information technology & management

viabiliit,

INTRO TO OPEN SOURCE

ILLINOIS INSTITUTE OF TECHNOLOGY

Managing Filesystems

Sean Hughes-Durkin

ITMO/IT-O 456 Fall 2017

Information Technology & Management Programs

School of Applied Technology

Objectives

At the end of this lesson students should be able to:

- Find files and directories on the filesystem
- Describe and create linked files
- Explain the function of the Filesystem Hierarchy Standard
- Use standard Linux commands to manage files and directories

Objectives

At the end of this lesson students should be able to:

- Modify file and directory ownership
- Define and change Linux file and directory permissions
- Identify the default permissions created on files and directories
- Apply special file and directory permissions
- Modify the default access control list (ACL)

Filesystem Hierarchy Standard (FHS)

- Standard set of directories on UNIX & Linux systems
  - Standard file and subdirectory contents
  - Simplifies the task of finding specific files
  - Gives Linux software developers the ability to locate files on any Linux system
    - Allows creation of non-distribution-specific software

Filesystem Hierarchy Standard (FHS)

- Comprehensive understanding of standard type of Linux directories valuable when locating and managing files and directories
- 3.0 release as of 6/3/2015 **http://www.linuxbase.org/betaspecs/fhs/**
  - Previous version is 2.3, dated 1/29/04 **http://refspecs.linuxfoundation.org/fhs.shtml**

Filesystem Hierarchy Standard (FHS)

- Summary of changes from 2.3 to 3.0
- The /run path is now specified, and reflects many of the common practices of current Linux distributions.
- A number of references to software current in 2004, the last release of the FHS, have been updated. For example, references to XFree86 have been removed, and character set examples have been altered to use Unicode rather than ISO charsets.
- Proper usage of a number of directories have been clarified, including /opt, /usr/local, and /srv.

Filesystem Hierarchy Standard (FHS)

Table 4-1: Linux directories defined by FHS

Directory	Description
/bin	Contains binary commands for use by all users (sysadmin,users)
/boot	Contains the Linux kernel and files used by the boot loader
/dev	Contains device files
/etc	Contains system-specific configuration files
/home	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	Empty directory used for accessing (mounting) disks such as floppy disks and CD-ROMs
/media	Contains subdirectories which are used as mount points for removable media such as floppy disks, cdroms and zip disks
/opt	Stores additional software programs (Solaris-style, instead of /usr/local)
/proc	Contains process and kernel information
/root	The root user's home directory
/run	Contains system information data describing the system since it was booted

Filesystem Hierarchy Standard (FHS)

Directory	Description
/sbin	Contains system binary commands (used for administration)
/tmp	Holds temporary files created by programs
/usr	Contains most system commands and utilities—will contain the following directories: **/usr/bin**—user binary commands **/usr/games**—educational programs and games **/usr/include**—C program header files **/usr/lib**—libraries **/usr/local**—local programs **/usr/sbin**—system binary commands **/usr/share**—files that are architecture independent **/usr/src**—source code **/usr/X11R6**—the XWindow system
/usr/local	Location for most additional programs
/var	Contains log files and spools

Table 4-1: Linux directories defined by FHS

cp Copy a file


```
cp file1 file2
```

- Copies a file
- This would copy file1 to file2 (or overwrite file2 with file1)
- If file2 is a directory name, it will copy the file into that directory
  - Arguments are a list of files
    - `cp file1 file2 file3 dst_dir/`
  - Can use wildcards

cp

Copy a file

```
[root@localhost itm]# touch file1
[root@localhost itm]# echo "Some Contents" > file1
[root@localhost itm]# cat file1
Some Contents
[root@localhost itm]# cp file1 file2
[root@localhost itm]# cat file2
Some Contents
[root@localhost itm]# ll
total 8
-rw-r--r--. 1 root root 14 Jan 31 13:36 file1
-rw-r--r--. 1 root root 14 Jan 31 13:36 file2
[root@localhost itm]# mkdir dst_dir
[root@localhost itm]# cp file1 file2 dst_dir/
[root@localhost itm]# ll dst_dir/
total 8
-rw-r--r--. 1 root root 14 Jan 31 13:37 file1
-rw-r--r--. 1 root root 14 Jan 31 13:37 file2
```

mv Move a file

mv file1 file2

- Renames or moves a file
  - If file2 is a directory name, moves the file to the new directory; i. e. **mv file1 News/** would move file1 to your News directory
- Pathnames can be absolute or relative
  - For multiple files, can use wildcards in pathname
    - **mv file* News/**
    - **mv /home/user/file* /home/user/News/**

mv

Move a file

```
[root@localhost itm]# touch file1
[root@localhost itm]# echo "Some Contents" > file1
[root@localhost itm]# cat file1
Some Contents
[root@localhost itm]# ll
total 4
-rw-r--r--. 1 root root 14 Jan 31 13:38 file1
[root@localhost itm]# mv file1 file2
[root@localhost itm]# cat file2
Some Contents
[root@localhost itm]# ll
total 4
-rw-r--r--. 1 root root 14 Jan 31 13:38 file2
[root@localhost itm]# touch file1
[root@localhost itm]# mkdir dst_dir
[root@localhost itm]# mv file1 file2  dst_dir/
[root@localhost itm]# ll dst_dir/
total 4
-rw-r--r--. 1 root root  0 Jan 31 13:39 file1
-rw-r--r--. 1 root root 14 Jan 31 13:38 file2
[root@localhost itm]# mv dst_dir/ new_dst_dir/
[root@localhost itm]# ll
total 4
drwxr-xr-x. 2 root root 4096 Jan 31 13:39 new_dst_dir
```

`rm` Remove a file

`rm filename`

- Type **`rm filename`** and hit enter (but beware: when you hit enter, it's gone for good!)
  - Arguments are a list of files
  - Can use wildcards
- Interactive mode can be configured
  - Add alias rm='rm -i' to .bashrc
    - Default alias in Fedora
  - Use **`-f`** option to override

Managing Files and Directories

- **`mkdir`** command
  - Creates new directories
  - Arguments specify directory's absolute or relative pathname
- Interactive mode
  - Prompts user before overwriting files
  - **`-i`** option (if not the default; normally is)
  - **`-f`** option (force): Overrides interactive mode

Managing Files and Directories

- Recursive
  - Referring to itself and its own contents
  - Recursive copy command copies the directory and all subdirectories and contents
  - Recursive search includes all subdirectories in a directory and their contents
  - Use **`-r`** or **`-R`** option

Managing Files and Directories


```
[root@localhost itm]# mkdir dst_dir
[root@localhost itm]# touch dst_dir/file{1..5}
[root@localhost itm]# mkdir dst_dir/sub_dst_dir
[root@localhost itm]# ll -R
.:
total 4
drwxr-xr-x. 3 root root 4096 Jan 31 14:14 dst_dir

./dst_dir:
total 4
-rw-r--r--. 1 root root    0 Jan 31 14:14 file1
-rw-r--r--. 1 root root    0 Jan 31 14:14 file2
-rw-r--r--. 1 root root    0 Jan 31 14:14 file3
-rw-r--r--. 1 root root    0 Jan 31 14:14 file4
-rw-r--r--. 1 root root    0 Jan 31 14:14 file5
drwxr-xr-x. 2 root root 4096 Jan 31 14:14 sub_dst_dir

./dst_dir/sub_dst_dir:
total 0
[root@localhost itm]# cp dst_dir/ new_dst_dir/
cp: omitting directory 'dst_dir/'
[root@localhost itm]# ll
total 4
drwxr-xr-x. 3 root root 4096 Jan 31 14:14 dst_dir
```

