

More on Assignment 1

CS110 Discussion Section 2

Agenda for Today

- Questions on Assignment 1 coding part?
- Hash Function: Checksum
- Naming Layers
- Symbolic Links
- Modern File Systems

Cryptographically secure hash function

- Hash function $F(\text{object}) \rightarrow \text{hash}$
 - One way: $F^{-1}(\text{hash})$ not known
 - Not known how to find another object Y different from the given object X that has the same hash value
 - Example: $F(\text{object}) \rightarrow \text{object}$
- Other nice names:
 - Message digest,
 - digital fingerprint

Assignment checksum is SHA1

- Widely used 160bit secure hash function
- Example: 2 files one bit different

```
t1.c: int main(void) { printf("Hello World\n");  
      return 0; }
```

```
t2.c: int main(void) { printf("Helln World\n");  
      return 0; }
```

Sha1(t1.c) =

0x52db68568747a4a34f3a4f2720f77d73a32ddce7

Sha1(t2.c) =

0x6144168a5a47891e2bc2b8a45ce947af8561419c

90 bits different!

Some example uses of SHA1

- Certifying info
 - Publish SHA1 hashes of distributed file
 - Recipients can tell if it has been modified
 - Name objects by SHA1
 - Easy check to see if name resolved to right object
 - Great if you don't trust your name service
- Very likely you read the files correctly if you get the hash correct
- Microsoft checks to see if they know about a device driver by its hash
- Publish hash of invention disclosure
 - Alternative to patenting

Naming Layers

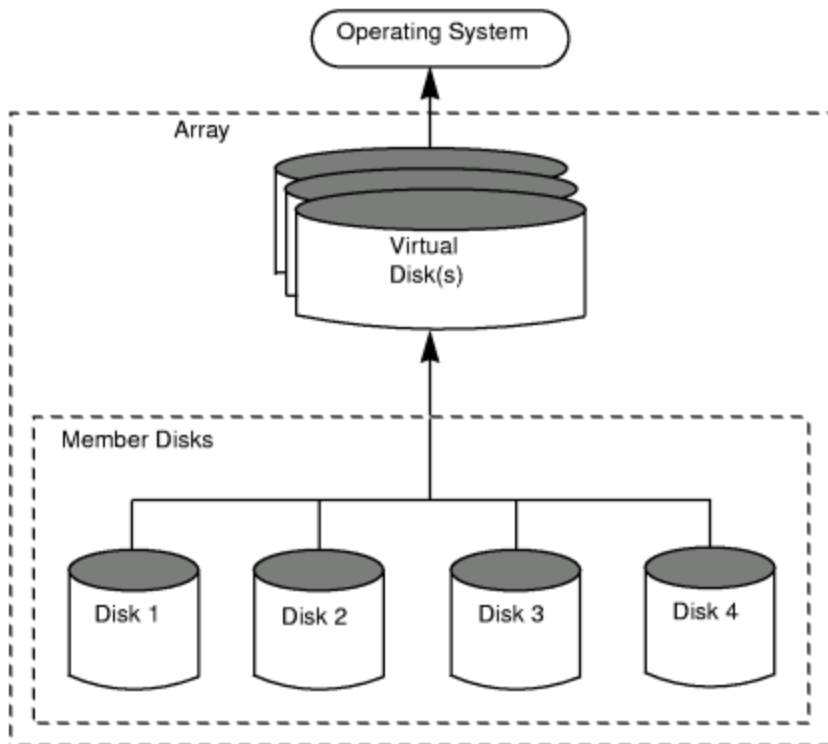
File system name spaces

Many OSes use a separate file system per disk volume

- Examples:
 - VMS disk: [dir.subdir] file.ext;version
 - DISK01:[DIRA.DIRB]FILE.TXT;2
 - Overloaded with disk and version info
 - DOS disk: pathame
 - C:\DIR2\DIRB\FOO.TXT
- Mostly hidden in name resolver by default
 - Environment holds current working disk volume and directory
- Disadvantages:
 - Clunky - Might need to remember which disk something is on
 - Absolute pathnames change if object moves between disks 7

