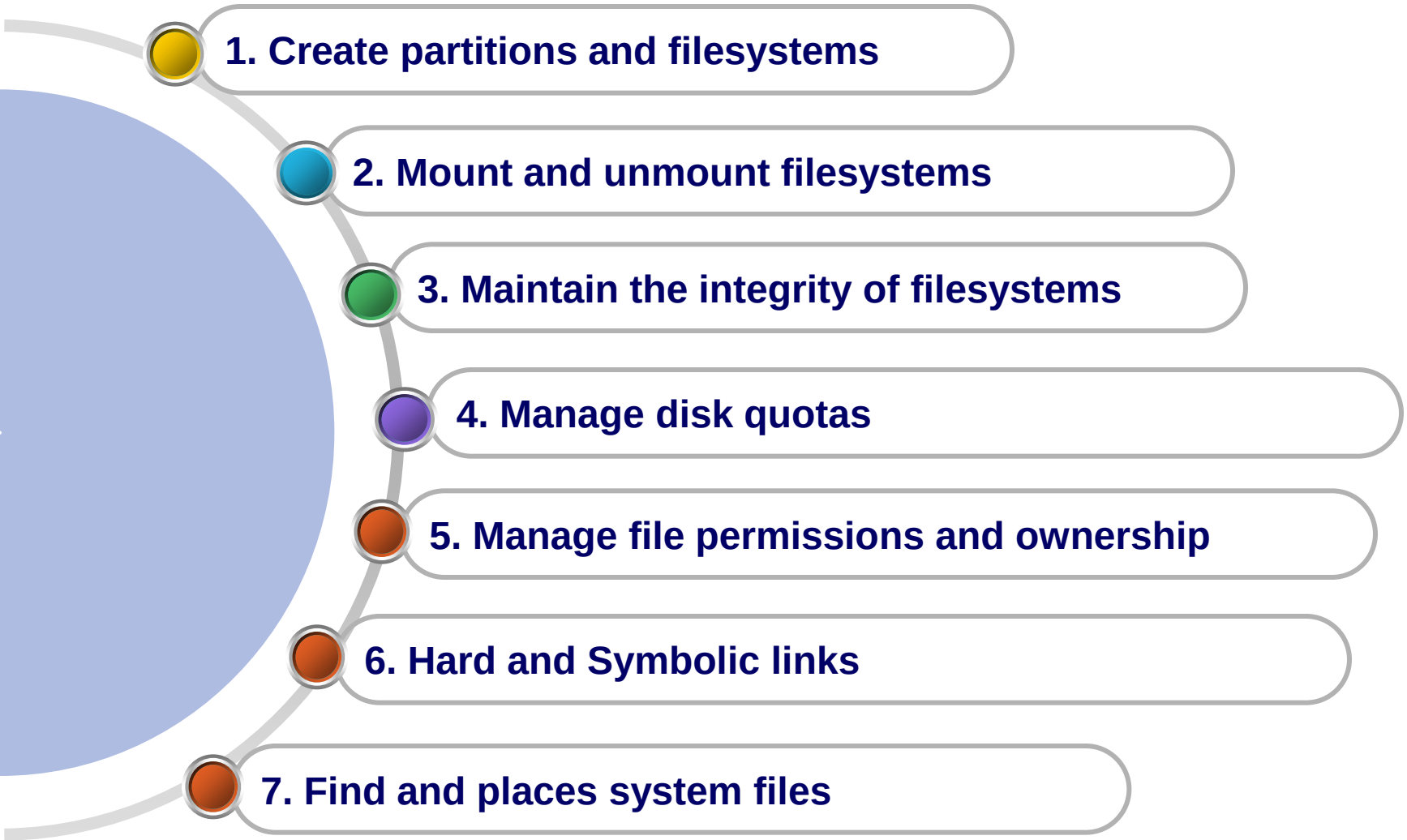
LPIC-1 TRAINING COURSE

Topic 104: Devices, Linux Filesystems & FHS

Contents


1. Create partitions and filesystems
2. Mount and unmount filesystems
3. Maintain the integrity of filesystems
4. Manage disk quotas
5. Manage file permissions and ownership
6. Hard and Symbolic links
7. Find and places system files

Objectives

- Configure disk partitions and create filesystems or swap space on media
- Maintain a standard filesystem or journaling filesystem
- Configure the mounting of a filesystem
- Manage disk quotas for user
- Control file access through the proper use of permission and ownership
- Create and manage hard and symbolic links to a file
- Understand the Filesystem Hierarchy Standard (FHS), including typical file locations and directory classifications

