CSCI 3150 Tutorial on Assignment 2

LI Runhui

Dept. of Computer Science and Engeering,
Chinese University of Hong Kong

November 4, 2012

Tutorials for Assignment 2

- **Nov 5th & 8th** Commandline tools to help you setup a FAT32 file system.
- **Nov 12th & 15th** The architecture of FAT32 file system and how to do recovery.
- **Nov 19th & 22th** How to test your program.

Outline

- Use a Ram Disk
- Use **dd** command to create a file pretending to be a disk
- Use **mkfs.vfat** command to format a disk
- Use **dosfsck** to look into details of a file system.
- Use **mount** to mount a disk
- All the commands are for **Linux**, they are not guaranteed to perform in other systems.
- You need root privilege to run these commands, that's why in the rest slides most commands have **sudo** in front of them.

RAM Disk

- RAM disk is a portion of RAM treated as if this part of memory is a disk.
- The I/O of RAM disk is much faster.
- Data will lost when the machine is powered off, just as the data in memory.

RAM Disks in Your System

- There are already some RAM disks created in your system. Try call the command

```
$ ls /dev/ram*
/dev/ram0   /dev/ram12  /dev/ram2  /dev/ram6
/dev/ram1   /dev/ram13  /dev/ram3  /dev/ram7
/dev/ram10  /dev/ram14  /dev/ram4  /dev/ram8
/dev/ram11  /dev/ram15  /dev/ram5  /dev/ram9
```

- The results may vary in different systems.

dd

- **dd**: a low-level copying command.
- **dd** is short for "disk duplication", while in the linux, the disk is presented as a file. Thus **dd** can also be used for copying files.
- We can use **dd** to generate a "clean" file, more details are shown in the next slide.

dd

sudo dd if=/dev/zero of=/dev/ram1 bs=64M count=1

- **if=**{**input file**} (e.g., /dev/zero, /dev/urandom, etc.)
- **of=**{**output file**}
- **bs=**{**block size**}
- **count=**{**# of blocks**}
- **skip=**{**# of blocks to skip at the start of input file**}

dd

- Create a file filled with $'\backslash 0'$

```
$ sudo dd if=/dev/zero of=/dev/ram1 bs=64M count=1
1+0 records in
1+0 records out
67108864 bytes (67 MB) copied, 0.0912389 s, 736 MB/s
$ sudo dd if=/dev/zero of=/dev/ram1 bs=256K count=1
1+0 records in
1+0 records out
262144 bytes (262 kB) copied, 0.000594406 s, 441 MB/s
```

mkfs.vfat

- **mkfs.vfat**: create FAT file systems
- For example, we are making /dev/ram1 an FAT32 file system
  $ sudo mkfs.vfat -F 32 /dev/ram1
- We can also make a regular file an FAT 32 file system
  $ sudo mkfs.vfat -F 32 mydisk

mkfs.vfat

mkfs.vfat -F 32 -f 2 -S 512 -s 1 -R 32 /dev/ram1

- **-F** type of FAT (e.g., FAT16, FAT32)
- **-f** number of FATs
- **-S** number of bytes per sector (i.e., the basic unit of disk)
- **-s** number of sectors per cluster (i.e., the basic unit for data storage in FAT32 file system)
- **-R** number of reserved sectors

dosfsck

- **dosfsck** - check and repair FAT file systems
- We can use **dosfsck -v** to look into details of an FAT file system
- Call the following command:
  $ sudo dosfsck -v /dev/ram1
  $ sudo dosfsck -v mydisk

```
$ sudo dosfsck -v /dev/ram1
dosfsck 3.0.12 (29 Oct 2011)
dosfsck 3.0.12, 29 Oct 2011, FAT32, LFN
Checking we can access the last sector of the filesystem
Boot sector contents:
System ID "mkdosfs"
Media byte 0xf8 (hard disk)
       512 bytes per logical sector
       512 bytes per cluster
        32 reserved sectors
First FAT starts at byte 16384 (sector 32)
         2 FATs, 32 bit entries
    516608 bytes per FAT (= 1009 sectors)
Root directory start at cluster 2 (arbitrary size)
Data area starts at byte 1049600 (sector 2050)
    129022 data clusters (66059264 bytes)
63 sectors/track, 255 heads
         0 hidden sectors
    131072 sectors total
Checking for unused clusters.
Checking free cluster summary.
/dev/ram1: 0 files, 1/129022 clusters
```

mount

- **mount** - mount a file system
- A file system needs to be mount to a **mount point** (a directory) before we can use it.
- Remember to **unmount** the system after use the file system after usage

mount

- Let us first create a mount point
  $ sudo mkdir /mnt/rd
- Then we can mount the /dev/ram1 to the mount point
  $ sudo mount -t vfat -o loop /dev/ram1 /mnt/rd
- After that, you can use the file system we just created.
- To unmount the file system, call
  $ sudo umount /mnt/rd

mount

- You may run into situations where the device is busy.

```
$ sudo umount /mnt/rd/
umount: /mnt/rd: device is busy.
        (In some cases useful info about processes that
         the device is found by lsof(8) or fuser(1))
```

mount

- Use **fuser** to help solve this problem.
- Call **fuser -m** to see which process is preventing the unmount

```
$ sudo fuser -m /mnt/rd
/mnt/rd:                     1234e
```

- Kill the process.
  $ sudo kill 1234
- Unmount the file system.
  $ sudo umount /mnt/rd

Q & A

- Thank You!