RHCE
BOOT CAMP

A whole week of geeky fun!

redhat®

CERTIFIED
ENGINEER

Rev: 2012-11-28

ABOUT THE INSTRUCTOR

- Nathan Isburgh
  - instructor@edgecloud.com
- Unix user 15+ years, teaching it 10+ years
- RHCE, CISSP
- Forgetful, goofy, patient :)

ABOUT THE CLASS

- 40 hour boot-camp style prep course
- Monday-Thursday: Lecture and labs
- Friday: Practice exam
- Hours:
  - 8:30am - 5:00pm
  - Lunch: 11:45am - 1:00pm

ABOUT THE COLLEGE

- Rackspace Parking Sticker = good to go
- Breakroom downstairs - labeled “Laundry”
- Sodas - bottles in machine ($1.25) or cans in mini-fridge ($0.50)
- Cafeteria
- Do not speed!
- No smoking anywhere.

ABOUT THE STUDENTS

- Name?
- Time served, I mean employed, at Rackspace?
- Department?
- Unix skill level?
- First attempt at RHCE?
- What are you most worried/interested about with RHEL?

EXPECTATIONS OF STUDENTS

- Strong foundation in basic Linux use and administration
- Ask Questions!
- Complete the labs
- Email if you're going to be late/miss class
- Have fun
- Learn something
- Pass your exam!

ABOUT RHCE EXAM

- The RHCE exam is 100% hands-on. You will have a computer and a list of tasks to accomplish. When you are finished, an automatic grader will check your machine to be sure that everything is set up correctly.
- There are no questions, only tasks.
- You will have 2.0 hours and access to all RHEL 6 Server software.
- You will not have internet access.

RHCE CERTIFICATION

- Note that RHCE certification is not granted until both RHCSA and RHCE on RHEL 6 have been passed.
- RHCSA = local machine settings, basic network access
- RHCE = network services

TO PASS EXAM:

- Details specific to RHEL v. 6
- Basic System Administration
- Configuration of all major services / daemons
- Implementation of basic security / traffic filtering technologies
- **Locating Local Reference Materials (--help, man)**

```
slideshow.end();
```

RHCE
BOOT CAMP

The Boot Process

OVERVIEW

- The boot process gets a machine from the useless off state to the feature rich operating system we all know and love

- Requires cooperation between hardware and software to correctly hand off processing

- Akin to the life cycle of a human - birth, newborn, infant, toddler, teen, adult

BIRTH

- Power switch flipped on
- Electricity flows from wall, through power supply where it gets converted to the levels necessary for the computer, and on to the motherboard, drives, CPU and more
- Completely unaware of the world or even what's attached to the motherboard.

INFANT

- BIOS - Basic Input/Output System - CPU looks for instructions starting at a specific address, which happens to be where BIOS resides. BIOS initializes and starts the....

- POST - Power On Self Test - A simple set of tests that BIOS performs to verify basic functioning of attached hardware.

- Like an infant, extremely limited understanding of world

- Searches for valid MBR, loads the software found there and transfers control to the...

TODDLER

- Boot Loader - Special software installed to the MBR of the boot partition which selects and loads the kernel.
- Can be configured to immediately load the default OS, or can offer choice to user
- Slightly better understanding of world - can read linux filesystems, sometimes includes powerful debugging and configuration support.
- Main job: select and load kernel, transfer control to kernel

TEENAGER

- Dreaded teenager age: knows a lot about the world, but doesn't contribute a thing. Still pretty useless.
- Kernel loads and initializes. Device drivers are loaded and initialized. Basic hardware checks performed.
- The First Process is *created from nothing*: `init`

ADULT

- init loads the inittab, specifying what the default runlevel should be, then additional configuration files specify what software needs to be started. init starts running all of the specified startup scripts at this point.

- Services are started by init, including network configurations, X Windows, network services, databases, etc.

- At this point, the machine is finally becoming useful: otherwise, an adult

- Eventually, login processes are started and the boot process is complete!

MORE ON INIT

- RHEL 6 marks Red Hat's departure from the old style SystemV initialization framework. Time to [mostly] forget about inittab!

- RHEL 6 now uses Upstart to handle startup, shutdown and service management.

  - `http://upstart.ubuntu.com`

- The only configuration `/etc/inittab` provides anymore is what the default runlevel should be, as Upstart supports the notion of runlevels to ease transition from SysV style initialization to Upstart.

UPSTART

- The configuration files for Upstart are under:
  - `/etc/init`
- Files in this directory detail configuration for certain global events, like ctrl-alt-delete, as well as maintaining TTY gettys, handling runlevels and more.
- A runlevel defines what services are running...

RUNLEVELS

- Runlevels:
  - S: System startup
  - 0: OS stopped, machine halted (usually powers off as well)
  - 1: Single user mode - for maintenance
  - 2: Multiuser, no NFS shares
  - 3: Full multiuser, TUI
  - 4: Unused
  - 5: Full multiuser, GUI
  - 6: Reboot

RUNLEVELS

- `telinit`: Signal the init process to change the current runlevel

- Switching runlevels is fairly uncommon - generally only used if system maintenance needs to be performed

- Runlevels can be used to control what services a machine provides, and can sometimes be useful to quickly reconfigure a machine for a new task

UPSTART OVERVIEW

- So the basic flow of operation for Upstart is as follows:
  - At bootup, the kernel starts `/sbin/init`. After `/sbin/init` loads configuration files and is ready, the first event is emitted: **`startup`**
  - The **`startup`** event causes `/etc/init/rcS.conf` to fire, which in turn runs the familiar `/etc/rc.d/rc.sysinit`. After `rc.sysinit` finishes, `rcS.conf` uses `/etc/inittab` to determine the default runlevel, then runs `telinit` to that runlevel.

UPSTART OVERVIEW

- `telinit` emits the **`runlevel`** event, which fires off `/etc/init/rc.conf`

- `rc.conf` fires off the familiar `/etc/rc.d/rc` script with the appropriate runlevel to fire off all of the startup scripts in the appropriate `/etc/rcX.d` directory.

- **WHEW!**

- All of this, mainly so that the transition to Upstart is relatively painless for the system administrators more comfortable with SysV initialization.

INIT SCRIPTS

- What is actually running in a given runlevel is defined by the init scripts for that level.
- That standard location for the init scripts is:
  - `/etc/rcX.d`
  - Where the `X` corresponds to the runlevel
- For example, `/etc/rc5.d` contains all of the init scripts that, combined, provide runlevel 5 service

RC DIRECTORIES

- The files in the rc directories start with either an S or a K:
  - S means to start the service, ie run the command with "start" as an argument
  - K means to kill the service, ie run the command with "stop" as an argument
- After the S or K, there is a two digit number which is used for ordering the execution of the scripts

ENTERING A RUNLEVEL

- So when the init process “enters” a runlevel, the steps are:
  - Run all of the Kill scripts, in order, with “stop” as an argument
  - Run all of the Start scripts, in order, with “start” as an argument

INIT SCRIPTS

- If you look closely, you will see that `/etc/rcX.d` actually holds a collection of symbolic links
- The actual script files are stored in `/etc/init.d`
- The main reason for this is so that there is only one copy of each init script, reducing the chance that a script change won't be reflected in all runlevels.
- You can run the scripts directly, or use the `service` command to start/stop various components of the OS.

MANAGING INIT SCRIPTS

- You can manage the links to the init scripts manually, or you can use the `chkconfig` command to get the job done:

- **`chkconfig --list`**

  - List all processes and display their default status in each run-level.

- **`chkconfig [--level levels] name <on|off|reset>`**

  - This command will modify the chkconfig configuration for a particular service, setting it on/off for the given runlevels.

GRUB

- Grand Unified Boot Loader
- Recall that GRUB is responsible for the initial kernel load at boot time.
- Using GRUB, an administrator can control what kernel is loaded, what options are passed to the kernel, as well as initial ramdisk configurations.

GRUB CONFIGURATION

- GRUB's configuration file is `/boot/grub/grub.conf`, which is configured as follows:

```
default=0

timeout=10

splashimage=(hd0,0)/grub/splash.xpm.gz

title RedHat Enterprise Linux

    root (hd0,0)

    kernel /vmlinuz ro root=LABEL=/

    initrd /initrd
```

GRUB SHELL

- Command mode – Pressing “c” while the boot menu is displayed will provide the user with the GRUB shell, where a limited set of commands can be used to explore the filesystem before booting. A full list of the commands available can be found by pressing Tab while in command mode.

- Editing mode – Pressing “e” while the boot menu is displayed will provide the user with the opportunity to edit a line in GRUB’s configuration file.

- Append mode – Pressing “a” while the boot menu is displayed will allow the user to append to the kernel line for the default kernel in GRUB’s configuration file

- Esc – can be pressed at any time to return you to the previous menu

BOOTING TO A GIVEN RUNLEVEL

- Using GRUB, add a number to the end of the kernel command line to override the default runlevel.

- Also, adding the letter “**`s`**” or the word “**`single`**” to the end of the command line is very important: this boots into single user mode, which by default, will not require a password to obtain a root shell.

- Very important!

LAB

1. Reboot your machine into the single user runlevel and verify root access without a password.

2. Review a few of the `init.d` scripts

3. Review the configuration files in `/etc/init`

```
slideshow.end();
```

RHCE
BOOT CAMP

Package Management

RPM

- Redhat Package Manager
- RPM's provide full software packaging features: pre-install scripts, post-install scripts, dependencies, meta information, and an installed software database to name a few.
- The RPM system maintains a database of all installed software on a machine - this is useful for tracking and updating reasons, as well as dependency verification and software management.

RPM

- `rpm`: The Redhat Package Manager tool. Provides interface to RPM system, performing queries, installs, upgrades, uninstalls and general database maintenance operations.
  - `-i` option: install the given package
  - `-q` option: query the database
  - `-e` option: erase the given package from the system

RPM QUERIES

- Below are just a few examples of the types of queries you can run against the RPM database.
  - **`rpm -qa`** Queries for the names of all installed rpms.
  - **`rpm -qi`** Queries the rpm database for package information.
  - **`rpm -qf`** Determines which rpm a file is associated with.
  - **`rpm -ql`** Queries the rpm database to determine which files are associated with a particular rpm.
- With any of these commands, you can add the `-p` option to run the command against a package before it is installed.