1. Create partitions and filesystems

Block devices and partitions

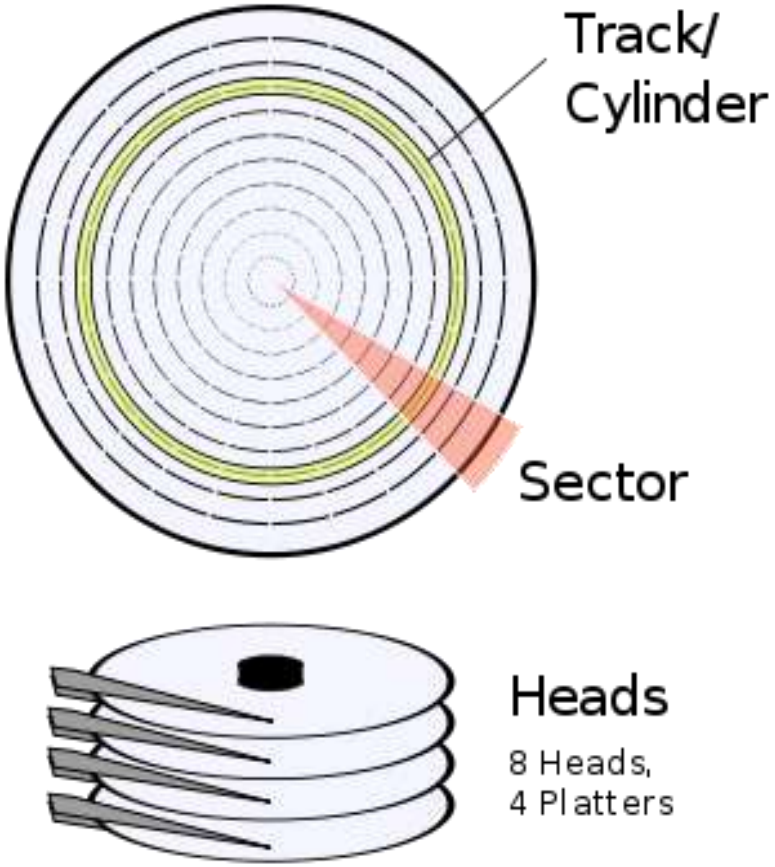
- ***Block* devices**: correspond to storage device that can be formmated in fixed-size blocks
  - Can be accessed randomly
  - *Block* vs *Character* device

```
[ian@echidna ~]$ ls -l /dev/loop1 /dev/null /dev/sd[ab] /dev/sr0 /dev/tty0
brw-rw----. 1 root disk   7,  1 2010-06-14 07:25 /dev/loop1
crw-rw-rw-. 1 root root   1,  3 2010-06-14 07:25 /dev/null
brw-rw----. 1 root disk   8,  0 2010-06-14 07:25 /dev/sda
brw-rw----. 1 root disk   8, 16 2010-06-14 07:25 /dev/sdb
brw-rw----+ 1 root cdrom 11,  0 2010-06-14 07:25 /dev/sr0
crw--w----. 1 root root   4,  0 2010-06-14 07:25 /dev/tty0
```

- ***Partition***: a physical division of an harddisk
  - Primary, Extended & Logical Partitions

Harddisk Geometry

- ***Track***: Areas on a hard disk that form a concentric circle of sectors
- ***Sector***: Smallest unit of data storage on a hard disk
  Cylinder: Series consisting of the same concentric track on all of the metal platters inside a HDD
- ***Block***: Combination of sectors, commonly used by filesystem commands



Displaying partition information

- Partition information is stored in a ***partition table*** on the disk
- **fdisk** can list, create, change, delete partition by editing ***partition table***

```
[root@localhost ~]# fdisk -l /dev/sda

Disk /dev/sda: 21.4 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1          13      104391   83  Linux
/dev/sda2              14        1288    10241437+  83  Linux
/dev/sda3            1289        1549     2096482+  82  Linux swap / Solaris
/dev/sda4            1550        2610     8522482+   5  Extended
/dev/sda5            1550        2186     5116671   83  Linux
```

Partition with *fdisk*

Warning

- Backup important data before you start
- Do not change partitions that are in use
- Know your tool: **fdisk** does not commit any changes until you tell it to
- Stop if you do make a mistake

Filesystem types

- ***Filesystem***: method for storing and organizing files
  - Each partition can have own filesystem
- ***Journaling***: allow much faster recovery after a system crash
  - Journaling filesystem is preferred over a non-journaling
- Linux supported filesystem types: ***ext2, ext3 (*), ReiserFS(*), XFS(*), swap(**), vfat***

Creating filesystems

- **mkfs** create filesystem
  - Syntax:
    `mkfs –t fstype [-l label] /dev/sdXY`
  - actually a frontend to several filesystem-specific commands: **`mkfs.ext3`**, **`mkfs.reiserfs`**...
    - **`ls /sbin/mk*`**
- **mkswap** create swap space
  - **`mkswap /dev/sdXY`**

Working with filesystem

- Set filesystem label: **`e2label, xfs_adm -L, reiserfstune, dosfslabel`**
- View filesystem UUID: **`blkid`**
- Convert ext2 to ext3 (add journal): **`tune2fs`**
- Change filesystem characteristics: **`tune2fs, xfs_adm, reserfstune`**

