CMP325 Operating Systems Lecture 27

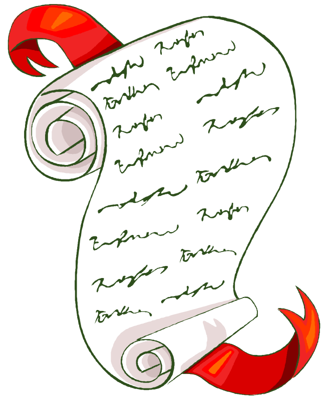
Disk Formatting & File System Mounting

Muhammad Arif Butt, PhD

Note:
Some slides and/or pictures are adapted from course text book and Lecture slides of
- Dr Syed Mansoor Sarwar
- Dr Kubiatowicz
- Dr P. Bhat
- Dr Hank Levy
- Dr Indranil Gupta

For practical implementation of operating system concepts discussed in these slides, students are advised to watch and practice following video lectures:
OS with Linux:
https://www.youtube.com/playlist?list=PL7B2bn3G_wfBuJ_WtHADcXC44piWLRzr8
System Programming:
https://www.youtube.com/playlist?list=PL7B2bn3G_wfC-mRpG7cxJMnGWdPAQTViW

Today's Agenda



- Overview of File System
- File System Types
- Low level vs High level formatting
- Information about Block devices
- Disk Formatting Tools
- Overview of File System Mounting
- Displaying mounted file systems
- Mounting/Unmounting a file system
- Boot Time Mounting
- Maintaining Integrity of File System

Formatting a Hard Disk

Introduction To Files

- All computer applications need to store and retrieve information. A running process can store information within its own address space. One problem with this is that the space is often inadequate for various applications (banking, airline reservations). Second problem with keeping information within a process address space is that when the process terminates the information is lost. Third problem is that it may be necessary for multiple processes to access the information at the same time.
- With this we come up to the requirements of Long Term Information Storage:
  - Capable of storing a very large amount of information.
  - Information must survive the termination of the process using it.
  - Multiple processes must be able to access the information concurrently.
- The solution to all these problems is to store information on disks and other external media in units called Files.
- Files are managed by operating systems. How a file is structured, named, used, protected and implemented are major topics in operating system design.

Files Files Every Where

- Suppose we are developing an application program. A program, which we prepare, is a file. Later we may compile this source program file and get an object code file or an executable file. When we store images from a web page we get an image file. If we store some music in digital format it is an audio file. So, in almost every situation we are engaged in using a file. So we can say that:
- Irrespective of the content, any organized information is a file. So be it a telephone numbers list or a C++ program or an executable code or a web image we think of it always as a file. This formlessness and disassociation from content was emphasized first in Unix.
- The formlessness essentially means that files are arbitrary bit (or byte) streams

Overview of File System

- From a user point of view, a **file** is the smallest unit of allocation on a secondary storage medium.
- A **file system** is a piece of code that provides an abstraction mechanism to users and programs to organize their files on secondary storage medium (hard disks). A file system provide a way to store information on the disk and read it back later w/o the knowledge of the user of how and where the information is stored and how the disks actually work.
- We can say that a file system is like a library system (e.g., a Dewey Decimal system), in which card drawers are used to index all the books.

File System Services

- **Creating files:** by allocating free disk blocks/sectors to a newly created file and later keep track of all the blocks of that file
- **Reading/Updating:** by symbolic names, without the need for the user to know on which data blocks this file resides
- **Deleting files:** by deallocating file data blocks on hard disk to the free list and removing the symbolic name associated with that file
- **Moving files:** from one directory to another directory, or file system or may be to other disks
- **Concurrency control:** Several programs may read/write files at the same time without any conflict
- **Security:** Owner of files should be able to specify type of accesses (read, write, execute) to allow only authorized users to access to their files
- **Persistence:** Files must persist even when power is switched off
- **Journaling:** In journaling file systems the changes are first written to a journal on disk. The journal is flushed regularly writing the changes in the file system. Journaling keep the file system in consistent state, therefore, you don't need a file system check after an unclean shutdown or power failure.
- **Optimize performance:** Every file system has max file size and partition size support
- **Provision of I/O interface routines**

File System Types

- A Microsoft user normally do not have to worry about the file systems, with FAT32 or NTFS being the default
- On the contrary, a Linux user has a variety of flavors of file systems to choose from. Different file systems have different characteristics / attributes. So being a Linux user we must know some details of different file systems
- To read information about different file systems, please go through the man page of fs

```
$ man fs
```

- Similarly to display the list of currently loaded file system drivers on your Linux machine:

```
$ man fs
```

- Students need to search for the max file size, max partition size and journaling facility available in file systems like fat32, ntfs, ext, ext2, ext3, ext4, hpfs, reiserfs, iso9660, JFS, minix, msdos, smb, xfs, zfs, xfs zfs,…

DISK FORMATTING

Low-level formatting, or physical formatting

- During LLF, the program divides the disk's tracks into a specific number of sectors, creating the inter-sector and inter-track gaps and recording the sector header and trailer information. The program also fills each sector's data area with a dummy byte values.
- All IDE & SCSI drives uses a technique called **Zoned Bit Recording**, which writes variable number of sectors per track. The outer tracks hold more sectors than the inner tracks because the outer tracks have longer circumference. Drives without Zoned Bit Recording store the same amount of data on every cylinder, even though the tracks of the outer cylinders may be twice as long as those of the inner cylinders. The result is wasted storage capacity.

