Two Install Sets available
- Server Spin
  - Red Hat Enterprise Linux
  - Red Hat Enterprise Linux Advanced Platform
- Client Spin
  - Red Hat Enterprise Linux Desktop
  - Workstation Option
  - Multi-OS Option




# Red Hat Network

- A comprehensive software delivery, system management, and monitoring framework
  - *Update Module* : Provides software updates
    - Included with all Red Hat Enterprise Linux subscriptions
  - *Management Module* : Extended capabilities for large deployments
  - *Provisioning Module* : Bare-metal installation, configuration management, and multi-state configuration rollback capabilities
  - *Monitoring Module* provides infrastructure health monitoring of networks, systems, applications, etc.

# Other Red Hat Supported Software

- Global Filesystem
- Directory Server
- Certificate Server
- Red Hat Application Stack
- JBoss Middleware Application Suite




# Notes on Internationalization

- Red Hat Enterprise Linux supports nineteen languages
- Default language can be selected:
  - During installation
  - With **system-config-language**
    - System->Administration->Language
- Alternate languages can be used on a per-command basis:

```
$ LANG=en_US.UTF8 date
```

- Language settings are stored in `/etc/sysconfig/i18n`




# The Fedora Project

- Red Hat sponsored open source project
- Focused on latest open source technology
  - Rapid four to six month release cycle
  - Available as free download from the Internet
- An open, community-supported proving ground for technologies which may be used in upcoming enterprise products
- Red Hat does not provide formal support

# Classroom Network

| | Names | IP Addresses |
|---|---|---|
| Our Network | example.com | 192.168.0.0/24 |
| Our Server | server1.example.com | 192.168.0.254 |
| Our Stations | station*X*.example.com | 192.168.0.*X* |
| Hostile Network | cracker.org | 192.168.1.0/24 |
| Hostile Server | server1.cracker.org | 192.168.1.254 |
| Hostile Stations | station*X*.cracker.org | 192.168.1.*X* |
| Trusted Station | trusted.cracker.org | 192.168.1.21 |

# Objectives

- Understand system and service initialization
- Integrate new filesystems
- Understand advanced partitioning schemes
- Perform filesystem management tasks
- Set up networking
- Perform user and group administration
- Automate tasks with at, cron, and anacron
- Set up core services: Logging, Printing, X Window system
- Manage software packages with **yum** and **rpm**
- Install the system interactively and with Kickstart
- Perform basic troubleshooting




# Audience and Prerequisites

- Audience: Linux or UNIX users, who understand the basics of Red Hat Linux, that desire further technical training to begin the process of becoming a system administrator.
- Prerequisites: RH033 Red Hat Linux Essentials or equivalent experience with Red Hat Enterprise Linux.




# Unit 1

# System Initialization

RH133-RH133-RHEL5-en-1-20070321



# Objectives

Upon completion of this unit, you should be able to:

- Discuss the boot sequence
- Understand GRUB's role
- Understand init's role
- Control System V services





# Boot Sequence Overview

- BIOS Initialization
- Boot Loader
- Kernel initialization
- **init** starts and enters desired run level by executing:
  - `/etc/rc.d/rc.sysinit`
  - `/etc/rc.d/rc` and `/etc/rc.d/rc?.d/`
  - `/etc/rc.d/rc.local`
  - X Display Manager if appropriate

# Boot Loader Components

- Boot Loader
  - 1st Stage - small, resides in MBR or boot sector
  - 2nd Stage - loaded from boot partition
- Minimum specifications for Linux:
  - Label, kernel location, OS root filesystem and location of the initial ramdisk (*initrd* )
- Minimum specification for other OS:
  - boot device, label

# GRUB and grub.conf

- GRUB "the GRand Unified Bootloader"
  - Command-line interface available at boot prompt
  - Boot from ext2/ext3, ReiserFS, JFS, FAT, minix, or FFS file systems
  - Supports MD5 password protection
- `/boot/grub/grub.conf`
- Changes to `grub.conf` take effect immediately
- If MBR on `/dev/hda` is corrupted, reinstall the first stage bootloader with:
  - `/sbin/grub-install /dev/hda`

# Starting the Boot Process: GRUB

- Image selection
  - Select with space followed by up/down arrows on the boot splash screen
- Argument passing
  - Change an existing stanza in menu editing mode
  - Issue boot commands interactively on the GRUB command line

# Kernel Initialization

- Kernel boot time functions
  - Device detection
  - Device driver initialization
  - Mounts root filesystem read only
  - Loads initial process (**init**)

# init Initialization

- **init** reads its config: `/etc/inittab`
  - initial run level
  - system initialization scripts
  - run level specific script directories
  - trap certain key sequences
  - define UPS power fail / restore scripts
  - spawn gettys on virtual consoles
  - initialize X in run level 5

# Run Levels

- **init** defines run levels 0-6, S, emergency
- The run level is selected by either
  - the default in `/etc/inittab` at boot
  - passing an argument from the boot loader
  - using the command **init** ***new_runlevel***
- Show current and previous run levels
  - **/sbin/runlevel**




# /etc/rc.d/rc.sysinit

- Important tasks include:
  - Activate **udev** and `selinux`
  - Sets kernel parameters in `/etc/sysctl.conf`
  - Sets the system clock
  - Loads keymaps
  - Enables swap partitions
  - Sets hostname
  - Root filesystem check and remount
  - Activate RAID and LVM devices
  - Enable disk quotas
  - Check and mount other filesystems
  - Cleans up stale locks and PID files

# /etc/rc.d/rc

- Initializes the default run level per the `/etc/inittab` file `initdefault` line such as `id:3:initdefault:`

- `l0:0:wait:/etc/rc.d/rc 0`

```
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3 (default)
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6
```

# System V run levels

- Run level defines which services to start
  - Each run level has a corresponding directory:
    - `/etc/rc.d/rc`*X*`.d`
  - The System V init scripts reside in:
    - `/etc/rc.d/init.d`
  - Symbolic links in the run level directories call the `init.d` scripts with a start or stop argument



# /etc/rc.d/rc.local

- Run after the run level specific scripts
- Common place for custom modification
- In most cases it is recommended that you create a System V *init* script in
- `/etc/rc.d/init.d` unless the service you are starting is so trivial it doesn't warrant it. Existing scripts can be used as a starting point.

# Controlling Services

- Utilities to control default service startup
  - **system-config-services**: graphical utility that requires an X interface
  - **ntsysv**: ncurses based utility usable in virtual consoles
  - **chkconfig**: a fast, versatile command line utility that works well and is usable with scripts and Kickstart installations
- Utilities to control services manually
  - **service**: immediately start or stop a standalone service
  - **chkconfig** immediately starts and stops **xinetd**-managed services

# End of Unit 1

- Questions and Answers
- Summary
  - System BIOS
  - GRUB
  - init
  - chkconfig and service

# Unit 2

# Package Management

RH133-RH133-RHEL5-en-1-20070321



redhat

# Objectives

Upon completion of this unit, you should be able to:

- Install and remove RPM packages
- Query packages and verify their state
- Manage packages using **yum**
- Understand the relationship between **yum** and **rpm**
- Configure **yum** to connect to a RHN Satellite Server
- Create a private **yum** repository
- Configure **yum** to connect to a private repository
- Configure and use Red Hat Network

# RPM Package Manager

- RPM Components
  - local database
  - **rpm** and related executables
  - RPM frontends such as yum
  - package files
- Primary Functions
  - install/remove
  - query
  - verify

# Installing and Removing Software

- Primary RPM options:
  - Install: **rpm -i**, **--install**
  - Upgrade: **rpm -U**, **--upgrade**
  - Freshen: **rpm -F**, **--freshen**
  - Erase: **rpm -e**, **--erase**
- Output Options: **-v**, **-h**
- URL support: `ftp://` (with globbing), `http://`
- Many other install-options are available to address special cases.