Managing Files and Directories

```
[root@localhost itm]# cp -R dst_dir/ new_dst_dir/
[root@localhost itm]# ll -R
.:
total 8
drwxr-xr-x. 3 root root 4096 Jan 31 14:14 dst_dir
drwxr-xr-x. 3 root root 4096 Jan 31 14:16 new_dst_dir

./dst_dir:
total 4
-rw-r--r--. 1 root root    0 Jan 31 14:14 file1
-rw-r--r--. 1 root root    0 Jan 31 14:14 file2
-rw-r--r--. 1 root root    0 Jan 31 14:14 file3
-rw-r--r--. 1 root root    0 Jan 31 14:14 file4
-rw-r--r--. 1 root root    0 Jan 31 14:14 file5
drwxr-xr-x. 2 root root 4096 Jan 31 14:14 sub_dst_dir

./dst_dir/sub_dst_dir:
total 0

./new_dst_dir:
total 4
-rw-r--r--. 1 root root    0 Jan 31 14:16 file1
-rw-r--r--. 1 root root    0 Jan 31 14:16 file2
-rw-r--r--. 1 root root    0 Jan 31 14:16 file3
-rw-r--r--. 1 root root    0 Jan 31 14:16 file4
-rw-r--r--. 1 root root    0 Jan 31 14:16 file5
drwxr-xr-x. 2 root root 4096 Jan 31 14:16 sub_dst_dir
```

Managing Files and Directories

- **`rmdir`** command
  - Removes directories
  - Arguments are a list of directories
  - Can use wildcards
  - Interactive mode by default
    - Use **`-f`** option to override
  - Directory must be empty to be removed
    - To delete a directory and all contents (subdirectories and files), use **`rm -r`**

Managing Files and Directories

- **Alias** command will display any aliases you have defined for your profile.

sean@localhost:~

File Edit View Search Terminal Help

```
[sean@itmo456 ~]$ alias
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
alias ls='ls --color=auto'
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot --show-ti
lde'
```

Managing Files and Directories

Command	Description
mkdir	Creates directories
rmdir	Removes empty directories
mv	Moves/renames files and directories
cp	Copies files and directories full of files (with the –r option)
alias	Displays BASH shell aliases
rm	Removes files and directories full of files (with the –r option)

Table 4-2: Common Linux file management commands

Finding Files: **`locate`** command

- Fastest method to search for files in the Linux directory tree
  - Receives full or partial filename as argument
  - Shortcut to **`slocate`** (or secure locate) command
- Uses indexed database of all files on system
  - To update the database use **updatedb** command
- Often returns too much information to display on the screen, as it searches all files on the filesystem; use with **`less`** or **`more`**

Finding Files: **locate** command

```
[sean@itmo456 ~]$ locate passwd
locate: can not stat () `/var/lib/mlocate/mlocate.db': No such file or directory
[sean@itmo456 ~]$ su -c "updatedb"
Password:
[sean@itmo456 ~]$ locate passwd
/etc/passwd
/etc/passwd-
/etc/passwdqc.conf
/etc/pam.d/passwd
/etc/security/opasswd
/usr/bin/gpasswd
/usr/bin/grub2-mkpasswd-pbkdf2
/usr/bin/lppasswd
/usr/bin/passwd
/usr/bin/smbpasswd
/usr/bin/vino-passwd
/usr/bin/vncpasswd
/usr/lib/firewalld/services/kpasswd.xml
/usr/lib64/libpasswdqc.so.0
/usr/lib64/libreoffice/share/config/soffice.cfg/svx/ui/passwd.ui
/usr/lib64/samba/libsmbpasswdparser.so
/usr/lib64/samba/pdb/smbpasswd.so
/usr/lib64/security/pam_passwdqc.so
/usr/lib64/security/pam_unix_passwd.so
```

Finding Files: **find** command

- Recursively search for files starting from a specified directory
  - Slower than locate command, but more versatile
- Format: **find <start directory> -criteria <what to find>**
  - e.g., **find /root –name project**

Finding Files: **`find`** command

- Can use many different file attributes as a search option
- If using wildcard metacharacters, ensure that they are interpreted by the **`find`** command
  - Place wildcards in quotation marks
- To reduce search time, specify subdirectory to be searched

Finding Files

Criteria	Description
-amin -x **-amin +x**	Searches for files that were accessed less than x minutes ago Searches for files that were accessed more than x minutes ago
-atime -x **-atime +x**	Searches for files that were accessed less than x days ago Searches for files that were accessed more than x days ago
-empty	Searches for empty files or directories
-fstype x	Searches for files if they are on a certain filesystem x (where x could be ext2, ext3, etc.)
-group x	Searches for files that are owned by a certain group or GID (x)
-inum x	Searches for files that have an inode number of x
-mmin -x **-mmin +x**	Searches for files that were modified less than x minutes ago Searches for files that were modified more than x minutes ago
-mtime -x **-mtime +x**	Searches for files that were modified less than x days ago Searches for files that were modified more than x days ago

Table 4-3: Common criteria used with find command

Finding Files

Criteria	Description
-name x	Searches for a certain filename x (x may contain wildcards)
-regexp x	Searches for certain filenames using regular expressions instead of wildcard metacharacters
-size -x **-size x** **-size +x**	Searches for files with a size less than x Searches for files with a size of x Searches for files with a size greater than x
-type x	Searches for files of type x where x is: • **b** for block files • **c** for character files • **d** for directory files • **p** for named pipes • **f** for regular files • **1** for symbolic links (shortcuts) • **s** for sockets
-user x	Searches for files owned by a certain user or UID (x)

Table 4-3: Common criteria used with find command

Finding files by Name or Size

- Find files by name
  - Searches filesystem for files' name
  - Example: **`find /etc -name passwd`**
- Find files by size
  - Searches filesystem for files' size
  - M=megabyte G=gigabyte
  - Example:

```
find /usr/share/ -size +10M
```

Finding files by User or Group

- Search filesystem by file owner or group
- User Example:

```
find /home -user christine -ls
```

- Group Example:

```
find /home -group ntp -ls
```

- Note the **-ls** option produces a long listing of the found files

Finding files by Permission

- Search filesystem for files with particular permissions set
- Permissions denoted by three-digit numbers
- Example: **`find /bin -perm 755 -ls`**

More Finding Files by Permission

- Hyphen (-) in front of the permission number:
  - means all three bits listed must match
  - Example:

```
find /home/Christine/ -perm -222 -type d –ls
```

  - Note the **-type** option allows you to search for only files (**f**) or only directories (**d**)
- Plus sign (**+**) in front of permission number:
  - means any of the three bits can match
  - Example: **find readonly -perm +222 -type f**

File Timestamps

- Date and time stamps are stored for each file when
  - Created
  - Accessed
  - Contents are modified
  - Metadata is changed, including:
    - Owner
    - Group
    - Timestamp
    - Filesize
    - permissions

Search for Files by Date and Time

- Can search for when a file was
  - accessed
  - changed
  - Metadata was changed
- By days
  - `-atime`
  - `-ctime`
  - `-mtime`
- By minutes
  - `-amin`
  - `-cmin`
  - `-mmin`

Finding Files by Date and Time

- The time is denoted by number and
  - Hyphen (-) to indicate time from now to the denoted number of minutes/days ago
  - Or plus sign (+) to indicate the minimum number of minutes/days ago and older
  - Or no marks for an exact match
- Example
  - Find a file that was changed in the /etc directory at least 10 minutes ago: **`find /etc/ -mmin -10`**

Using not and or When Finding Files

- To further refine searches the **`not`**, **`and`**, & **`or`** options can be used
- **`-not`** option
  - Allows **`find`** to match one item and not the other
  - Example:

```
find /home/ -user joe -not -group joe -ls
```

Using not and or When Finding Files

- **`-and`** option
  - Allows **`find`** to match one item and the other
  - Example: **`find /home/ -user joe -and -group ann –ls`**
- **`-or`** option
  - Allows **`find`** to match one item or the other
  - Example: **`find /home/ \(-user joe -o -group ann\) -ls`**