Other tools

- Partitioning tools: **`cfdisk, sfdisk, parted, gparted`**
- Logical Volume Manager (**LVM**)
- Redundant Array of Independent Disks (**RAID**)
- More filesystems: IBM *Journaled File System (**JFS**)*, *B-Tree file system (**btrfs**)*, ***cramfs**…*

2. Mount and unmount filesystem

Mounting filesystem

- Created filesystems is not usable until it is ***mounted*** at a ***mountpoint***
  - ***mountpoint*** must be exist before mounting
- Example:
  - List all mounted filesystem: **`mount`**
  - Mount filesystem:
    `mount /dev/sda2 /media/dos`
  - Mount filesystem with explicit type:
    `mount -t vfat /dev/sda2 /media/dos`
  - Remount filesystem with read-only option
    `mount -o remount,ro /media/dos`

/etc/fstab

- Contains the necessary information to automate mounting partitions
- Listed filesystems can be mount using only ***device name*** or ***mountpoint***
- Syntax:

```
root@pinguino:~# cat /etc/fstab
# /etc/fstab: sttic file system information.
#
#<file system> <mount point>   <type>  <options>       <dump>  <pass>
proc            /proc           proc    defaults        0       0
/dev/hda6       /               reiserfs defaults       0       1
/dev/hda2       /boot           ext3    defaults        0       2
/dev/hda8       /dos            vfat    defaults        0       0
/dev/hda7       /home           xfs     defaults        0       2
/dev/hda1       /media/hda1     ntfs    defaults        0       0
/dev/hda5       none            swap    sw              0       0
/dev/hdc        /media/cdrom0   udf,iso9660 user,noauto     0       0
/dev/fd0        /media/floppy0  auto    rw,user,noauto  0       0
```

Unmounting filesystem

- Removable media should be unmounted before removing
- Make sure there are no running processes that have open files on the filesystems
  - **`lsof`** determine what files are open
- Example:
  - **`lsof /media/dos`**
  - **`umount /media/dos`**

Swap space

- Swap space does not have a mountpoint
- Swap space defined in **/etc/fstab** is usually enabled in boot process
- Swap space can be manually controlled with **`swapon`** and **`swapoff`** commands
- View currently enable swap devices: **`cat /proc/swaps`**

3. Maintain the integrity of filesystems

Checking filesystems

- Filesystem may be corrpupted if system crashes or loses power
- Main tool for checking filesystem is **fsck**
  - really a front-end for various filesystem type (**dosfsck, e2fsck, fsck.ext2, fsck.ext3, fsck.msdos, fsck.xfs…**)
- Example:
  - **fsck /dev/sda7**
  - **fsck.ext3 LABEL=EXT3PARTITION**
  - **e2fsck UUID=7803f979-ffde-4e7f-891c**
- Do not attempt to check a mounted filesystem

Monitoring free space

- **df** displays information about mounted filesystem
  - Example: **`df -TH`**
- **tune2fs** inspect information about *ext2*, *ext3*, *ext4* filesystem
  - Example: **`tune2fs -l /dev/sda3`**
- **du** displays information about files/directories's size
  - Example: **`du -hs /tmp`**
    `du -shc /usr/*`

Repairing filesystems

- Filesystems in ***/etc/fstab*** are automatic checked and repair at boot-time
  - Failed automatic boot-time check will dump you into a single user shell to run **fsck** manually
- Manually check require the filesystem to be unmounted or mounted as *read-only*
  - root filesystem should be mounted *read-only* in single mode or booting recovery system
- Example: **`fsck -p /dev/sdb1`**

Advanced tools

- Ext2 and Ext3: **`tune2fs, dumpe2fs, debugfs`**
- ReiserFS: **`reiserfstune, debugreiserfs`**
- XFS: **`xfs_info, xfs_growfs, xfs_admin, xfs_repair, xfs_db`**

Exercise

1. Turn off your Linux Virtual Machine and add a new 10GB disk to it
2. Turn your VM back on. How can you find out your newly added disk?
3. Use **fdisk** to devide your new disk into 5 partitions, 2GB each.
4. Create the new folder **/tmp/mymount** and copy some files into it.
5. Create an **ext2** filesystem in the first partition, then manually mount it on **/tmp/mymount**. Where do the old files in this folder go?
6. Copy **/usr/bin/xclock** to **/tmp/mymount** and run this copy of **xclock** in background. Can you unmount **/tmp/mymount** now? Why?
7. Find out what process is accessing **/tmp/mymount**. Stop them and unmount this filesystem.
8. Convert the first partition to **ext3** filesystem then run **fsck** to check this filesystem. Verify that your files in this filesystem remain intact.
9. Create an **ext3** file system in the second partition, then manually mount it on **/sharefs** with *read-only* option. Verify that you cannot write to this filesystem.
10. Configure your system to automatically mount the second partition as read-write on reboot. Verify your work.