# Updating a Kernel RPM

- Make sure to install kernel updates
- Do not use **rpm -U** or **rpm -F** !
  - **rpm -ivh kernel-version.arch.rpm**
  - Boot new kernel to test
  - Revert to old kernel if a problem arises
  - **rpm -e kernel-oldversion** if no problems

# rpm Queries

- Syntax:
  - **rpm -q what_packages what_information**
- Installed Package Options:
  - **rpm -qa** lists installed packages
  - **rpm -qf filename** shows owning package
  - **rpm -qi package_name** general information
  - **rpm -ql package_name** lists files in package
- Uninstalled Package Options:
  - **rpm -qip package_file.i386.rpm**
  - **rpm -qlp package_file.i686.rpm**

# rpm Verification

- Installed RPM File Verification:
  - **rpm -V <package_name>**
  - **rpm -Vp <package_file>.i386.rpm**
  - **rpm -Va**
- Signature verification BEFORE package install:
  - **rpm --import RPM-GPG-KEY**
  - **rpm -K <package_file>.i386.rpm**

# About yum

- Front-end to **rpm**
  - Designed to resolve package dependencies
  - Can locate packages across multiple repositories
- Replacement for **up2date**




# Using yum

- Install/Remove/Update
  - **yum install** *package...*
  - **yum remove** *package...*
  - **yum update** *[package...]*




# Searching packages/files

- Searching packages
  - **yum search** *searchterm*
  - **yum list** *(all|available|extras|installed|recent|updates)*
  - **yum info** *packagename*
- Searching files
  - **yum whatprovides** *filename*

# Configuring Additional Repositories

- Create a file in `/etc/yum.repos.d` for your repository
- Required information
  - [*repo-name*]
  - name=*A nice description*
  - baseurl=*http://yourserver.com/path/to/repo*
  - enabled=*1*
  - gpgcheck=*1*



# Creating a private repository

- Create a directory to hold your packages
- Make this directory available by **http**/**ftp**
- Install the **createrepo** RPM
- Run **createrepo -v /package/directory**
- This will create a `repodata` subdirectory

and the needed support files



# Red Hat Network

- Centralized platform for systems mangement
  - provides Red Hat software packages
  - shows if errata are available for systems
  - can update many systems at once
  - allows full life cycle management
- Webbased management interface
- Uses HTTPS for all transactions

# Red Hat Network Server

- `rhn.redhat.com` or local Satellite/Proxy
  - Web based management of machines
  - RHN Proxy caches RHN traffic
  - RHN Satellite provides an autonomous RHN
- RHN Accounts
  - RHN Users for registration of machines and web based management
  - System ID for automatic authentication of systems

# Entitlements

- Grant access to software channels
  - Base Channel
  - Child Channel(s)
- Define level of service
  - Update
  - Management
  - Provisioning
  - Monitoring

# Red Hat Network Client

- Registration
  - Run **rhn_register**
  - Select the updates location (RHN or local satellite/ proxy)
  - Enter Account information
- Interactive usage
  - **yum** plugin for downloading packages from RHN
  - Configuration in `/etc/yum/pluginconf.d/rhn-plugin.conf`
- Remote management
  - **rhnsd** polls RHN every four hours
  - **rhn_check** polls imediately

# End of Unit 2

- Questions and Answers
- Summary
  - What are the primary functions of RPM?
  - What **rpm** options should be used to install a kernel RPM?
  - Package-management with yum
  - Relationship between **yum** and **rpm**
  - Using **yum** with RHN
  - Creating a private repository
  - Configuring repositories
  - How does Red Hat Network work?



# Unit 3

# Kernel Services





# Objectives

Upon completion of this unit, you should be able to:

- Understand the purpose and organization of the kernel
- Understand how to load and configure kernel modules.
- Know how to configure the kernel using /proc and sysctl
- Explore hardware devices available on the system

# The Linux Kernel

- The kernel constitutes the core part of the Linux operating system.
- Kernel duties:
  - System initialization: detects hardware resources and boots up the system.
  - Process scheduling: determines when processes should run and for how long.
  - Memory Management: allocates memory on behalf of running processes.
  - Security: Constantly verifies filesystem permissions, SELinux contexts and firewall rules.
  - Provides buffers and caches to speed up hardware access.
  - Implements standard network protocols and filesystem formats.
- Documentation available in the `kernel-doc` RPM package

# Kernel Images and Variants

- Architectures supported: x86, x86_64, IA64/ Itanium, PowerPC64, s390x.
- Three kernel versions available for x86:
  - Regular: one or more processors but 4GB of RAM or less
  - PAE: multiple processors and up to 64G of RAM
  - Xen: needed for virtualization
- Kernels always installed under `/boot/vmlinuz-*`

# Kernel Modules

- Modules are small kernel extensions that may be loaded and unloaded at will
- Can implement drivers, filesystems, firewall, and more
- Are located under `/lib/modules/$(uname -r)/`
- Compiled for a specific kernel version and are provided with the kernel RPM.
- Third party modules may be added



# Kernel Module Utilities

- **lsmod** provides a list of loaded modules
- **modprobe** can load and unload modules
- **modinfo** displays information about any available module
- `/etc/modprobe.conf` used for module configuration:
  - Parameters to pass to a module whenever it is loaded
  - Aliases to represent a module name
  - Commands to execute when a module is loaded or unloaded

# Managing the initrd Image

- The initial RAM disk provides modules loaded early in the boot process.
- This file is located under `/boot/initrd-$(uname -r).img`
- Extra modules sometimes need to be added due to:
  - New hardware added to the system. i.e. SCSI controller
  - New features needed such as USB devices.
  - Module needs to load automatically at boot time.
- Use **mkinitrd** and the **--with** option to rebuild with an extra module:

```
mkinitrd --with=module_name /boot/initrd-$(uname -r).img \
$(uname -r)
```

# Accessing Drivers Through /dev

- Files under /dev used to access drivers
- Reading from and writing to those files are valid operations:
  - Read from serial port: **cat /dev/ttyS0**
  - Write to serial port: **echo "Message" > /dev/ttyS0**
- Three file attributes determine which driver to access:
  - Device type (character or block)
  - Major number
  - Minor number

# Device Node Examples

- Block Devices
  - `/dev/hda`, `/dev/hdc` - IDE hard disk, CDROM
  - `/dev/sda`, `/dev/sdb` - SCSI, SATA, or USB Storage
  - `/dev/md0`, `/dev/md1` - Software RAID
- Character Devices
  - `/dev/tty[0-6]` - virtual consoles
  - `/dev/null`, `/dev/zero` - software Devices
  - `/dev/random`, `/dev/urandom` - random Numbers

# Managing /dev With udev

- udev manages files stored under /dev/
- Files only created if corresponding device is plugged in
- Files are automatically removed when device is disconnected
- udev statements under /etc/udev/rules.d/ determine:
  - Filenames
  - Permissions
  - Owners and groups
  - Commands to execute when a new device shows up

# Adding Files Under /dev

- The right way to add a /dev entry involves udev:
  - Create a new file under /etc/udev/rules.d/
  - Insert a statement such as:

```
KERNEL=="sda", NAME="usbkey" , SYMLINK="usbstorage"
```

  - This creates a device file named usbkey and a symlink named usbstorage next time /dev/sda gets plugged.
- Files can be added manually with **mknod**:

```
mknod /dev/usbdevice b 8 0
```

- **mknod** not persistent!