High-level formatting or Logical formatting

- To use a disk to hold files, the operating system needs to record its own data structures on the disk. File systems are used to create and locate the files you save on your hard disk. So after creating a partition you must format it with a specific file system. Some examples of files systems are FAT-12, 16, 32, NTFS, ext2, ext3, nfs, JFS, hpfs/ISO9660, ReiserFS.

Formatting a Disk Partition

- We already know how to partition a hard disk using the fdisk utility. For a review, you can print the partition table of your hard disk using following command:

```
# fdisk –l /dev/sda
```

- After we are done with partitioning our hard disk, we need to place a file system on each partition. This is called high level formatting or simply formatting
- To see the different file system build commands available on your Linux machine, use following command:

```
$ ls /sbin/*mk*fs
```

```
mkfs
mke2fs (mkfs.ext2, mkfs.ext3, mkfs.ext4)
mkntfs (mkfs.ntfs)
mkfs.fat (mkfs.msdos, mkfs.vfat)
mkfs.minix
```

Information About Block Devices

- Before we use these file system build commands, let us dig out some information about the existing file systems that resides on different partitions of our hard disk. We can use `lsblk` command for this which is used to list information about all available block devices attached with our system

```
$ lsblk /dev/sda
```

- To display information of our interest only, we can use the -o option to `lsblk` command

```
$ lsblk –o name,type,fstype,parttype,size,mode /dev/sda
```

- You can always change certain attributes of a file system using different shell commands, e.g., e2label us used to assign labels to disk partitions

```
$ e2label /dev/sda5 “arif007”
```

Formatting a Disk Partition

- To format a disk partition, the existing file system on that partition must not be mounted. So to unmount a file system on the second logical partition of a SCSI disk, use the following command:

```
# umount /dev/sda6
```

- Now let us format this partition to vfat file system

```
# mkfs.vfat /dev/sda6
```

- Now let us mount and confirm

```
# mount /dev/sda6 /opt
# lsblk –o name,fstype,size,mountpoint /dev/sda
```

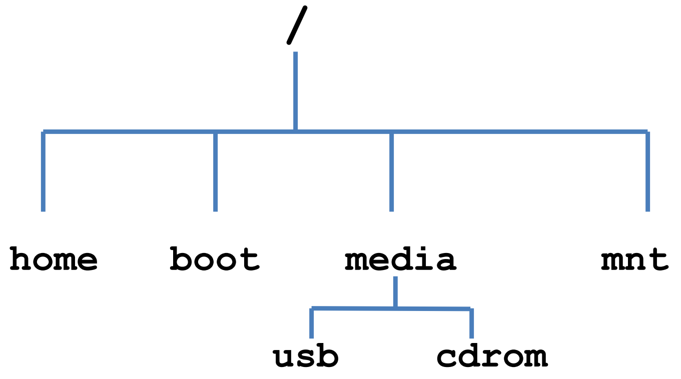
Mounting a File System

Overview of File System Mounting

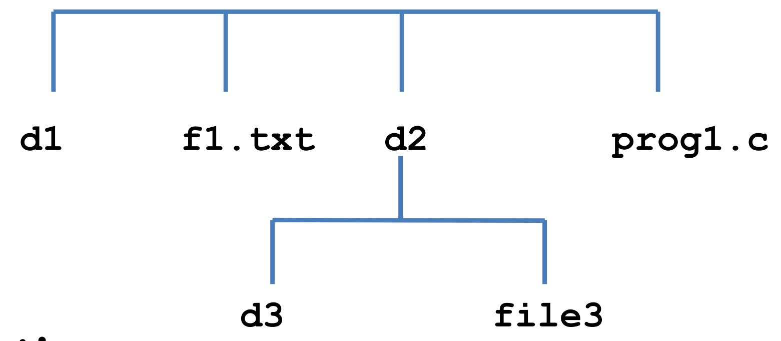
- A UNIX like file system is best visualized as a tree rooted at /. We have filesystems on different partitions (/home. /boot. /opt, …) and devices (USB flash drive, CD)
- The root file system (/) is mounted as part of initialization process. To make other file systems usable you must mount them at a mount point
- A **mount point** is simply a directory where the file system on a device is grafted into the tree
- **Mounting** is the process of making the foreign file system look like part of the main tree.