Hints to Exercise

1. Turn off your Linux Virtual Machine and add a new 10GB disk to it
2. Turn your VM back on. How can you find out your newly added disk? **`fdisk -l`**
3. Use **fdisk** to devide your new disk into 5 partitions, 2GB each? **`fdisk /dev/sdb`**
4. Create the new folder **/tmp/mymount** and copy some files into it.
   `mkdir /tmp/mymount`
   `cp /var/log/* /tmp/mymount`
5. Create an **ext2** filesystem in the first partition, then manually mount it on **/tmp/mymount**. Where do the old files in this folder go?
   `mkfs -t ext2 /dev/sdb1`
   `mount /dev/sdb1 /tmp/mymount`
6. Copy **/usr/bin/xclock** to **/tmp/mymount** and run this copy of **xclock** in background. Can you unmount **/tmp/mymount** now? Why?
   `cp /usr/bin/xclock /tmp/mymount`
   `/tmp/mymount/xclock &`
7. Find out what process is accessing **/tmp/mymount**. Stop them and unmount this filesystem.
   `lsof /tmp/mymount`
   `kill -9 <xclock's PID>`
   `cd / && umount /tmp/mymount`
8. Convert the first partition to **ext3** filesystem then run **fsck** to check this filesystem. Verify that your files in this filesystem remain intact: **`tune2fs -j /dev/sdb1; fsck /dev/sdb1`**
9. Create an **ext3** file system in the second partition, then manually mount it on **/sharefs** with *read-only* option. Verify that you cannot write to this filesystem.
   `mkfs -t ext3 /dev/sdb2`
   `mkdir /sharefs`
   `mount -o ro /dev/sdb2 /sharefs`
   `touch /sharefs/test`
10. Configure your system to automatically mount the second partition as read-write on reboot. Verify your work.
    `cp /etc/fstab /etc/fstab.bak`
    `echo "/dev/sdb2 /sharefs ext3 defaults 0 2" >> /etc/fstab`

4. Manage disk quotas

Quotas

- Allow you to control disk usage by user or by group
- Quotas must be enabled and managed by root
- Enable quotas requires filesystem to be mounted with ***usrquota*** or ***grpquota*** option

```
/dev/hda2       /boot           ext3    defaults,usrquota,grpquota    0   2
/dev/hda7       /home           xfs     defaults,usrquota,grpquota    0   2
```

- For XFS: Quotas information is stored in filesystem metadata
- For non-XFS: Quotas information is stored in the ***aquota.user*** and ***aquota.group*** binary files

Step 1: Enabling Quotas

- Mount filesystems with ***usrquota*** and/or ***grpquota*** option
  - Eg: **`mount -o usrquota,grpquota /dev/sdb1 /share`**
  - Add ***usrquota*** or ***grpquota*** to filesystems in ***/etc/fstab*** and remount the filesystems
- Run **quotacheck** to checks and creates the required ***aquota.user*** and ***aquota.group*** files
  - Syntax: **`quotacheck [options] [filesystem]`**
    Options:
    `-a` for all filesystems in **/etc/fstab** that are automount
    `-u` for user quotas (default)
    `-g` for group quotas
    `-v` for verbose output
  - Example: **`quotacheck -ug /share`**

Step 2: Setting Quota Limits

- **edquota** set quota for a particular user or group

```
Disk quotas for user ian (uid 1000):
  Filesystem          blocks       soft       hard     inodes     soft     hard
  /dev/hda2                0          0          0          0        0        0
  /dev/hda7             2948          0          0        172        0        0
```

  - View block size: **tune2fs -l device**

- Usages:
  - Editing user quotas: **edquota -u username**
  - Editing group quotas: **edquota -g groupname**
  - Copying quotas: **edquota -p protousr username**
  - Set grace period: **edquota -t**

Step 3: Turn on Quota Checking

- Run **quotaon** to turns on quota checking
- Syntax:

  `quotaon [options] [filesystem]`

  - Options:

    `-a` for all filesystems in **/etc/fstab** that are automount

    `-u` for user quotas (default)

    `-g` for group quotas

    `-v` for verbose output

- Example:

  `quotaon -ug /share`

Step 4: Checking Quotas

- Displays quotas information:
  quota
  quota -v [username]
- Generating quota reports:
  repquota -uga
  repquota –ug mountpoint
- Warning users who are over quota:
  warnquota

Exercise

1. Enable quota on the **/sharefs** filesystem you created in the previous exercise without rebooting your machine.
2. Run **`chmod a+rwx /sharefs`** to eable world-writeable in **/sharefs**
3. Run **`quotacheck`** to create **aquota.user** and **aquota.group**. Verify that these files exist.
4. Create a new user account named **user1** (**`useradd user1 && passwd user1`**). Set quotas for this user to restrict him in the **/sharefs** filesystem so that:
    1. **user1** cannot use more than 100MB in this filesystem. He'll get a warning when using more than 80MB
    2. **user1** canot create more than 10 files/folders in this filesystem. He'll get a warning when creating more than 5 files/folders
5. Turn on quota in **/sharefs** filesystem.
6. **su** to **user1** and try to create as much files as possible in **/sharefs** filesystem and see how quota works
7. Generating quota report on **/sharefs** filesystem with **`repquota`**