RPM INSTALLATION VERIFICATION

- In addition to storing information about where a package is installed, `rpm` also stores permissions, file sizes, md5sums, and ownership information. This information can be easily referenced to see if anything has been changed.
  - **`rpm -Va`** Verifies all installed packages.
  - **`rpm –Vi <package>`** Verifies given package.
- Rackspace Best Practice Example
  - `rpm –Va | grep ^..5`

RPM VERIFY OUTPUT

- **S** File Size differs
- **M** Mode differs (includes permissions and file type)
- **5** MD5 sum differs
- **D** Device major/minor number mismatch
- **L** readLink(2) path mismatch
- **U** User ownership differs
- **G** Group ownership differs
- **T** mTime differs
- **C** SELinux Context differs

EXTRACT RPM CONTENTS

- Use this technique to make a clean working copy of the files and directories that would be installed with a package.
  - `cd /temp/dir`
  - `rpm2cpio /path/to/package | cpio -i –d –m`
- This would allow you to:
  - Replace one corrupted file without un-installing and then re-installing a package
  - Compare original configuration files versus modified files in the running system to quickly locate changed lines, for example with the ‘diff’ utility

YUM

- yum: Yellowdog Updater Modified
  - Supports package installation over the network through repositories.
  - RPM backend
  - Simple interface

REPOSITORIES

- Repositories of packages must be listed in files in the `/etc/yum.repos.d` directory with names ending in `.repo` and having a format like:
  - `[label-for-repo]`
  - `name = descriptive text`
  - `baseurl = protocol://path/to/directory/of/packages`
- Access to the Red Hat Network, including any Satellite Servers, is implemented through a plugin to the yum tool itself and not as a repository definition in the above format.

LAB

1. Connect to http://server1.example.com and read the information there.

2. Download the OpenOffice archive from `server1` and choose an appropriate location to extract all its RPMs

3. Install the `createrepo` package and use it to to turn your collection of OpenOffice packages into a yum repository

4. Add that repository to your local yum configuration

5. Using yum, install the “openoffice.org3-writer” package, and/or any others from your new repository

```
slideshow.end();
```

RHCE
BOOT CAMP

System Administration

redhat®

CERTIFIED
ENGINEER

INSTALLATION

- Installing RHEL 6 is a straightforward process when performed interactively. I expect every single person in here can install RHEL 6 from media.

- Unattended install using a Kickstart file is another matter entirely, though.

KICKSTART FILES

- Fortunately, Kickstart files are *extremely simple* to understand and create.
- A Kickstart file is a flat text file which answers all of the installation questions automatically. Therefore, logically, it contains details on:
  - Partitioning and filesystems
  - Software packages
  - Users, Groups, Passwords
  - Features, networking and more

KICKSTART FILES

- There are three primary means of creating a Kickstart file:
  - From scratch
  - From an existing Kickstart file (perhaps from a recent install?)
  - Using `system-config-kickstart`

LAB

1. Examine `/root/anaconda-ks.cfg`

2. Install and run `system-config-kickstart` and create a simple kickstart file to install a basic desktop RHEL 6 machine.

NETWORK CONFIGURATION

- There are two main approaches to configuring a machine for network access:
  - Static configuration
  - Dynamic configuration
- Static configuration uses set parameters for the configuration, which is known by the machine and the network and never changes. Generally used with servers.
- Dynamic configuration configures network machines on the fly, where a service on the network provides all configuration parameters to a machine when it joins the network. Generally used with workstations.

DYNAMIC CONFIGURATION

- Dynamic configuration is the easiest to use.
- The machine just needs to set up it's interfaces with the DHCP protocol.
- DHCP: Dynamic Host Configuration Protocol.
- A lease is obtained from the DHCP server, providing all network configuration details for the client. The lease expires after some amount of time and is renewed by the client to maintain network access.

STATIC CONFIGURATION

- Static configuration requires four configuration parameters in order to allow full network functionality:
  - IP Address
  - Netmask
  - Default Gateway or Router
  - DNS Server(s)

DNS?

- Domain Name Service: This is the glue between network names and IP addresses.

- Remember: Humans like names, computers like numbers. DNS is a service like so many others, mapping names to numbers and numbers to names. Mostly a convenience.

- Also provides for email functionality, geographic load balancing and limited service failover capabilities.

STATIC CONFIGURATION

- The first two components of static configuration are IP address and netmask.
- These provide LAN-level access
- To view current address:
  - `ip addr list`

GATEWAYS

- The third configuration parameter is the default gateway.
- Provides access to *inter-networking*, or moving from just the local LAN to other LAN's
- To see the current routing entries:
  - `ip route show`

DNS SERVERS

- Final piece of configuration information.
- List of one or more IP addresses which provide the DNS service, allowing name to IP address mapping
- To view current nameservers, see:
  - `/etc/resolv.conf`
- Also consider `/etc/nsswitch.conf`

STATIC CONFIGURATION

- Once all four pieces of information are configured on the system, full network service will be available.
- To test local connectivity, try pinging the gateway
- To test inter-networking connectivity, try pinging `8.8.8.8` or some other external IP address.
- To test name resolution, try pinging `google.com` or another public DNS name.

CHANGING NETWORKING

- To change the IP address, hostname, netmask and gateway, you have to edit two configuration files:
  - `/etc/sysconfig/network-scripts/ifcfg-em1`
  - `/etc/sysconfig/network`

/ETC/SYSCONFIG/NETWORK

```
NETWORKING={yes|no}

HOSTNAME=<fqdn>

NISDOMAIN=<nis domain name>
```

IFCFG-* FILES

- To configure a device to use dhcp, the ifcfg file should contain the following:

```
DEVICE=em1

BOOTPROTO=dhcp

ONBOOT=yes
```

IFCFG-* FILES

- To configure a device with static settings, the ifcfg file should contain the following:

```
DEVICE=em1

BOOTPROTO=none

IPADDR=<ip>

NETMASK=<netmask> ( or PREFIX=<net bits> )

ONBOOT=yes

GATEWAY=<gateway ip>

DNS1=<nameserver ip>

DOMAIN=<search domain>
```

NETWORK MANAGER

- In RHEL 6, Network interfaces are now handled via Network Manager. Some notable commands/tools:
  - `nmcli` - simple CLI to Network Manager
  - `nm-connection-editor` - excellent GUI tool for managing all network connections.
- On the test, you should decide if you are going to use Network Manager or not, and if so, only use NM and don't edit the config files by hand. Otherwise, disable NM and edit the files by hand. Your choice!

NAT CONFIGURATION

- NAT Configuration, `eth0` outside, `eth1` inside:

```
sysctl -w net.ipv4.ip_forward=1 >> /etc/sysctl.conf

iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

iptables -A FORWARD -i eth0 -o eth1 -m state --state
RELATED,ESTABLISHED -j ACCEPT

iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT

service iptables save
```

STATIC ROUTES

- Static routes are configured via:
  - `/etc/sysconfig/network-scripts/route-eth0`
    - `192.168.0.0/24 via 192.168.0.1 dev eth0`
- `/etc/sysconfig/network-scripts/route-eth1`
  - `10.10.10.0/24 via 10.10.10.1 dev eth1`

LAB

1. Determine your current network settings (which were assigned by DHCP) and change your machine to a static network configuration using these settings.

2. When you are satisfied with your configuration, restart the network service to put your changes into effect.

3. Test your connectivity to `server1` to make sure you are still online.

4. Refer back to DHCP settings if necessary to correct any mistakes in your static configuration.

5. Once complete, switch everything back to DHCP.

CRON

- `crond` is the cron daemon. Cron provides for the ability to execute commands on a regular basis.

- Generally used to run hourly, daily and weekly type system maintenance scripts.

- Also useful to run reports, cleanup jobs and much, much more.

SYSTEM CRONS

- `/etc/crontab` and `/etc/cron.d/*` define the system cron jobs.
- `/etc/anacrontab` defines system cron jobs that are run even if the machine was not running when the job normally executes.
- Many distributions use the `run-parts` script to execute all scripts found in `/etc/cron.hourly`, `/etc/cron.daily`, etc on the appropriate schedule.
  - `/etc/anacrontab` defines the times for each schedule: daily, weekly, monthly
  - Due to limitations in `anacrontab`, the hourly scripts are configured to run via `/etc/cron.d/0hourly`

USING CRON

- Cron is controlled through crontab files.
- There are system-wide crons as discussed previously.
- Every user has their own crontab, accessible through the `crontab` command

CRONTAB

- `crontab`: View, edit or remove crontabs
  - The `-l` option prints the crontab. The `-e` option opens the crontab for editing. The `-r` option removes the crontab.
- Root can work with the crontab for any user by specifying the username on the command line:
  - `crontab -e -u bob`

CRONTAB SYNTAX

- There are two main components to a crontab entry:
  - The timespec specifies when the command should be run
  - The command is what gets executed every time the timespec is matched

CRONTAB TIMESPECS

- The timespec is broken down into 5 fields, separated by spaces:
  - `minute hour day-of-month month day-of-week`
- Each field can contain a number, a range of numbers, a comma-separated list of numbers, an asterisk or a number slash division rate
- Mostly self-explanatory - some examples will help...

TIMESPEC EXAMPLES

- `0 23 * * *` *11pm every day*
- `30 * * * 1-5` *30 minutes after every hour, M-F*
- `0 7 1 * *` *7am, first of every month*
- `* * * * *` *Every single minute*
- `0,10,20,30,40,50 * * * *` *Every 10 minutes*
- `*/5 8-17 * * 1-5` *Every 5 minutes, 8am-5pm, M-F*

EXAMPLE CRONTAB

```
01 4 * * * /usr/local/bin/restart-webserver
00 8 1 * * /usr/bin/mail-report boss@mycompany.com
*/5 * * * * /monitor/bin/check-site -e admin@mycompany.com -o /var/log/check.log
```

- There are various additional options and features available to the cron system. Check the man pages for reference:
  - `cron`, `crontab` (sections 1 and 5)

LAB

1. Create a cronjob for the user root that checks the amount of available space on the system every Friday at 12:34pm.

2. Create a cronjob as a regular user that lists the contents of `/tmp` at 3:54am on Sunday, January 2. Hint: not possible with just cron syntax - you will have to do a tiny amount of scripting to complete this one.

LOGS

- One of the easiest places to find the cause of a problem is in the log files.

- Log files store informational messages from software. The types of messages include debug information, status information, warnings, errors and more.

- Some applications manage their own log files. Others use the system-wide logging package...

SYSLOG

- `rsyslog` - The system logger. A framework consisting of a library, a daemon, a configuration file and logs.
- Any application can use the library and log messages through `rsyslog` with simple function calls.
- Log messages consist of 3 parts:
  - Facility
  - Level
  - Message

SYSLOG

- The facility describes what part of the operating system generated the message, and is selected by the software:
  - `auth, authpriv, cron, daemon, ftp, kern, lpr, mail, news, security, syslog, user, uucp, local0-local7`
- The level represents the importance of the message, and is also chosen by the software:
  - `emergency, alert, critical, error, warning, notice, info, debug`

/ETC/RSYSLOG.CONF

- `/etc/rsyslog.conf` defines where all of the log messages should go. Destinations include files, screens of logged in users, console, other syslog servers. Additional configuration is available as well.