Overview of File System Mounting

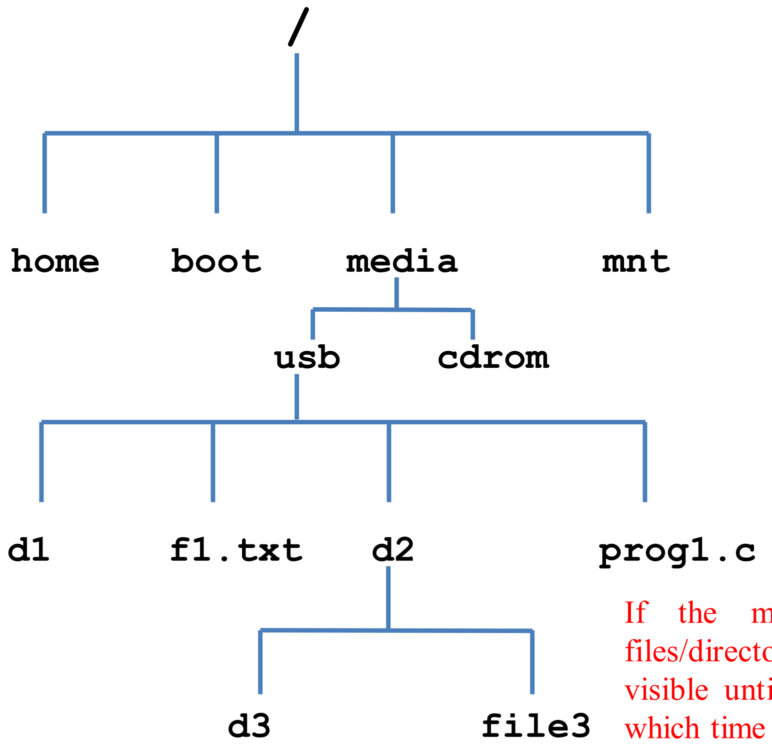
Existing File System Tree on Disk



Un-mounted File System on USB stick



New Tree After Mounting



If the mount point directory already contained files/directories they are not lost, but are no longer visible until the mounted file system is unmounted, at which time they become visible again

Mounting a File System

- On Linux systems, the mount command is used to attach a file system on some device to the big file tree.
- The general syntax of mount command is:

```
# mount [-t fstype] [device] [mount point]
```

- If you want to get information about the currently mounted file systems you can view the contents of the file /proc/mounts or /etc/mtab. You can also use the mount command without any options for this purpose

Un-Mounting a File System

- All mounted filesystems are usually unmounted automatically by the system when it is rebooted or shutdown
- A user may unmount file systems manually, while removing writable media like USB flash drives or memory cards
- To do this use the umount command by specifying either the device name or the mount point as argument.
- The general syntax of mount command is:

```
# umount [devicenname / mountpoint]
```

- A file system cannot be unmounted if a process has open some files on it or has its working directory there. A lazy unmount overrides this using –l option
- If you find your USB flash device busy, may be because some process has opened files on it. You can use lsof (list open files) or fuser command to check this out

Permanent / Boot Time Mounting

- The /etc/fstab file contains six fields of information about the file systems that the system can mount
- It is read by programs like fsck(8), dump(8), mount(8), and umount(8) sequentially to perform their tasks
- It is maintained by system administrator
- Each file system is described on a separate line each having space or tab separated six fields
- Students are advised to read the details by reading the man page of fstab from section 5
- The six fields of this file are:
  - **File System**
  - **File System Type**
  - **Mount Point**
  - **Mount Options**
  - **Dump Frequency**
  - **fsck Order**

File System Integrity

Integrity of File System

- Whenever our system crashes or looses power, Linux may not be able to cleanly unmount the file systems
- This may leave the file systems in inconsistent state with some changes completed and some not
- To view the file integrity check commands available on your system, give the following command

```
$ ls /sbin/*fsck*
fsck
e2fsck (fsck.ext2, fsck.ext3, fsck.ext4)
fsck.fat (fsck.msdos, fsck.vfat)
fsck.minix
fsck.nfs
```

- When system is booted fsck(8) check the root (/) file system to perform consistency check and repairs if required. This is done automatically because the fsck field, (6th field) of /etc/fstab file normally contains a 1 for the root (/) file system
- Some journaling file systems (reiserfs, xfs), might have a value of zero inn the fsck field of /etc/fstab file, to show that the file system consistency check and repair is done by journaling code

Summary

Always use extreme care when using tools mentioned in this lecture
Data loss may be only a key stroke away

We're done for now, but Todo's for you after this lecture…

- Go through the related video lectures # 18 and 19.
- Practice, practice and practice…