Possible solution

- Make multiple disks look like a single disk to the file system

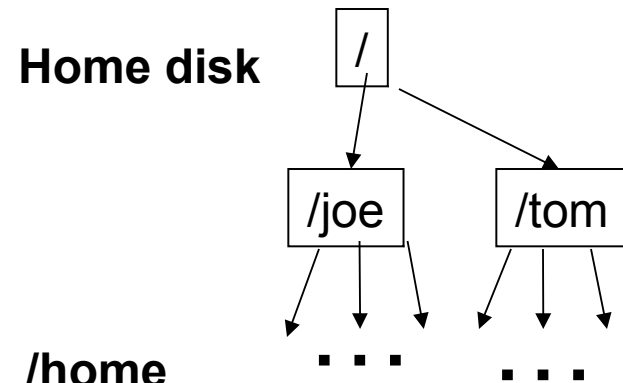
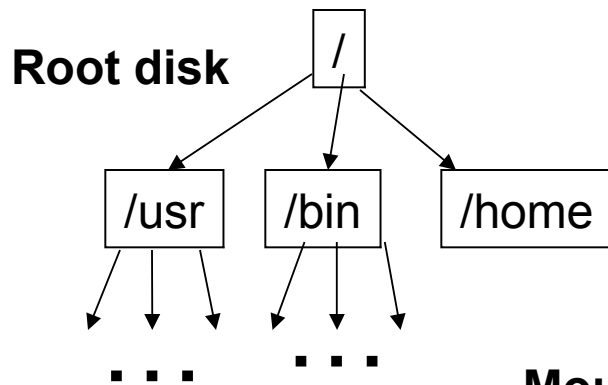


- Gang together multiple disks and export the **abstraction** of a single larger disk. Build a file system on it. As long as larger disk looks like a disk (READ/WRITE sector works) everything works.
- Idea is now called RAID
Redundant array of inexpensive disks
Includes redundancy to tolerate disk failures.

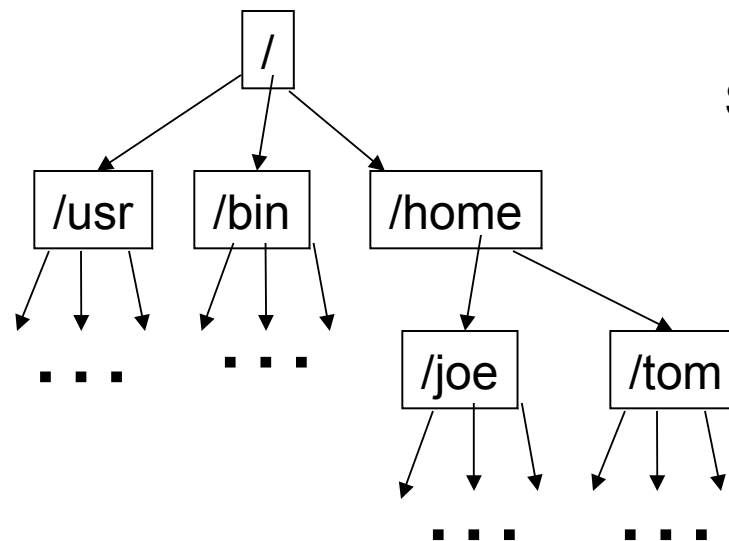
Unix approach

- Hide disk organization from file system name space
 - Each disk has its own file system/name space
 - Uses mount system call to splice together `mount(diskname, mountpoint, ...)`.
Place the file system on *diskname* started at the *mountpoint* pathname in the host file system
 - Exports the abstraction of a single name space
 - Mount table kept in volatile memory
 - Rebuilt on each reboot
 - Spooky if used on a non-empty mountpoint
 - Hides previous sub-tree at the mountpoint (previous files are still there, you just can't see them)

Mount point example



Mount Home Disk on `/home`



Single file system

Issues with mount points

- Mostly hide disk partitioning from user
 - Over-simplifying
- Some exceptions:
 - Disk filling up
 - See `df` command
 - Can create a file in a mounted disk but not in that filled disk
 - Could be viewed as a benefit as well
 - Links don't work across disk types (hard link)
 - Recall a link is a `<name, inumber>`
 - Inumbers unique inside each file system namespace
 - File move command is implemented using `link()/unlink()`.
 - Ugly when an abstraction hides something but then fails:
 - `mv /usr/bin/cmd /bin`
 - `mv /usr/bin/cmd /home/tom/bin`