- Basic rule format:

  - `facility.level  destination`

- Examples:

  - `*.err  /dev/console`

  - `mail.*  /var/log/maillog`

  - `*.info;mail.none;authpriv.none  /var/log/messages`

/VAR/LOG

- `maillog`: messages from the email subsystem
- `secure`: authentication and security messages
- `cron`: cron messages
- `boot.log`: boot messages
- `messages`: catch-all
- `dmesg` : hardware and kernel events generated before syslogd

REMOTE LOGGING

- Setting up remote logging with rsyslog is trivial:
  - Make sure a hostname is set up on each machine
  - Make sure server firewall has holes for port 514 udp/tcp

REMOTE LOGGING SERVER

- On the server, add to `rsyslog.conf`:
  - `$ModLoad imudp.so`
  - `$UDPServerRun 514`
  - `$ModLoad imtcp.so`
  - `$InputTCPServerRun 514`
- Restart `rsyslogd`

REMOTE LOGGING CLIENT

- On the client, add to `rsyslog.conf`:
  - `*.*  @loghost.fqdn   # for udp`
  - `*.*  @@loghost.fqdn  # for tcp`
- Restart `rsyslogd`
- Consider using the Action Queue parameters to improve reliability. See bottom of `rsyslog.conf` for example.

LOGS

- As mentioned earlier, not all software uses the syslog framework to handle it's logging. Quite a bit of software manages it's own logs.

- This can make it difficult to track down all of the log locations on an unfamiliar system. The best way to handle this is to start from the init scripts...

LOCATING APPLICATION LOGS

- To track down the log file location for an application, you need to find it's configuration file so you can see where the logs are being written.

- Of course, finding the configuration file might be just as difficult, so it's best to start at the source.

- `init` starts all of the system services, and so there is an init script somewhere that is starting up the application in question.

- The init script almost always references the configuration file

LOCATING APPLICATION LOGS

- Now that the configuration file location is known, it only takes a few moments to scan through it and find out where logs are being written.

- As for the format of the log file, that's completely dependent on the application. Some will be similar to syslog, others, like Apache or Qmail, will be completely foreign looking.

- Fortunately, a little common sense and judicious application of Google Ointment will get the information you seek.

MAINTAINING LOGS

- `/etc/logrotate.conf`
  - This is the main configuration file for logrotate.
- `/etc/logrotate.d/`
  - EVERYTHING in this directory will be parsed as if it is a logrotate configuration file. Usually, applications such as Apache and Sendmail will have configuration files in this directory to control how their logs will be rotated.
- `logrotate -vf /etc/logrotate.conf`
  - Can be run as root at any time to force log rotation and check for errors.

TROUBLESHOOTING

- There will be some basic troubleshooting objectives on the exam, mostly to test basic knowledge of how permissions should work, SELinux and locating error messages in log files.

- Mentioned here are a few useful tools to remember

TOP

- `top`: Self-updating tool displays combination summary at top, followed by ordered list of processes. Fully customizable.
  - The summary includes uptime information, memory breakdowns, CPU utilization and process state summaries
- The process display can be customized and sorted to suit need

```
top - 16:39:32 up 682 days, 10:41,  2 users,  load average: 0.01, 0.00, 0.00
Tasks: 118 total,   1 running, 116 sleeping,   1 stopped,   0 zombie
Cpu(s):  0.1%us,  0.0%sy,  0.0%ni, 99.8%id,  0.0%wa,  0.0%hi,  0.0%si,  0.1%st
Mem:    262316k total,   258024k used,     4292k free,     7380k buffers
Swap:   524280k total,    74564k used,   449716k free,    67808k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
    1 root      15   0 10316  648  592 S    0  0.2   0:06.24 init
    2 root      RT   0     0    0    0 S    0  0.0   0:04.88 migration/0
    3 root      34  19     0    0    0 S    0  0.0   0:00.19 ksoftirqd/0
```

DF

- `df`: lists filesystem utilization
  - Breaks down size and use information for each mounted filesystem
  - -h is useful option to display in “human-friendly” format

```
[root@dev1 ~]# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda1             9.4G  7.2G  1.8G  81% /
none                  129M     0  129M   0% /dev/shm
[root@dev1 ~]#
```

LDD, LDCONFIG

- `ldd`: List library dependencies
- `ldconfig`: Update library location database
- `/etc/ld.so.conf` and `/etc/ld.so.conf.d/*.conf` for list of pathnames to search for libraries, creates database for dynamic linker

```
[root@dev1 ~]# ldd /bin/bash
    libtermcap.so.2 => /lib64/libtermcap.so.2 (0x00002ac044572000)
    libdl.so.2 => /lib64/libdl.so.2 (0x00002ac044775000)
    libc.so.6 => /lib64/libc.so.6 (0x00002ac044979000)
    /lib64/ld-linux-x86-64.so.2 (0x00002ac044357000)
[root@dev1 ~]# cat /etc/ld.so.conf.d/mysql-x86_64.conf
/usr/lib64/mysql
[root@dev1 ~]# ldconfig
[root@dev1 ~]#
```

NICE LEVEL

- The nice level represents one influence on the calculations the kernel uses when assigning priorities.
- Originally designed and named to allow users to be “nice” to other users of the system by assigning a higher nice value to an intensive process, which in turn lowers it’s priority.
- Ranges from -20 to 19. Default nice level is 0.
- Only root can assign negative nice values.
- See `nice` and `renice` commands

LAB

1. Take a few minutes to browse through the various logs in `/var/log`. Familiarize yourself with the kinds of information available.

2. Browse the man page for `rsyslog.conf`

3. Find where the audit service keeps its log and add a corresponding new entry to your logrotate configuration. Force a rotation to see everything work.

4. Remove the audit logrotate configuration and restart the auditd service.

5. Locate the PIDs of the highest memory and highest CPU utilization processes. Play with their nice levels.

6. Work with a neighbor to set up remote logging from your station to theirs, and theirs to yours. Verify using `logger`. Careful of log loops!

SHELL SCRIPTING

- Shell scripting involves placing a series of shell commands in a file for later re-use.
    - Simple shell scripts simply run command after command, as if the user typed them in at the command line
    - More complex shell scripts actually make decisions about what commands need to be run, and might even repeat certain sequences to accomplish some task
- Scripts start executing at the top and stop when there are no more commands to execute or when `exit` is called.

EXAMPLE SHELL SCRIPT

- Here is an example of a very simple shell script:

```
echo “Hello, what is your name?”
read NAME
echo “Hello $NAME, it’s nice to meet you!”
echo -n “The current time is: “
date
```

- Using the `echo` command, this script asks a question.
- The `read` command accepts input from the user and stores it in the environment variable `NAME`
- The script finishes up with a couple more echo statements, greeting the user and announcing today’s date

SHELL SCRIPTING

- If we put the example in a file called `myscript`, we can execute the script as:
  - `bash myscript`
- `bash` will open `myscript` and execute each line as if the user had typed it in manually.

```
[root@localhost ~]# bash myscript
Hello, what is your name?
Linus
Hello Linus, it's nice to meet you!
The current time is: Sun Nov 29 09:39:33 CST 2009
[root@localhost ~]#
```

INTERPRETERS

- In the previous example, we put five commands in a regular file and fed the filename to `bash` on the command line, which in turn executed the commands.

- Running in this way, `bash` operated as an interpreter. Reading each line of the file, `bash` would interpret the words and perform some action.

- There are many interpreted languages available for scripting, including all shells, `python`, `ruby`, `perl`, etc.

EXECUTING SCRIPTS

- To run a script, feed the file to the appropriate interpreter:
  - `bash mybashscript`
  - `perl myperlscript`
- This works fine, but sometimes it's more user-friendly to allow the script to be run directly, removing the need for an external call to the interpreter...
  - `./mybashscript`
  - `myperlscript`

SHEBANG

- This is accomplished with the shebang (#!). Also known as a hash bang, pound bang or hashpling.

- When the kernel is asked to execute a file, it must either be machine code, or a file that starts with the shebang sequence. If the first two characters of the file are a hash mark and an exclamation mark, the rest of the line is expected to be an absolute pathname for an interpreter, which will then be invoked to “run” the file as a script.

SHEBANG

- So, add an appropriate shebang to the example:

```
#!/bin/bash
echo “Hello, what is your name?”
read NAME
echo “Hello $NAME, it’s nice to meet you!”
echo -n “The current time is: “
date
```

- Then add execute permissions and the script can be run directly:

```
[root@localhost ~]# chmod 755 myscript
[root@localhost ~]# ./myscript
Hello, what is your name?
Linus
Hello Linus, it’s nice to meet you!
The current time is: Sun Nov 29 09:39:33 CST 2009
[root@localhost ~]#
```

DECISIONS

- More advanced problems require the script to make decisions. There are two basic ways to make decisions with shell scripts:
  - `if` statements
  - `case` statements

TEST COMMAND

- Before we continue talking about decisions, we need to talk about the `test` command. This command actually performs the comparisons necessary to ask a question, such as:
  - `"string1" = "string2"` *Returns true if string1 is identical to string2*
  - `VAR -le 45` *Returns true if VAR is numerically less than or equal to 45*
- See the man page for `test` for additional details

IF STATEMENTS

- Basic syntax:

```
if list;

  then list;

  [ elif list; then list; ]

  ...

  [ else list; ]

fi
```

IF EXAMPLE

```
#!/bin/bash
echo “Hello, what is your name?”
read NAME
if [ “$NAME” = “Linus” ]
then
  echo “Greetings, Creator!”
elif [ “$NAME” = “Bill” ]
then
  echo “Take your M$ elsewhere!”
  exit
else
  echo “Hello $NAME, it’s nice to meet you!”
fi
echo -n “The current time is: “
date
```

- This script will now base it’s response based on what name the user provides

CASE STATEMENTS

- Basic syntax:

```
case word in
  pattern [| pattern] ) list;;
  ...
esac
```

CASE EXAMPLE

```
#!/bin/bash
echo “Hello, what is your name?”
read NAME
case $NAME in
  “Linus” )
    echo “Greetings, Creator!”
    ;;
  “Bill” )
    echo “Take your M$ elsewhere!”
    exit
    ;;
  * )
    echo “Hello $NAME, it’s nice to meet you!”
esac
echo -n “The current time is: “
date
```

- This script also bases it’s response based on what name the user provides, but does so using a case statement instead of a large if statement

LOOPING

- Sometimes a certain sequence of commands need to be run repeatedly, either for a set number of times or while some condition is true. This is accomplished with:
  - `while` loops
  - `for` loops

WHILE LOOPS

- Basic syntax:

```
while list;

  do list;

done
```

WHILE EXAMPLE

```
#!/bin/bash
echo “Hello, what is your name?”
read NAME
while [ “$NAME” != “Linus” ]
do
  echo “I don’t know that person, what is your name?”
  read NAME
done
echo “Greetings, Creator!”
echo -n “The current time is: “
date
```

- This script will loop until the name typed is “Linus”

FOR LOOPS

- Basic syntax:

```
for (( expr1 ; expr2 ; expr3 ))

  do list;

done
```

FOR EXAMPLE

```
#!/bin/bash
echo “Hello, what is your name?”
read NAME
for (( I=0 ; I<3 ; I++ ))
do
  echo “Hello $NAME!!”
done
echo -n “The current time is: “
date
```

- This excitable script repeats your name 3 times before giving you the date and time

SCRIPTING

- There is of course quite a bit more to shell scripting than can be covered in this course. There are a few more structures you can use for looping, and dozens of special metacharacters for achieving all kinds of results.

