

CS 140: Operating Systems Lecture 28: Course Review

Mendel Rosenblum

Operating Systems

- ◆ What does an OS do:
 - Manage and protect hardware resources.**
 - Export useful, abstractions to access resources:**
E.g. Process, Signals, Virtual Memory, Files, Sockets
- ◆ Typical “kernel” organization:
 - OS kernel runs in most privileged processor mode.**
 - Full access to all hardware resources**
 - Runs everything else in “user” mode.**
 - Restricted access to resource**

Topic Processes

- ◆ Implementation:
 - state, creation, dispatching, context switch.
- ◆ Synchronization:
 - Race conditions, inconsistencies.**
 - Mutual exclusion, Critical sections.**
 - Semaphores: P() and V().**
 - Producer & Consumer problems.
 - Scheduling problems
 - Semaphore implementations.**
 - Atomic operations: interrupt disable, test-and-set.
 - Monitors & Condition Variables**
 - Deadlock.**

State of the Art: Processes

- ◆ Pretty much all OS environments have:
 - Address space + one or more threads of control**
 - Synchronization primitives – Semaphores**
 - Language environments with monitors (Java, C#)**
- ◆ Modern architectures support for non-blocking ops
 - Compare and swap**
 - Research: Transactional memory**
- ◆ Problems still exist
 - Deadlock – Careful programming, orderings.**
 - Race conditions – Source of bugs, Eraser-like checking**

CPU Scheduling, Memory management

- ◆ Allocation -- Non-preemptible resources.
- ◆ Scheduling -- Preemptible resources.
 - FIFO, Round-robin, STCF**
 - Adaptive: Exponential Queue, Fair Share**
 - Lottery**
- ◆ Linking, static relocation.
 - Segments: Code, heap, stack.**
 - Linker passes, relocation info, cross-references.**
- ◆ Dynamic memory management:
 - First fit, Best fit.**
 - Implementation: Bitmaps, Pools, Garbage collection.**

State of the Art: CPU/Memory

- ◆ CPU scheduling – still a research area
 - Quality of Service**
 - Realtime**
 - Guarantees**
- ◆ Growth of code has forced dynamically linking
 - Static linking only at module level**
- ◆ Garbage collection is still controversial for OS work
 - Norm: Implicit memory management (alloc/free)**
 - Reference counts**

Virtual memory management

- ◆ Base and bounds.
- ◆ Paging.
- ◆ Segmentation.
- ◆ Page & segmentation.
- ◆ TLBs.
- ◆ Demand paging.
- ◆ Page replacement: LRU, clock.
- ◆ Thrashing, Working sets.

State of the art: VM

- ◆ All OS virtual memory look the same:
Software segments with demand paged memory
- ◆ Paging algorithms less important:
Just buy enough memory.
Infinitely big programs handle own memory (e.g. DB)
- ◆ All architecture support paged virtual memory
OSes assume address space abstraction.
Ease fragmentation of main memory.
- ◆ Research:
Big pages for dealing with TLB overheads

File systems, Disks

- ◆ Addressing:
Sequential, random, keyed.
- ◆ Storage management:
Contiguous allocation.
Linked files.
Indexed files. (Multi-level Unix style index).
Crash Recovery (Logging, shadowing)
- ◆ Block cache
- ◆ Freelist, bitmaps.
- ◆ Naming:
Descriptor organization, directories.
- ◆ Disk scheduling: FIFO, SSTF, Scan.

State of the art: File System

- ◆ Pretty much every OS supports a hierarchical FS
Write-ahead logging for performance and crash recovery.
- ◆ Trade off buffer cache and virtual memory
Memory mapped files example.
- ◆ Research: Content based retrieval
Fast search – FS needs to interpret file contents.
- ◆ Disk scheduling implemented but not used too much.

Protection & security:

- ◆ Authentication:
Passwords, keys.
- ◆ Authorization determination:
Access matrix
Access lists.
Capabilities.
- ◆ Access enforcement:
Security kernel.
- ◆ Attack methods:
Abuse of privileges.
Trojan Horse.
Listener.
Spoiler.
Worm or virus.

Security Defenses

- ◆ Logging.
- ◆ Caller identification.
- ◆ Principle of minimum privilege.
- ◆ Correctness proofs.
- ◆ Encryption:
Private key: DES
Public key systems: RSA
Digital signatures.
Message Digests.

State of the art: Security and Protection

- ◆ Huge issue today:
 - Authentication – passwords, too weak.**
 - Authorization – ACL on files, some capabilities.**
 - Access enforcement – Need no bugs!**
- ◆ Implications – Beware of complexity
 - Most features not most secure.**
- ◆ Encryption being deployed
- ◆ Research – Attestation how to deploy

Networks & communication

- ◆ Link-level
 - Point-to-point, bus, encoding**
 - Ethernet**
- ◆ Network Level.
 - Datagrams**
 - Virtual circuits**
 - IP**
- ◆ End-to-End.
 - TCP – Acks, sliding window**
 - Distributed consensus**

State of the art: Networking

- ◆ Clearly the biggest impact on OS area
- ◆ Distributed system:
 - New problems: Latency, bandwidth, failure, and trust.**
 - Breaks everything.**

Major concepts

- ◆ Locality:
- ◆ Scheduling:
 - best algorithms know future, but we use past instead.**
- ◆ Layering:
 - synchronization, network protocols, file systems, etc.**
- ◆ Caching:
 - translation look aside buffers, file system, etc.**