Hints to Exercise

1. Enable quota on the **/sharefs** filesystem you created in the previous exercise without rebooting your machine.
   - *Add "**usrquota,grpquota**" after "**defaults**" to the line of **/dev/sdb2** in **/etc/fstab** then run:*
   mount -o remount /sharefs
2. Run **chmod a+rwx /sharefs** to eable world-writeable in **/sharefs**
   chmod a+rwx /sharefs
3. Run **quotacheck** to create **aquota.user** and **aquota.group**. Verify that these files exist.
   quotacheck -ug /sharefs
4. Create a new user account named **user1** (**useradd user1 && passwd user1**). Set quotas for this user to restrict him in the **/sharefs** filesystem so that:
   1. **user1** cannot use more than 100MB in this filesystem. He'll get a warning when using more than 80MB
   2. **user1** canot create more than 10 files/folders in this filesystem. He'll get a warning when creating more than 5 files/folders

```
useradd user1 && passwd user1
tune2fs -l /dev/sdb2 | grep "block size"
edquota -u user1
/dev/sdb2        0          80000      100000     0          5          10
```

5. Turn on quota in **/sharefs** filesystem.
   quotaon -ug /sharefs
6. **su** to **user1** and try to create as much files as possible in **/sharefs** filesystem and see how quota works
   su - user1
   cp /tpm/* /sharefs
7. Generating quota report on **/sharefs** filesystem with **repquota**
   repquota -ug /sharefs

5. Manage files permission and ownership

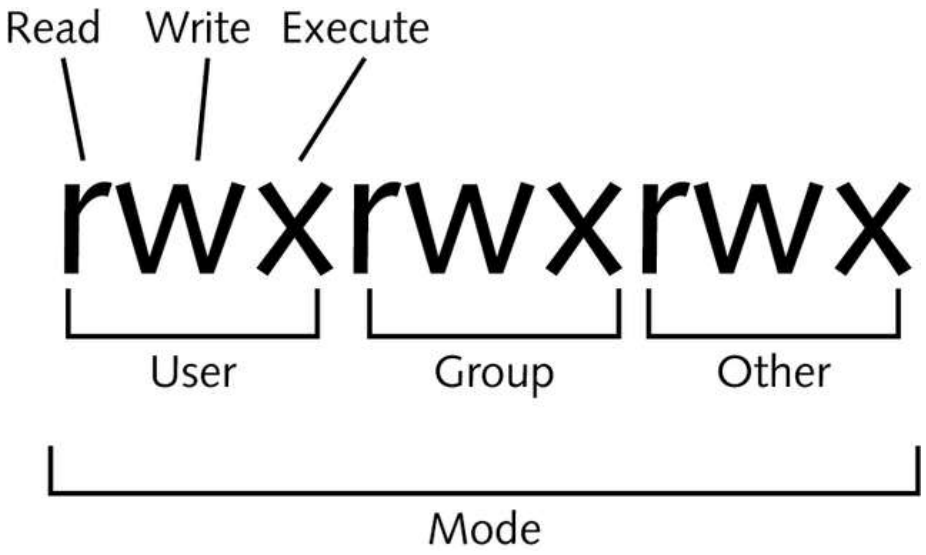
File ownership and permissions

- Every file has one owner and one group associated with it
- View file ownership & permissions: **`ls -l`**
- Change file ownership (only ***root*** can do)
  - Changing file's owner: **`chown user filenames`**
  - Changing file's owner and group:

    `chown user: filenames`

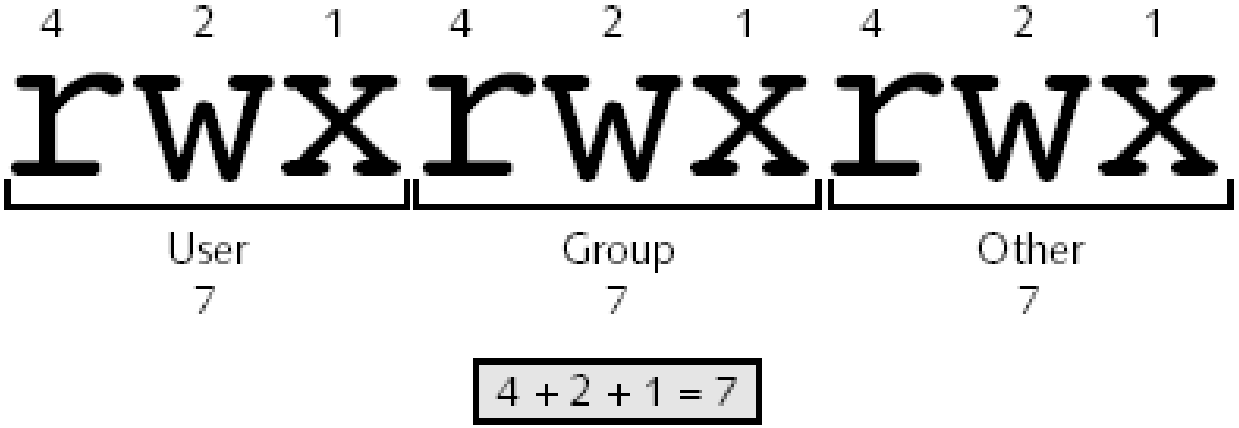
    or: **`chown user:group filenames`**
  - Changing file's group*:* **`chgrp group filenames`**
- File permission are specified seperately for
  - file's owner
  - member of file's group
  - everyone else