- With this introduction, though, you should be able to read through light shell scripts and have a handle on what's going on, as well as be able to write simple ones on your own.

EXERCISES

- Write a simple shell script that prints out the message “Hello world.” Make the script executable and verify it works correctly by running it as “./myscript”

- Browse through the man page on ‘bash’, focusing in a bit on the various scripting elements.

- Devise a program which uses one or more **if** statements, changing behavior based on user input or command line arguments. Output a warning to stderr if there are no arguments passed to your script on the command line.

```
slideshow.end();
```

RHCE
BOOT CAMP

Filesystem Administration

PARTITIONING

- What is partitioning?
  - Splitting up a hard drive into organizable chunks
- Why?
  - Isolates filesystem corruption
  - Simplifies/speeds backups
  - Allows optimizing filesystems to tasks

FDISK

- `fdisk`: partitioning tool.
  - Works on one disk at a time, allows for viewing and manipulating partition table.
    - Online help (hit ‘m’) makes tool easy to use
- At boot, the kernel loads a copy of the partition table into memory. Most partition editing commands only update the partition table on the drive, and not in memory. As such, the command `partprobe` was run to update the information that the kernel has in memory. **Partprobe does not work in RHEL 6.**

MKFS

- `mkfs`: format a device to create a new filesystem
  - "Paints the parking stripes" for the filesystem structure
  - For ext*, creates superblock, block groups, superblock copies, bitmaps and inode tables and creates basic structure on disk
  - Through `-t` option, mkfs can create different types of filesystems

EXT2

- Benefits
  - Default file system for pre - 7.x versions of Red Hat
  - Heavily tested / Rock solid stability
- Drawbacks
  - Does not have a journal
  - File system check (fsck) required to mount a “dirty” file system
  - System offline and unavailable while fsck is running

EXT3

- Benefits
  - Default file system of the old 7.x Red Hat to RHEL 5.x releases
  - Based on proven stability of Ext2
  - Has journal for increased reliability
- Drawbacks
  - Inodes allocated when file system is created (other file systems create them dynamically as they are needed)
  - Not as efficient as other file systems when dealing with lots of small files

EXT4

- Benefits
  - Default file system of RHEL 6.x releases and newer
  - Built from a series of extensions to ext3
    - Many improvements over ext3, including larger scales, timestamps, performance and more
- Drawbacks
  - Inodes allocated when file system is created (other file systems create them dynamically as they are needed)
  - Delayed allocation can potentially lead to data loss (patches in place)

JOURNALING

- Journaling - How does it help?
- Deleting a file in Linux requires two steps:
  1. The file's directory entry must be removed.
  2. The file's inode must be marked as free in the free space map.
- If step 1 happens before a crash, an inode will be orphaned and the file will be lost.
- If step 2 happens first before a crash, the inode will be marked free and will possibly be overwritten.
- Journaling keeps a journal of the changes that are planned for the file system ahead of time. The journal can then replay the changes in the journal at any time to keep the file system clean.

FILESYSTEM INTEGRITY CHECKS

- `fsck`: Filesystem Check
  - Generally only run when a filesystem needs it:
    - Mount count
    - Last check
    - Dirty
- Checks all of the filesystem structures for accuracy and completeness

FILE SYSTEM TOOLS

- `e2label`: View/set filesystem label
- `tune2fs`: View/set filesystem attributes
- `mount/umount`: You better know these already. :)

FSTAB

- `/etc/fstab` is parsed during boot by `rc.sysinit` to determine what file systems should be mounted and how. After boot, this file is referenced by the `mount` command.

- The file is space delimited and organized as follows:

```
device  mount_point  fs_type  options  dump  fsck
```

LAB

1. Using `fdisk` or your favorite RHEL6 partitioning tool of choice, create a new 100MB partition.

2. Create a new filesystem on this partition using ext4, a blocksize of 1k, and a reserve space of 2%. Confirm settings with `tune2fs`. Mount the new filesystem as `/u01` and set it to mount at boot.

3. Un-mount the `/u01` filesystem and force an integrity check. Re-mount the `/u01` filesystem. Use `e2label` to set the filesystem label on `/u01` to '`/u01`'.

EXTENDED ATTRIBUTES

- The Linux Extended filesystems support attributes that affect how data can be manipulated.
- The `chattr` command can change these file system attributes.
- The `lsattr` command will list the file system attributes.
- Extended attributes can only be set by the root user, unless the `user_xattr` mount option is in effect.

COMMON EXTENDED FILE ATTRIBUTES

- `i` Immutable. The file can not be changed. By anyone. Period.
- `a` Append-only. File can only be opened for appending.
- Many of the others are experimental and/or esoteric. Surprising? ;)

ACL's

- The Linux Extended Filesystem supports access control lists, which allow for more flexible permissions than standard file system permissions.
- ACL's can be listed with the `getfacl` command.
- They can be modified with the `setfacl` command.
- To use ACLs, a file system must have the `acl` mount option.
- Use `dumpe2fs -h <block device node>` to see default mount options.

ACL EXAMPLES

- `setfacl -m u:bob:w memo.txt`
- `setfacl -x g:ru report.txt`
- `setfacl -m g:ru:r another-report.txt`
- See the man page on `acl`.

SELINUX

- Every process or object has an SELinux context:
  - `identity:role:domain/type`
- The SELinux policy controls:
  - What identities can use which roles
  - What roles can enter which domains
  - What domains can access which types

SELINUX

- Adding the `-Z` option to several commands will show how they are running in regards to SELinux:
  - `ps -Z` lists the process contexts
  - `ls -Z` lists the file contexts
- To change the context of a file, you can use the `chcon` command:
  - `chcon -R --reference=/var/www/html <file>`
- SELinux will log all policy violations to `/var/log/audit/audit.log` as AVC (access vector cache) denials.

CONTROLLING SELINUX

- The tool `system-config-selinux` can be used to configure SELinux.
- The file `/etc/sysconfig/selinux` can be edited.
- The command `getenforce` will show the current SELinux mode, and `setenforce` will allow you to change the mode.
- To change the SELinux mode during boot, you can pass the `enforcing=0` option to the kernel in GRUB.
- See also the members of the "`policycoreutils`" and "`setroubleshoot`" packages.

ADDITIONAL SELINUX TOOLS

- `restorecon` Will restore default filesystem contexts from policy.
- `getsebool` View SELinux boolean(s)
- `setsebool` Set SELinux boolean
- `seinfo` View SELinux information - types, domains, roles, etc.

LAB

1. With SELinux enforcing, configure a website to be served from `/srv`

2. Don't focus on advanced Apache settings, accomplish this in the simplest way possible: just change the global `DocumentRoot`. Further Apache exercises will follow later in class.

3. Populate a simple `index.html` file. Plain text is acceptable.

4. The `setroubleshoot` tool is useful here. Don't be confused by any typos in its output.

```
slideshow.end();
```

RHCE
BOOT CAMP

Users and Groups

USERS AND GROUPS

- Users and Groups define access to the operating system through the file permission scheme.
- Root is the super user, and the only user with special permissions
- Every user is a member of at least one group, which is called their primary group. The main purpose of this primary relationship is to define group owner of created files.
- Users can have a secondary group membership in as many groups as needed. These secondary relationships exist to broaden a user's access to the files on the system.

CONFIG FILES

- User information is stored in two files:
  - `/etc/passwd`
  - `/etc/shadow`
- Group information is stored in one file:
  - `/etc/group`

/ETC/PASSWD

- List of user records, one per line, with columns separated by colons. Format:
- `login:x:userid:groupid:gecos:homedir:shell`
- Examples:
  - `root:x:0:0:root:/root:/bin/bash`
  - `mysql:x:27:27:MySQL Server:/var/lib/mysql:/bin/bash`

/ETC/SHADOW

- Similar colon-separated-column list of records:
- `login:password:`*password aging fields*
- Aging fields track dates for password resets, locks, etc
- Examples:
  - root:pB8msP1fCbCqc:13904:0:99999:7:::
  - nisburgh:vRoPw6a/jQsp.:14466:0:99999:7:::

/ETC/GROUP

- Same colon-separated-column list of records format
- `groupname:grouppassword:groupid:secondarymembers`
- Group passwords allow temporary management to a group, are rarely used and not set up by default
- Examples:
  - `daemon:x:2:root,bin,daemon`
  - `apache:x:48:jack,nisburgh`

MANAGEMENT

- While it is possible to edit the three files directly, it's easier and safer to use the management commands to create, modify and delete users and groups:
  - `useradd, usermod, userdel`
  - `groupadd, groupmod, groupdel`

USERADD

- `useradd`: Add a new user to the system

- Accepts various arguments to control the settings on the user account. Most common is the -g option to specify the primary group of the user, and the -G option to list secondary group memberships. Examples:

  - `useradd lisa`

  - `useradd -g clowns -G trouble,simpson bart`

USERMOD, USERDEL

- `usermod`: Modify a user's settings. Example:
  - `usermod -G detention bart`
- `userdel`: Remove a user from the system. Main option to consider is `-r`, which tells `userdel` to remove the user's home and spool directories. Example:
  - `userdel moe`

GROUP COMMANDS

- `groupadd`: Adds a new group to the system. Example:
  - `groupadd bullies`
- `groupmod`: Mainly used to rename a group. Example:
  - `groupmod -n mktg mkg`
- `groupdel`: Remove a group. Example:
  - `groupdel microsoft`

PASSWORDS

- `passwd`: Change login password.
- Root can change the password for any user on the system
- Root can also setup password aging, allowing for timed password resets and account disabling (or use `chage`)
- `passwd` is also the preferred way to lock a user account:
  - `passwd -l mary`

PASSWORD AGING

- To set the maximum lifetime for a user's password:
  - `passwd -x days login`
- When a user's password has expired, you can set the number of days it can remain expired before disabling the account completely:
  - `passwd -i days login`

IMPORTANT USER ENVIRONMENT FILES

- `/etc/skel` — default template for a newly-added user's home directory
- `/etc/profile` — sets environmental variables used by all users
- `/etc/profile.d` — contains scripts specific to certain rpms
- `/etc/bashrc` — contains global aliases and system settings
- `~/.bashrc` — contains user aliases and functions
- `~/.bash_profile` — contains user environment settings and can be set to automatically start programs at login

LAB

1. Create a new group '`dev`'. Create a new user '`alice`' as a secondary member of the '`dev`' group, with a description of "`Alice from Dev`" and a default shell of '`/bin/csh`'. Use the `passwd` command to set a password for `alice`, then log in as `alice` and verify her access.