Finding files and Executing Commands

- The **-exec** option:
  - once file is found, a command can be executed upon it
  - Example: **find /etc/ -mmin -10 -exec echo "Here it is:{}" \;**
- The **-ok** option:
  - similar to -exec option, except it will ask before it executes the command on the file
  - Example: **find / -user joe -ok rm {} \;**

Finding Files: **find** command


```
[sean@itmo456 ~]$ find /usr/ -name passwd
/usr/bin/passwd
/usr/share/bash-completion/completions/passwd
/usr/share/doc/passwd
find: '/usr/share/polkit-1/rules.d': Permission denied
find: '/usr/libexec/initscripts/legacy-actions/auditd': Permission denied
[sean@itmo456 ~]$
[sean@itmo456 ~]$ find /usr/ -name "*passwd"
/usr/bin/lppasswd
/usr/bin/gpasswd
/usr/bin/vino-passwd
/usr/bin/vncpasswd
/usr/bin/smbpasswd
/usr/bin/passwd
/usr/share/bash-completion/completions/gpasswd
/usr/share/bash-completion/completions/smbpasswd
/usr/share/bash-completion/completions/passwd
/usr/share/bash-completion/completions/chpasswd
/usr/share/bash-completion/completions/ldappasswd
/usr/share/bash-completion/completions/htpasswd
/usr/share/doc/passwd
find: '/usr/share/polkit-1/rules.d': Permission denied
/usr/sbin/chpasswd
/usr/sbin/lpasswd
find: '/usr/libexec/initscripts/legacy-actions/auditd': Permission denied
[sean@itmo456 ~]$
```

Finding Files: **find** command


```
[sean@itmo456 ~]$ find /boot -type d
/boot
/boot/extlinux
/boot/lost+found
find: '/boot/lost+found': Permission denied
/boot/grub2
/boot/grub2/fonts
/boot/grub2/i386-pc
/boot/grub2/themes
/boot/grub2/themes/system
/boot/grub2/locale
/boot/efi
/boot/efi/EFI
/boot/efi/EFI/BOOT
/boot/efi/EFI/fedora
/boot/efi/EFI/fedora/fonts
/boot/efi/System
/boot/efi/System/Library
/boot/efi/System/Library/CoreServices
[sean@itmo456 ~]$
```

Finding Files: `which` command

- **`which`** command
  - Used to locate files that exist within directories listed in the PATH variable
  - If file not found, lists directories searched
- PATH variable
  - Lists directories on system where executable files are located
  - Allows executable files to be run without specifying absolute or relative path

Finding Files: **which** command


```
[sean@itmo456 ~]$ which ifconfig
/usr/sbin/ifconfig
[sean@itmo456 ~]$
[sean@itmo456 ~]$ which passwd
/usr/bin/passwd
[sean@itmo456 ~]$
[sean@itmo456 ~]$ which which
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot --show-ti
lde'
        /usr/bin/alias
        /usr/bin/which
[sean@itmo456 ~]$
[sean@itmo456 ~]$ which ll
alias ll='ls -l --color=auto'
        /usr/bin/ls
[sean@itmo456 ~]$
[sean@itmo456 ~]$ which vi
/usr/bin/vi
[sean@itmo456 ~]$
[sean@itmo456 ~]$ which ping
/usr/bin/ping
[sean@itmo456 ~]$
[sean@itmo456 ~]$ which test1
/usr/bin/which: no test1 in (/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/s
bin:/home/sean/.local/bin:/home/sean/bin)
[sean@itmo456 ~]$
```

Linking Files

- Files may be linked to another in one of two ways
- **Symbolic link**
  - One file is a pointer or a shortcut to another file (aka symlink)
- **Hard link**
  - Two files share the same data

Linking Files

- To better understand how files are linked, you must understand how files are stored on a filesystem
- On a structural level, a filesystem has three main sections:
  - The superblock
  - The inode table
  - Data blocks

Superblock

- The superblock contains general information about the filesystem including:
  - Number of inodes
  - Number of data blocks
  - Size of data blocks

The Inode Table and Inodes

- Inode table consists of inodes (information nodes)
- Each inode describes a file or directory
  - Unique inode number
  - File size in bytes
  - Number of data blocks in the file
  - An array of (usually 15) data block pointers
  - Various timestamp attributes (date last modified, etc.)
  - File type and access rights/permissions
  - Owner and group identifiers

Example Inode File Types

- Regular file
  - Need data blocks when it starts to have data
- Directory file
  - Special kind of file whose data blocks store filenames with corresponding inode numbers
    - Each directory structure contains inode number, entry length, name length, file type, & file name
    - Variable length structure, padded to be a multiple of 4

http://www.inf.fu-berlin.de/lehre/SS01/OS/Lectures/Lecture16.pdf; Freie Universität Berlin Institut

Example Inode File Types

- Symbolic link
  - Up to 60 characters are stored in the data block pointer array of the inode structure for "fast" symbolic links
  - If longer than 60 characters, a data block is required

Data Blocks

- Data blocks store:
  - The data that makes up the file
    - Directory data blocks store a list of the file and directory names contained in the directory with inode numbers
  - The filename
- Data blocks are referenced by the inode
- In operating-system-neutral language, these are *file allocation units*

Linking Files

- **`ln`** (link) command
  - Creates hard and symbolic links
- **`ln`** requires two arguments:
  - The existing file to link
  - The target file that will be created as a link to the existing file
  - Use **`-s`** option to create symbolic links
- Hard linked files **share inodes**
- Data blocks in symbolically linked files contain pathname to target file

Linking Files

- Hard linked files share the same inode and inode number
  - Must reside on the same filesystem
- To remove hard linked files, delete one of the linked files
  - Reduces the link count for the file
- Symbolic linked files do not share the same inode and inode number with their target file