# Kernel Configuration With /proc

- /proc used to get or set kernel configuration
- Virtual filesystem: files not stored on hard disk
- Entries not persistent: modifications get reinitialized after a reboot
- Used to display process information, memory resources, hardware devices, kernel memory, etc.
- Can be used to modify network and memory subsystems or modify kernel features
- Modifications apply immediately

# /proc Examples

- Read-only files:
  - /proc/cpuinfo
  - /proc/1/*
  - /proc/partitions
  - /proc/meminfo
- Read-Write entries under `/proc/sys/`:
  - `/proc/sys/kernel/hostname`
  - `/proc/sys/net/ipv4/ip_forward`
  - `/proc/sys/vm/drop_caches`
  - `/proc/sys/vm/swappiness`

# sysctl : Persistent Kernel Configuration

- **sysctl** adds persistence to `/proc/sys` settings
- Statements added to `/etc/sysctl.conf` automatically reflected under `/proc` after a reboot.
- Configuration maintained or monitored using the **sysctl** command:
  - List all current settings: **sysctl -a**
  - Reload settings from `sysctl.conf`: **sysctl -p**
  - Set a `/proc` value dynamically: **sysctl -w net.ipv4.ip_forward=1**

# Exploring Hardware Devices

- A snapshot of all connected devices is maintained by HAL: Hardware Abstraction Layer
- **hal-device** lists all devices in text mode.
- **hal-device-manager** displays all devices on a graphical window.
- **lspci** and **lsusb** list devices connected to the PCI and USB buses, respectively.
- The `/proc` and `/sys` filesystems also contain bus and device specific information.

# Monitoring Processes and Resources

- Information available under `/proc/` can be hard to understand.
- Interfaces are available to format the data and make it more accessible:
  - Memory: **free**, **vmstat**, **swapon -s**, **pmap**
  - Processes: **ps**, **top**, **gnome-system-monitor**
  - Kernel state: **uname**, **uptime**, **tload**

# Unit 4

# System Services

RH133-RH133-RHEL5-en-1-20070321



# Objectives

Upon completion of this unit, you should be able to:

- Understand the importance of time syncronization
- Configure System Logging
- Setup the X Window System
- remotely administer the system
- Automate tasks with cron
- Configure printing




# Network Time Protocol

- Workstation hardware clocks tend to drift over time without correction
- Many application require accurate timing
- Time synchronization makes system logs easier to analyze
- NTP counters the drift by manipulating the length of a second
- NTP clients should use three time servers
- Config file: `/etc/ntp.conf`
- Config tool: **system-config-date**

# System Logging

- Centralized logging daemons: **syslogd**, **klogd**
- Log file examples:
  - `/var/log/dmesg`: Kernel boot messages
  - `/var/log/messages`: Standard system error messages
  - `/var/log/maillog`: Mail system messages
  - `/var/log/secure`: Security, authentication, and xinetd messages
- Application log files and directories also reside in `/var/log`

# syslog Configuration

- **syslog** System V initialization script in `/etc/rc.d/init.d` controls both the syslogd and the klogd daemons
- `/etc/syslog.conf`
  - Configures system logging
- `/etc/sysconfig/syslog`
  - Sets switches used when starting syslogd and klogd from the System V initialization script

# XOrg: The X11 Server

- Foundation for the Red Hat Enterprise Linux graphical user interface(GUI)
- Open Source implementation of X11
- Client / Server Architecture
- Core server with dynamically loaded modules
  - drivers: ati, nv, mouse, keyboard, etc.
  - extensions: dri, glx, and extmod
- Font Rendering
  - Native server: **xfs**
  - Fontconfig/Xft libraries

# XOrg Server Configuration

- Typically configured after installation
- Post-install configuration:
  - Best results while in runlevel 3!
  - **system-config-display**
    - options:
      - **--noui**
      - **--reconfig**
  - stored in `/etc/X11/xorg.conf`

# XOrg in runlevel 3

- Two methods to establish the environment
  - `/usr/X11R6/bin/xinit`
  - `/usr/X11R6/bin/startx`
- Environment configuration
  - `/etc/X11/xinit/xinitrc` and `~/.xinitrc`
  - `/etc/X11/xinit/Xclients` and `~/.Xclients`
  - `/etc/sysconfig/desktop`

# XOrg in runlevel 5

- Environment established by **/sbin/init**
- Environment configuration
  - `/etc/inittab`
    - `/etc/X11/prefdm`
  - `/etc/sysconfig/desktop`
    - DESKTOP defines the window manager
    - DISPLAYMANAGER defines the display manager
  - `/etc/X11/xdm/Xsession`
    - `/etc/X11/xinit/xinitrc.d/*`
    - `~/.xsession` or `~/.Xclients`

# Remote X Sessions

- X protocol communication is unencrypted
- Host-based sessions implemented through the **xhost** command
- User-based sessions implemented through the Xauthority mechanism
- **sshd** may automatically install **xauth** keys on remote machine
  - Tunnels X protocol over secure encrypted **ssh** connection

# SSH: Secure Shell

- encrypted remote shell
- frequently used for remote system administration
- can copy files securely
- can execute commands remotely

```
# ssh root@host 'ifconfig eth0'
```

- can tunnel X11 and other TCP based network traffic
- supports key based authentication



# VNC: Virtual Network Computing

- Allows to access or share a complete desktop over the network
- Uses significantly less bandwidth as pure remote X connections
- Server
  - Individual users can start a VNC server with the command: **vncserver**
  - Runs `$HOME/.vnc/xstartup` upon startup
  - Requires a VNC password which should not be identical to the system password
  - Servers can automatically be started via `/etc/init.d/vncserver`
- Client
  - connects to a remote VNC server with **vncviewer *host*:*screen***
  - Unique screen numbers distinguish between multiple VNC servers running on the same host
  - supports tunneling through SSH: **vncviewer -via user@host localhost:1**

# cron

- Used to schedule recurring events
- Use crontab to edit, install, and view job schedules
- Syntax
  - **`crontab [-u user] file`**
  - **`crontab [-l|-r|-e]`**
    - **-l** lists crontab
    - **-r** removes crontab
    - **-e** edits crontab using `$EDITOR`

# Controlling Access to cron

- Restrict / allow user access to cron
  - `/etc/cron.allow`
  - `/etc/cron.deny`
- Contains usernames to allow / deny access

# System crontab Files

- Different format than user crontab files
- Master crontab file `/etc/crontab` runs executables in
  - `/etc/cron.hourly`
  - `/etc/cron.daily`
  - `/etc/cron.weekly`
  - `/etc/cron.monthly`
- `/etc/cron.d/` directory contains additional system crontab files



# Daily Cron Jobs

- **tmpwatch**
  - Cleans old files in specific directories
  - Keeps `/tmp` from filling up
- **logrotate**
  - Keeps log files from getting to large
  - Highly configurable in `/etc/logrotate.conf`
- **logwatch**
  - provides a summary about system activity
  - reports suspicious messages
  - Configuration file: `/etc/log.d/conf/logwatch.conf`

# The anacron System

- **anacron** runs **cron** jobs that did not run when the computer is down
  - Assumes computers are not up continually
  - Vital for laptops, desktops, workstations, and other systems that are not up continually
  - Useful for servers that need to be taken down temporarily
- Configuration file: `/etc/anacrontab`
  - Field 1: If the job has not been run in this many days...
  - Field 2: wait this number of minutes after reboot and then run it
  - Field 3: job identifier
  - Field 4: the job to run

# CUPS

- uses the Internet Printing Protocol (IPP)
  - allows remote browsing of printer queues
  - based on HTTP/1.1
  - Uses PPD files to describe printers
  - Configuration files
    - `/etc/cups/cupsd.conf`
    - `/etc/cups/printers.conf`
  - Configuration tools
    - **system-config-printer**
    - Web based on `http://localhost:631`
    - Commandline management with **lpadmin**

# End of Unit 4

- Questions and Answers
- Summary
  - System Logging
  - system-config-display
  - Remote Administration tools: **ssh** and **vnc**
  - Task Automation
  - What are the tools to configure **cups**?