Intepreting the permission mode



Permission	Definition for Files	Definition for Directories
Read	Allows a user to open and read the contents of a file	Allows a user to list the contents of the directory (if he has also been given execute permission)
Write	Allows a user to open, read, and edit the contents of a file	Allows a user the ability to add or remove files to and from the directory (if he has also been given execute permission)
Execute	Allows a user the ability to execute the file in memory (if it is a program file) and shell scripts	Allows a user the ability to enter the directory and work with directory contents

Octal permission mode



Mode (one section only)	Corresponding Number
rwx	4 + 2 + 1 = 7
rw-	4 + 2 = 6
r-x	4 + 1 = 5
r--	4
-wx	2 + 1 = 3
-w-	2
--x	1
---	0

Changing permissions

- **chmod**: change mode (permissions) of files or directories

Category	Operation	Permission
u (user)	+ (adds a permission)	r (read)
g (group)	- (removes a permission)	w (write)
o (other)	= (makes a permission equal to)	x (execute)
a (all categories)		

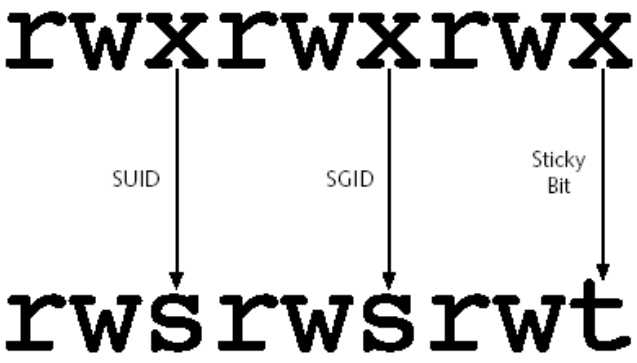
- Examples:
  - **chmod u+x g-w o= /tmp/testfile.txt**
  - **chmod a+rw /tmp/testfile.txt**
  - **chmod 644 /tmp/testfile.txt**
  - **chmod –R a-x /tmp**

Special permissions

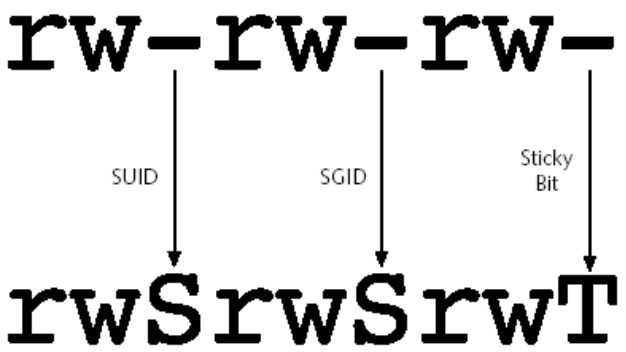
- **SUID** (Set User ID)
  - user who executes the file becomes owner of the file during execution
  - only applicable to binary compiled programs
- **SGID** (Set Group ID)
  - user who executes the file becomes member of the file's group during execution
- **Sticky bit**
  - only applicable to directories
  - ensure that a user can only delete his/her own files

Special permissions (cont')

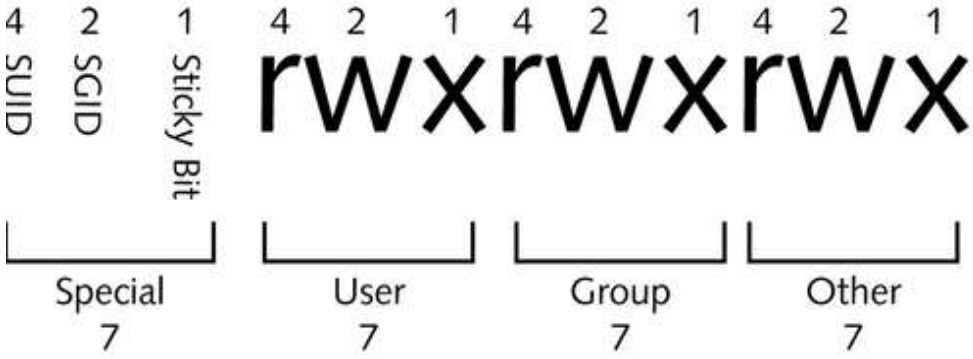
Execute permissions is set



Execute permissions is unset



Octal representation



Setting special permissions

- Still using **chmod** command
- Examples:
  - Set SUID:
    `chmod u+s /tmp/testfile`
  - Set SGID:
    `chmod g+s /tmp/testfile`
  - Set sticky bit:
    `chmod +t /tmp/`
  - Set SUID and SGID:
    `chmod 6644 /tmp/testfile`