Linking Files

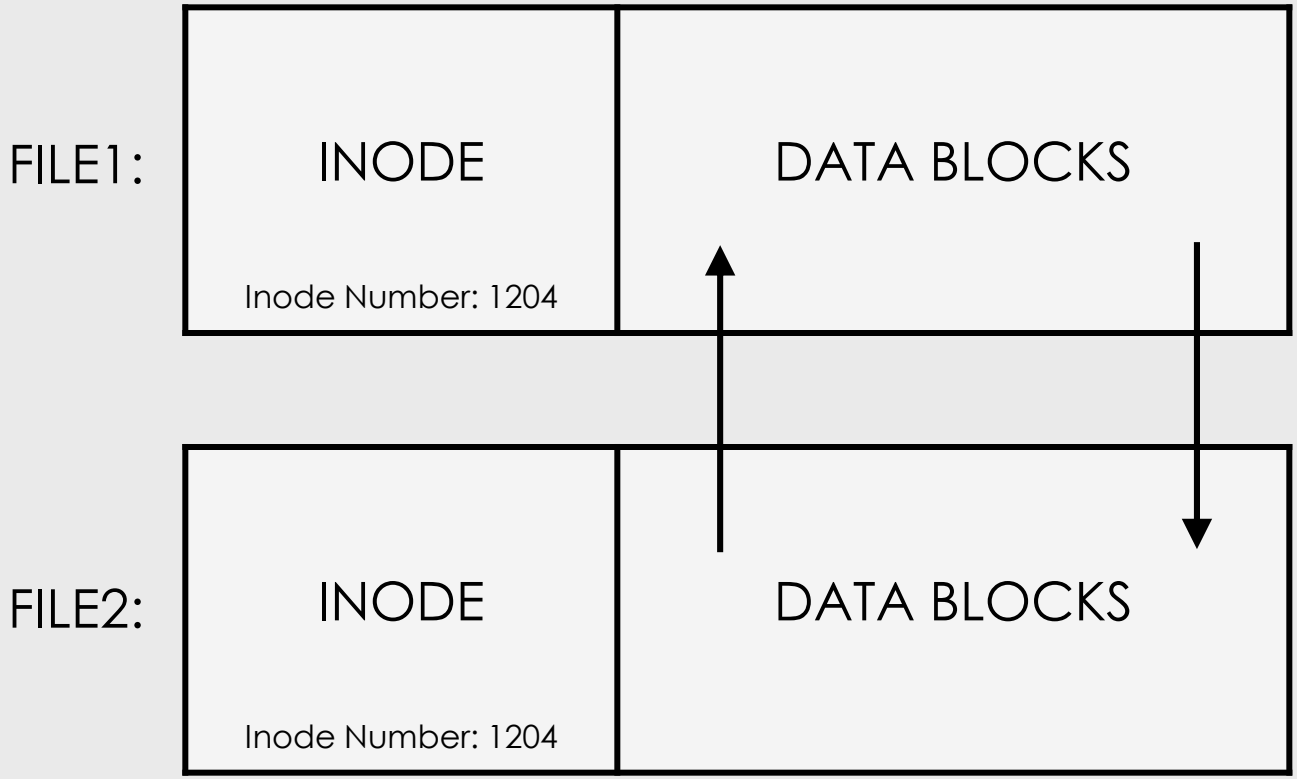

Figure 4-1: The structure of hard-linked files

Linking Files

```
[sean@itmo456 ~]$ ll recipe.txt
-rw-rw-r--. 1 sean sean 28 Sep 20 17:18 recipe.txt
[sean@itmo456 ~]$
[sean@itmo456 ~]$ ln recipe.txt /shared/shared_recipe.txt
[sean@itmo456 ~]$
[sean@itmo456 ~]$ ll recipe.txt
-rw-rw-r--. 2 sean sean 28 Sep 20 17:18 recipe.txt
[sean@itmo456 ~]$
[sean@itmo456 ~]$ ll -i recipe.txt
552069 -rw-rw-r--. 2 sean sean 28 Sep 20 17:18 recipe.txt
[sean@itmo456 ~]$
[sean@itmo456 ~]$ ll -i /shared/shared_recipe.txt
552069 -rw-rw-r--. 2 sean sean 28 Sep 20 17:18 /shared/shared_recipe.txt
[sean@itmo456 ~]$
[sean@itmo456 ~]$ ln recipe.txt /tmp/recipe.txt
ln: failed to create hard link '/tmp/recipe.txt' => 'recipe.txt': Invalid cross-
device link
[sean@itmo456 ~]$
[sean@itmo456 ~]$ df -h
Filesystem                Size  Used Avail Use% Mounted on
devtmpfs                  1.8G     0  1.8G   0% /dev
tmpfs                     1.8G  148K  1.8G   1% /dev/shm
tmpfs                     1.8G  920K  1.8G   1% /run
tmpfs                     1.8G     0  1.8G   0% /sys/fs/cgroup
/dev/mapper/fedora-root    11G  3.9G  6.1G  39% /
tmpfs                     1.8G  292K  1.8G   1% /tmp
/dev/sda1                 477M  115M  334M  26% /boot
/dev/sr0                  953M  953M     0 100% /run/media/sean/Fedora-Live-Deskt
op-x86_64-20-1
[sean@itmo456 ~]$
```

Linking Files

- Symbolic linked file is a pointer to the target file
  - Data blocks in the linked file contain only a pathname for the target file
    - Linked file & target file have different sizes
  - Editing symbolic linked file actually edits the target file
- If the target file is deleted, symbolic link serves no function

Linking Files

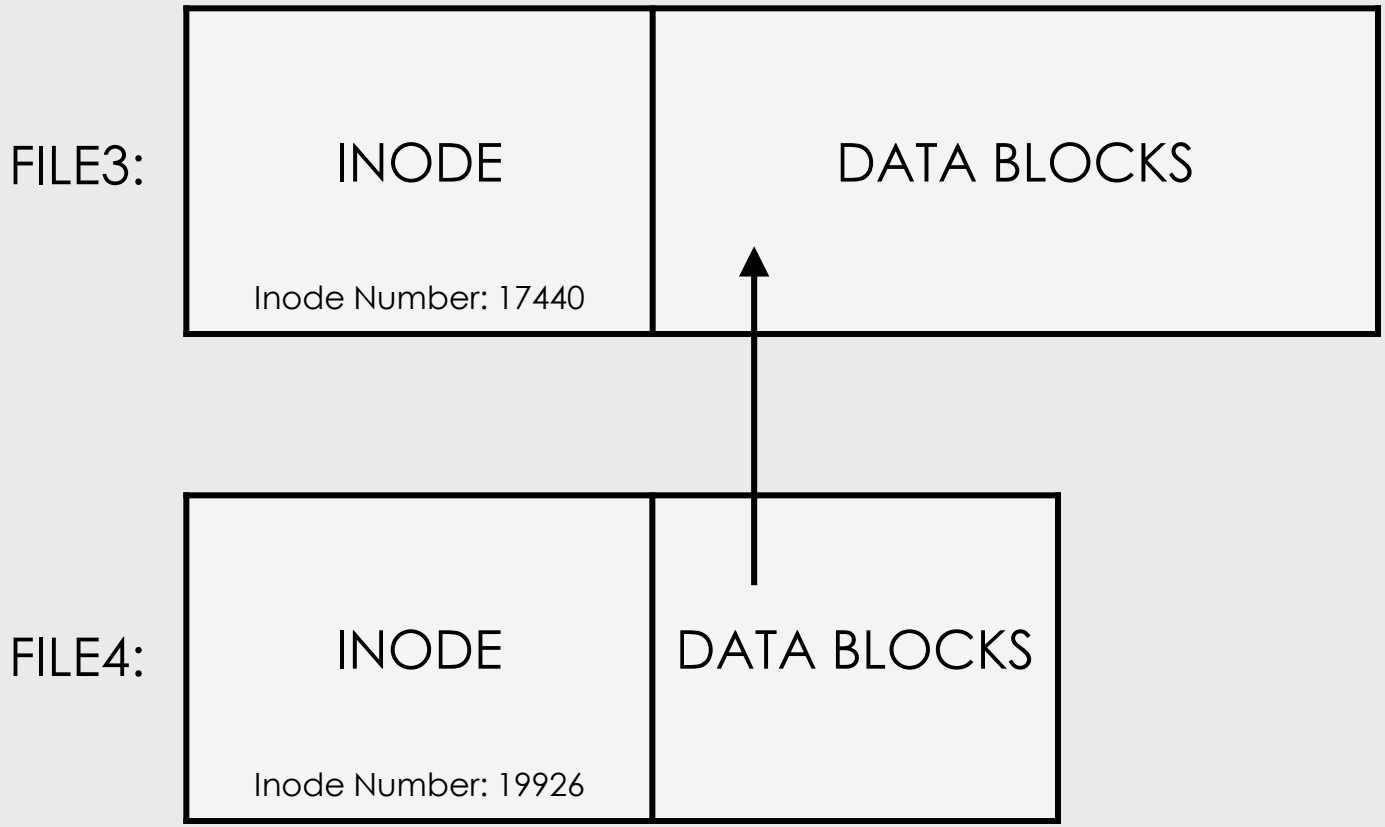

Figure 4-2: The structure of symbolically linked files

Linking Files

- To create a symbolic link, use the **`-s`** option with the **`ln`** command
  - Two arguments: existing file to link and target file to create as a link to existing file
  - Arguments can be relative or absolute pathnames, as with hard links
- Use the **`ll`** command to view both hard link and symbolic link files
- Symbolic links need not reside on the same filesystem as their target

