Filesystem: The Basics

CSE 231

Instructor: Arani Bhattacharya

Files are organized in a hierarchy

- What is a file?
  - Logically contiguous space for storing any type of data

- All file systems follow the hierarchical model
- Files are organized hierarchically into folders/directories
- The files themselves have a name
  - Optionally consists of two parts
  - First part is used to denote the actual name used
  - Second part is used to denote the type of file (such as image, slide, etc.)
  - Note that this division of filenames is ONLY a convention on Linux

Windows Filesystem Hierarchy

Windows uses a very simple hierarchy, with a single disk divided into three filesystems by default -- named C:, D:, and E:

Advantage: The hierarchy is easy to understand

Disadvantages:

- Most files tend to go straight to C:, since both system files and user files go there
- User files are kept very deep in the hierarchy, making it cumbersome to find them

Linux Filesystem Hierarchy

Linux uses a much more complex hierarchy

- **/ – The Root Directory**
  - All content is stored in this directory
- **/bin -- Essential User Binaries**
  - Like Firefox, ls, bash
- **/boot -- File needed to boot the system**
  - All GRUB files

Linux Filesystem Hierarchy(2)

- **/dev -- Device files**
  - Linux has a somewhat non-intuitive system of dealing with I/O devices
  - For Linux, all I/O devices are handled as “special files”
  - Example: /dev/sda represents one hard disk, /dev/sr0 represents CD-ROM
  - There are also “pseudo-files” to deal with “virtual devices”, such as /dev/null, /dev/random and so on
- **/etc -- Configuration files**
  - All configuration files are stored here as text files
  - Example: network configuration, sudo permissions of individual users, etc.

Linux Filesystem Hierarchy(3)

- **/home -- Home folders for each user**
  - Personal files of each user
- **/lib -- Libraries installed by users**
  - Libraries (both static and dynamic) needed by different installed programs
- **/media -- Removable Media**
- **/mnt – Temporary Mount Points**

Linux Filesystem Hierarchy(4)

- **/proc – Kernel & Process Files**
  - Shows data about the kernel
  - Example: What is the processor and memory configuration?
  - How much memory is taken by each process?
- **/root – Home directory of root user**
- **/tmp – Temporary Files**
- **/var – Variable Data Files**
  - Used to store data required by different packages
- **/opt – Optional Packages**
  - Used by proprietary software packages

Disks and Partitions

- Disk space is divided into partitions. Each partition consists of its own file system
- A partition decides how data should be organized in the space
- Common file systems used include:
  - Linux: ext4, BtrFS
  - Windows: NTFS
  - CDROM: iso9660
  - USB drive: FAT32
- We will discuss the internal organization in the next class

Mount Points

Since all files under all file systems must come under the overall file system hierarchy, how do we map the content?

Mountpoint: An empty directory where a disk is mounted, i.e. all the contents of the disk can be accessed by entering this directory. Process of mapping the disk to this directory is called “mounting”.

Example: Suppose you insert a CD. How do you access it?

You create an empty directory called (say) mydir, and then mount the disk using mount command (internally uses mount system call).

```
~> mkdir mydir && mount -t iso9660 /dev/sr0 mydir
```

How does system keep track of mounted filesystems?

The /etc/fstab keeps track of available disk partitions, mount points and filesystem type

```
susel:~ # cat /etc/fstab
/dev/sda1            swap                 swap       defaults              0 0
/dev/sda2            /                    ext3       acl,user_xattr        1 1
proc                 /proc                proc       defaults              0 0
sysfs                /sys                 sysfs      noauto                0 0
debugfs              /sys/kernel/debug    debugfs    noauto                0 0
usbfs                /proc/bus/usb        usbfs      noauto                0 0
devpts               /dev/pts             devpts     mode=0620,gid=5       0 0
# /dev/sr0      /cdrom  iso9660 ro,nosuid,nodev,uid=0    0 0
/dev/sdc1            /novi_disk                    ext3       acl,user_xattr,usr
quota,grpquota        2 0
```

How does system keep track of mounted filesystems?

The /etc/mtab keeps track of only mounted file systems

```
proc /proc proc rw 0 0
sysfs /sys sysfs rw 0 0
devpts /dev/pts devpts rw,gid=5,mode=620 0 0
/dev/sda1 /boot ext3 rw 0 0
tmpfs /dev/shm tmpfs rw 0 0
none /proc/sys/fs/binfmt_misc binfmt_misc rw 0 0
sunrpc /var/lib/nfs/rpc_pipefs rpc_pipefs rw 0 0
```

File Attributes

```
struct stat {
    dev_t     st_dev;         /* IDs of device on which file resides */
    ino_t     st_ino;         /* I-node number of file */
    mode_t    st_mode;        /* File type and permissions */
    nlink_t   st_nlink;       /* Number of (hard) links to file */
    uid_t     st_uid;         /* User ID of file owner */
    gid_t     st_gid;         /* Group ID of file owner */
    dev_t     st_rdev;        /* IDs for device special files */
    off_t     st_size;        /* Total file size (bytes) */
    blksize_t st_blksize;     /* Optimal block size for I/O (bytes) */
    blkcnt_t  st_blocks;      /* Number of (512B) blocks allocated */
    time_t    st_atime;       /* Time of last file access */
    time_t    st_mtime;       /* Time of last file modification */
    time_t    st_ctime;       /* Time of last status change */
};
```

File Ownership & Permissions

File has its user id and group id, and permissions for three different operations, each represented by an octal number:

1. Read -- 4
2. Write -- 2
3. Execute -- 1

Also, by looking at user and group, three types of users:

1. User
2. Group
3. Others

The three octal numbers together represent the overall permissions of the file

Do directories also have permissions?

Yes, for directories, the permissions imply the following:

1. Read -- ls and reading files and subdirectories is allowed
2. Write -- creating and/or removing files is possible (but only if execute permission is also set)
3. Execute -- existing files in directory can be accessed

Changing file owner and permission

In both bash shell and C:

1. Changing file owner -- chown command/system call
2. Changing file permission -- chmod command/system call

Finding all the file attributes

1. stat(const char *pathname, struct stat *statbuf)
2. lstat(const char *pathname, struct stat *statbuf)
3. fstat(int fd, struct stat *statbuf)

Directories

- A special file kept in file system
- Directory itself stores the list of files and subdirectories
- File information are stored internally in i-nodes
- The exact organization of i-nodes depends on the filesystem
- Having a file within a directory is simply mentioning the i-node corresponding to a file within the directory file

Soft Links & Hard Links

- Soft link is similar to a shortcut
  - Internally, create a new file (i.e. inode) and point to the original file's inode
  - ln -s original_file_name link_name
  - Changing the location of the original file makes the soft link invalid

- Hard link is different
  - The inode itself should store two or more distinct locations
  - Changing the location of one location does not affect the hard links

File Locking

- Note that reading and writing to files can lead to race conditions
  - Relatively common

- Different OSes use different techniques
  - Windows -- Locks all files whenever a process opens it for reading or writing
  - Linux -- Provides special system calls called flock and fcntl
    int flock(int fd, int operation);
    fcntl(fd, cmd, &flockstr);