RHCSA
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 used to trick the kernel and force a reload. **`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 Linux extended filesystems, this means creating the 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`, 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`'.

AUTOMOUNT

- The `autofs` service can be configured to monitor certain directories and automatically mount a file system when a call is made to files in that directory.

- When `autofs` starts, it parses the configuration file `/etc/auto.master` to determine which directories it should be monitoring. Each directory can then have its own configuration file determining how each file system should be mounted, or the default file `/etc/auto.misc` can be used.

AUTO.MASTER

- Basic format for `auto.master`:
- Path Config file
- `/misc /etc/auto.misc`
- This tells `automountd` to “watch” the `/misc` pathname for activity, and if activity is observed, consult `/etc/auto.misc` for instructions.

AUTOMOUNT PATH CONFIG FILES

- Basic syntax:
- path options mount device
- `nfs -fstype=nfs,ro nfsserver:/share/nfs`
- This tells `automountd` to dynamically mount the nfs share "`/share/nfs`" on `nfsserver` when access is attempted on the "`nfs`" pathname under a watched pathname (`/misc` for example)

LAB

1. Configure your server to automatically mount `/share` as an NFS share from `server1` to `/server1/share` when a process changes directories there.

EXTENDED ATTRIBUTES

- The Linux Extended filesystem supports 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.
- Most 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`

QUOTAS

- Quotas are used to limit how many filesystem resources are available to a user.
- Inodes and space are controllable.
- Hard and soft limits are available, with grace periods.
- Enabling quotes is an involved process...

ENABLING QUOTAS

- `usrquota` and `grpquota` options must be enabled on the filesystem mount
- Run `quotacheck -mavug`
- Two files will be created at the root of the filesystem: `aquota.user` and `aquota.group`
- Turn on quotas by running `quotaon` with the mount point as argument.
- Now you can use `edquota` to set up the quotas
- See man pages: `quota, repquota, edquota, quotaon, quotacheck`

LAB

1. Create a quota for the user `student` with:
   - a block soft limit of `100M` and a hard limit of `150M`
   - a soft inode limit of `30` and a hard inode limit of `100`
2. Create a quota for the group `gdm` so that its members collectively have:
   - a block soft limit of `200M` and a hard limit of `300M`
   - a soft inode limit of `50` and a hard inode limit of `200`

DISK ENCRYPTION

- Disk encryption is supported under Linux via the Device Mapper functionality introduced in the 2.6 kernel.
- The Device Mapper allows arbitrary device path mapping.
- Disk encryption is most commonly implemented with the dm-crypt Device Mapper module, supporting transparent device encryption.
- dm-crypt supports a simple, internal encryption specification, as well as the more common LUKS, Linux Unified Key Setup.

LUKS

- LUKS is an open standard disk encryption specification.
- LUKS is a preferred standard due to it's broad compatibility and secure implementation.
- Using `cryptsetup`, a LUKS encrypted device can be created, accessed and modified.

CRYPTSETUP

- To create a new LUKS encrypted device:
  - `cryptsetup luksFormat <device>`
- Then, to establish access to the device:
  - `cryptsetup luksOpen <device> <mapname>`
- This command will verify the password and setup a new dm-crypt device mapper mapping of:
  - `<device> -> dm-crypt(LUKS) -> <mapname>`
- Creating `/dev/mapper/mapname`

CRYPTSETUP

- After the `/dev/mapper/mapname` is in place, all operations operate on the mapper device:
  - `mkfs -t ext4 /dev/mapper/mapname`
  - `mount /dev/mapper/mapname /crypt`
- To remove access to an encrypted device, unmount the filesystem if it's mounted, then:
  - `cryptsetup luksClose mapname`

LUKS PERSISTENCE

- To make a LUKS encrypted device available at boot time, use the `/etc/crypttab` file:
  - `<mapname> <device> [keyfile] [options]`
- To create an insecure keyfile:
  - `echo -n 'your pass phrase' > /etc/keyfile`
- To create a secure keyfile:
  - `dd if=/dev/urandom of=/etc/keyfile bs=1k count=4`
  - `cryptsetup luksAddKey <device> /etc/keyfile`

LAB

1. Create a new 100M partition, then set up a LUKS encrypted ext4 filesystem on the partition which will be persistent across reboots.

2. Reboot your machine to verify the LUKS filesystems prompt for the passphrase and become accessible automatically after bootup.

3. Browse through the man pages on `cryptsetup` and `crypttab`.

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.

LABELING

- The SELinux policy includes a specification for default contexts on all common pathnames in a standard Linux filesystem, known as the default filesystem labels.

- Relabeling involves using the defaults from the policy and applying the contexts to files. The tool for relabeling is:

  - `restorecon [-R] <path> [path...]`

- `restorecon` can work on individual pathnames as well as recursively apply contexts to a pathname.

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`.

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();
```