Overview of an Operating System

- Main components of an operating system
- Papers that we will be covering in this course

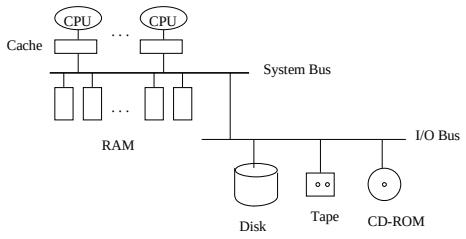
Today's Operating Systems

- UNIX:
  - BSD 4.x --> AIX, HP-UX, …
  - System V --> Solaris, ...
  - --> Linux
  - Mach --> OSF/1 --> DEC OSF/1…
- IBM VM/370, VAX/VMS --> Windows NT
- DOS --> Windows 3.x --> Windows 95 --> Windows 98 ...
- MacOS, OS/2, PalmOS, …

Reasons for Operating Systems

- An abstraction layer for the hardware
  - Application programming interfaces
    - file I/O API
    - networking API
    - memory management API
- A resource manager
  - Processes
  - Virtual memory
  - Information storage and protection

Hardware Components



Purposes of Operating Systems

- Provide an easier-to-use machine
  - Shield programmers and users from physical device details
  - Provide memory models for programs and compilers
- Manage resources
  - An "orderly and controlled" allocation of CPU, RAM, Network, I/O
  - Protection

Components in an OS

- Process management
- Memory management
- File system and I/O management
- Networking and distributed systems

Processes

- Process: execution of a program
- Process hierarchies
  - e.g. init -> login/shell -> tcsh -> cp ...
- Process states: ready, running, blocked
- Process scheduling
  - non-preemptive scheduling
  - preemptive scheduling

Process Table

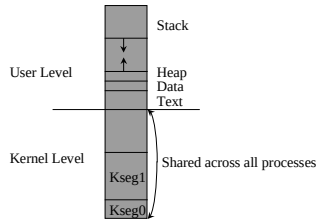
One entry per process, including

- process PID, parent PID, real/effective uid, etc.
- process state, signal status, signal bits
- timers, alarm clocks
- pointer to u area

u area includes

- file descriptor table, current directory, root
- system call result, system call error
- region tables or page tables

Process Address Space



Process Context

- user-level context
  - text, data, user stack, and shared memory, etc.
- register context
  - program counter, status register, stack pointer, general-purpose registers
- system-level context
  - process table entry, u area, region/page tables, kernel stack

Interrupts

Steps in processing an interrupt:

1. Hardware saves current process's PC, etc.
2. Hardware loads new PC from interrupt vector
3. Assembly language routine saves register, then sets up new stack
4. C procedure services the interrupt, then returns to the assembly code
5. Scheduler decides which process to run next
6. Assembly language procedure starts up current process

Threads

- Motivation
  - concurrency
  - parallelism
  - overhead concerns
- Like a process, but doesn't own an address space
  - Private resources: program counter, stack, register contexts, lock states
  - Shared resources: address space, open files, user credentials, quotas

Kernel Support for Threads

- Light-weight processes: a kernel-supported thread at user level
- User threads: thread abstraction at user-level only; kernel has no knowledge
  - example: pthreads
- Kernel threads: created and used by the kernel

The Need for Synchronization

- Processes sharing regions of memory
- Threads interacting with each other

Concurrency & Synchronization

- Synchronization mechanisms:
  - Hoare, "Communicating Sequential Processes"
  - Hoare, "Monitors: An Operating System Structuring Concept"
  - Lampson & Redell, "Experience with Processes and Monitors in Mesa"
  - Lauer & Needham, "On the Duality of Operating System Structures"
- Thread scheduling:
  - Anderson et al "Scheduler Activation"

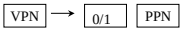
Memory Management

Management of DRAM

- No swapping, no paging
- Swapping
- Virtual memory (or Paging)
  - establish and maintain virtual address to physical address mapping
  - decide what virtual pages are in memory

Virtual Memory Management

Page table (per process):



Necessary hardware support

Page fault handling

Page replacement

Virtual Memory Mechanisms

- Levy & Lipman, "Virtual Memory Management in VAX/VMS"
- Babauglu & Joy, "Converting a Swap-Based System to do Paging in an Architecture Lacking Page-Reference Bits"

More on Virtual Memory

Policy study
- Denning, "Working Set"
- Hennesey, "WS Clock"

Old Systems
- Bensousan, et al "The Multics Virtual Memory: Concepts and Design"
- Young, et al "The Duality of Memory and Communication in the Implementation of a Multiprocessor Operating System"

Software DSM

Distributed Shared Virtual Memory
- Li & Hudak, "Memory Coherence in Shared Virtual Memory Systems"
- Carter, Bennett & Zwaenepoel, "Techniques for Reducing Consistency-Related Communication in Distributed Shared-Memory Systems"

File System

User interface
- Example: a file is a sequential byte stream; directory hierarchy; path names
- File attributes
- Operations:
  - create/delete, open/close, read/write, seek, get/set attributes, rename
  - opendir/closedir, readdir, link/unlink

File System Implementation

- how to keep track of where files are on the storage device (for example, the disk)
- how to allocate disk blocks to files
- how to implement directories
- how to make file system reliable and maintain file system consistency
- improving file system performance

On File Systems

- A few words about disks
  - Ruemmler & Wilkes, "An Introduction to Disk Drive Modeling"
- Reliable storage
  - Patterson et al "A Case for Redundant Array of Inexpensive Disks (RAID)"
- Classic implementation
  - McKusick, et al "A Fast File System for UNIX"

More on File Systems

- Study on file access patterns
  - Baker et al "Measurement of a Distributed File System"
- More advanced implementation
  - Rosenblum & Ousterhout, "The Design and implementation of a Log-Structured File System"

Communication Mechanisms

Protocol implementation:

- TCP/UDP, IP, link layer

APIs

- Sockets
- RPC

Distributed Systems

- Distributed file systems
  - NFS
  - AFS
  - file systems over mobile environments
- WWW
- General distributed systems
  - Java-based systems
  - Various research issues

On Distributed Systems

Examples

- Birrell et al "Grapevine: An Exercise in Distributed Computing"

Implementation

- Birrell, Nelson "Implementing Remote Procedure Calls"

High performance networking

- Druschel, Peterson, "Fbufs: A High-Bandwidth Cross-Domain Transfer Facility"

Distributed System Examples

- Distributed file systems
  - Sandberg et al "Design and Implementation of the Sun Network File System" (NFS)
  - Howard et al "Scale and Performance in a Distributed File System" (AFS)
  - Kistler "Disconnected Operation in the Coda File System"
- WWW:
  - Fan et al "Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol"

Security and Protection

- Security models in OS
  - user based
  - capability based
- Security mechanisms
  - passwords
- Network security concerns

Security

- Authentication: how to know this person is Alice
  - Rivest et al "A Method for Obtaining Digital Signatures and Public Key Cryptosystems"
  - Steiner et al "Kerberos: An Authentication Service for Open Network Systems"
- Authorization: How to keep Alice from doing bad things
  - Wallach et al "Extensible Security Architecture for Java"

Case Studies

- Early Systems
  - Ritchie & Thompson, “The UNIX Timesharing System”
  - Redell et al “Pilot: An Operating System for a Personal Computer”
- Recent Systems
  - UNIX System V
  - Windows NT
- Research Systems: SPIN, Synthesis

Assignment Schedule

- 10/5: first assignment due;
- 10/19: Project proposal due;
- 11/16: Interim report due;
- 12/9: Project due;
- 12/9 and 12/14: Project presentation;