Linking Files

```
[sean@itmo456 ~]$ ll -i recipe.txt
552069 -rw-rw-r--. 2 sean sean 28 Sep 20 17:18 recipe.txt
[sean@itmo456 ~]$
[sean@itmo456 ~]$ ln -s /home/sean/recipe.txt /shared/linked_recipe.txt
[sean@itmo456 ~]$
[sean@itmo456 ~]$ ll -i recipe.txt
552069 -rw-rw-r--. 2 sean sean 28 Sep 20 17:18 recipe.txt
[sean@itmo456 ~]$
[sean@itmo456 ~]$ ll -i /shared/linked_recipe.txt
791 lrwxrwxrwx. 1 sean sean 21 Sep 20 17:33 /shared/linked_recipe.txt -> /home/s
ean/recipe.txt
[sean@itmo456 ~]$
[sean@itmo456 ~]$ ln -s recipe.txt /shared/badlink_recipe.txt
[sean@itmo456 ~]$
[sean@itmo456 ~]$ ll -i /shared/badlink_recipe.txt
798 lrwxrwxrwx. 1 sean sean 10 Sep 20 17:34 /shared/badlink_recipe.txt -> recipe
.txt
[sean@itmo456 ~]$
```

File and Directory Permissions

- All users must successfully login with a username and password to gain access to a Linux system
- Users are identified by their username & group memberships
  - All access to resources depends on username and group membership
  - Must have required **permissions**

File and Directory Ownership

- Primary group
  - Default group to which a user belongs
- During file creation, file's owner and group owner set to user's username and primary group
  - Same for directory creation

File and Directory Ownership

- **`whoami`** command
  - View current user name
- **`groups`** command
  - View group memberships and primary group
- **`touch`** command
  - Used to create new files
  - Originally used to update timestamp on a file (we still can use it that way)

File and Directory Ownership


```
[sean@itmo456 ~]$ whoami
sean
[sean@itmo456 ~]$
[sean@itmo456 ~]$ groups
sean
[sean@itmo456 ~]$
[sean@itmo456 ~]$ touch file1 file2 file3 file4 file5
[sean@itmo456 ~]$
[sean@itmo456 ~]$ touch test{1..5}
[sean@itmo456 ~]$
[sean@itmo456 ~]$ ll file* test*
-rw-rw-r--. 1 sean sean 0 Sep 20 17:37 file1
-rw-rw-r--. 1 sean sean 0 Sep 20 17:37 file2
-rw-rw-r--. 1 sean sean 0 Sep 20 17:37 file3
-rw-rw-r--. 1 sean sean 0 Sep 20 17:37 file4
-rw-rw-r--. 1 sean sean 0 Sep 20 17:37 file5
-rw-rw-r--. 1 sean sean 0 Sep 20 17:37 test1
-rw-rw-r--. 1 sean sean 0 Sep 20 17:37 test2
-rw-rw-r--. 1 sean sean 0 Sep 20 17:37 test3
-rw-rw-r--. 1 sean sean 0 Sep 20 17:37 test4
-rw-rw-r--. 1 sean sean 0 Sep 20 17:37 test5
[sean@itmo456 ~]$
```

File and Directory Ownership

- **`chown`** (change owner) command
  - Command used to change the owner and group owner of a file or directory
  - Takes two arguments at a minimum:
    - New group owner
    - Files or directories to change
  - Can use **`-R`** option for contents of directory

File and Directory Ownership

- Regular user
  - Cannot change ownership of a file or directory belonging to another user
- root user
  - Can change ownership of a file or directory belonging to another user
- Examples
  - To change user owner of a file or directory: **`chown joe memo.txt`**
  - To change user and group of a file or directory **`chown joe:joe memo.txt`**

File and Directory Ownership


```
[root@itmo456 sean]# chown mike file1
[root@itmo456 sean]# chown steve file2
[root@itmo456 sean]#
[root@itmo456 sean]# ll file*
-rw-rw-r--. 1 mike  sean 0 Sep 20 17:37 file1
-rw-rw-r--. 1 steve sean 0 Sep 20 17:37 file2
-rw-rw-r--. 1 sean  sean 0 Sep 20 17:37 file3
-rw-rw-r--. 1 sean  sean 0 Sep 20 17:37 file4
-rw-rw-r--. 1 sean  sean 0 Sep 20 17:37 file5
[root@itmo456 sean]#
```

File and Directory Ownership

- **`chgrp`** (change group) command
  - Changes group owner of a file or directory
  - Takes two arguments at a minimum:
    - The new group owner
    - The files or directories to change
- Usage for **`chown`** and **`chgrp`**
  - **`chown sam myfilelist.txt`**
  - **`chgrp –R users myfile/`**
  - **`chown sam:users myfilelist.txt`**

File and Directory Ownership


```
[root@itmo456 sean]# chgrp mike file1
[root@itmo456 sean]# chgrp steve file2
[root@itmo456 sean]# chown mike:mike file3
[root@itmo456 sean]# chown steve:steve file4
[root@itmo456 sean]#
[root@itmo456 sean]# ll file*
-rw-rw-r--. 1 mike  mike  0 Sep 20 17:37 file1
-rw-rw-r--. 1 steve steve 0 Sep 20 17:37 file2
-rw-rw-r--. 1 mike  mike  0 Sep 20 17:37 file3
-rw-rw-r--. 1 steve steve 0 Sep 20 17:37 file4
-rw-rw-r--. 1 sean  sean  0 Sep 20 17:37 file5
[root@itmo456 sean]#
```

Managing File & Directory Permissions

- **Mode**
  - Inode section that *stores permissions*
  - Three sections based on the user(s) that receive(s) the permission to that file or directory
    - User (owner) permissions
    - Group (group owner) permissions
    - Other (everyone on Linux system) permissions
      - Also called ***world***

Managing File & Directory Permissions

- Three regular permissions may be assigned to each of the user(s) referenced on the previous slide:
  - Read
  - Write
  - Execute

