INTRO TO OPERATING SYSTEMS (COM 3610)

PROFESSOR BEN WYMORE

Assignment: An Introduction to File Systems

Introduction

- The goal of assignment is to understand
  - basic file system design and implementation
  - file system testing
  - data serialization/de-serialization

- At the end of the assignment, you will feel like a file system expert!

Outline

- Background
  - Mounting file systems
- Assignment
  - Specification
  - Downloading and testing file system image
  - General FAT32 data structures
  - Endian-ness

Mounting File Systems

Unix File Hierarchy

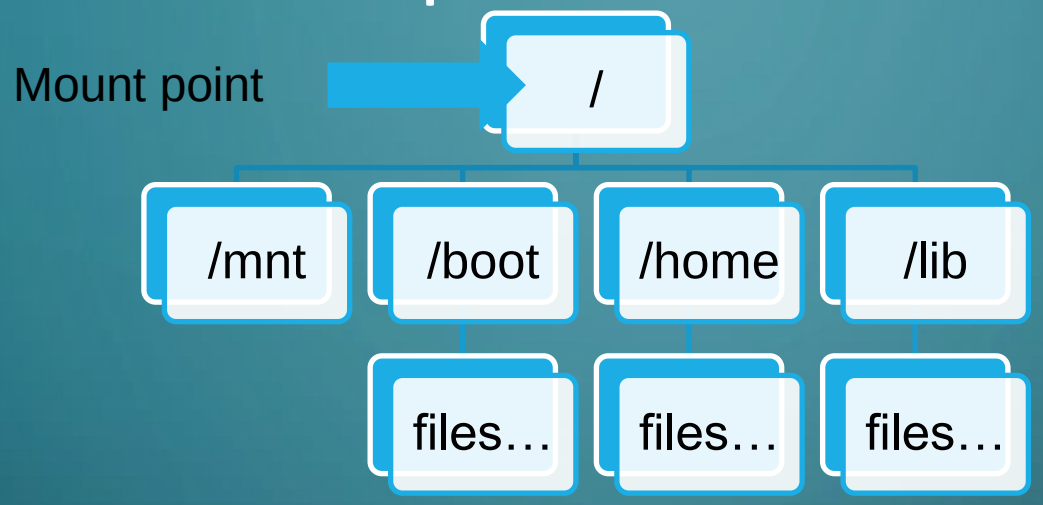
- All files accessible in a Unix system are arranged in one big tree
  - Also called the ***file hierarchy***
  - Tree is rooted (starts) at **/**
- These files can be spread out over several devices
- The **mount** command serves to attach the file system found on some device to the big file tree

‘mount’ command

- `mount`
- `mount <device> <mount directory>`

- Typing ‘mount’ without arguments shows you what is mounted and where
- Second example attaches a device or partition to a directory
  - Must have root privileges

Mount Example



The device sda partition 1 is mounted at “/”. All files and dirs below “/” come from this device.

Mount Example

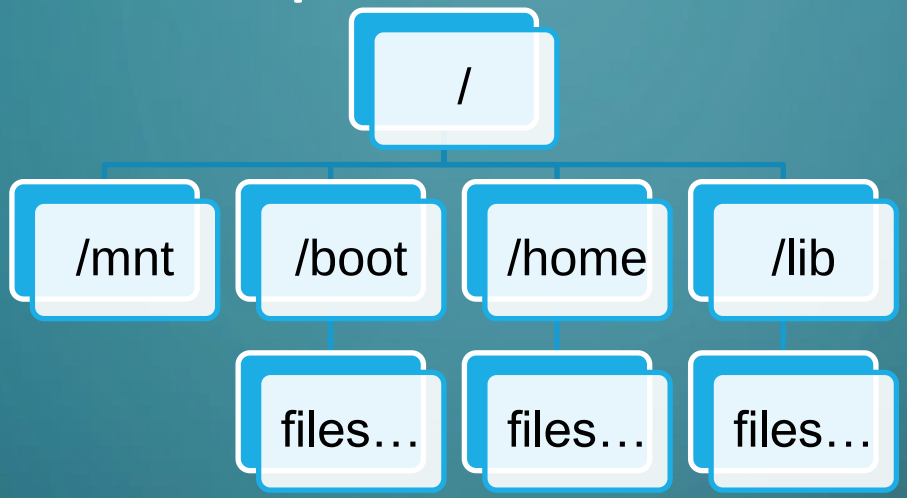
- Type command ‘mount’ without any arguments to see what is mounted and where

user@cop4610: ~

```
user@cop4610:~$ mount
/dev/sda1 on / type ext3 (rw,errors=remount-ro)
tmpfs on /lib/init/rw type tmpfs (rw,nosuid,mode=0755)
proc on /proc type proc (rw,noexec,nosuid,nodev)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
procbususb on /proc/bus/usb type usbfs (rw)
udev on /dev type tmpfs (rw,mode=0755)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
devpts on /dev/pts type devpts (rw,noexec,nosuid,gid=5,mode=620)
user@cop4610:~$
```