# Unit 5

# User Administration

RH133-RH133-RHEL5-en-1-20070321



# Objectives

Upon completion of this unit, you should be able to:

- Configure user and group accounts
- Modify File ownership and permissions
- Use "Special" permissions SUID / SGID / Sticky
- Configure Network Users with NIS and LDAP
- Set ACLs

# Adding a New User Account

- Most common method is **useradd**:
  - **useradd [*options*] *username***
- Running **useradd** is equivalent to:
  - editing `/etc/passwd`, `/etc/shadow`, `/etc/group`, `/etc/gshadow`
  - creating and populating home directory
  - setting permissions and ownership
- Set account password using **passwd**
- Accounts may be added in a batch with **newusers**

# User Private Groups

- When user accounts are created, a private group is also created with the same name
  - Users are assigned to this private group
  - User's new files affiliated with this group
- Advantage: Prevents new files from belonging to a "public" group
- Disadvantage: May encourage making files "world-accessible"

# Modifying / Deleting User Accounts

- To change fields in a user's `/etc/passwd` entry you can:
  - Edit the file by hand
  - Use **usermod [*options*] *username***
- To remove a user either:
  - Manually remove the user from `/etc/passwd`, `/etc/shadow`, `/etc/group`, `/etc/gshadow`, `/var/spool/mail`, etc.
  - Use **userdel [-r] *username***

# Group Administration

- Entries added to `/etc/group` and `/etc/gshadow`
  - **groupadd**
  - **groupmod**
  - **groupdel**

# Password Aging Policies

- By default, passwords do not expire
- Forcing passwords to expire is part of a strong security policy
- Modify default expiration settings in `/etc/login.defs`
- To modify password aging for existing users, use the **chage** command
  - **chage [*options*] *username***

# Switching Accounts

- Syntax
  - **su [-] [user]**
  - **su [-] [user] -c *command***
- Allows the user to temporarily become another user
  - Default user is root
- The "**-**" option makes the new shell a login shell

# sudo

- Users listed in `/etc/sudoers` execute commands with:
  - an effective user id of 0
  - group id of root's group
- An administrator will be contacted if a user not listed in `/etc/sudoers` attempts to use **sudo**



# Network Users

- Information about users may be centrally stored and managed on a remote server
- Two types of information must always be provided for each user account
  - Account information: UID number, default shell, home directory, group memberships, and so on
  - Authentication: a way to tell that the password provided on login for an account is correct

# Authentication Configuration

- **system-config-authentication**
  - GUI tool to configure authentication
  - For text-based tool, use **authconfig-tui**
  - Load `authconfig-gtk` RPM
- Supported account information services:
  - (local files), NIS, LDAP, Hesiod, Winbind
- Supported authentication mechanisms:
  - (NSS), Kerberos, LDAP, SmartCard, SMB, Winbind

# Example: NIS Configuration

- Must install `ypbind` and `portmap` RPMs
- Run **system-config-authentication**
  - Enable NIS to provide User Information
  - Specify NIS server and NIS domain name
  - Keep default authentication (through NSS)
- What does this actually do?
  - Five text-based configuration files are changed

# Example: LDAP Configuration

- Must install `nss-ldap` and `openldap` RPMs
- Run **system-config-authentication**
  - Enable LDAP to provide User Information
  - Specify server, the search base DN, and TLS
  - Enable LDAP to provide Authentication
- What does this actually do?
  - Five text-based configuration files are changed

# SUID and SGID Executables

- Normally processes started by a user run under the user and group security context of that user
- SUID and/or SGID bits set on an executable file cause it to run under the user and/or group security context of the file's owner and/or group

# SGID Directories

- Used to create a collaborative directory
- Normally, files created in a directory belong to the user's the default group
- When a file is created in a directory with the SGID bit set, it belongs to the same group as the directory

# The Sticky Bit

- Normally users with write permissions to a directory can delete any file in that directory regardless of that file's permissions or ownership
- With the sticky bit set on a directory, only the owner of a file can delete the file
- Example:

```
ls -ld /tmp
drwxrwxrwt  12 root  root  4096 Nov 2 15:44 /tmp
          ^
```

# Default File Permissions

- Read and write (not execute) for all is the default for files
- Read, write and execute is the default for directories
- **umask** can be used to withhold permissions on file creation
- Users' umask is 022
  - Files will have permissions of 644
  - Directories will have permissions of 755
  - May need to change to 002 for group colaboration



# Access Control Lists (ACLs)

- Grant rwx access to files and directories for multiple users or groups
  - **mount -o acl *`/directory`***
  - **getfacl *`file`*|*`directory`***
  - **setfacl -m u:gandolf:rwx *`file`*|*`directory`***
  - **setfacl -m g:nazgul:rw *`file`*|*`directory`***
  - **setfacl -m d:u:frodo:rw *`directory`***
  - **setfacl -x u:samwise *`file`*|*`directory`***

# SELinux

- Mandatory Access Control (MAC) -vs- Discretionary Access Control (DAC)
- A rule set called the *policy* determines how strict the control
- Processes are either restricted or unconfined
- The policy defines what resources restricted processes are allowed to access
- Any action that is not explicitly allowed is, by default, denied



# SELinux, continued

- All files and processes have a *security context*
- The context has several elements, depending on the security needs
  - user:role:type:sensitivity:category
  - `user_u:object_r:tmp_t:s0:c0`
  - Not all systems will display s0:c0
- **ls -Z**
- **ps -Z**
  - Usually paired with other options, such as **-e**

# SELinux: Targeted Policy

- The targeted policy is loaded at install time
- Most local processes are *unconfined*
- Principally uses the type element for *type enforcement*
- The security context can be changed with **chcon**
  - **chcon -t tmp_t /etc/hosts**
- Safer to use **restorecon**
  - **restorecon /etc/hosts**

# SELinux: Management

- Modes: Enforcing, Permissive, Disabled
  - Changing enforcement is allowed in the Targeted policy
  - **getenforce**
  - **setenforce 0 | 1**
  - Disable from GRUB with **selinux=0**
- `/etc/sysconfig/selinux`
- **system-config-securitylevel**
  - Change mode, Disabling requires reboot
- **system-config-selinux**
  - Booleans
- **setroubleshootd**
  - Advises on how to avoid errors, not ensure security!

