

Review 2

Dave Eckhardt
de0u@andrew.cmu.edu

Synchronization

- Exam will be closed-book
 - *But you may bring a 1-sided 8.5x11 sheet of notes*
 - 6 point font or larger
 - *Weakly* non-cumulative
 - Emphasis on new material
 - You will need to use some “old” knowledge

Synchronization

- About today's review
 - Mentioning key concepts
 - No promise of exhaustive coverage
 - Reading *some* of the textbook is advisable

Synchronization

- Faculty evaluation forms
 - *middle* of class?
- *Read bboards during final exam period*
- *Read bboards during final exam period*
- SCS Facilities hacking jobs

Core “Phase I” concepts

- Process model
 - You should be a memory-map expert
 - Kernel space, user space
 - Process vs. thread
- Mutual exclusion
 - mutex, cvar, what's inside, why
- Deadlock

IPC

- Communicating process on one machine
- Naming
 - Name server?
 - File system?
- Message structure
 - Sender id, priority, type
 - Capabilities: memory region, IPC rights
- Synchronization/queueing/blocking

IPC

- Group receive
- Copy/share/transfer
- A Unix surprise
 - `sendmsg()/recvmsg()` pass file descriptors!

RPC Overview

- RPC = Remote Procedure Call
- Extends IPC in two ways
 - IPC = Inter-Process Communication
 - OS-level: bytes, not objects
 - IPC restricted to single machine
- *Marshalling*
- Server location

RPC Overview

- Call semantics
 - Asynch? Batch? Net/server failure?
- Client flow, server flow
 - Stub routines, dispatch skeleton
- Java RMI

Marshalling

- Values must cross the network
- Machine formats differ
 - Integer byte order
 - www.scieng.com/ByteOrder.PDF
 - Floating point format
 - IEEE 754 or not
 - Memory packing/alignment issues

Marshalling

- Define a “network format”
 - ASN.1 - “self-describing” via in-line tags
 - XDR – not
- “Serialize” language-level object to byte stream
 - Rules typically recursive
 - Serialize a struct by serializing its fields in order
 - Implementation probably should *not* be

Marshalling

- Issues
 - Some types don't translate well
 - Ada has ranged integers, e.g., 44..59
 - Not everybody really likes 64-bit ints
 - Floating point formats are religious issues
 - Performance!
 - Memory speed \cong network speed
 - The dreaded “pointer problem”
 - See lecture notes

File System Interface

- Abstraction of disk/tape storage
 - Records, not sectors
 - Type information
- Naming
 - Directory tree
 - Complexity due to linking
 - Soft vs. hard links

File System Interface

- Mounting
- Ownership, permissions
- Semantics of multiple open()s

Operations on Files

- Create – locate space, enter into directory
- Write, Read – according to position pointer
- Seek – adjust position pointer
- Delete – remove from directory, release space
- Truncate
 - Trim data from end
 - Often all of it
- Append, Rename

File System Layers

- Device drivers
 - read/write(disk, start-sector, count)
- Block I/O
 - read/write(partition, block) [cached]
- File I/O
 - read/write(file, block)
- File system
 - manage directories, free space, mounting

Disk Structures

- Boot area (first block/track/cylinder)
- File system control block
 - Key parameters: #blocks, metadata layout
 - Unix: superblock
- Directories
- “File control block” (Unix: inode)
 - ownership/permissions
 - data location

Memory Structures

- In-memory partition tables
- Cached directory information
- System-wide open-file table
 - In-memory file control blocks
- Process open-file tables
 - Open mode (read/write/append/...)
 - “Cursor” (read/write position)

VFS layer

- Goal
 - Allow one machine to use multiple file system *types*
 - Unix FFS
 - MS-DOS FAT
 - CD-ROM ISO9660
 - Remote/distributed: NFS/AFS
 - Standard system calls should work transparently
- Solution
 - Insert a level of indirection!

VFS layer – file system operations

```
struct vfsops {  
    char *name;  
    int (*vfs_mount)();  
    int (*vfs_statfs)();  
    int (*vfs_vget)();  
    int (*vfs_unmount)();  
    ...  
}
```

