

Announcements

- Reading Chapter 18 (8th ed)
- MT#2 regrade requests due by Thursday
- Project #5 is due Th April 28th

Virtual Memory and File Cache

- Both need to contend for memory
- Possible solutions:
 - Fixed size allocation of buffer cache (I.e. 20% of memory)
 - Unified buffer cache and virtual memory system
 - All pages (memory and file buffer) compete for all of memory
 - Allows large processes or lots of file access as needed

Memory Mapped Files

- Can treat files like memory
 - Allows fast random access to files
 - Uses file cache to make operations fast
- Interface
 - Use mmap call to map file into memory (similar to open)
 - Use normal memory operations to access file (instead of read/write)
 - Use munmap to “close” file

Bad Blocks

- **Some blocks on a disk may not work**
 - could be bad from the start (when disk is installed)
 - could go bad during use
- **Two options to manage bad blocks**
 - disk drive maps the blocks to “replacement” blocks
 - special blocks that are held in reserve for this purpose
 - OS keeps track of where the bad blocks are located and avoids them
- **Replacement blocks**
 - can be located in tracks at one location, or around the disk
 - provide correct behavior, but change disk performance
- **Even if the disk re-maps bad blocks**
 - OS could lose data stored on disk
 - needs to be able to recover filesystem from partial update

Booting the OS

- How does the OS get loaded and started?
- Process is called booting
 - want to use the OS to load itself
 - but what loads the OS?
- ROM monitor
 - knows how to read from a fixed location on disk and jump into it
- Bootstrap program
 - knows how to load a program from the filesystem and jump into it
 - X86 PCs boot this way
- Alternative:
 - put more info into ROM about booting
 - MAC OS has most of the info in ROM
 - hard to change OS without changing ROMs

Booting the OS (cont.)

- Network Booting

- ROM knows how to request a boot packet from the network
 - once the packet is received, execute it
- useful for systems without local disks
- used by OS developers to ease edit/compile/boot cycles

Booting in GeekOS

- **PC Architecture**
 - Reads first sector on drive and then executes it
 - Hardware thinks it is a 16 bit 8088 processor at boot
 - Provides backwards compatibility
- **Boot Sector**
 - contains code to read
 - kernel.bin into memory
 - setup.bin into memory
 - uses bios to access drives
 - Includes a boot record to find kernel
- **Setup code**
 - Detects amount of memory
 - Moves processor to protected mode
 - Jumps to 32 bit code (and 32 bit mode)
 - Sets up initial kernel stack

GeekOS Booting Notes

- **Kernel and setup files**
 - Are normal files in what ever filesystem we have
 - Bootinfo record in boot sector tells how to find them
 - Must be in contiguous blocks on disk
 - A restriction in the boot sector code
- **Once booted**
 - Boot sector is ignored by main filesystem
 - Rest of disk is available to be used as desired
- **Have special utility to write boot sector**
 - Gosfs has a call `GOFs_BootInfo`

Swap Space

- Where is swap space located?
 - Is it a “normal” file in the filesystem?
 - Is it in a special location on disk?
- “normal” file
 - simple, just looks like a file
 - easy to change size
 - use normal tools
 - slow since it requires all of the filesystem overhead
- separate disk partition
 - faster
 - harder to change size (need a new partition)

Backups

- Disks can fail, so need to provide a way to copy them
- Need to plan for disasters too
 - What if the building burns down?
- Two types of backups
 - full backup (all of the data on disks)
 - incremental (data that has changed since last backup)
 - can mark changed files with a field
 - can use the date of the file compared to the last backup
 - permits several levels of backup
 - may want multiple levels of incremental (day, week changes)

Backups

- Does the system need to be shutdown for backups?
 - what if a file is moved during a backup?
 - it could get copied 0, 1, or 2 times.
 - easy answer is to shutdown the machine for backup
 - more typical setup:
 - Compute backup set
 - Backup files
 - Compute new backup set
 - Add any files that were missed