2. Set a maximum password lifetime of 4 weeks for the `alice` account. Look at the `passwd`, `shadow` and `group` files.

3. Configure the users `guido`, `linus`, and `richard`. Set all their passwords to "`linux`".

4. Make these users part of the `ru` group in a secondary capacity.

5. Configure the directory `/home/linux` so that each user from the `ru` group can read, create, and modify files.

6. Configure the directory `/home/linux/work` so that each user can create and read files, but only the file's owner can delete.

7. Use ACL's to allow `alice`, not in `ru`, read/write access to the `work` folder and all created sub objects.

PAM

- Applications which are compiled against `libpam.so` may use PAM's modules to customize how individual applications verify their users. Each application has its own configuration file in `/etc/pam.d`
- The first field of the configuration file indicates how the module will be used:

Authentication management (`auth`)	Establishes the identity of a user.
Account management (`account`)	Allows or denies access to the account.
Password management (`password`)	Enforces password management policies.
Session management (`session`)	Starts, stops, and records each session.

PAM

- The second field of the configuration file indicates the effect that the module will have on the application:
  - **`Required`** If this module fails, access will not be granted, but all other modules will still be run.
  - **`Requisite`** If this module fails, access will not be granted and no other modules will be run.
  - **`Sufficient`** If this module succeeds, access will be granted and no other modules will be run.
  - **`Optional`** The result of this module is ignored.

PAM

- The third field of the configuration file indicates the name of the actual PAM module to be used for the config line in question.

- Side note:

  - The config file `system-auth` is a collection of many PAM modules commonly used by many authentication services. You will see it included by many of the other configuration files. *Do not modify this file directly*.

PAM

- **`pam_unix`** Authenticates users by UNIX password
- **`pam_securetty`** Only allows root to log in from secure terminals listed in `/etc/securetty`
- **`pam_nologin`** Will not allow any non-root user to login if `/etc/nologin` exists
- **`pam_time`** Can be configured to allow/deny access based on the system time
- Helpful PAM documentation can be found in:
  - `/usr/share/doc/pam-<version>`

LAB

1. Using PAM, prevent "`guido`" from being able to login on Virtual Console 2. `Guido` should still be able to login elsewhere.

   Hint: Configure the `pam_access` module.

2. Set up the `pam_time` module to restrict `linus` so he can only login between 8am and 5pm Monday through Friday, and block out all non-root users from logging in midnight to 2am Sundays for a maintenance period.

NIS

- NIS Servers can be configured to centrally manage system and account information. These servers can share the contents of `/etc/passwd`, `/etc/shadow`, `/etc/group`, and several other files among any number of clients.
- To configure a client, you must install the `ypbind` and `rpcbind` RPMs, and then you can run `system-config-authentication`.
- This command will make the proper entries in:
  - `/etc/sysconfig/network`
  - `/etc/yp.conf`
  - `/etc/nsswitch.conf`
  - `/etc/pam.d/system-auth`

LAB

1. Configure your server to authenticate against `server1.example.com` using NIS.

2. You should then be able to log in to your server as `station#` (where `#` is your station number) with the password: `station#`

LDAP

- LDAP Servers can also be configured to centrally manage system and account information. LDAP is much more secure and flexible than a default NIS configuration, and as such is becoming much more popular.
- To configure a client, you must install the `nss-pam-ldap` and `openldap` RPMs, and then you can run **`system-config-authentication`**.
- This command will make the proper entries in:
  - `/etc/ldap.conf`
  - `/etc/openldap/ldap.conf`
  - `/etc/nsswitch.conf`
  - `/etc/pam.d/system-auth`

KERBEROS

- Kerberos is a secure authentication method which never needs to send passwords over the network, except in the case of changing a password, which is handled with strong encryption.

- All that is needed for a client to set up Kerberos authentication is:

  - Realm

  - KDC - Key Distribution Center

  - Admin Server (often same server as KDC)

LAB

1. Disable NIS authentication and verify you can no longer authenticate as `station#`.

2. Configure your server to authenticate against `server1.example.com` using LDAP and Kerberos passwords. KDC/Admin server: server1.example.com, realm: EXAMPLE.COM

3. You should then be able to log in to your server as `station#` (where # is your station number) with the password: `station#`

```
slideshow.end();
```

RHCE
BOOT CAMP

Kernel Features

IMPORTANT KERNEL DIRECTORIES

- `/boot` — contains the `vmlinuz` and `initrd` required to boot the system
- `/proc` — virtual file system for seeing “into” the kernel

/PROC/*

- The `/proc` folder contains copious amounts of information useful for troubleshooting. Some examples:
  - `/proc/meminfo` Memory utilization breakdown
  - `/proc/devices` Mapping major numbers to drivers
  - `/proc/dma` dma channel assignments
  - `/proc/ioports` io port assignments
  - See the manpage for `proc` for more information and descriptions

/PROC/*

- Also in the `/proc` folder is detailed information on every process on the system.
  - Details on process status, environment, commandline, and more can be obtained
- Read the `proc` manpage - tons of information available through `/proc`

SYSCTL

- `sysctl`: Get/set kernel parameters
  - `sysctl -w kernel.pid_max=65535`
  - `sysctl -a`
  - `sysctl -w vm.swappiness=100`
- Also, you can view/edit runtime values under `/proc/sys`
- To make changes permanent, edit `/etc/sysctl.conf`

LAB

1. Configure your server to have an open file limit of `524288` files.
2. Configure your server to refuse any ping requests.
3. Configure your server to forward ipv4 packets.
4. Make all of these changes persistent across reboots.

MODULAR

- The Linux kernel is modular, allowing functional blocks of software to be added and removed on the fly via the modules mechanism.

- Modules encompass functions such as:
  - Device drivers
  - Kernel features - firewalls, RAID, LVM
  - Filesystems

LSMOD

- `lsmod`: Prints all of the currently loaded modules

```
[root@dev1 ~]# lsmod
Module                  Size  Used by
ipv6                  264608  20
binfmt_misc            14096  1
dm_multipath           21136  0
parport_pc             31724  0
lp                     16576  0
parport                42252  2 parport_pc,lp
usbcore               129724  1
ext3                  125968  1
jbd                    61928  1 ext3
raid10                 23808  0
raid456               119840  0
xor                    10512  1 raid456
raid1                  24064  0
raid0                  10752  0
multipath              11776  0
linear                  9088  0
dm_mirror              23016  0
dm_snapshot            18872  0
dm_mod                 55752  3 dm_multipath,dm_mirror,dm_snapshot
processor              26412  0
fuse                   42160  1
[root@dev1 ~]#
```

RMMOD

- `rmmod`: Removes (unloads) a loaded modules
  - Can not unload a module that is a dependency of another module
  - Can not unload in-use modules

INSMOD

- `insmod`: Loads a module into the kernel.
  - Full pathname required
  - Does not handle dependencies automatically

MODPROBE

- `modprobe`: Intelligent module handler
  - Can load/unload modules
  - Automatically handles dependencies
  - Only need to specify name of module, not full path, when loading
  - `depmod`: Rebuilds module dependency lists

SOFTWARE RAID

- Software RAID can all be configured, monitored, and modified with the `mdadm` command.
- To create a RAID array, you can run the following command:
  - `mdadm -C <RAID dev> -l <LEVEL> -n <# DISKS> <partitions>`
- To verify the RAID array, use either of the following commands:
  - `mdadm --detail <RAID device>`
  - `cat /proc/mdstat`

LAB

1. Create a RAID-5 array on your machine consisting of:
   - 4 partitions
   - each 512MiB in size
   - one of which should be reserved for use as a hot spare
2. Format this array with ext4 and mount it with support for user quotas so that it will persist across reboots.

LVM

- The Logical Volume Manager
  - Abstracts the physical hardware into logical drive spaces which can be dynamically grown/shrunk and span disparate physical devices
  - Simplifies hard drive management as it abstracts away the details of the underlying storage devices.
  - Adds a small amount of overhead to the VFS layer, slightly reducing performance.

LVM TERMINOLOGY

- **Physical Volume (pv)** — A physical volume is simply the partition/RAID device for the LVM space.
- **Physical Extent (pe)** — A physical extent in a chunk of disk space. Can be any size, but default to 4M.
- **Volume Group (vg)** — A volume group is a collection of physical volumes.
- **Logical Volume (lv)** — A logical volume is a grouping of physical extents from your physical volumes. This logical volume is where you can format a file system.

LVM BASIC IDEA

- To create a space suitable for `mkfs`, three steps must occur:
  - `pvcreate`: Create a physical volume
  - `vgcreate`: Create a volume group on PV
  - `lvcreate`: Create a logical volume on VG
- See also `pvdisplay`, `vgdisplay`, `lvdisplay`

PVCREATE

- Easiest of the LVM tools:

- `pvcreate /dev/sda4`

VGCREATE

- In basic form, you need to provide a name:

- `vgcreate VolGroup00 /dev/sda4`

- Note that `/dev/sda4` is actually a physical volume created with `pvcreate` - not just a device

LVCREATE

- `lvcreate -n myvol -L 10G VolGroup00`

- Creates a new logical volume called `myvol`, 10 gigs in size pulled from the `VolGroup00` Volume Group.

RESIZING LV'S

- `vgextend <volume group name> <new PV path>`
  - Add a new physical volume to a volume group
- `lvextend {-l <+extents>| -L <+size>} <lv>`
  - Grow a logical volume
  - NOTE: Use the + to give the amount of additional space added, otherwise specify the total desired size to end up with.

RESIZING LV'S

- `resize2fs <logical volume>`
  - Once the lv has been extended, you will need to extend the file system
  - You can grow the file system while it is mounted, but before shrinking it must first be unmounted.
- `lvresize -r {-l <+extents>| -L <+size>} <lv>`
  - Resizes logical volume **and** filesystem at same time!

