Operating systems fundamentals - B08

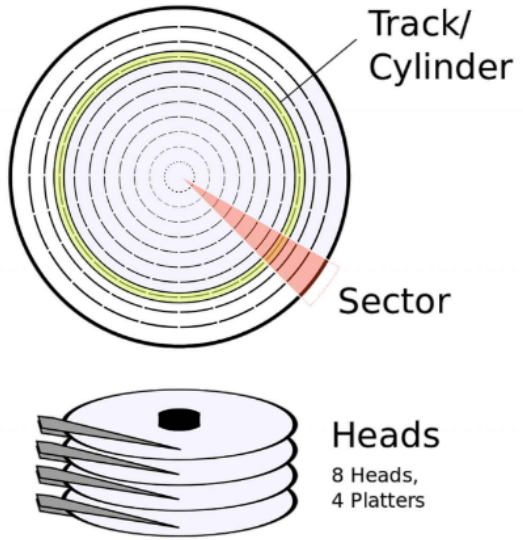
David Kendall

Northumbria University

Introduction

- Knowledge of tools and techniques for working with disks and file systems is important
  - for the average user
  - particularly important for the computer forensics and cybersecurity investigator
- This session looks at
  - disk storage structure
  - useful commands
    - `fdisk`
    - `dd`
    - `mount`
    - `find`

Disk geometry



Disk geometry

- A *hard drive* is made up of a number of *platters*
- Data is stored on the top and bottom surfaces of the platter and is read/written by a read/write *head*
- Each platter is divided into a number of *tracks* - concentric circles
- Each track is divided into number of *sectors*
- Platters are stacked on top of each other and so the stacked tracks seem to form *cylinders*

Logical Block Addressing (LBA)

- The disk geometry led to a sector addressing scheme based on *cylinder/head/sector (C/H/S)*
- As hard drive sizes increased the addressing scheme ceased to represent the physical geometry of the disk
- C/H/S addressing is now regarded as obsolete but the terminology still appears in many utility programs
- The modern addressing scheme is known as *logical block addressing (LBA)* and identifies each sector by a number between 0 and $TOTALSECTORS - 1$
- A C/H/S address can be translated into a LBA using the formula

$$A = (C * N_{heads} + H) \cdot N_{sectors} + (S - 1)$$

where A is the Logical Block Address, N_{heads} is the number of heads per cylinder and $N_{sectors}$ is the number of sectors per track

- Most disk drives now have 512 bytes per sector

Hard disk partitions

- A hard disk drive can be divided into a number of *partitions*
- Each partition can have its own characteristics: size, file system type, bootable, etc.
- An early method of creating multiple partitions involved writing the partition information to a *master boot record (MBR)* stored in sector 0.
- A MBR has space for 4 partition entries
- Extended partitions can be created to allow more than 4 partitions
- As the BIOS has been replaced by the *Unified Extensible Firmware Interface (UEFI)*, a new partition scheme (GUID partitions) has been introduced to go with it
- A GUID partition has a Globally Unique IDentifier
- The GUID partitions are stored in a GUID Partition Table (GPT), with space for 128 partitions
- Each partition starts at some logical block address (sector number) and consists of a specified number of sectors

Reading the partition information with `fdisk`

```
cgdk2@red:~$ sudo fdisk /dev/sda

Command (m for help): p

Disk /dev/sda: 640.1 GB, 640135028736 bytes
255 heads, 63 sectors/track, 77825 cylinders, total 1250263728 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x0000cd29

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048   772561766   386279859+  83  Linux
/dev/sda2       772562942  1250263039   238850049    5  Extended
/dev/sda5      1228443648  1250263039    10909696   82  Linux swap / Solaris
/dev/sda6       772562944  1228443647   227940352   83  Linux

Partition table entries are not in disk order

Command (m for help):
```

Copying an entire drive (dd)

- Copying a partition

```
dd if=/dev/sda1 of=/dev/sdb1 bs=64K conv=noerror,
sync status=progress
```

- Copying an entire drive

```
dd if=/dev/sdX of=/dev/sdY bs=64K conv=noerror,
sync status=progress
```