Interpreting the Mode

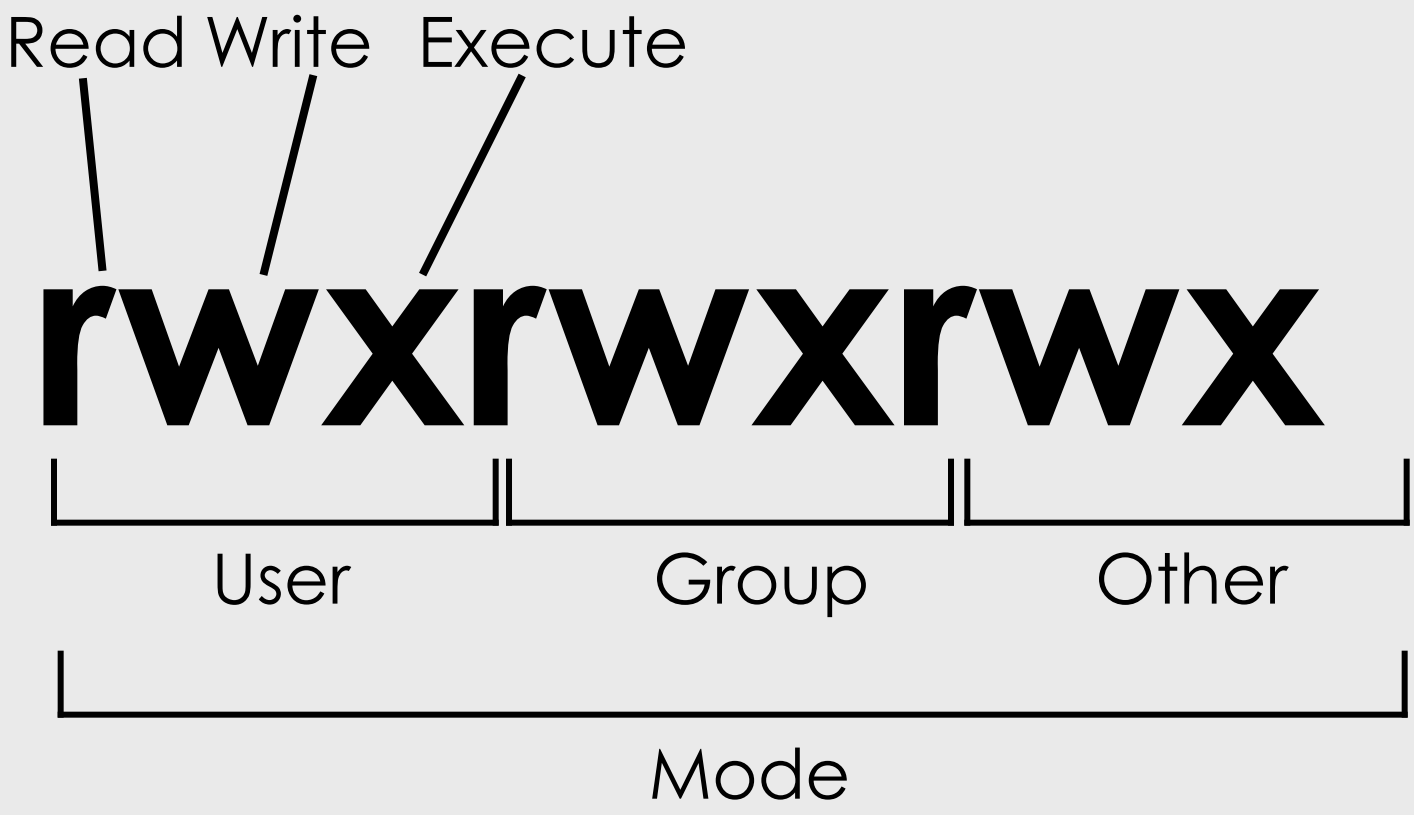

Figure 4-3: The structure of a mode

Interpreting the Mode

- User
  - When used in the mode of a certain file or directory, it refers to the *owner* of that file or directory
- Owner
  - User whose name appears in a long listing of a file or directory and who has the ability to change permissions on that file or directory
- Other
  - When used in the mode of a certain file or directory, refers to all users on the system
- Permissions are not additive

Interpreting Permissions

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 a directory
Write	Allows a user to open, read and edit the contents of a file	Allows a user to add or remove files from the directory (if they've also been given execute permission)
Execute	Allows a user to execute the file in memory (if it is a program file) and shell scripts	Allows a user to enter the directory and work with directory contents

Table 4-4: Linux permissions

Changing Permissions

- **`chmod`** (change mode) command
  - Changes mode (permissions) of a file or directory
  - Two arguments at a minimum:
    - First specifies criteria used to change the permissions
    - Remaining arguments indicate filenames to change
  - Permissions stored in a file or directory inode as binary powers of two

Changing Permissions

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)		

Table 4-5: Criteria used within the chmod command

Changing Permissions

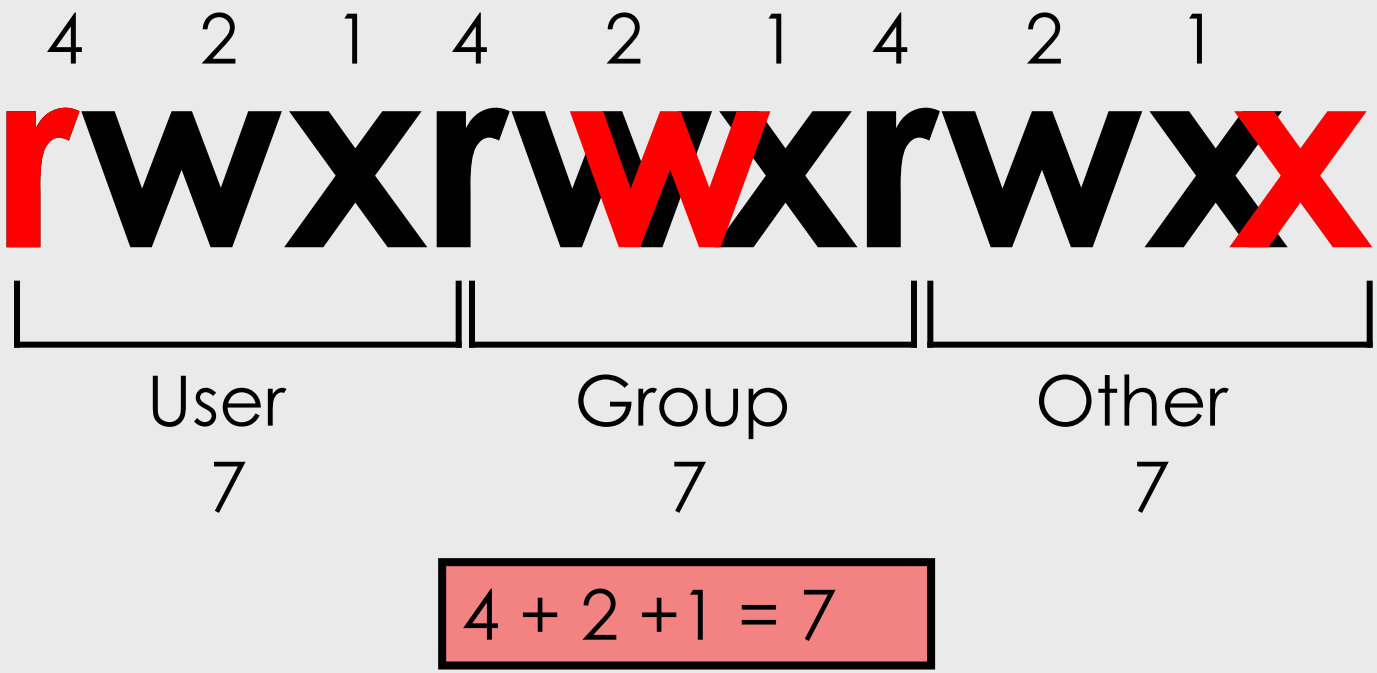

Figure 4-4: Numeric representation of the mode

Changing Permissions

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**

Table 4-6: Numeric representation of the permissions in a node

Changing Permissions


```
[root@itmo456 sean]# ll file*
-rw-rw-r--. 1 mike  mike  0 Sep 20 17:37 file1
-rw-rw-r--. 1 steve steve 0 Sep 20 17:37 file2
-rw-rw-r--. 1 mike  mike  0 Sep 20 17:37 file3
-rw-rw-r--. 1 steve steve 0 Sep 20 17:37 file4
-rw-rw-r--. 1 sean  sean  0 Sep 20 17:37 file5
[root@itmo456 sean]# chmod 754 file1
[root@itmo456 sean]# chmod 777 file2
[root@itmo456 sean]# chmod 555 file3
[root@itmo456 sean]# chmod +x file4
[root@itmo456 sean]# ll file*
-rwxr-xr--. 1 mike  mike  0 Sep 20 17:37 file1
-rwxrwxrwx. 1 steve steve 0 Sep 20 17:37 file2
-r-xr-xr-x. 1 mike  mike  0 Sep 20 17:37 file3
-rwxrwxr-x. 1 steve steve 0 Sep 20 17:37 file4
-rw-rw-r--. 1 sean  sean  0 Sep 20 17:37 file5
[root@itmo456 sean]#
```

Default Permissions

- New files given **`rw-rw-rw-`** permissions by default
- New directories given **rwxrwxrwx**
- Umask
  - Alters permissions on all new files and directories by taking select default file and directory permissions away
  - Only applies to newly created files and directories
    - Never used to modify permissions of existing files and directories

Default Permissions

- **`umask`** command
  - Displays the umask
- Changing the umask
  - Use a new umask as an argument to the **`umask`** command

