CS 3204
Operating Systems

Lecture 32

Godmar Back

Announcements

- Project 3 due Thursday, April 13, 11:59pm
- Office Hours today (Monday) 3-4pm
- 4pm today talk in McB 655 – open to the public
  - "On Efficient Buffer Cache Management"
- Additional office hours Tuesday 3-4pm
- Will assign TAs to keep labs open this week every evening until at least 7pm
- Recommended reading
  - Chapter 11.1-11.5, 11A
  - Chapter 12, in particular 12.7

Buffer Cache Prefetching

- Would like to bring next block to be accessed into cache before it's accessed
  - Exploit "Spatial locality"
- Must be done in parallel
  - use daemon thread and producer/consumer pattern
- Note: next(n) not always equal to n+1
  - although we try for it – via clustering to minimize seek times
- Don't initiate read_ahead if next(n) is unknown or would require another disk access to find out

```
b = cache_get_block(n, _);
cache_read_block(b);
cache_readahead(next(n));
```

```
queue q;
cache_readahead(sector s) {
   q.lock();
   q.add(request(s));
   signal qcond;
   q.unlock();
}
cache_readahead_daemon() {
  while (true) {
    q.lock();
    while (q.empty())
      qcond.wait();
    s = q.pop();
    q.unlock();
    read sector(s);
  }
}
```

Filesystems

Files vs Disks

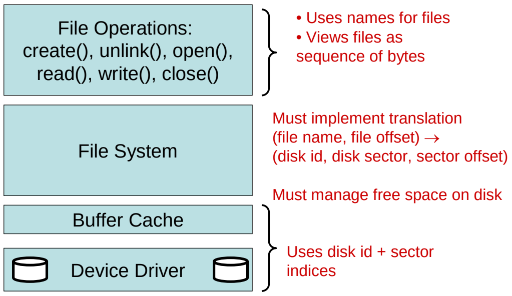
File Abstraction

- Byte oriented
- Names
- Access protection
- Consistency guarantees

Disk Abstraction

- Block oriented
- Block #s
- No protection
- No guarantees beyond block write

Overview


File Operations:
create(), unlink(), open(),
read(), write(), close()
• Uses names for files
• Views files as sequence of bytes
File System
Must implement translation (file name, file offset) → (disk id, disk sector, sector offset)
Must manage free space on disk
Buffer Cache
Device Driver
Uses disk id + sector indices

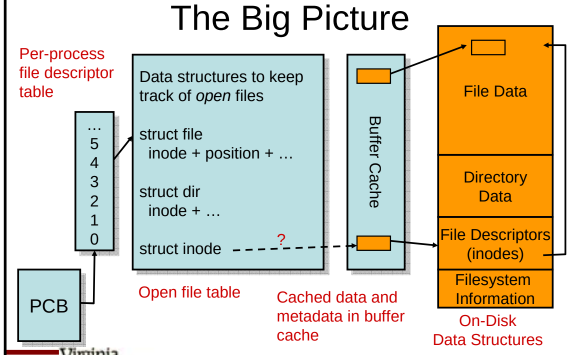
Filesystem Requirements

- Naming
  - Should be flexible, e.g., allow multiple names for same files
  - Support hierarchy for easy of use
- Persistence
  - Want to be sure data has been written to disk in case crash occurs
- Sharing/Protection
  - Want to restrict who has access to files
  - Want to share files with other users

FS Requirements (cont'd)

- Speed & Efficiency for different access patterns
  - Sequential access
  - Random access
  - Sequential is most comment & Random next
  - Keyed access (not usually provided by OS)
- Minimum Space Overhead
  - Disk space needed to store metadata is lost for user data
- Twist: all metadata that is required to do translation must be stored on disk
  - Translation scheme should minimize number of additional accesses for a given access pattern
  - Harder than, say page tables where we assumed page tables themselves are not subject to paging!

The Big Picture



Per-process file descriptor table

…
5
4
3
2
1
0

PCB

Data structures to keep track of *open* files

struct file
inode + position + …

struct dir
inode + …

struct inode

?

Open file table

Buffer Cache

Cached data and metadata in buffer cache

File Data

Directory Data

File Descriptors (inodes)

Filesystem Information

On-Disk Data Structures

Steps in Opening & Reading a File

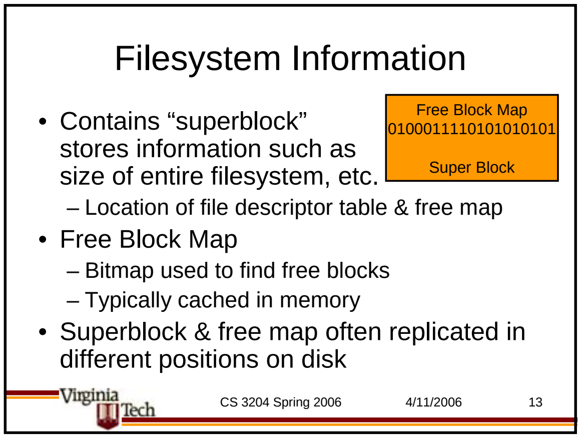
- Lookup (via directory)
  - find on-disk file descriptor's block number
- Find entry in open file table (struct inode list in Pintos)
  - Create one if none, else increment ref count
- Find where file data is located
  - By reading on-disk file descriptor
- Read data & return to user

Open File Table

- inode – represents file
  - at most 1 in-memory instance per unique file
  - #number of openers & other properties
- file – represents one or more processes using an file
  - With separate offsets for byte-stream
- dir – represents an inode of a directory file
- Generally:
  - None of data in OFT is persistent
  - Reflects how processes are currently using files
  - Lifetime of objects determined by open/close
    - Reference counting is used

File Descriptors ("inodes")

- Term "inode" can refer to 3 things:
  1. in-memory inode
     - Store information about an open file, such as how many openers, corresponds to on-disk file descriptor
  2. on-disk inode
     - Region on disk, entry in file descriptor table, that stores persistent information about a file – who owns it, where to find its data blocks, etc.
  3. on-disk inode, when cached in buffer cache
     - A bytewise copy of 2. in memory
- Q.: Should in-memory inode store a pointer to cached on-disk inode?



Filesystem Information

- Contains “superblock” stores information such as size of entire filesystem, etc.
  - Location of file descriptor table & free map
- Free Block Map
  - Bitmap used to find free blocks
  - Typically cached in memory
- Superblock & free map often replicated in different positions on disk

Free Block Map
01000111101010101O1

Super Block