The *umask*

- New files given ***rw-rw-rw*** *(666)* mode by default
- New directory given ***rwxrwxrwx*** *(777)* mode by default
- ***umask*** value change default permissions for new directories or files

	New Files	New Directories
Permissions assigned by system	rw-rw-rw-	rwxrwxrwx
- umask	0 2 2	0 2 2
= resulting permissions	rw-r--r--	rwxr-xr-x

	New Files	New Directories
Permissions assigned by system	rw-rw-rw-	rwxrwxrwx
- umask	0 0 7	0 0 7
= resulting permissions	rw-rw----	rwxrwx---

Exercise

You need 3 accounts for this exercise: root and two user account, each in a different group

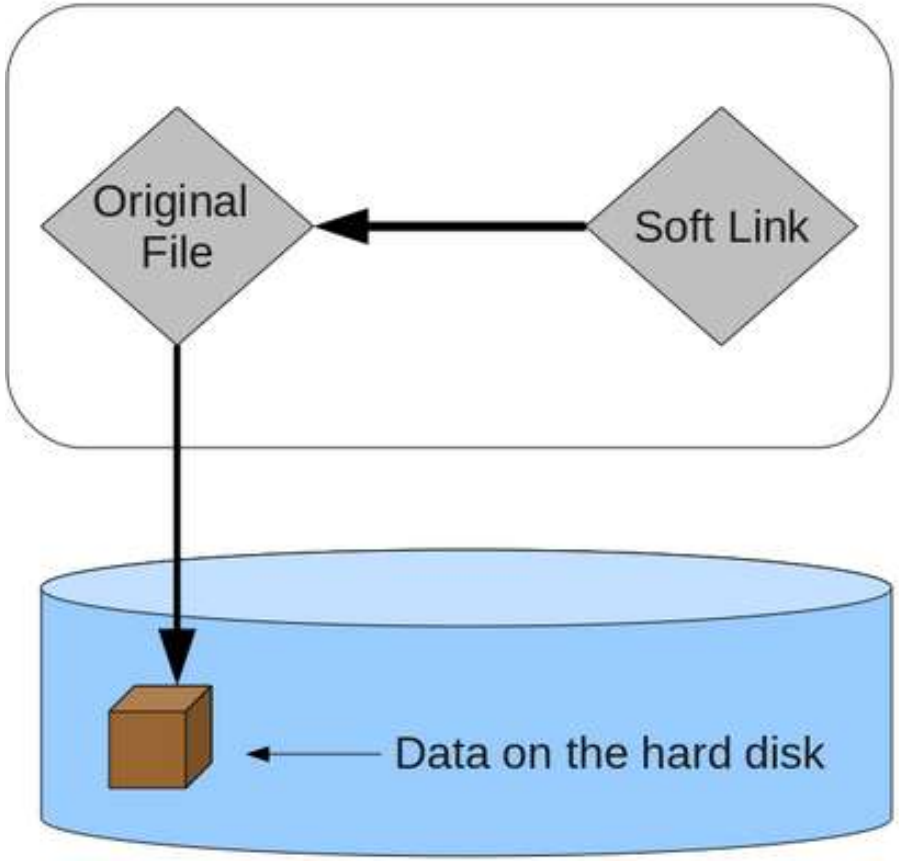
1. Log in 3 times using 3 virtual terminal, once at **root**, once as **user1** and once as **user2**.
2. As **root**, create a scratch directory - say **/tmp/scratch**
3. As **root**, give all users read and write access to scratch directory
4. In the **user1** and **user2** login session, change to scratch directory
5. As **user1**, copy a short text file to the scratch directory
6. As **user1**, set **0644** permissions on the file. Type **`ls -l`** and verify that the permission string in the first column matches this value: **-rw-r--r--**
7. As **user2**, try to access the file using **cat**. The file should appear on the screen
8. As **user2**, try to change the name of the file, then type **`ls`** to see whether this file's name is changed or not. Note that **user2** doesnt own the file but can rename it because **user2** can write to the directory in which the file resides
9. As **user2**, try to change the mode of the file to **0600**. The system should respond with an **`Operation not permitted`** error.
10. As **user2**, try to delete the file. The system should permit the deletion because **user2** can write to the directory in which the file resides.
11. As **user1**, repeat step 5 to re-create the test file
12. As **user1**, give the file more restrictive permissions like **0640**. Type **`ls -l`** to verify
13. As **user2**, repeat steps 7-10. The cat operation should fail but steps 8-10 should produce the same results