# End of Unit 5

- Questions and Answers
- Summary
  - User and Group accounts
  - File ownership and permissions
  - Extended file modes: SUID / SGID / Sticky
  - Switching accounts with su
  - **umask** and the UPG scheme
  - Shell environment
  - Setup NIS and LDAP
  - Use ACLs
  - Configure and troubleshoot SELinux



# Unit 6

# Filesystem Management

RH133-RH133-RHEL5-en-1-20070321



# Objectives

Upon completion of this unit, you should be able to:

- Understand filesystem hierarchy
- Manage virtual memory
- Add new drives and partitions
- Mount NFS filesystems




# Overview: Adding New Filesystems to the Filesystem Tree

- Identify Device
- Partition Device
- Make Filesystem
- Label Filesystem
- Create entry in `/etc/fstab`
- Mount New Filesystem

# Device Recognition

- Master Boot Record ( MBR ) contains:
  - Executable code to load operating system
  - Space for partition table information, including:
    - Partition id or type
    - Starting cylinder for partition
    - Number of cylinders for partition

# Disk Partitioning

- An extended partition points to additional partition descriptors
- Total maximum number of partitions supported by the kernel:
  - 63 for IDE drives
  - 15 for SCSI drives
- Why partition drives?
  - containment, performance, quotas, recovery




# Managing Partitions

- Create partitions using:
  - **fdisk**
  - **sfdisk**
  - GNU **parted** - advanced partition manipulation (create, copy, resize, etc.)
- **partprobe** - reinitializes the kernel's in-memory version of the partition table

# Making Filesystems

- **mkfs**
- **mkfs.ext2**, **mkfs.ext3**, **mkfs.msdos**
- Specific filesystem utilities can be called directly
  - **mke2fs [*options*] *device***

# Filesystem Labels

- Alternate way to refer to devices
- Device independent
  - **e2label** ***special_dev_file [fslabel]***
  - **mount** ***[options]*** **LABEL=*fslabel* *mount_point***
- **blkid** can be used to see labels and filesystem type of all devices.

# tune2fs

- adjusts filesystem parameters
  - reserved blocks
  - default mount options
  - **fsck** frequency
- View current settings with **dumpe2fs**

# Mount Points and /etc/fstab

- Configuration of the filesystem hierarchy
- Used by **mount**, **fsck**, and other programs
- Maintains the hierarchy between system reboots
- May use filesystem volume labels in the device field
- The **mount -a** command can be used to mount all filesystems listed in the /etc/fstab



# Mounting Filesystems with mount

- **mount [options] *device mount_point***
  - `-t` *`vfstype`* (normally not needed)
  - `-o options`
  - Default options: `rw`, `suid`, `dev`, `exec`, `acl`, and `async`

# Unmounting Filesystems

- **umount [*options*] *device* | *mount_point***
- You cannot unmount a filesystem that is in use
  - Use **fuser** to check and/or kill processes
- Use the **remount** option to change a mounted filesystem's options atomically
  - **mount -o remount,ro /data**

# mount By Example

- Sample filesystem requirements met using options:
  - Disabling execute access
  - Mounting a filesystem image
  - Mounting a PC-compatible filesystem
  - Disabling access time updates
  - Setting up a **mount** alias

# Handling Swap Files and Partitions

- Swap space is a supplement to system RAM
- Basic setup involves:
  - Create swap partition or file
  - Write special signature using **mkswap**
  - Add appropriate entries to `/etc/fstab`
  - Activate swap space with **swapon -a**

# Mounting NFS Filesystems

- Makes a remote NFS filesystem work as though it were a local filesystem
- `/etc/fstab` can be used to specify persistent network mounts
- NFS shares are mounted at boot time by **/etc/init.d/netfs**
- Exports can be mounted manually with the **mount** command.

```
# mkdir /mnt/server1
# mount -t nfs server1:/var/ftp/pub /mnt/server1
```

# Automounter

- System administrator specifies mount points controlled by automounter daemon process in `/etc/auto.master`
- The automounter monitors access to these directories and mounts the filesystem on demand
- Filesystems automatically unmounted after a specified interval of inactivity
- Enable the special map `-host` to "browse" all NFS exports on the network
- Supports wildcard directory names

# Direct Maps

- Direct maps include absolute path names
- Does not obscure local directory structure
- Example:

```
/etc/auto.master:
/-  /etc/auto.direct

/etc/auto.direct:
/foo            server1:/export/foo
/usr/local/  server1:/usr/local
```

# gnome-mount

- automatically mounts removable devices
- integrated with the HAL (Hardware Abstraction Layer)
- replaces fstab-sync

# End of Unit 6

- Questions and Answers
- Summary
  - What tools are available for partitioning?
  - What two ways can swap space be implemented?
  - Mount NFS

# Unit 7

# Advanced Filesystem Management

RH133-RH133-RHEL5-en-1-20070321



# Objectives

Upon completion of this unit, you should be able to:

- Setup filesystem quotas
- Setup and manage software Raid devices
- Configure Logical Volumes
- setup LVM Snapshots
- perform backups




# Configuring the Quota System

- Overview
  - Implemented within the kernel
  - Enabled on a per-filesystem basis
  - Individual policies for groups or users
    - Limit by the number of blocks or inodes
    - Implement both soft and hard limits
- Initialization
  - Partition mount options: `usrquota, grpquota`
  - Initialize database: **quotacheck**

# Setting Quotas for Users

- Implementation
  - Start or stop quotas: **quotaon**, **quotaoff**
  - Edit quotas directly: **edquota *username***
  - From a shell:

    **setquota *username* 4096 5120 40 50 */foo***

  - Define prototypical users:

    **edquota -p *user1 user2***

# Reporting Quota Status

- Reporting
  - User inspection: **quota**
  - Quota overviews: **repquota**
  - Miscellaneous utilities: **warnquota**

# What is Software RAID?

- Multiple disks grouped together into "arrays" to provide better performance, redundancy or both.
- **mdadm** - provides the administration interface to software RAID.
- Many "RAID Levels" supported, including RAID O, 1 and 5.
- Spare disks add extra redundancy
- RAID devices are named, `/dev/md0`, `/dev/md1`, `/dev/md2`, `/dev/md3` and so on.

# Software RAID Configuration

- Create and define RAID devices using **mdadm**

```
mdadm -C /dev/md0 -a yes -l 1 -n 2 -x 1 /dev/sda1 /dev/sdb1 /dev/sdc1
```

- Format each RAID device with a filesystem

```
mke2fs -j /dev/md0
```

- Test the RAID devices
- **mdadm** allows you to check the status of your RAID devices

```
mdadm --detail /dev/md0
```

# Software RAID Testing and Recovery

- Simulating disk failures

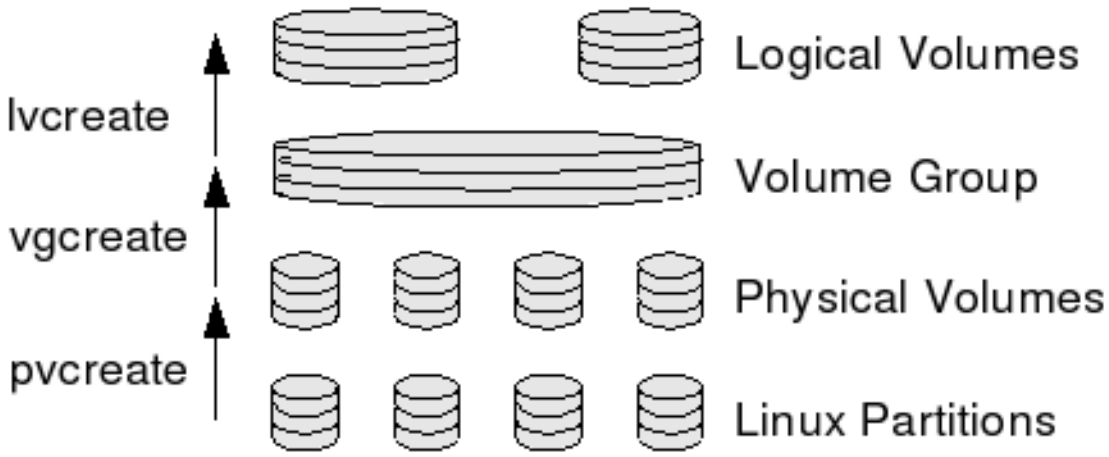
```
mdadm /dev/md0 -f /dev/sda1
```

- Recovering from a software RAID disk failure
  - replace the failed hard drive and power on
  - reconstruct partitions on the replacement drive
  - **`mdadm /dev/md0 -a /dev/sda1`**
- **mdadm**, `/proc/mdstat`, and syslog messages




