**Agenda:**

* Linux File System

**Linux File System**

**File Systems**

* ext – Extended file system, designed for Linux systems, obsolete
* ext2 – 2nd revision extended file system, 32TB max volume
* ext3 – Journaled version of ext2, same limits, “feature-rich” version of ext2
  + Journaling is a tool that allows a filesystem to recover from an error, such as corrupted files
* ext4 – Designed for large file systems (up to 1 exbibyte, file sizes up to 16 tebibytes
* xfs – Developed by SGI, up to 16 exbibyte volume, up to 8 EiB file size
* What’s the difference between tebibyte and terabyte? Terabyte is base 10, tebibyte is base 2

**Filesystem Hierarchy Standard (FHS)**

* /bin – Essential command binaries that need to be available in single user mode; for all users
* /boot – Boot loader files, e.g., kernels, initrd
* /dev – Essential devices
* /etc – Host-specific system-wide configuration files
* /home – Users’ home directories, containing saved files, personal settings, etc
* /lib – Libraries essential for the binaries in /bin/ and /sbin/
* /media – Mount points for removable media such as CD-ROMs (appeared in FHS-2.3)
* /mnt – Temporarily mounted filesystems
* /opt – Optional application software packages
* /proc – Virtual filesystem providing process and kernel information as files. In Linux, corresponds to a procfs mount
* /root – Home directory for the root user
* /run – Run-time variable data: Information about the running system since last boot, e.g., currently logged-in users and running daemons
* /sbin – Essential system binaries (sbin = superuser binaries)
* /srv – Site-specific data which are served by the system
* /tmp – Temporary files (see also /var/tmp). Often not preserved between system reboots, and may be severely size restricted
* /user – Secondary hierarchy for read-only user data; contains the majority of (multi-)user utilities and applications
* /var – Variable files – files whose content is expected to continually change during normal operation of the system – such as logs, spool files, and temporary e-mail files

**Physical vs Logical**

* It all starts at /
* Mountpoints – directory that connects to physical hard drive space
* Anything that does not go in its own mountpoint goes under the root (/)
* /etc/fstab

**/etc/fstab**

* The file that makes the file system work
* fstab is a configuration file that contains information of all the partitions and storage devices connected to your computer. The file is located in /etc. You must be root to edit this file

*<device> <mountpoint> <filesystemtype><options> <dump> <fsckorder>*

/dev/hdb5 / ext2 defaults 1 1

/dev/hdb2 /home ext2 defaults 1 2

* First field (**/dev/hdb5**) is the physical device or remote filesystem to be mounted
* Second field (**/**) specifies the mount point where the filesystem will be mounted
* Third field (**ext2**) is the type of filesystem on the device from the first field
* Fourth field (noatuo, ro, user) is a (**default**) list of options which mount should use when mounting the filesystem
* Fifth Field (**0**) is used to by dump (backup util) to decide if a filesystem should be backed up. If zero, then dump will that fs, (1) means don’t ignore. **Normally 0**
* Sixth field if (**0**) then the filesystem will not be checked. 1 is file sys errors corrected, 2 is system should be rebooted. **Normally 0**

**fstab Options**

* **auto** – Mount automatically at boot, or when the command mount –a is issued
* **noauto** – Mount only when you tell it to
* **exec** – Allow execution of binaries on the filesystem
* **noexec** – Disallow execution of binaries on the filesystem
* **ro** – Mount the filesystem read-only
* **rw** – Mount the filesystem read-write
* **user** – Allow any user to mount the filesystem. This automatically implies noexec, nosuid, nodev, unless overridden
* **users** – Allow any user in the users group to mount the filesystem
* **nouser** – Allow only root to mount the filesystem
* **owner** – Allow the owner of device to mount
* **sync** – I/O should be done synchronously
* **async** – I/O should be done asynchronously
* **dev** – Interpret block special devices on the filesystem
* **nodev** – Don’t interpret block special devices on the filesystem
* **suid** – Allow the operation of suid, and sgid bits. They are mostly used to allow users on a computer system to execute binary executables with temporarily elevated privileges in order to perform a specific task
* **nosuid** – Block the operation of suid, and sgid bits
* **noatime** – Don’t update inode access times on the filesystem. Can help performance
* **nodiratime** – Do not update directory inode access times on the filesystem. Can help performance
  + For last two points, see <https://wiki.archlinux.org/index.php/fstab#atime_options>
* **relatime** – Update inode access times relative to modify or change time. Access time is only updated if the previous access time was earlier than the current modify or change time. (similar to noatime, but doesn’t break mutt or other applications that need to know if a file has been read since the last time it was modified). Can help performance (see above link)
* **discard** – Issue TRIM (see <https://wiki.archlinux.org/index.php/Solid_state_drive#TRIM>) commands to the underlying block device when blocks are freed. Recommended to use if the filesystem is located on an SSD
* **flush** – The vfat option to flush data more often, thus making copy dialogs for progress bars to stay up until all data is written
* **nofail** – Mount device when present but ignore if absent. This prevents errors being reported at boot for removable media
* **defaults** – the default mount options for the filesystem to be used the default options for ext4 are: rw, suid, dev, exec, auto, nouser, async