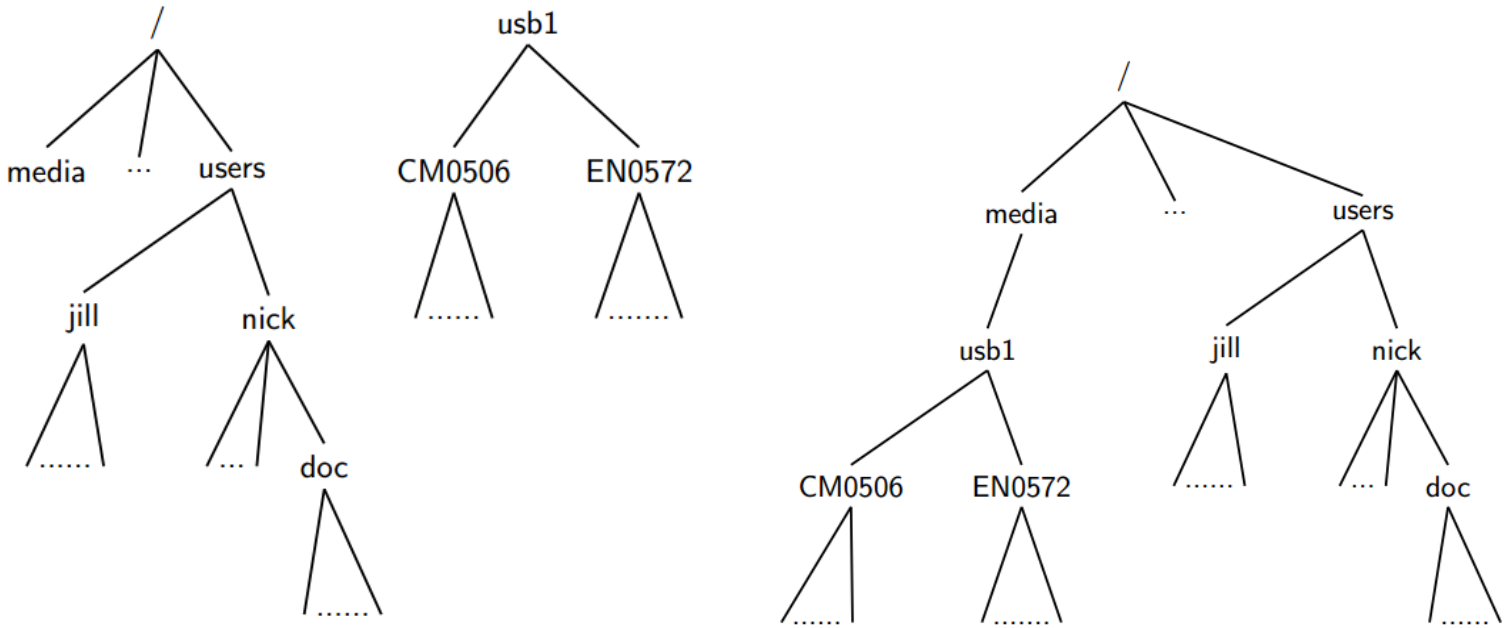
- Create a disk image

```
dd if=/dev/sdX conv=sync,noerror bs=64K |
gzip -c  > /path/to/backup.img.gz
```

- Cloning a Virtualbox virtual disk

```
VBoxManage clonemedium ubuntu.vdi
ubuntu.img --format RAW
```

Mounting a hard disk



Unmounted mounted

- In a Unix system, different devices can be *mounted* into the same file space
- The diagram shows an example of mounting a USB stick into the file space of a Unix system
- A disk drive can be read from or written to in a Unix system until it has been mounted

Using the `mount` command

- We use the `mount` command to mount a device
- We need a *mount point* for the device
- The mount point is just an empty directory, e.g. `/media`
- We specify the device and the mount point, e.g.
  ```
  sudo mount /dev/sdb1 /media
  ```
- Notice here that the USB drive appears as the *block device* `/dev/sdb1`
- A *file type* can be specified explicitly using the `-t` switch, but Linux usually does a good job of automatically determining the file type, e.g.
  ```
  sudo mount -t ext4 /dev/sdb1 /media
  ```
- View the mount table by using the `mount` command without any arguments.

Mounting a disk image

- In order to mount a disk image, we need to make it appear to Unix as a block device
- We can do this by specifying the `loop` option, e.g.
  ```
  sudo mount -o loop disk.img /media
  ```
- This works only if we have a single partition occupying the whole disk
- If there are multiple partitions, we need to also specify the *offset* of the start of the partition in bytes, e.g.
  ```
  sudo mount -o loop,offset=1048576 disk.img /media
  ```
- `fdisk` can be used to discover the starting sector of the partition
- You can use the arithmetic shell expression `$(( ... ))` to calculate the byte offset number, e.g.
  ```
  offset=$(( 2048 * 512 ))
  ```
- Options `ro` and `noatime` can be used to mount the disk image as read-only with no access time modifications allowed.

Using the `find` command to access file metadata

- `find` is a powerful command that can be used for a variety of file system tasks
- The basic use is to find a file matching a glob or a regular expression, e.g.
  ```
  find ~/courses -name "*.pdf" -print
  ```
- The `find` command has many other uses, e.g.
  ```
  find /bin -printf "%Ax;%AT"
  ```
  will print the last access date and time for all files in the `/bin` directory
- Use `man find` for more options

The `tar` command

The `tar` command stores and extracts files from a tape or disk archive. `tar` is a flexible command and has numerous options. As always, you can use `man` to find out about the full set of options. For now you should just practice the most frequent use cases.

- Create a compressed archive of a directory
  ```
  tar zcf somedir.tgz somedir
  ```
- Examine the contents of a compressed archive
  ```
  tar ztf somedir.tgz
  ```
- Extract the contents of a compressed archive
  ```
  tar zxvf somedir.tgz
  ```