# What is Logical Volume Manager (LVM)?

- A layer of abstraction that allows easy manipulation of volumes. Including resizing of filesystems
- Allows reorganization of filesystems across multiple physical devices
  - Devices are designated as Physical Volumes
  - One or more Physical Volumes are used to create a Volume Group
  - Physical Volumes are defined with Phyisical Extents of a fixed size
  - Logical Volumes are created on Physical Volumes and are composed of Physical Extents
  - Filesystems may be created on Logical Volumes



# Creating Logical Volumes

- Create physical volumes

```
pvcreate /dev/hda3
```

- Assign physical volumes to volume groups

```
vgcreate vg0 /dev/hda3
```

- Create logical volumes from volume groups

```
lvcreate -L 256M -n data vg0
mke2fs -j /dev/vg0/data
```

# Resizing Logical Volumes

- Growing Volumes
  - **lvextend** can grow logical volumes
  - **resize2fs** can grow EXT3 filesystems online
  - **vgextend** adds new physical volumes to an existing volume group.
- Shrinking volumes
  - Filesystem must be reduced first
  - Requires a filesystem check and cannot be performed online
  - **lvreduce** can then reduce the volume.
  - Volume Groups can be reduced with:

```
pvmove /dev/hda3
vgreduce vg0 /dev/hda3
```

# Logical Volume Manager Snapshots

- *Snapshots* are special Logical Volumes that are an exact copy of an existing Logical Volume at the time the snapshot is created
- Snapshots are perfect for backups and other operations where a temporary copy of an existing dataset is needed
- Snapshots only consume space where they are *different* from the original Logical Volume
  - Snapshots are allocated space at creation but do not use it until changes are made to the original Logical Volume or the Snapshot
  - When data is changed on the original Logical Volume the older data is copied to the Snapshot
  - Snapshots contain only data that has changed on the original Logical Volume or the Snapshot since the Snapshot was created.





# Using LVM Snapshots

- Create Snapshot of existing Logical Volume

```
# lvcreate -l 64 -s -n databackup /dev/vg0/data
```

- Mount Snapshot

```
#  mkdir -p /mnt/databackup
# mount -o ro /dev/vg0/databackup /mnt/databackup
```

- Remove Snapshot

```
# umount /mnt/databackup
# lvremove /dev/vg0/databackup
```

# Archiving tools: tar

- **tar** can backup to a file or tape device
- supports GZIP and BZIP2 compression
- can preserve file permissions, ownership and timestamps
- supports extended attributes
- uses **rmt** to write to a remote tape device

# Archiving Tools: dump/restore

- Back up and restore ext2/3 filesystems
  - Does not work with other filesystems
  - dump should only be used on unmounted filesystems or filesystems that are read-only.
- Can do full or incremental backups
- Examples:

```
dump -0u -f /dev/nst0 /dev/hda2
restore -rf /dev/nst0
```

# Archiving Tools: rsync:

- Efficiently copies files to or from remote systems
- Uses secure **ssh** connections for transport
  - **rsync *.conf barney:/home/joe/configs/**
- Faster than **scp** - copies differences in like files

# End of Unit 7

- Questions and Answers
- Summary
  - Filesystem quotas
  - Configuration of Software RAID
  - Software RAID recovery
  - Configuration of Logical Volumes
  - LVM Snapshots
  - Backup Tools

# Unit 8

# Network Configuration





# Objectives

Upon completion of this unit, you should be able to:

- configure IP interfaces
- setup routes
- understand name resolution
- setup IPv6





# Network Inferfaces

- Networking scripts refer to logical interface names:
  - Ethernet: `eth0, eth1 ...`
  - Dial-up: `ppp0, ppp1 ...`
  - Loopback: `lo`
- Display network interfaces by using:
  - **ifconfig -a**
  - **ip link [show]**




# Driver Selection

- All drivers for network interface cards are built as modules
- `/etc/modprobe.conf` maps logical names to specific modules:
  - `alias eth0 3c59x`
- Secondary "card selection" can be specified in the interface configuration file, `/etc/sysconfig/network-scripts/ifcfg-eth0`
  - `HWADDR=00:0D:60:FB:CA:61`

# Speed and Duplex Settings

- Modules are configured to autonegotiate, by default
- Mismatches can cause intermittent to no communcation
- Manually overridden using:
  - **ethtool**
  - `ETHTOOL_OPTS` in `ifcfg-`*`ethX`*
  - `options` or `install` in `/etc/modprobe.conf` for older interface modules

# IPv4 Addresses

- View configuration with:
  - **ifconfig**
  - **ip addr [show]**

# Dynamic IPv4 Configuration

- Interface configuration defined in:
  - `/etc/sysconfig/network-scripts/ifcfg-`*`ethX`*
  - Dynamic with line of: `BOOTPROTO=dhcp`
- Zero Configuration Networking
  - Uses `169.254.0.0/16`
  - Disabled with line of: `NOZEROCONF=yes` in `/etc/sysconfig/network-scripts/ifcfg-`*`ethX`*
- Use **ifdown *`device`*; ifup *`device`*** to apply configuration changes




# Static IPv4 Configuration

- Interface configuration defined in:
  - `/etc/sysconfig/network-scripts/ifcfg-`*`ethX`*
- Static with lines of:
  - `BOOTPROTO=none`
  - `IPADDR=`*`10.0.0.1`*
  - `NETMASK=`*`255.255.255.0`*

# Device Aliases

- Useful for virtual hosting
- Bind multiple IP addresses to a single NIC
  - `eth1:1`
  - `eth1:2`
  - `eth1:3`
- Create a separate interface configuration file for each device alias:
  - `ifcfg-ethX:xxx`
  - Must use static networking

# Routing Table

- Defines path to all systems
- View table with:
  - **route**
  - **netstat -r**
  - **ip route [show]**

# Default Gateway

- Used when no route entry is matched
- Might be obtained dynamically with DHCP
- Can be statically configured:
  - With a line of: `GATEWAY=`*`10.53.0.254`*
  - Globally in: `/etc/sysconfig/network`
  - OR, per interface in the interface configuration file: `/etc/sysconfig/network-scripts/ifcfg-`*`ethX`*

# Configuring Routes

- To control traffic flow when there is more than one router
- Static routes defined per interface
  - `/etc/sysconfig/network-scripts/route-`*`ethX`*
  - Uses **ip route add** syntax
- Dynamic routes learned via daemon(s)
  - **quagga**
  - Support for various forms of `RIP`, `OSPF`, and `BGP`