LAB

1. Add logical volume management on top of your raid array. Use a physical extent size of 32MB.

2. Use half the available space for a logical volume formatted with ext4 and mounted persistently across reboots.

3. Take a snapshot of this logical volume and check the file system for errors.

4. Assuming none are found, reset the counter for days and mounts until a check is forced on the original file system.

5. Copy some data onto the LV, then expand it and the filesystem by 50MB. `fsck`, then re-mount the filesystem and verify it's contents.

6. Now try shrinking the filesystem by 200MB.

SWAP SPACE

- Swap space allows the kernel to better manage limited system memory by copying segments of memory onto disk
  - Performance gains
  - “Expanded” memory space

`mkswap`	Create a new swap space for use by the kernel
`swapon/swapoff`	Enable/disable a swap area
`/proc/swaps`	Lists current swap areas

LAB

1. Add 500MB of swap space to your system using a device.

2. Add 500MB of swap space to your system using a swap file.

```
slideshow.end();
```

RHCE
BOOT CAMP

File Sharing Services

NFS

- The Network File Service, or NFS, is used to share data with other servers.
- For this service to work properly, `portmap` and `nfs-utils` rpms must be installed, and `portmap` and `nfs` must be running.
- The command `rpcinfo` can be run to confirm that these services are running on a remote server:
  - `rpcinfo -p server1`
- The directories to be shared are listed in `/etc/exports`

/ETC/EXPORTS

- The directories to be shared are listed in `/etc/exports`
- `/etc/exports` should be configured as follows:
  - `<shared directory> <who>(<how>)`
- Note the **lack** of space between the who and the parenthesis for how. Be very careful about this!
- Example:
  - `/to/be/shared  station*.example.com(rw)`

EXPORTS NETWORK SPECIFICATIONS

- The host/network to be shared to can be specified in a number of convenient ways:
  - **Host** — Just a single host (given by name/ip)
  - **Netgroup** — NIS netgroup, expressed as `@group`
  - **Wildcards** — Using the asterisk, match based off hostnames plus wildcards, as `*.example.com`
  - **IP Networks** — Specify with IP/netmask or CIDR notation:

    `192.168.1.0/24`

    `192.168.1.0/255.255.255.0`

EXPORTFS

- **`exportfs –r`** refreshes the server share list
- **`exportfs –a`** exports all shares in /etc/exports
- **`exportfs –u`** un-exports a share name
- **`showmount -e server1`** shows shares on server1

NFS PERSISTENCE

- NFS mounts can be made persistent across reboots by adding the following to `/etc/fstab`:
  - `server1:/share  /server1/share  nfs  defaults 0 0`

LAB

1. Create a new user.

2. Configure your anonymous NFS user to use this new UID.

3. Grant read/write access to a directory this user owns to our class network, except for server1, who should get read-only access.

4. Mount the NFS share from your neighbor, and add it to their `fstab`.

VSFTPD

- VSFTPd is the default ftp server
- The primary configuration file is `/etc/vsftpd/vsftpd.conf`
- Provides two levels of user access:
  - **Anonymous:** by default these users are chrooted to `/var/ftp` for security
  - **User:** these users authenticate with a username/password and can download any file they can read and can upload into any directory in which they have write access
- Individual users can be denied by placing their names in:
  - `/etc/vsftpd/ftpusers`

LAB

1. Configure VSFTPd to only allow the user `richard` to ftp to your server.

2. Make sure that `richard` is chrooted to his home directory upon login.

3. Configure your FTP server to allow anonymous access, chrooted to `/srv`

SAMBA

- SAMBA is an open source implementation of Windows networking protocols. With SAMBA, it is possible to:
  - Provide file and print services for various Microsoft Windows clients
  - Integrate with a Windows Server domain as a Primary Domain Controller (PDC) or as a Domain Member.
  - Be part of an Active Directory domain.

SAMBA

- SAMBA provides the following services in Linux:
  - Authentication and authorization of users (Active Directory)
  - File and printer sharing
  - Name resolution
  - Browsing (Wins or NetBios)

GETTING SAMBA GOING

- Some packages what should be installed for SAMBA to work as desired:

• **samba**	provides basic software for sharing files and printers
• **samba-client**	allows server to connect to windows shares (also includes the smbclient command, which functions like a command-line ftp client)
• **samba-common**	contains samba configuration files

GETTING SAMBA GOING

- For SAMBA to work properly, the following services must be running:
  - `smbd` (SMB/CIFS Server) for authentication and authorization and file and printer sharing
  - `nmbd` (NetBIOS name server) for resource browsing and possibly as a wins server

CONFIGURING SAMBA

- The main configuration file for SAMBA is:
  - `/etc/samba/smb.conf`
- This file is **very** well commented and has examples for just about anything that you need to do.
- Once you have made a configuration change, you can test it with the `testparm` command.

SAMBA USERS

- To have a SAMBA user, that user must first be created in `/etc/passwd`
- The command `smbpasswd -a` can then be used to add a user to the password database under `/etc/samba/` for SAMBA authentication.

SAMBA SHARES

- To see the SAMBA shares a user has access to, you use `smbclient` as follows:
  - `smbclient -L <server> -U <user>%<passwd>`
- To mount a share, you use the UNC path:
  - `mount.cifs //server/share /mount/point -o username=<user>`
- To configure this mount to happen at boot, add the following to `fstab`:
  - `//server/share /mount/point cifs  credentials=/etc/samba/pub.cred 0 0`
- (where `/etc/samba/pub.cred` is a file that only root can read which contains usernames and passwords)

LAB

1. Configure SAMBA to share your `/srv` directory only to one neighbor who must log in with the SAMBA username of `richard`.

2. Make this share read-only for the SAMBA user `guido`.

3. Mount the share from your neighbor. Configure it to mount automatically at boot time. Use a credentials file to store the account information securely.

```
slideshow.end();
```

RHCE
BOOT CAMP

Web Services

APACHE CONFIGURATION

- The main apache configuration file is `httpd.conf` and is found in `/etc/httpd/conf/`. This configuration file stores the core configuration of the web server.

- In Apache 2, the `/etc/httpd/conf.d` directory stores configurations that are specific to a particular Apache module. All files in this directory ending in `.conf` will be parsed as a configuration file.

APACHE CONFIGURATION

- You can find this example Apache VirtualHost definition at the bottom of `httpd.conf`:

```
<VirtualHost _____>
    ServerName name
    ServerAlias alias
    DocumentRoot path
    CustomLog /path/to/access_log combined
    ErrorLog /path/to/error_log
</VirtualHost>
```

- The `NameVirtualHost` directive **must be used** to specify an IP that can host multiple websites.

LAB

1. Configure two websites on your server. “X” represents your station #.

2. `wwwX.example.com` should be served from `/var/www/html` and should also respond to requests for the short hostname `wwwX`.

3. `vhostX.example.com` should be served from `/home/linus/html` and should also respond to requests for the short hostname `vhostX`.

4. Both should be listening on your primary ip address, but `wwwX.example.com` should be the default site.

SECURING APACHE

- Apache support access control through `allow` and `deny` directives:
    - `allow from <host|network|ALL>`
    - `deny from <host|network|ALL>`
- These can be applied in the given order:
    - `order allow,deny` — Allows explicitly allowed clients and **denies everyone else**. Anyone matching both the allow and deny are denied.
    - `order deny,allow` — Denies explicitly denied clients and **allows everyone else**. Anyone matching both the allow and deny are allowed.

SECURING APACHE

- These access control directive are applied through a per-Directory or per-File basis.

- The `allow`, `deny` and `order` directives are placed inside one of the following tags:

    - `<Directory>`

    - `<File>`

LAB

1. Reconfigure your two websites such that:

   - `wwwX.example.com` is accessible to everyone except for the person sitting to your left.

   - `vhostX.example.com` is only accessible to the person sitting to your right.

CGI SCRIPTING

- Scripting involves making Apache *execute* a file and return it's *output*, as opposed to simply returning the file itself.
- There is an entire framework for facilitating this operation, and allowing the webserver to communicate basic information to script through the use of environment variables, and sometimes input.
- This is known as CGI scripting, or Common Gateway Interface scripting.

BASIC SCRIPTING

- Some of the simplest scripting requires only a shell script. Consider:

```
#!/bin/bash

echo -e “Content-type: text/html\n”

echo “<h1>Hello world!</h1>”
```

BASIC SCRIPTING

- If we put the appropriate execute permissions on the script, then we can see it output the expected content at the command line:

```
# chmod +x myscript

# ./myscript

Content-type: text/html



<h1>Hello world!</h1>
```

BASIC SCRIPTING

- If this file is placed in a location identified to Apache as supporting executables (CGI scripts), then we have a working CGI!

LAB

1. Install `httpd-manual` if you have not already done so.

2. Look up the `ScriptAlias` directive in the manual.

3. Use this directive and your simple shell script to create a simple, dynamic webpage. Maybe have it report the current date and time with the `date` command.

SQUID

- Squid is designed to cache internet objects and can act as a proxy server for HTTP, FTP, and many other types of requests.

- Squid is highly flexible and powerful, but for the RHCE exam, you only need to demonstrate the ability to set it up and proxy web services, possibly denying access to a given subnet.

- The configuration file for Squid is

  `/etc/squid/squid.conf`

KEY SQUID SETTINGS

- **`http_port`** *3128 by default*
- **`visible_hostname`** *the hostname Squid broadcasts*

KEY SQUID SETTINGS

- Access control in squid is handled via ACL definitions coupled with access definitions, as:

```
acl mynet src 192.168.0.0/255.255.255.0

acl yournet src 192.168.1.0/255.255.255.0

http_access allow mynet

http_access deny yournet
```

- Look for “HERE” in the config file. This is the best place for new ACL entries.

LAB

1. Configure your server to offer Squid proxy service to the person sitting on your right, but not to the person sitting on your left.

2. This service should listen on port 8080.

```
slideshow.end();
```

RHCE
BOOT CAMP

Network Services

OPENSSH

- OpenSSH is the open source version of SSH, and is used by most UNIX variants for secure remote administration.

- This service is configured in `/etc/ssh/sshd_config`.

OPENSSH CONFIGURATION

- OpenSSH Configuration Parameters of interest:
  - `Port`
  - `ListenAddress`
  - `PermitRootLogin`
  - `PubkeyAuthentication`
  - `Subsystem sftp`

XINETD

- `xinetd` is the extended internet services SUPER daemon. :)
- This service acts as a super daemon by listening on key ports for certain types of requests.
- When a request is received, `xinetd` starts the appropriate service and then hands of the request so that it can be handled correctly.
- `xinetd` is configured in `/etc/xinetd.conf`, the services that it controls are configured in `/etc/xinetd.d/`

LAB

1. Configure your box to allow both the '`root`' and '`student`' users to login locally, but not over ssh.

2. Configure an anonymous rsync service to share the contents of your `/srv` directory. See the man page for `rsyncd.conf`.