Directories

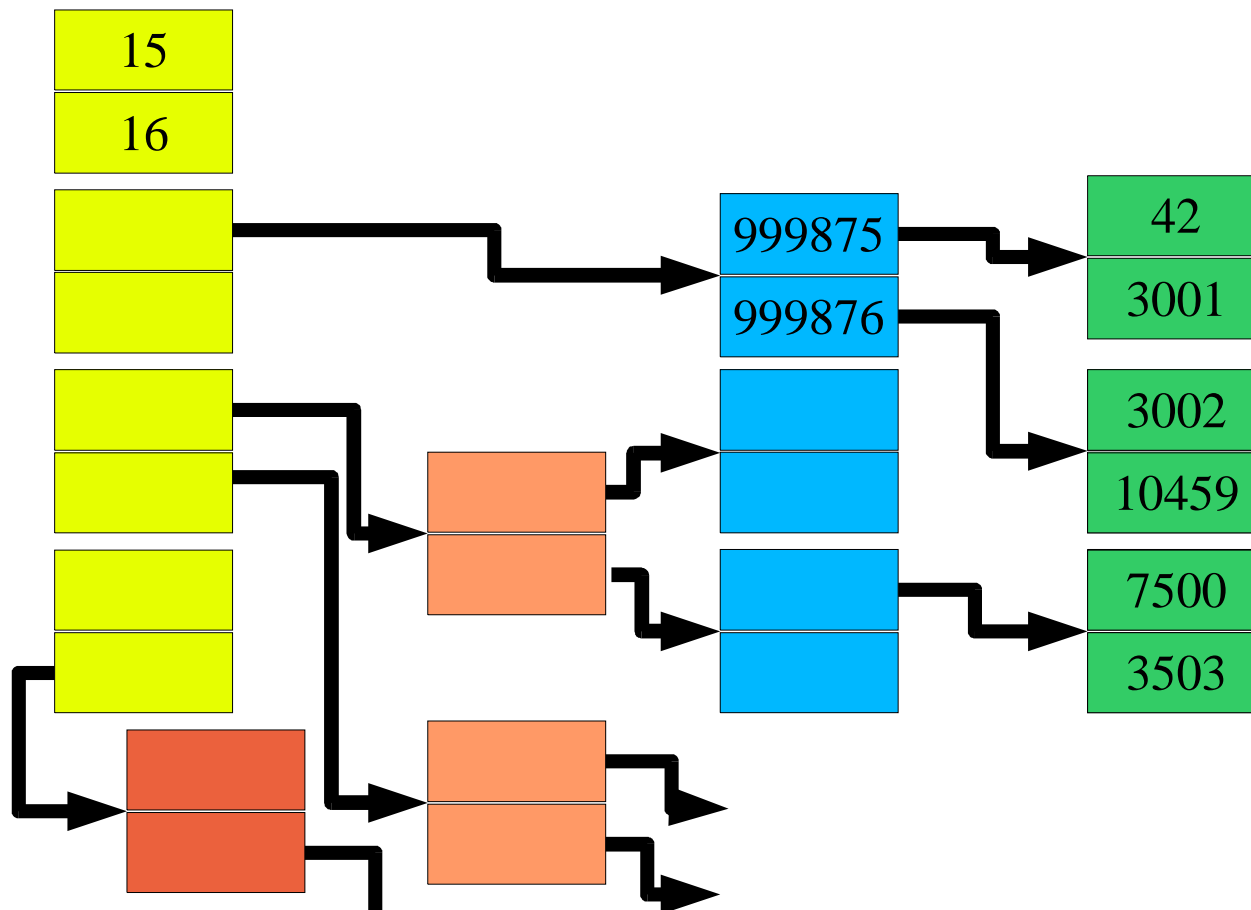
- External interface
 - `vnode = lookup(vnode, name)`
- Traditional Unix FFS
 - List of (name,inode #) - not sorted
 - Names are variable-length
 - Lookup is linear
 - How long does it take to delete N files?
- Common alternative: hash-table directories

Allocation - FAT

7
2
5
-1
3
-1
0
-1

hello.java	0
dir.c	1
sys.ini	4

Unix Index Blocks



Cache tricks

- Read-ahead

```
for (i = 0; i < filesize; ++i)  
    putc(getc(infile), outfile);
```

- System observes sequential reads

- can pipeline reads to overlap “computation”, read latency

- Free-behind

- Discard buffer from cache when next is requested

- Good for large files

- “Anti-LRU”

Recovery

- System crash...now what?
 - Some RAM contents were lost
 - Free-space list on disk may be wrong
 - Scan file system
 - Check invariants
 - Unreferenced files
 - Double-allocated blocks
 - Unallocated blocks
 - Fix problems
 - Expert user???

NFS & AFS

- VFS interception
- NFS & AFS
 - Architectural assumptions & goals
 - Namespace
 - Authentication, access control
 - I/O flow
 - Rough edges

NFS Assumptions, goals

- Workgroup file system
 - Small number of clients
 - Very small number of servers
- Single administrative domain
 - All machines agree on “set of users”
 - ...which users are in which groups
 - Client machines run mostly-trusted OS
 - “User #37 says read(...)”