# Verify IP Connectivity

- **ping**
  - Network packet loss and latency measurement tool
- **traceroute**
  - Displays network path to a destination
- **mtr**



# Defining the Local Host Name

- View/Set local hostname with **hostname**
- Initially defined in `/etc/sysconfig/network`:
  - `HOSTNAME=`*`stationX.example.com`*
- Might "pull" name from network
  - **dhclient** daemon
  - "Reverse DNS Lookup"

# Local Resolver

- Resolver performs forward and reverse lookups
- `/etc/hosts`
  - Local database of hostname to IP address mappings
  - Useful for small isolated networks
  - Normally, checked before DNS




# Remote Resolvers

- `/etc/resolv.conf`
  - Domains to search
  - Strict order of name servers to use
  - May be updated by **dhclient**
- `/etc/nsswitch.conf`
  - Precendence of DNS versus `/etc/hosts`

# Verify DNS Connectivity

- Verify name servers using:
  - **nslookup** (deprecated)
  - **host**
  - **dig**

# Network Configuration Utilities

- **system-config-network**
  - **system-config-network-gui**
  - **system-config-network-tui**
- Profile Selection
  - **system-config-network-cmd**
  - `netprofile` kernel argument

# Transparent Dynamic Configuration

- **NetworkManager** service
- **nm-applet**

# Implementing IPv6

- Kernel **ipv6** module enables stateless autoconfiguration
- Additional configuration implemented by **/etc/rc.d/init.d/network** initialization script
  - `NETWORKING_IPV6=yes` in `/etc/sysconfig/network`
  - `IPV6INIT=yes` in `/etc/sysconfig/network-scripts/ifcfg-`*`ethX`*

# IPv6: Dynamic Interface Configuration

- Two ways to dynamically configure IPv6 addresses:
  - Router Advertisement Daemon
    - Runs on (Linux) Default Gateway - **radvd**
    - Only specifies prefix and default gateway
    - Enabled with `IPV6_AUTOCONF=yes`
    - Interface ID automatically generated based on the MAC address of the system
  - DHCP version 6
    - **dhcp6s** supports more configuration options
    - Enabled with `DHCPV6C=yes`



# IPv6: StaticInterface Configuration

- `/etc/sysconfig/network-scripts/ifcfg-`*`ethX`*
  - `IPV6ADDR=<ipv6_address>[/prefix_length]`
  - Device aliases unnecessary...
  - `IPV6ADDR_SECONDARIES=<ipv6_address>[/prefix_length] [...]`

# IPv6: Routing Configuration

- Default Gateway
  - Dynamically from **radvd** or **dhcpv6s**
  - Manually specified in `/etc/sysconfig/network`
    - `IPV6_DEFAULTGW=<IPv6_address[%interface]>`
    - `IPV6_DEFAULTDEV=<interface>` - only valid on point-to-point interfaces
- Static Routes
  - Defined per interface in `/etc/sysconfig/network-scripts/route6-`*`ethX`*
    - Uses **ip -6 route add** syntax
    - `<ipv6_network/prefix> via <ipv6_routeaddress>`

# New and Modified Utilities

- ping6
- traceroute6
- tracepath6
- ip -6
- host -t AAAA *hostname6.domain6*



# End of Unit 8

- Questions and Answers
- Summary
  - Where are drivers linked to specific interfaces?
  - Where is a static IP address defined?
  - Where is the default route set?
  - Where is the list of nameservers stored?

# Unit 9

# Installation





# Objectives

Upon completion of this unit, you should be able to:

- Know important command line switches
- Understand different installation methods
- Create advanced partition layouts
- Understand Kickstart's role
- Create Kickstart files

# Anaconda, the Red Hat Enterprise Linux Installer

- Supports different modes
  - Kickstart offers automated Installation
  - Upgrade performs an update of an existing Red Hat Enterprise Linux installation
  - Rescue Mode allows troubleshooting of unbootable systems
- Consists of two stages:
  - First stage starts the installation
  - Second stage performs the installation

# First Stage: Starting the Installation

- The first stage consists of a installation kernel and an `initrd.img`
- Can be started with any supported boot loader
- Tasks of the First Stage:
  - Initializes the Installer
  - Parses command line arguments
  - Autodetects hardware
  - Loads additional drivers
  - Selects language, keyboard layout and installation method
  - Sets up networking if required for installation

# First Stage: Boot Media

- Supported boot media:
  - `boot.iso` or Installation CD/DVD
  - USB drive containing `bootimg.img`
  - Network boot with PXE
  - Other bootloaders such as GRUB
  - Boot floppies no longer supported
- Boot media can modified for custom installations

# Accessing the Installer

- Graphical Installation
  - Default installation type
  - Useful Switches: `lowres`,`resolution`, `skipddc`
- VNC based Installation
  - Activate with **vnc** and protect the session with **vncpassword=*password***
  - Set network parameters with **ip=*IP Address*** and **netmask=*Network Mask***
- Text based Installation
  - Started with the **text** switch
  - Menu-based terminal interface
- Serial Installation
  - Used automatically when no graphic card is detected
  - Enable with: **serial=*device***

# First Stage: Installation Method

- Available Installation Methods:
  - Local CDROM
  - Hard drive
  - NFS image
  - FTP
  - HTTP
- Media sets:
  - Two Sets available: Client and Server
  - Can be downloaded from Red Hat Network
  - May contain packages from additional layered products
  - An "Installation Key" must be entered to unlock additional content
  - Extra packages can also be installed after installation through RHN.

# Network Installation Server

- Necessary for network-based installs
- Often faster than CDROM-based installation methods
- Provides an easy distribution platform for the enterprise
- Shares the RedHat directory via NFS, FTP and/or HTTP
- Can be used as a **yum** repository

# Second Stage: Installation Overview

- Language and keyboard selection
- Installation Key
- Disk partitioning
- Bootloader configuration
- Network and time zone configuration
- Package selection

# Configuring File Systems

- Must select mount points, partition sizes, and file system types in the installer
  - Can set up manually or automatically
- There are many layouts which may be used
  - `/` must include `/etc`, `/lib`, `/bin`, `/sbin`
  - Swap space is typically 2x physical RAM
  - Typical mount points: `/boot`, `/home`, `/usr`, `/var`, `/tmp`, `/usr/local`, `/opt`

# Advanced Partitioning

- Software RAID
  - Create new partitions and select Software RAID as "filesystem" type
  - Combine RAID partitions into a RAID device with RAID
- LVM
  - Select Physical Volume to create physical volumes
  - LVM creates a Volume Group
  - Add creates new Logical Volumes

# Package Selection

- A default set of packages is automatically installed
- Select Customize now to change the default set of packages
- Customizing is necessary to add support for additional languages
- Anaconda automatically resolves package dependencies
- Package set can easily customized after install with **yum** or **system-config-packages**

# First Boot: Post-Install Configuration

- Configure X Window System if necessary
- Firewall and SELinux Setup
- Kdump setup
- Set date and time
- Register with Red Hat Network and get updated RPMs
- Setup users
- Configure sound card
- Install additional RPMs or Red Hat documentation from CDROM



# Kickstart

- Scripted Installation method
- Supports all Anaconda features
- `/root/anaconda-ks.cfg` is automatically created during Install
- Configuration utility: **system-config-kickstart**
- Syntax-checker: **ksvalidator**