NTP

- The Network Time Protocol is a very useful and accurate method to keep your system clock synchronized with time servers around the world. This is important because:
  - Timestamps in log files across machine will line up, allowing for proper analysis and comparison
  - Cron jobs run at the right time
  - Knowing the correct time just makes for a happy server

LAB

1. Enable NTP on your machine, and use `0.pool.ntp.org` and `1.pool.ntp.org`.

2. Use the `ntpq` command to figure out how far off your machine's clock is from true time.

```
slideshow.end();
```

RHCE
BOOT CAMP

BIND

CONFIG FILES

- BIND basically has two types of configuration files:
  - BIND configuration file, specific to BIND and it's features
  - Database files, or zone files, which contain DNS resource records used to describe all of the DNS information needed in a domain

RESOURCE RECORDS

- A resource record contains the DNS *information* about a domain.

- There are several types of resource records, including address records (A), mail exchangers (MX) and name servers (NS).

- Every domain has at least 2 resource records, an SOA and an NS. But that wouldn't be a very useful domain, so there are usually quite a few more records, defining addresses, mail exchangers, canonical names and more.

SOA

- Start Of Authority: This resource record defines authority for a zone.

```
domain IN SOA nameserver adminemail (
        serial
        refresh
        retry
        expire
        negativettl
)
```

We'll discuss these later!

SOA EXAMPLE

```
rackspace.com. IN SOA ns1.rackspace.com. dnsadmin.rackspace.com. (
   2009123004   ; Serial number
   3h           ; Refresh interval
   1h           ; Retry interval
   1w           ; Expires
   1h           ; Negative TTL
)
```

NS

- Name Server: Defines authoritative nameservers for the zone.

  zone **IN NS** *nameserver*

- Example:

  ```
  rackspace.com. IN NS ns1.rackspace.com.
  ```

A

- Address: Maps hostnames to IP addresses

  hostname **IN A** *ipaddress*

- Example:

```
ns1.rackspace.com. IN A 192.168.1.5
```

CNAME

- Canonical Name: Maps *alias* hostnames to their *canonical* counterparts.

  aliashostname **IN CNAME** *canonicalhostname*

- Example:

  ```
  ns.rackspace.com. IN CNAME ns1.rackspace.com.
  ```

CANONICAL?

- In layman terms, canonical is another way of saying “real”, “absolute” or “official”.
- So a canonical name refers to the “official” name for a host.
- Creating an alias for a host means that you have to decide on the canonical name, which would be some A record.
- When a resolver performs an A query on a CNAME, the nameserver looks up the canonical name to find out the address to return.

WHY NOT JUST USE A?

- There are basically two reasons to use CNAME records instead of just lots of A records.
  - First, ease of maintenance. If you need 10 names for one machine, defining them with CNAME is easiest if you then need to change the IP address of the machine. Only one change instead of 11.
- Second, *canonicalization*. Some services, most notably sendmail, will convert all aliases into their canonical names. This simplifies mail configuration.

PTR

- Pointer: Maps an IP address back to a name, specifically the canonical name.

ipaddressdomain **IN PTR** *canonicalhostname*

- Example:

```
5.1.168.192.in-addr.arpa. IN PTR ns1.rackspace.com.
```

PTR

- Remember, there is only one PTR record for a given IP address, and it should always point to the *canonical hostname*.

- Also, as a side note, make sure your mail servers map both directions exactly. This is important for proper authentication:

  - `mailer.mydomain.com. -> 192.168.1.50`

  - `50.1.168.192.in-addr.arpa. -> mailer.mydomain.com`

MX

- Mail Exchanger: Defines hosts responsible for incoming email for the named zones.

```
zone IN MX preference mailhandler
```

- Example:

```
rackspace.com. IN MX 10 mail1.rackspace.com.
```

MX RECORDS

- MX records allow for enhanced mail routing functionality.
- When an email is shipped out, the server canonicalizes the delivery address. So, for example, bob@ns.rackspace.com becomes bob@ns1.rackspace.com.
- Then the server looks up the MX records for ns1.rackspace.com, choosing the record with the **lowest** preference and attempting delivery there. If delivery fails, the next lowest is attempted.
- This allows for backup email servers!

PUTTING IT ALL TOGETHER

db.rackspace.com:

```
$TTL 1h
rackspace.com. IN SOA ns1.rackspace.com. dnsadmin.rackspace.com. (
   2009123004   ; Serial number
   3h           ; Refresh interval
   1h           ; Retry interval
   1w           ; Expires
   1h           ; Negative TTL
)

rackspace.com. IN NS ns1.rackspace.com.
ns1.rackspace.com. IN A 192.168.1.5
mail1.rackspace.com. IN A 192.168.1.20
ns.rackspace.com. IN CNAME ns1.rackspace.com.
rackspace.com. IN MX 10 mail1.rackspace.com.
```

TOGETHER...

db.192.168.1:

```
$TTL 1h
1.168.192.in-addr.arpa. IN SOA ns1.rackspace.com. dnsadmin.rackspace.com. (
   2009123004   ; Serial number
   3h           ; Refresh interval
   1h           ; Retry interval
   1w           ; Expires
   1h           ; Negative TTL
)

1.168.192.in-addr.arpa. IN NS ns1.rackspace.com.
5.1.168.192.in-addr.arpa. IN PTR ns1.rackspace.com.
20.1.168.192.in-addr.arpa. IN PTR mail1.rackspace.com.
```

SO WHAT ELSE?

- In addition to the zone files for your domains, you need a couple more zone files to get BIND up and running.
  - Loopback address
  - Root hints

LOOPBACK ADDRESS?

- Someone has to take responsibility for loopback address requests! It's simple enough. `db.127.0.0`:

```
$TTL 1w
0.0.127.in-addr.arpa. IN SOA ns1.rackspace.com. dnsadmin.rackspace.com. (
   2009123004   ; Serial number
   3h           ; Refresh interval
   1h           ; Retry interval
   1w           ; Expires
   1h           ; Negative TTL
)

0.0.127.in-addr.arpa. IN NS ns1.rackspace.com.
1.0.0.127.in-addr.arpa. IN PTR localhost.
```

ROOT HINTS!

- The root hints tell the nameserver where those DNS Root Servers are located, so that requests for hosts outside of your authoritative zones can be resolved.
- This one is the simplest to put together. You don't even have to write it!
- Simply ftp the db.cache file from ftp.rs.internic.net/domain

WHEW

- Finally. All of the zone files are put together and ready. Final step? Configuring BIND.

- The config file is generally `/var/named/chroot/etc/named.conf`

NAMED.CONF GENERAL

```
acl “clients” { 192.168.0.0/24;};

acl “servers” { 192.168.1.2; 192.168.1.3;};

options {

    directory “/var/named”;

    forwarders { 192.168.0.254; };

    allow-query { clients; };

    allow-transfer { servers; };

};
```

NAMED.CONF ZONES

```
zone “example.com” IN {

     type master;

     file “example.com.zone”;

};

zone “example.com” IN {

     type slave;

     file “slave.example.com.zone”;

     masters { 192.168.2.254; };

};
```

LAB

1. Configure your machine to act as the authoritative nameserver for a "`demoX.example.com`" domain and a "`rhceX.example.com`" domain, where X is your station number.

2. For both domains, configure these records:

   - "`www`", "`mail`", and "`ns`" should all resolve to your IP address
   - "`web`" should resolve to "`www`"
   - "`mail`" should be listed as the primary MTA for the domain
   - "`ns`" should be listed as the DNS server for the domain

3. Also configure your machine to respond to reverse DNS lookups, such that your own IP address will resolve to "`www.rhceX.example.com`"

4. Set up your machine to forward any other DNS requests to server1.

```
slideshow.end();
```

RHCE
BOOT CAMP

Email Services

POSTFIX - DEFAULT RHEL6 MTA

- Postfix was designed from the ground up to be a replacement for Sendmail.
- The Postfix group has the following goals for their product:
  - It should be more efficient than Sendmail.
  - It should be more secure than Sendmail.
  - It should be easier to administer than Sendmail.
  - It should be 100% Sendmail compatible.
- To accomplish these goals, Postfix is made up of many individual programs which each handle a particular aspect of mail transfer. These programs are managed by a supervisory `master` daemon.

CONFIGURING POSTFIX

- Postfix's configuration file is `/etc/postfix/main.cf`
- The directives in this file can be changed manually, or `postconf -e` can be run to apply them from the command line. For example, the following are the most common of the changes that can be made:
  - `postconf –e "myorigin = redhat.com"`
  - `postconf –e "mydestination = redhat.com mail.redhat.com"`
  - `postconf –e "mynetworks = 192.168.0.0/24, 127.0.0.1"`
  - `postconf –e "inet_interfaces = all"`
  - `postconf -e "relayhost = server1.example.com"`
- `postconf -n` can then be called to check your configuration for errors

SENDMAIL - ALT. RHEL6 MTA

- What is Sendmail?

- Sendmail is an extremely popular mail transfer agent (MTA) used by default on many UN*X distributions to handle the receipt and delivery of emails.

- Sendmail has been around a very long time, and still carries some configuration thorns from previous decades

- Namely, using `m4` for a configuration “language”

SENDMAIL

- MUA versus MTA
  - A mail user agent (MUA) is a program that users run to read, reply to, compose, and dispose of email (such as Outlook, Mozilla Mail, Eudora, etc). You can have many different MUA's installed and running on one machine.
  - A mail transfer agent (MTA) is a program that delivers mail and transports it between machines. Usually, there is only one MTA running on a machine at any particular time.
- LDA (Local Delivery Agent)
  - Once the MTA receives a message, it determines if the message is intended for a local or remote recipient. If the message is intended for a remote location, the message is then passed off to the appropriate MTA. If the message is local, it will be passed to the LDA.

SENDMAIL CONFIGURATION

- **`/etc/mail/sendmail.cf`**
  - Main configuration file. This file is parsed at each successful start-up.
- **`/etc/mail/sendmail.mc`**
  - Configuration changes should always be written here.
- **`/etc/mail/local-host-names`**
  - This file contains a list of domain names that the server will handle mail for.
- **`/etc/aliases`**
  - This file specifies redirects for one user to another address or group of addresses.

ALTERNATIVES!