Default Permissions

	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

Figure 4-5: Performing a umask 022 calculation

Default Permissions

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

Figure 4-6: Performing a umask 007 calculation

umask

```
[root@itmo456 sean]# umask
0022
[root@itmo456 sean]# ll -d Desktop/ file1
drwxr-xr-x. 2 sean sean 4096 Sep 20 17:13 Desktop/
-rwxr-xr--. 1 mike mike    0 Sep 20 17:37 file1
[root@itmo456 sean]#
[root@itmo456 sean]# umask 0777
[root@itmo456 sean]# mkdir umask
[root@itmo456 sean]# touch umask.txt
[root@itmo456 sean]# ll -d umask umask.txt
d---------. 2 root root 4096 Sep 20 19:03 umask
----------. 1 root root    0 Sep 20 19:04 umask.txt
[root@itmo456 sean]#
[root@itmo456 sean]# umask 0124
[root@itmo456 sean]# mkdir umask2
[root@itmo456 sean]# touch umask2.txt
[root@itmo456 sean]# ll -d umask2 umask2.txt
drw-r-x-wx. 2 root root 4096 Sep 20 19:04 umask2
-rw-r---w-. 1 root root    0 Sep 20 19:05 umask2.txt
[root@itmo456 sean]# umask 0022
[root@itmo456 sean]# umask
0022
[root@itmo456 sean]#
```

Special Permissions

- Read, write, and execute are the regular file permissions used to assign security to files
- Three more special permissions that you may optionally use on file and directories:
  - SUID (Set User ID)
  - SGID (Set Group ID)
  - Sticky bit

Defining Special Permissions: SUID

- SUID
  - If set on a file, user who executes the file becomes owner of the file during execution
  - No functionality when set on a directory
- SUID can only be applied to binary compiled programs
  - Cannot be used on shell scripts

Defining Special Permissions: SGID

- SGID
  - Applicable to files and directories
  - If set on a file, user who executes the file becomes member of the file's group during execution
  - If a user creates a file in a directory with SGID set, the file's group owner is set to be the directory's group owner and not the user's primary group

Defining Special Permissions

- Sticky bit
  - Previously used to lock files in memory
  - Currently only applicable to directories
  - Ensures that a user can only delete files that are his/her own files
  - Prevents deletion of files in writable directories by a non-owner
    - Example: the **`/tmp`** directory

Setting Special Permissions

- Mode of a file displayed using **`ls -l`** command does not have a section for special permissions
- Special permissions require execute
  - Mask the execute permission when displayed using the **`ls -l`** command
- May be set even if file or directory does not have execute permission
  - Via chmod command
    - Adds an extra digit at front of permissions argument

Setting Special Permissions

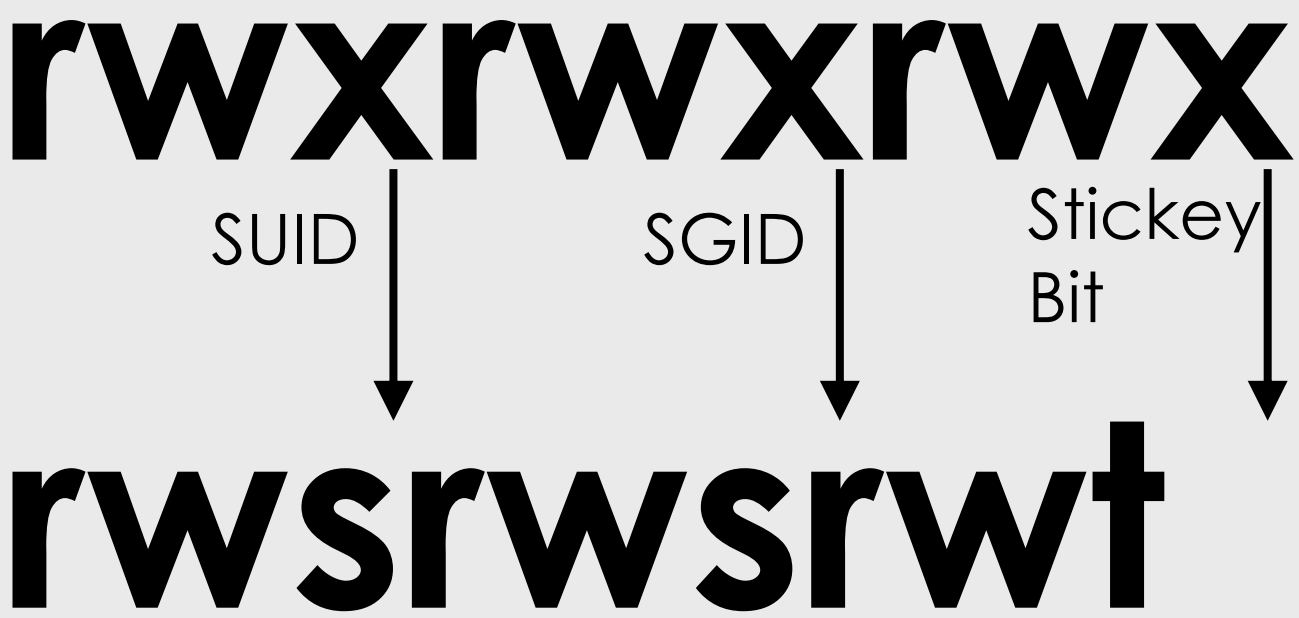

Figure 4-7: Representing special permissions in the mode

Setting Special Permissions

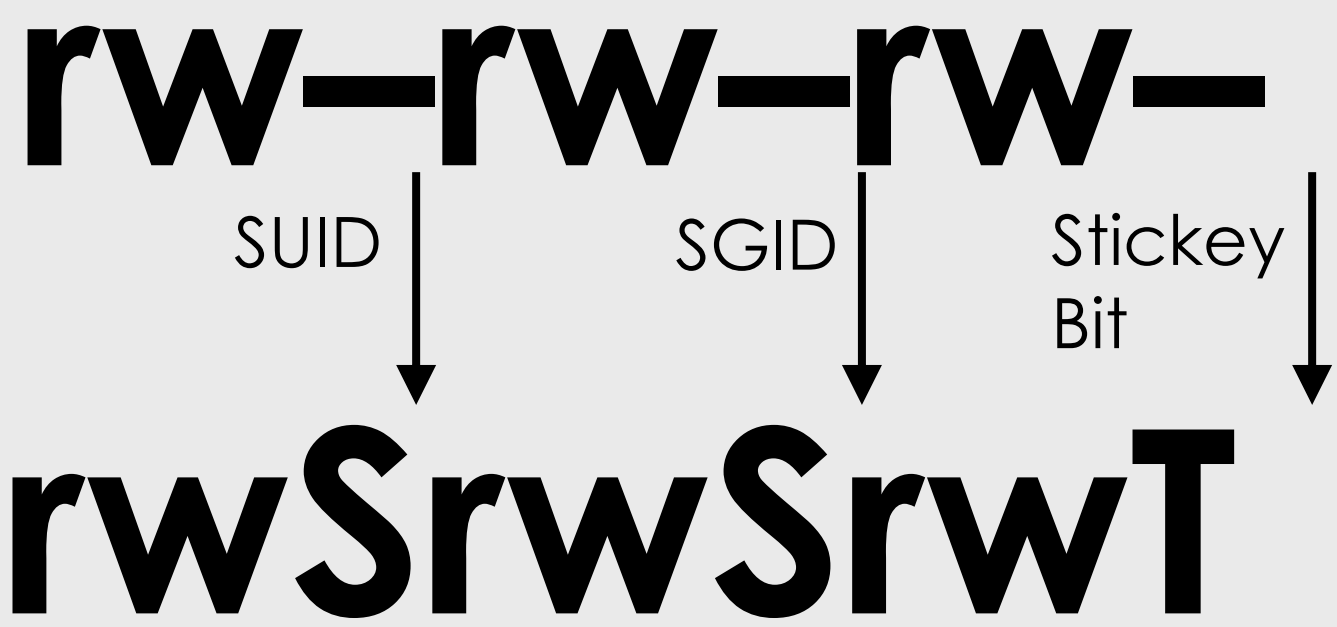

Figure 4-8: Representing special permissions in the absence of execute permissions