# Starting a Kickstart Installation

- Anaconda enters Kickstart mode, when the `ks` boot option is specified
- `ks` queries DHCP for the Kickstart location
- `ks=`*`url`* gets the file via HTTP, FTP, or NFS
- From a local medium: `ks=floppy`, `ks=cdrom`, or `ks=hd:`*`device`*`:/path/to/file`



# Anatomy of a Kickstart File

- Commands section
  - Configures the system
  - Ommited directives are prompted to the user
- `%packages` Section
  - Selects packages and groups for installation
  - Dependcies are always resolved
- Scripts section(s)
  - Optional section to customize the system
  - `%pre` scripts are run before installation
  - `%post` scripts are run after installation



# Kickstart: Commands Section
## Starting the Installation

- Installation Mode
  - `install` performs a fresh install.
  - `upgrade` upgrades an existing installation.
- Installation Method:

```
cdrom
url --url url
nfs --server host --path directory
harddrive --partition=device --dir=/path/to/install_tree
```

# Kickstart: Commands section
## Important Directives

- Required Directives
  - Must be specified, otherwise the installer configures them interactively
  - Localization options: `keyboard`, `lang`, `timezone`
  - Authentication: `rootpw`, `authconfig`
  - Bootloader: `bootloader`
- Optional Directives
  - Network: `network [options]`
  - Security: `firewall`, `selinux`, `services`
  - Installer behaviour: `firstboot`, `poweroff|reboot`, `interactive`, `text`

# Kickstart: Packages Section

- Add single packages with `package_name` without any version number
- Add package groups with `@package_group`
- Remove packages from the list: `-package_name`
- Use wildcards to specify multiple packages
- Dependencies are always resolved
- Add support for additional languages with `@lang-support`
- Packages from layered products can be installed when an installation key is specified by with the `key` directive in the commands section.

# Kickstart: %pre, %post

- %pre gives you the first word
  - executes as a bash shell script
  - executes after Kickstart file is parsed
- %post gives you the final word
  - Can specify interpreter (bash is default)
  - chrooted by default, but may be run without chroot



# End of Unit 9

- Questions and Answers
- Summary
  - Steps of the installation
  - Important Anaconda switches
  - **system-config-kickstart**
  - **ksvalidator**

# Unit 10

# Virtualization with Xen





# Objectives

Upon completion of this unit, you should be able to:

- Define Virtualization
- Understand Xen Terminology
- Xen Tools

# Virtualization with Xen

- Advantages of Virtualization
  - Effective resource usage
  - Managability
  - Security
- Key Concepts of Xen
  - Small Hypervisor
  - First "Domain" manages the system
  - supports full and para virtualization



# Hardware Considerations

- Minimum Requirements:
  - Processor with PAE support
  - 256 MiB RAM per domain
  - 6 GiB hard drive per domain
- Additional Considerations:
  - CPU with VT/SVM for Full Virtualization
  - Shared Storage for Live Migration
  - Actual Storage needs will vary by application



# Preparing Domain-0

- Install the Domain0 as normal
- Boot the Xen hypervisor
- Start the **xend** management daemon



# Virtual Resources

- CPU
  - uses VCPUs (Virtual CPUs)
  - need not map directly to real CPUs
- Storage
  - Block devices
  - Simple files
- Network Devices
  - Bridged or routed to Domain0
  - By default mapped to `xenbr0`

# Domain-U Configuration

- Defined Per Domain-U
- Virtual Block Devices
- CPUs
- Networking
- `/etc/xen/`*`domain`*



# Installing a new Domain-U

- **virt-manager**
  - Graphical frontend for managing domains
  - Provides a wizard for setting up new domains
  - Command line alternative: **xm**
- Define name of the domain
- Select storage type and number of CPUs
- Specify the location of the installer and optionally a kickstart file

# Domain Management with xm

- Command line management tool
- Controlling domains
  - **xm <create|destroy>**
  - **xm <pause|unpause>**
  - **xm <save|restore>** ***filename***
  - **xm <shutdown|reboot>**
- Monitoring
  - **xm list**
  - **xentop**
  - **xen console**

# Activating Domains on boot

- **xendomains** Sys-V init script
- Starts/stops Domain-Us
- must link domain config files to `/etc/xen/auto`

# End of Unit 10

- Questions and Answers
- Summary
  - Xen Terminology
  - xm commands
  - xendomains

# Unit 11

# Troubleshooting



# Objectives

Upon completion of this unit, you should be able to:

- Develop a strategy for troubleshooting
- Fix problems in different areas of the Linux system
- Boot the system into various runlevels
- Use the Rescue environment

# Method of Fault Analysis

- Characterize the problem
- Reproduce the problem
- Find further information
- Eliminate possible causes
- Try the easy things first
- Backup config files before changing

# Fault Analysis: Gathering Data

- Useful commands
  - **history**
  - **grep**
  - **diff**
  - **find -cmin -60**
  - **strace *command***
  - **tail -f *logfile***
- Generate additional information
  - `*.debug` in syslog
  - **--debug** option in application



# Things to Check: X

- Never debug X while in runlevel 5!
- Try **system-config-display** first
- **X -probeonly**
- Is /home or /tmp full, or has the user reached a hard quota?
- Is **xfs** running?

# Things to Check: Networking

- Hostname resolution
  - **dig www.redhat.com**
- IP configuration
  - **ifconfig**
- Default gateway
  - **route -n**
- Module specification
- Device activation

# Order of the Boot Process

- Bootloader configuration
- Kernel
- **/sbin/init**
  - Starting init
- **/etc/rc.d/rc.sysinit**
- **/etc/rc.d/rc**, `/etc/rc.d/rc?.d/`
  - Entering runlevel X
- **/etc/rc.d/rc.local**
- X

# Filesystem Corruption

- Common after crash or improper shutdown
- ext2 mounted for writing marked "dirty"
  - If not mounted or mounted read-only, "clean"
  - if not mounted and "dirty", may be corrupted
  - repair requires exhaustive check
- ext3 usually marked "clean"
  - journal indicates if recovery is needed
  - only need to check files recorded in journal

# Filesystem Recovery

- If / has journal, kernel examines it at boot
- **/etc/rc.d/rc.sysinit** runs **fsck** on filesystems marked in /etc/fstab
- fsck is a front end to other programs
- A failed fsck must be run manually

# Recovery Run-levels

- Pass run-level to init
  - on boot from GRUB splash screen
  - from shell prompt using: **init** or **telinit**
- Runlevel 1
  - Process `rc.sysinit` and `rc1.d` scripts
- Runlevel s, S, or single
  - Process only `rc.sysinit`
- emergency
  - Run **sulogin** only

# Rescue Environment

- Required when root filesystem is unavailable
- Non-system specific
- Boot from CDROM (boot.iso or CD #1)
- Boot from diskboot.img on USB key

# Rescue Environment Utilities

- Disk Maintenance Utilities
- Networking Utilities
- Miscellaneous Utilities
- Logging: `/tmp/syslog` or `/tmp/anaconda.log`

# Rescue Environment Details

- Filesystem reconstruction
  - Anaconda will ask if filesystems should be mounted
  - `/mnt/sysimage/*`
  - `/mnt/source`
  - `$PATH` includes hard drive's directories
- Filesystem nodes
  - System-specific device files provided
  - **mknod** knows major/minor #'s

# End of Unit 11

- Questions and Answers
- Summary
  - What are some things to check for
  - X problems?
  - Services problems?
  - Networking problems?
  - Boot problems?
  - How might you repair an ext2 filesystem?
  - What are some alternate boot methods?