Root “/” file system mounted

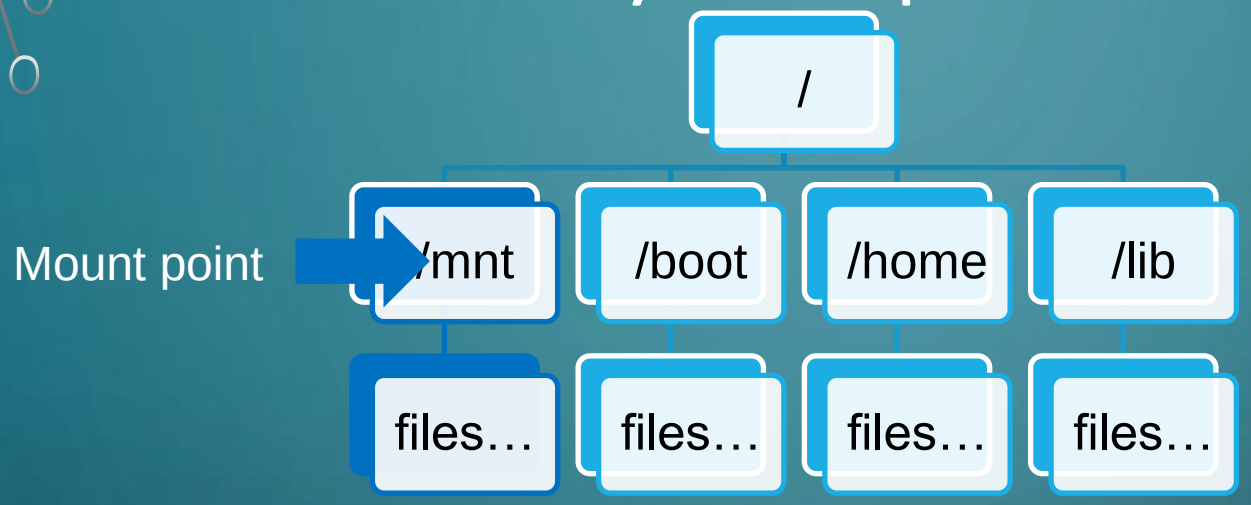
Mount Example



/dev/sdb1

Now suppose we attach a thumb drive and want our thumb drive files accessible under /mnt…

File Hierarchy Example



/dev/sdb1

Files from the thumb drive are now accessible under /mnt

Mount Example

- The ‘mount’ command can dynamically attach new devices to new mount points

user@cop4610: ~

```
user@cop4610:~$ sudo mount /dev/sdb1 /mnt
[sudo] password for user:
user@cop4610:~$ mount
/dev/sda1 on / type ext3 (rw,errors=remount-ro)
tmpfs on /lib/init/rw type tmpfs (rw,nosuid,mode=0755)
proc on /proc type proc (rw,noexec,nosuid,nodev)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
procbususb on /proc/bus/usb type usbfs (rw)
udev on /dev type tmpfs (rw,mode=0755)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
devpts on /dev/pts type devpts (rw,noexec,nosuid,gid=5,mode=
/dev/sdb1 on /mnt type vfat (rw)
user@cop4610:~$
```

Thumb drive mounted here

Un-mount Command

- `umount <dir>`

- In our example where the thumb drive was mounted at /mnt, we can issue
  - `$> umount /mnt`
  - Requires root privileges

Loopback File Systems

- A loopback file system is an image of a complete file system within a single file
- All the sector contents are concatenated in order within the file (one big char array).
- Linux can mount one of these like any other filesystem.

```
mount -o loop disk1.iso /mnt/disk
```

- You will use one of these for your project
- You will NOT be mounting it, unless you want to explore its contents from the Linux shell.

Figuring out names of devices

- /etc/fstab – Has list of devices and file systems that get auto-mounted on boot
- ‘dmesg’ command shows output when plugging in a dynamic device

Assignment

- You will create a user-space utility to manipulate a FAT32 file system image
- Utility must understand a few basic commands to allow simple file system manipulation
- Utility must not corrupt the file system and should be robust

This assignment will be written and tested using Java 17.0.10 on the VM

- Do the following:

```
sudo apt update
sudo apt install openjdk-17-jdk-headless
java -version
```

- You should see:

```
openjdk 17.0.10 2024-01-16
OpenJDK Runtime Environment (build 17.0.10+7-Ubuntu-120.04.1)
OpenJDK 64-Bit Server VM (build 17.0.10+7-Ubuntu-120.04.1, mixed
mode, sharing)
```

File System Image

- Manipulation utility will work on a pre-configured FAT32 ***file system image***
  - Actually a file
- File system image will have raw FAT32 data structures inside
  - Just like looking at the raw bytes inside of a disk partition

File System Image

- Your FAT32 manipulation utility will have to
  - Open the FAT32 file system image
  - Read parts of the FAT32 file system image and interpret the raw bytes inside to service your utility's file system commands…

…just like a file system!