Setting Special Permissions

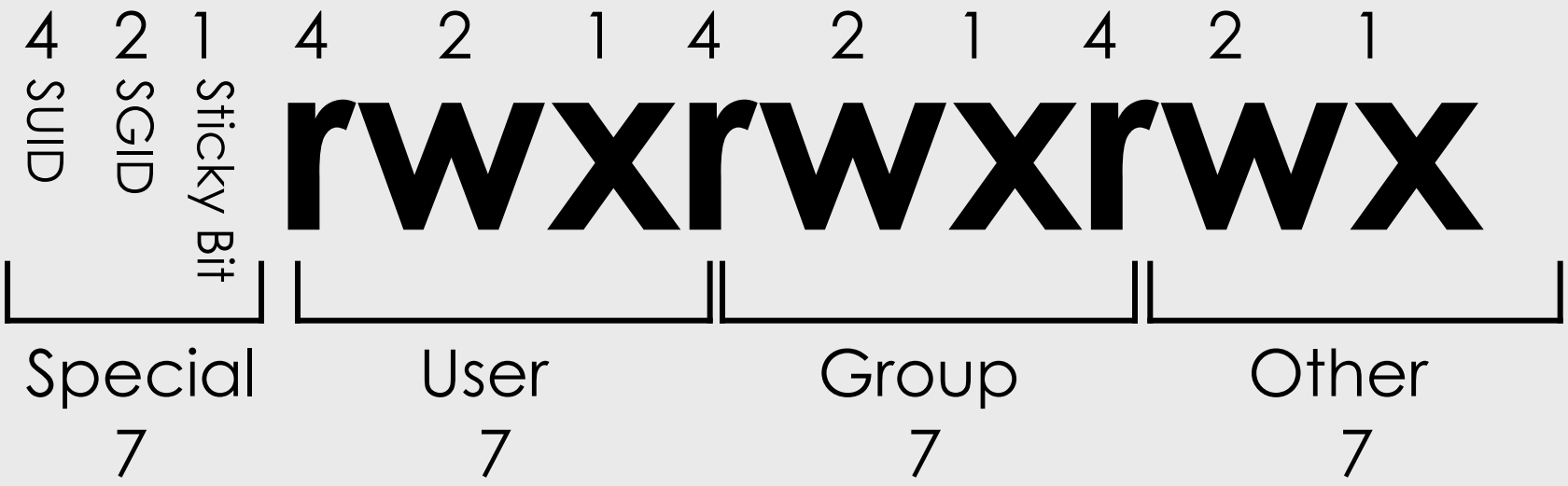

Figure 4-9: Numeric representation of regular and special permissions

Setting Special Permissions


```
[root@itmo456 sean]# ll -d test* Downloads/
drwxr-xr-T. 2 sean sean 4096 Sep 20 16:14 Downloads/
-rwsr-sr-x. 1 sean sean    0 Sep 20 17:37 test1
-rwsr-xr-x. 1 sean sean    0 Sep 20 17:37 test2
-rw-rw-r--. 1 sean sean    0 Sep 20 17:37 test3
-rw-rw-r--. 1 sean sean    0 Sep 20 17:37 test4
-rw-rw-r--. 1 sean sean    0 Sep 20 17:37 test5
[root@itmo456 sean]# chmod 6755 test1
[root@itmo456 sean]# chmod 4755 test2
[root@itmo456 sean]# chmod 2755 test3
[root@itmo456 sean]# chmod 1754 Downloads/
[root@itmo456 sean]# ll -d test* Downloads/
drwxr-xr-T. 2 sean sean 4096 Sep 20 16:14 Downloads/
-rwsr-sr-x. 1 sean sean    0 Sep 20 17:37 test1
-rwsr-xr-x. 1 sean sean    0 Sep 20 17:37 test2
-rwxr-sr-x. 1 sean sean    0 Sep 20 17:37 test3
-rw-rw-r--. 1 sean sean    0 Sep 20 17:37 test4
-rw-rw-r--. 1 sean sean    0 Sep 20 17:37 test5
[root@itmo456 sean]#
```

Setting Access Control Lists (ACL)

- Access control list (ACL):
  - a list of users or groups that you can assign permissions to
- **`setfacl`** (set file ACL) command: used to modify ACL entries for a particular Linux file or directory
  - Use the `-m` option to modify the ACL
  - e.g: **`setfacl -m u:bob:r-- doc1`**
- **`getfacl`** (get file ACL) command: used to list all ACL entries for a particular Linux file or directory

Setting Access Control Lists (ACL)


```
[sean@itmo456 ~]$ touch /tmp/secret.txt
[sean@itmo456 ~]$ chmod 660 /tmp/secret.txt
[sean@itmo456 ~]$ ll /tmp/secret.txt
-rw-rw----+ 1 sean sean 0 Sep 20 19:15 /tmp/secret.txt
[sean@itmo456 ~]$ setfacl -m u:mike:r-- /tmp/secret.txt
[sean@itmo456 ~]$ ll /tmp/secret.txt
-rw-rw----+ 1 sean sean 0 Sep 20 19:15 /tmp/secret.txt
[sean@itmo456 ~]$ getfacl /tmp/secret.txt
getfacl: Removing leading '/' from absolute path names
# file: tmp/secret.txt
# owner: sean
# group: sean
user::rw-
user:mike:r--
group::rw-
mask::rw-
other::---

[sean@itmo456 ~]$
```

Managing Filesystem Attributes

- Linux has file attributes that can be set
  - These attributes work outside Linux permissions and are filesystem-specific
- **`lsattr`** (list attributes) command: used to list filesystem attributes for a Linux file
- **`chattr`** (change attributes) command: used to change filesystem attributes for a Linux file
- Immutable attribute (i): prevents the file from being modified in any way
  - Not even root user can modify

Managing Filesystem Attributes

```
[root@itmo456 ~]# ll file1.txt
-rw-r--r--. 1 root root 6 Sep 20 19:21 file1.txt
[root@itmo456 ~]# lsattr file1.txt
-------------e-- file1.txt
[root@itmo456 ~]#
[root@itmo456 ~]# echo hello > file1.txt
[root@itmo456 ~]# cat file1.txt
hello
[root@itmo456 ~]#
[root@itmo456 ~]# chattr +i file1.txt
[root@itmo456 ~]# ll file1.txt
-rw-r--r--. 1 root root 6 Sep 20 19:22 file1.txt
[root@itmo456 ~]# lsattr file1.txt
----i--------e-- file1.txt
[root@itmo456 ~]#
[root@itmo456 ~]# echo test > file1.txt
-bash: file1.txt: Permission denied
[root@itmo456 ~]#
[root@itmo456 ~]# chattr -i file1.txt
[root@itmo456 ~]# echo test > file1.txt
[root@itmo456 ~]#
[root@itmo456 ~]# cat file1.txt
test
[root@itmo456 ~]#
```

Summary

- The Linux directory tree obeys the Filesystem Hierarchy Standard
  - Allows system files to be located in standard directories
- Many file management commands exist
- You can find files using different commands
  - **`locate`**: search preindexed database
  - **`which`**: search PATH variable
  - **`find`**: search for file based on criteria

Summary

- Files can be linked two different ways
  - Symbolic link: a file serves as a pointer to another
  - Hard links: one file is a linked duplicate of another
- Each file and directory has an owner and a group owner
  - Owner can change permissions and grant ownership
- Permissions can be set on the owner of a file, members of the group of the file, and everyone on the system (other)

Summary

- Three regular file and directory permissions (read, write, execute) and three special file and directory permissions (SUID, SGID, sticky bit)
- Permissions can be changed using `chmod`
- New files and directories receive default permissions from the system

Summary

- The root user has all permissions to all files and directories on the Linux filesystem
  - Root user can change the ownership of any file or directory on the Linux filesystem
- The default ACL on a file or directory can be modified to include additional users or groups
- Filesystem attributes can be set on Linux files to provide low-level functionality such as immutability

The End…

- Questions?