6. Hard and symbolic links

Soft link vs hard link

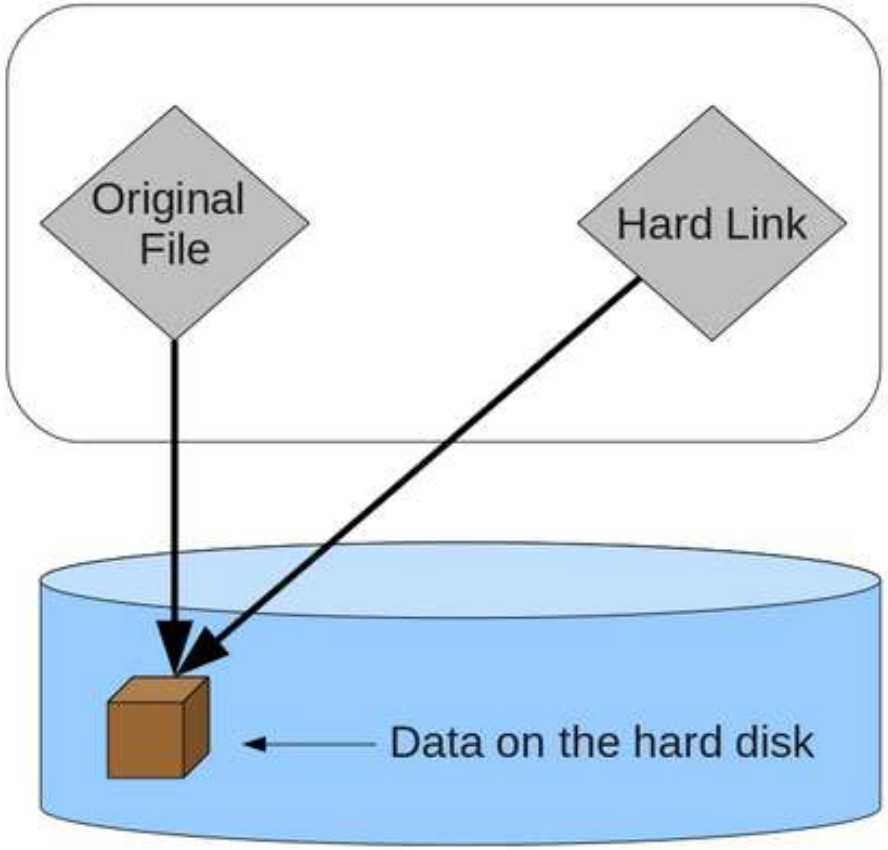
Soft Linking

A soft link is a file that is a pointer to another file. That other file points to the data on the hard disk. A soft link behaves similar to a Windows shortcut.



Hard Linking

A hard link is a direct pointer to the data on the hard disk. A hard link is identical to the original file, and any modifications you make to the hard linked version are made to the original as well, since you are modifying the same physical space on the hard disk.



Using Hardlink

- Use **ln** command to create hardlink to an existing file (not to a directory)
  - Syntax: **`ln sourcefile hardlink`**
- Hardlink must be in the same file system as orgininal file
- **`ls -l`** showing the number of hardlinks to a file
- File is not deleted until all hardlinks are deleted
- **find** command can be used to find hardlink
  - Example: **`find / -samefile /tmp/myfile`**

Using Softlink

- Use **`ln -s`** command to create softlink (symlink) to an existing file or directory
  - Syntax: **`ln -s sourcefile softlink`**
- Softlink can be create across file systems
- **`ls -l`** show the target of the link
- Deleting softlink does not affect the target file
- If the target file is moved or deleted, the softlink will be broken

7. Find and place system files

Filesytem Hierachy Standard

- Shareable vs. unshareable files:
  - *Shareable files* can be located on one system and used on another
  - *Unshareable files* must reside on the system on which they are used
- Static vs. variable files:
  - *Static files* change only through system administration intervention (documentation, libraries, binaries)
  - *Variable files* are subject to change by users and by system processes (mail, logs, databases, userdata)

	Shareable	Unshareable
Static	/usr /opt	/etc /boot
Variable	/var/mail /var/spool/news	/var/run /var/lock

FHS directories in the root filesystem

Directory	Description
/bin	Contains binary commands for use by all users
/boot	Contains the Linux kernel and files used by the boot loader
/dev	Contains device files
/etc	Contains system-specific configuration files
/home	Is the default location for user home directories
/lib	Contains shared program libraries (used by the commands in /bin and /sbin) as well as kernel modules
/mnt	Is the empty directory used for accessing (mounting) disks, such as floppy disks and CD-ROMs
/opt	Stores additional software programs
/proc	Contains process and kernel information
/root	Is the root user's home directory
/sbin	Contains system binary commands (used for administration)
/tmp	Holds temporary files created by programs
/usr/local	Is the location for most additional programs
/var	Contains log files and spools

which, type and *locate* commands

- **which** search your PATH and find out which command will be executed
- **type** command tell you how a command string will be evaluated for execution
- **locate** commands searches for matching files in a database
  - database is created or updated using the **updatedb** command

Thank You !

BACKUP SLIDES