NFS Assumptions, goals

- “Stateless” file server
 - Files are “state”, but...
 - Server *exports* files without creating extra state
 - No list of “who has this file open”
 - No “pending transactions” across crash
 - Result: crash recovery “fast”, protocol “simple”
- Some “stateful” operations
 - File locking
 - Handled by separate service outside of NFS

AFS Assumptions, goals

- Global distributed file system
 - Uncountable clients, servers
 - “One AFS”, like “one Internet”
 - Why would you want more than one?
- Multiple administrative domains
 - username@*cellname*
 - davide@cs.cmu.edu de0u@andrew.cmu.edu

AFS Assumptions, goals

- Client machines are un-trusted
 - Must *prove* they act for a specific user
 - Secure RPC layer
 - Anonymous “system:anyuser”
- Client machines have disks
 - Can cache whole files over long periods
- Write/write and write/read sharing are rare
 - Most files updated by one user, on one machine

AFS Assumptions, goals

- Support *many* clients
 - 1000 machines could cache a single file
 - Some local, some (very) remote

AFS Callbacks

- Observations
 - Client disks can cache files indefinitely
 - Even across reboots
 - Many files nearly read-only
 - Contacting server on each open() is wasteful
- Server issues *callback promise*
 - If this file changes in 15 minutes, I will tell you
 - *callback break* message
 - 15 minutes of free open(), read()

Disk scheduling

- Spinning platter/waving arm model
- Seek time vs. rotational latency
- FCFS, SSTF, SCAN, LOOK, C-SCAN, C-LOOK, SPTF, WSPTF
- Fairness, mean response time, variance, starvation
- Freeblock scheduling
 - Concept

Disk Array Overview

- Historical practices
 - Striping, mirroring
- The reliability problem
 - More disks \Rightarrow *frequent* array failures
 - *Cannot* tolerate $1/N$ reliability
- Parity, ECC, why parity is enough
 - Erasure channels
 - Good terminology to display at parties

Disk Array Overview

- RAID “levels” (really: flavors)
 - Understand RAID 0, 1, 4 vs. 5
 - What they're good for, why

Host Naming Overview

- Three names for your PC
 - Why?
- Two resolution protocols
 - DNS, ARP

Three names for ~~your~~ my PC

- PIPER.NECTAR.CS.CMU.EDU

- What's a “nectar”?

- What's a “piper”?

- 128.2.194.80

- 00-20-AF-D9-FD-CA

- *All* are globally unique

- Won't one do?

Questions about names

- Who uses the name?
 - For what?
- Who owns/defines the namespace?
- How long is the name valid?

Three names for my PC

- User specifies host name
- Data packet sent to IP address
- Last-hop router must know MAC address
- Two lookup problems
 - Name -> IP address: global, pretty stable
 - IP address -> MAC address: local, variable
- Two protocols
 - DNS – multi-level tree
 - ARP – local broadcast

Lamport Clocks Overview

- Light cones
- Meeting for beer
- “Happened before” partial order
- Logical clocks
 - “Happened before” partial order
 - Potential causality
 - Another definition of concurrency

Lamport Clocks Overview

- Advanced techniques
 - Total orders
 - Fair distributed mutual exclusion

Protection Overview

- Protection vs. Security
 - Inside vs. outside “the box”
- Objects, operations, domains
- Access control (*least privilege*)
- 3 domain models
- Domain switch (setuid example)
- Multics ring architecture

Protection Overview

- Access Matrix
 - Concept and real-world approaches
- Capability revocation

Mobile Code Safety Outline

- Motivation
 - Packet-filter example
- Careful hardware, careful interpreter
- Trusted compiler / signed code
- Java 2-level verification
 - Low-level properties via *byte-code verifier*
 - High-level properties via pluggable security modules
- Software fault isolation, proof-carrying code

Mobile Code Safety Metrics

- Performance?
 - Time to start running
 - Throughput while running
- Safety policy
 - Static (language designer)
 - Dynamic (system administrator)
- Trust model
 - Trust people, programs, or proofs?

Mobile Code Safety Summary

- Careful hardware – could EROS make it happen?
- Careful interpreter – ok for slow applications
- Trusted compiler / Signed code – dubious
- Byte code verifier – has its uses, limits
- Software Fault Isolation – hmm...
- Proof-Carrying Code – hmm...

Plan 9, XOK, Eros

- “Incorporated by reference”
- Hard to test on these directly
 - Can I meet the challenge?

Security Overview

- Goals & threats
 - Authentication (impersonation)
 - Secrecy (theft, eavesdropping)
 - Integrity (cracking)
 - Signature (repudiation)
- TEMPEST (and low-tech snooping)

Security Overview

- Malware
 - Trojans, trapdoors
 - Buffer overflow
 - Viruses, worms
- Password files, salt
- Biometrics vs. cheating

Security Overview

- “Understand cryptography”
 - *Secure* hashing
 - One-time pad
 - Symmetric (private-key) crypto
 - Asymmetric (public-key) crypto
 - Has private keys and public keys
 - Kerberos
 - Symmetric crypto
 - Central server avoids the n^2 problem

Transactions

- A different kind of critical section
- "ACID" Transaction Model
 - Atomic, consistent, isolated, durable
- Unifying concept for system building
- Write-ahead logging, replay during restart
- Concurrency control (serializability)
- Distributed transactions, 2-phase commit

Preparation Suggestions

- Sleep well (*two* nights)
- Scan lecture notes
- Read any skipped textbook sections
- Understand the code you turned in
 - Even what your partner wrote

Preparation Suggestions

- Prepare a sheet of notes
- RoboCup, but not *too* much...
- Read comp.risks & Effective Java
 - Ok, after the exam will suffice
- Don't panic!
 - Budget time wisely during exam