- Alternatives can be used when many packages provide the same service.
- The executable that the Sendmail init script invokes is really just a symbolic link to another symlink in the `/etc/alternatives` directory.
- For example take a look at `/usr/sbin/sendmail`.
- In order to choose between Sendmail and postfix, we just change the symlink.
- This can be done with the following commands:
  - `alternatives —-display mta`
  - `alternatives -—config mta`
  - `alternatives —-set mta`

DOVECOT

- `dovecot` is the default POP/IMAP server for RHEL 6.
- The configuration files are under `/etc/dovecot`
- Usually, the only changes that need to be made are the enabling/disabling of the desired protocols in `dovecot.conf` and possibly specifying where user mailboxes are located. See:

  `/etc/dovecot/conf.d/10-mail.conf`

MUTT

- `mutt` is a full-featured MUA for your terminal. You can use it to test pop3s and imaps:
  - `mutt -f protocol://server`

LAB

1. Configure Postfix to receive mail for `stationX.example.com`, and store user mailboxes in Maildir format.

2. When mail is received for `ru@stationX.example.com`, that mail should be forwarded to the users `richard` and `linus`.

3. Configure `dovecot` to serve user Maildirs on both imap and pop3.

4. Generate a new key and self signed certificate (see `genkey, openssl, mkcert.sh, /etc/pki/tls/certs/Makefile` for help with this step) for use with ssl encrypted imaps and pop3s and then enable those protocols.

5. Test your secure mail server with `mutt`.

```
slideshow.end();
```

RHCE
BOOT CAMP

Network Security

TCP WRAPPERS

- TCP Wrappers was originally written to provide host based access control for services which did not already include it.
- It was one of the first “firewalls” of a sort. :)
- Before you can set up tcp_wrappers on a service, you have to check if the service supports it...

CHECKING TCP WRAPPER SUPPORT

- Determine which binary the application runs as. Check `init` script or:

```
# which sshd

/usr/sbin/sshd
```

- Check for `libwrap` support in the binary.

- If you see `libwrap` support in the output, then you can configure access to the service with tcp_wrappers.

```
# ldd /usr/sbin/sshd | fgrep wrap

libwrap.so.0 => /usr/lib/libwrap.so.0 (0x009c5000)
```

TCP WRAPPER OPERATION

- In addition to checking for the library, note that some services have configuration entries which enable/disable tcpwrapper support. Check these settings as well.

- If an application is compiled with support for `tcp_wrappers`, that application will check connection attempts against the `tcp_wrappers` configuration files:

    - `/etc/hosts.allow`

    - `/etc/hosts.deny`

TCP WRAPPER OPERATION

- These files are parsed in the following order:
    - The file `/etc/hosts.allow` is consulted. If the configuration of this file permits the requested connection, the connection is immediately allowed.
    - The file `/etc/hosts.deny` is consulted. If the configuration of this file does not permit the requested connection, the connection is immediately refused.
    - If the connection is not specifically accepted or rejected in either file, the connection is permitted.

TCP WRAPPER CONFIGURATION

- The basic syntax for these files is:
  - `<daemon>: <client>`
- For example, to disable `ssh` connections from `192.168.2.200`, add this line to `/etc/hosts.deny`:
  - `sshd: 192.168.2.200`

IPTABLES

- IPTables works at the kernel level, just above the network drivers, to provide several useful features.
  - Extremely powerful and flexible Layer 2 filtering engine.
  - NAT support
  - Port forwarding
  - And many more

IPTABLES RULE MATCHING

- The IPTables configuration is parsed from top to bottom.
- IPTables will respond based on the first match that it finds.
- If there is no specific match, the chain policy will apply.

IPTABLES TOOLS

- **`iptables`**: View/modify current firewall rules
- **`iptables-save`**: Script to save current firewall rules for use with iptables-restore
- **`iptables-restore`**: Restores iptables-save format firewall rules - useful to set up firewalls at boot
- Consider `iptables` init script for save/restore. Config file:

  `/etc/sysconfig/iptables`

IPTABLES RULES

- When creating a new rule, considerations include:
  - What chain should the rule apply to?
  - What is the traffic pattern to look for?
  - What should happen with the traffic?

IPTABLES CHAINS

- **`INPUT`**
  - This chain is responsible for filtering traffic destined for the local system.
- **`OUTPUT`**
  - This chain is responsible for handling outbound traffic.
- **`FORWARD`**
  - This chain is responsible for controlling traffic routed between different interfaces.

IPTABLES RULES

- Below are a few examples of possible IPTables match criteria:
  - incoming interface `-i`
  - protocol `-p`
  - source ip address `-s`
  - destination ip address `-d`
  - destination port `--dport`

IPTABLES RULES

- Finally, some examples of what to do with traffic when matched:
  - **`DROP`** Do not deliver, do not respond
  - **`REJECT`** Do not deliver, send reject notice
  - **`ACCEPT`** Deliver
  - **`LOG`** Just log the packet

IPTABLES RULES

- So to summarize the syntax:
    - `iptables`
    - What chain should the rule apply to?
        - `-A INPUT`
    - What is the traffic pattern to look for?
        - `-s 192.168.2.100`
    - What should happen with the traffic?
        - `-j REJECT`

LAB

1. Using `iptables`, configure your web server to NOT accept connections from the `192.168.1.0/24` network, EXCEPT for the ip address of whomever is sitting next to you. Work together to test the firewall settings, and remember, **web** server. :)

2. Browse through the man page for iptables.

3. Use `iptables` to allow ssh from the classroom network only.

```
slideshow.end();
```

RHCE
BOOT CAMP

Various Additional Topics

BUILDING RPMS

- Building an RPM can be simple or difficult, depending on if it's done incorrectly or correctly. ;)
- Fortunately, for the RHCE exam, you only need to know how to build a simple RPM that packages one file.
- Unfortunately, RPM's were designed to build and package software, so there are lots of extra details that need to be removed to package a simple file.

RPM PACKAGES

- Some important packages to install:

• `rpm`	Duh. ;)
• `rpmdevtools`	Very helpful
• `rpmlint`	Checks rpm's - can be handy

GETTING STARTED

- First, switch to a non-root user.
- Second, run `rpmdev-setuptree` to build a basic directory tree needed for rpm construction.
- `cd rpmbuild`
- Note the various folders - most important right now are `SOURCES` and `SPECS`.

SETTING UP SOURCES

- Setting up the SOURCES folder is a little involved:
  - `cd SOURCES`
  - `mkdir rhce-1.0`
  - `echo “test” > rhce-1.0/afile`
  - `tar czf rhce-1.0.tar.gz rhce-1.0`
- This will create your initial “source code” tarball

SETTING UP SPEC FILE

- Writing the spec file is the most difficult part.
- First, cd into the `SPECS` folder and create a template spec file by running:
  - `rpmdev-newspec rhce.spec`

SPEC FILES

- Spec files have a peculiar syntax:
  - There are tags that have short values associated with them, such as `Name` and `Version`
  - There are sections that are identified with a percent sign followed by a name, such as `%description` and `%prep`
  - There are “macros”, which behave similarly to environment variables: `%{version}` will substitute to the version number entered in the `Version` tag line.

SPEC FILES

- In your spec file, fill out the following areas:
  - `Name: rhce`
  - `Version: 1.0`
  - `Summary: RPM for RHCE`
  - `Group: Documentation`
  - `License: None`
  - `URL: http://www.redhat.com`
  - `Source0: rhce-1.0.tar.gz`

SPEC FILES

- You do not have any requirements, so just delete the lines:
  - `Requires`
  - `BuildRequires`
- For the Description, put a short, meaningful message:
  - `RHCE Exam RPM file`

SPEC FILES

- Remove the `%configure` macro, as this just tries to call `configure` for you automatically, which is not needed for our simple rpm.

- Also, remove the `make` lines - one is under `%build`, one is under `%install`. Same reasoning - we don't need make for our rpm.

- Under `%install`, below the `rm -rf`, add:

  - `mkdir -p $RPM_BUILD_ROOT`

  - `cp afile $RPM_BUILD_ROOT`

SPEC FILES

- Under `%files`, replace the `%doc` with:
  - `/afile`
- Verify all of your spec file contents

BUILD THE RPM

- Trial by fire! Build the rpm from the `rpmbuild` folder:
  - `rpmbuild -ba SPECS/rhce.spec`
- If you spec file is good, and your `SOURCES` tar file, you will have a new rpm under the `RPMS` folder
- Verify new rpm with:
  - `rpm -qlp RPMS/*/*.rpm`
- You should see the single pathname "`/afile`". Install if you wish.

LAB

1. Build a simple rpm that packages a file called “`I-rock-rpms`” and installs it to `/`.

2. Install your rpm and verify `/I-rock-rpms` exists.

ISCSI

- iSCSI is a neat protocol which allows for the transport of SCSI commands over standard network stacks, such as TCP/IP.

- In iSCSI parlance, a “target” is a server/device that accepts commands and relays them to a storage system. An “initiator” is a client which sends commands to a target.

- For the RHCE exam, all you need to know is how to set up an initiator.

ISCSI

- Required package: `iscsi-initiator-utils`
- This provides the `iscsid` and `iscsi` services. `iscsid` manages the low level iSCSI communications, and `iscsi` automatically logs in and out of targets.
- You usually only want to start/stop `iscsi`, as it will take care of `iscsid`.

ISCSI

- Once the iSCSI package is installed, connecting to a target is super simple:
  - `iscsiadm -m discovery -t st -p <ip>`
- If any targets are discovered, they will be printed back, as:
- **`192.168.1.100:3260,1 iqn.2011-04.com.example.server1:server1.target1`**
- This shows a single target on server1 is available.

ISCSI

- Once targets are discovered, they will be remembered. You can see your known targets with:
  - `iscsiadm -m node -o show`
- Once targets are found, start up the iscsi service:
  - `service iscsi start`
- Check dmesg to verify it finds and attaches the new SCSI devices.

ISCSI

- Once you identify the scsi device (`/dev/sdb` on our machines), you can partition, format and roll:
  - `fdisk /dev/sdb`
  - `mkfs /dev/sdb1`
  - *Add entry to fstab and mount!*

SCHEDULE FOR TOMORROW

- Exam starts at 9:00am **SHARP**
- Exam concludes at 11:00am.
- Lunch from 11:00am to 12:30pm
- Review exam on projector from 12:30pm until finished
- Final Q/A session
- Survey Monkey!
- Remember: Machine getting wiped today! Save what you need!

LAB

1. Check server1 for available iSCSI targets. You should see exactly one, and the target number will match your station number.

2. Attach the iSCSI device, partition it, build an ext4 filesystem and set it to mount at boot to `/iscsi`. Don't forget to chkconfig iscsi on!

3. Reboot you machine and verify the iSCSI filesystem comes up automatically.

```
slideshow.end();
```