**LVM – Logical Volume Manager**

* Physical volumes – disk – has a name: /dev/{sd or hd}{a letter: a, b…)}
* Volume groups – one or more physical volumes. Name usually vg\_{name}
* Logical volumes – partitions/”chunks” of a volume group
* Adding in physical volumes later, and linking them to volume groups and logical volumes, is possible
* GUI (deprecated) or command line
* <https://www.tecmint.com/add-new-disks-using-lvm-to-linux/>

**Adding a New Disk**

* fdisk is a command to manually configure disk partitioning, not used with LVM
* Usually use commands p, n, and w, in fdisk

**Creating a Mountable Partition**

* Use mkfs to take the raw partition created with fdisk and format it to be amountable partition

**Manually Mounting a Partition**

* You can mount the disk and ensure it is in working order before adding it permanently to fstab
* mount /dev/sdb/mnt/disk
* umount /mnt/disk
* Guide to disk add process: <https://www.tecmint.com/add-new-disk-to-an-existing-linux/>

|  |  |  |
| --- | --- | --- |
| Mount point | Size | fstype |
| /mnt/newdisk/p1 | 10 GB | xfs |
| /opt/newapp | 5 GB | ext3 |
| /opt/newapp2 | 4 GB | ext4 |

(in class – making partitions)

fstab file uses spaces [ ] to divide columns

**UUID vs “Text Name”**

* UUID (universally unique ID) is consistent no matter what the logical name is
* Find with blkid command
* Best choice when using SAN or hot-pluggable (USB) disk

**NFS**

* Network File System
* Mounted somewhat like a local disk in /etc/fstab

**CIFS/SMB**

* Standard Windows file sharing
* Need to make sure you have some support packages installed: sudo yum install samba-client samba-common cifs-utils will do so
* Add the entry to fstab (options to handle credentials)
* And then run the mount command

**ACLs – Access Control**

* The way to add multiple groups/users is by using setfacl this will allow for Access Control Lists against files and directories
* Add permissions to a user
  + setfacl –m “u:username:permissions”
  + Or: setfacl –m “u:uid:permissions”
* Add permissions to a group
  + setfacl –m “g:groupname:permissions”
  + Or: setfacl –m “g:gid:permissions”
* To remove all permissions: setfacl –b
* To check permissions:
  + getfacl filename
  + getfacl directoryname

**Inodes**

* What is an inode?
* How can we see an inode number?

**Links**

* Like shortcuts
* Hard link – must be on same physical partition, point to same inode
* Symbolic link – like a Windows shortcut, a new “file” that points

In class:

Reference Lab4 – AddDisk-LVM.docx from D2L

LVM – We are going to take 2 disks, put them into a volume group called vg\_data

Then we’ll put that into logical volumes called lv\_app and lv\_data

In terminal, switch to admin with su command, then type fdisk –l | grep /dev/sd to find partition names

Use command pvcreate to make partitions with LVM: we use pvcreate /dev/sdc and pvcreate /dev/sdd

Then make the volume group vg\_data and add the partitions to it with command vgcreate vg\_data /dev/sdc /dev/sdd note that you can use one vgcreate command and add both the partitions from the previous step at once; put a space between each one

vgdisplay <volume group name> to display data from the volume group.

In class: vgdisplay vg\_data

Create logical volumes: lvcreate –n <logical volume name> --size <number>{K,M,G} <volume group name>

In class: lvcreate –n lv\_app --size 12G vg\_data and lvcreate –n lv\_data --size 750M vg\_data

To see changes: lvdisplay <name of volume group> command. Note you’re entering the name of the volume group here, which will show all logical groups under that volume group

These logical volumes will be located at /dev/vg\_data/lv\_app and /dev/vg\_data/lv\_data

Make the mount points: mkdir /lvm then mkdir /lvm/app1 and mkdir /lvm/data these names aren’t important, just what Jay used

Then alter the fstab file, same as physical partitions: open the file with gedit /etc/fstab command

At the bottom of the fstab file:

# LVM vg\_data

/dev/vg\_data/lv\_app /lvm/app1 ext4 defaults 0 0

/dev/vg\_data/lv\_data /lvm/data ext3 defaults 0 0

Then mount with mount –a command, df –h will show the new partitions at the bottom of the report. These records were created when I ran it:

/dev/mapper/vg\_data-lv\_app 12G 41M 12G 1% /lvm/app1

/dev/mapper/vg\_data-lv\_data 725M 780K 686M 1% /lvm/data

For Lab 5: run command yum install tree to start