Symbolic Links

BSD Unix adds symbolic links

- Symbolic link: a special file with a pathname in it
 - Also called a *soft link*
 - An *indirect name*
 - Works between disks
 - Different from Unix FS link which is called a *hard link*
- Name resolver reads file and follows path inside
- Example:

```
ln -s /usr/bin/cmd /home/tom/bin/cmd
```

 - Creates a special file at /home/tom/bin/cmd that contains “/usr/bin/cmd”
 - Doing an open(“/home/tom/bin/cmd”) will cause the name lookup to open “/usr/bin/cmd”

Soft links are not hard links

■ *Dangling references*

- `ln -s /usr/bin/cmd /home/tom/bin/cmd`
- `rm /usr/bin/cmd`
 - `Open("/home/tom/bin/cmd")` fails with file not found
- Not so with hard links - *reference count* in inode
 - `ln /usr/bin/cmd /home/tom/bin/cmd`
 - Increments reference count in inode (`i_nlink`) by 1
 - `rm /usr/bin/cmd`
 - Decrement reference count in inode (`i_nlink`) by 1, `nlink != 0`

■ Allows links to directories

- Form file system naming into arbitrary graphs (with cycles)

```
mkdir foo
```

```
ln -s ../foo foo/foo
```

```
cd foo; cd foo; cd foo; cd foo; cd foo; ...
```

Propagation of Effects

Add symbolic links to Unix

- Obvious changes:
 - System call to create them
 - New file type added in inode
 - Name lookup function changes
- Less obvious changes:
 - Any program that transverses name hierarchy
 - Deal with cycles and dangling references
 - Add new lstat() system call
 - Random programs that get confused by soft links

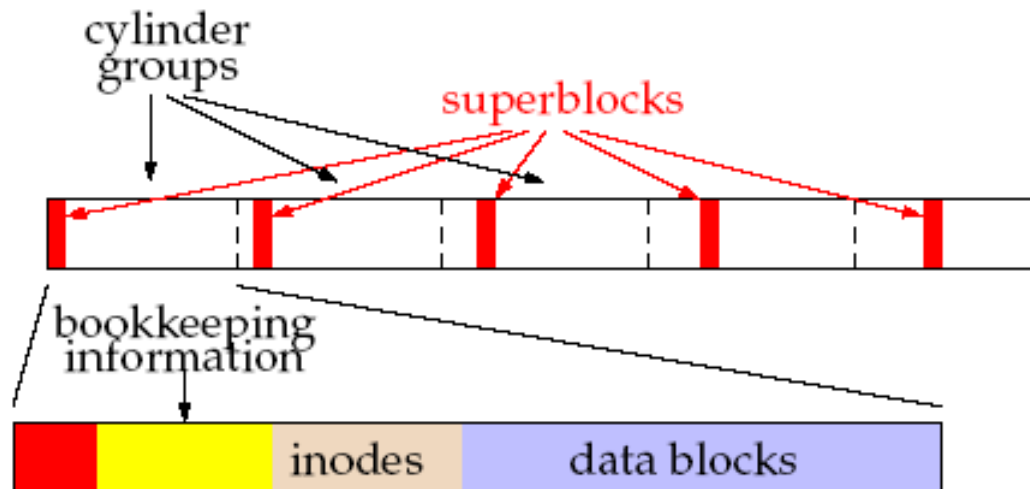
Some security issues

- Hash value of password stored
- Admin user privileges during some operation temporarily
- Symbolic link changed after check, and linked to some important system files, like passwd
- Passwd changed!!

Modern File Systems

BSD FFS

- **FFS disk layout:**



- **Each cylinder group has its own:**
 - Superblock, Bookkeeping information, Set of inodes, Data/directory blocks

BSD FFS

- **Change block size to at least 4K**
 - To avoid wasting space, uses “fragments” for ends of files
- **Cylinder groups spread inodes around disk**
 - Make inodes near datablocks
 - Avoid possible damage to the entire inode table
- **Bitmaps replace free list**
 - Better allocation of file blocks

BSD FFS

- **FS reserves space to improve allocation**
 - Tunable parameter, default 10%
 - Only superuser can use space when over 90% full
- **Usability improvements:**
 - File names up to 255 characters
 - Atomic *rename* system call