Chapter 1: Introduction

- Os is a resource allocator, manages all resources
- OS is a control program, controls executions of programs.
- An operating system is interrupt driven
- Interrupt handling, determines which type of interrupt has occurred
  polling or vectored interrupt system
- Two types of Dual-Core designs, UMA and NUMA
- Clustered Systems, like multiprocessor systems but multiple systems working together usually sharing storage via a storage-area network (SAN)
- Kernel Data structures: Binary Search Tree, Balanced Binary Search Tree, Hash map, Bitmap
- Computing Environments:
  - Software as a Service (SaaS)
  - Platorm as a Service (PaaS)
  - Infrastructure as a Service (IaaS)

Chapter 2: System Structures

User Interface:

- Command-Line(CLI), Graphics User Interface (GUI), Batch

Accounting:

- To keep track of which users use how much and what kinds of computer resources.

System Calls, mostly accessed by programs via high level Application Program Interface(API)

- Three most used: Win32 API, POSIX API(UNIX, Linux, MacOSx), Java API for JVM

Background Services:

- Launch at boot time
- Known as services, subsystems, daemons

Microkernel System Structure

- Mach example of microkernel, Mac OS X partly based on this.
- Communication takes place between user modules using message passing

Modules

- Most modern operating systems implement loadable kernel modules.

Hybrid Systems

- Linux and Solaris kernels in kernel address space, so monolithic, plus modular for dynamic loading of functionality
- Windows mostly monolithic, plus microkernel for different subsystem personalities
- Apple Mac OS X hybrid, layered, Aqua UI plus Cacoa programming environment

System Boot

- Bootstrap loader, stored in ROM or EEPROM locates the kernel, loads it into memory and starts it.
- Common bootstrap loader, GRUB allows selection of kernel from multiple disks, versions, kernel options

Chapter 3: Process Concept

An operating system executes a variety of programs:

- Batch system – jobs
- Time-shared systems – user programs or tasks

Multiple parts

- The program code, also called text section
- Current activity including program counter, processor registers
- Stack containing temporary data
- Function parameters, return addresses, local variables
- Data section containing global variables
- Heap containing memory dynamically allocated during run time

As a process executes, it changes state

- new: The process is being created
- running: Instructions are being executed
- waiting: The process is waiting for some event to occur
- ready: The process is waiting to be assigned to a processor
- terminated: The process has finished execution

Long-term scheduler (or job scheduler) –selects which processes should be brought into the ready queue

Short-term scheduler (or CPU scheduler) –selects which process should be executed next and allocates CPU

UNIX, process creation

- fork()system call creates new process
- exec()system call used after a fork()to replace the process'memory space with a new program

If no parent waiting, then terminated process is a zombie

If parent terminated, processes are orphans

Google Chrome Browser is multiprocess with 3 categories

- Browser process manages user interface, disk and network I/O
- Renderer process renders web pages, deals with HTML, Javascript, new one for each website opened RidbtitidikdtkI/Oiiiifftfitlit
- Runs in sandbox restricting disk and network I/O, minimizing effect of security exploits
- Plug-in process for each type of plug-in

Processes within a system may be independent or cooperating

- Independent process cannot affect or be affected by the execution of another process
- Cooperating process can affect or be affected by the execution of another process

Synchronization

Message passing may be either blocking or non-blocking

Blocking is considered synchronous

- Blocking send has the sender block until the message is received
- Blocking receive has the receiver block until a message is available

Non-blocking is considered asynchronous

- Non-blocking send has the sender send the message and continue
- Non-blocking receive has the receiver receive a valid message or null

Buffering

Queue of messages attached to the link; implemented in one of three ways

- Zero capacity – 0 messages, sender must wait for receiver (rendezvous)
- Bounder capacity – finite length of n messages, sender must wait if link full
- Unbounded capacity – infinite length, sender never waits

Communications in Client-Server Systems

- Sockets
- Remote Procedure Calls
- Pipes
- Remote Method Invocation (Java)

Sockets

A socket is defined as and endpoint for communication

Communication consists between a pair of sockets

Three types of sockets

- Connection-oriented (TCP)
- Connectionless(UDP)
- MulticastSocketclass–data can be sent to multiple recipients

Remote Procedure Calls

Remote procedure call (RPC) abstracts procedure calls between processes on networked systems

Stubs–client-side proxy for the actual procedure on the server

- The client-side stub locates the server and marshalls the parameters
- The server-side stub receives this message, unpacks the marshalled parameters, and performs the procedure on the server procedure on the server

Pipes

Acts as a conduit allowing two processes to communicate

Ordinary Pipes

- Producer writes to one end (the write-end of the pipe) (pp)
- Consumer reads from the other end (the read-end of the pipe)
- Communication is Unidirectional

Named Pipes

- More powerful than ordinary pipes
- Several processes can use the named pipe for communication
- Communication is bidirectional

Chapter 4: Multithread Programming

Process creation is heavy-weight while thread creation is light-weight

Multicore or multiprocessor systems putting pressure on programmers, challenges include:

- Dividing activities (tasks)
- Load balance
- Data splitting
- Data dependency
- Testing and debugging

Parallelism implies a system can perform more than one task simultaneously

- Data parallelism – distributes subsets of the same data across multiple cores, same operation on each
- Task parallelism – distributing threads across cores, each thread performing unique operation

User threads - management done by user-level threads library

Three primary thread libraries:

- POSIX Pthreads
- Win32 threads
- Java threads

Kernel threads - Supported by the Kernel

Multithreading Models

Many-to-One

Many user-level threads mapped to single kernel thread

One-to-One

Each user-level thread maps to kernel thread, ex Linux, Solaris 9 and later

Many-to-Many

Allows many user level threads to be mapped to many kernel threads, Solaris prior version 9

Two-level

Similar to Many-to-Many, except that it allows a user thread to be bound to kernel thread like in One-to-One, Solaris 8 and earlier

Implicit Threading

Creation and management of threads managed by compilers instead of programmers.

Three methods explored

- Thread Pools
- OpenMP
- Grand Central Dispatch

Thread Cancellation

Two different approaches:

- Asynchronous cancellation terminates the target thread immediately
- Deferred cancellation allows the target thread to periodically check if it should be cancelled

Thread-local storage (TLS) allows each thread to have its own copy of data

Chapter 5: Scheduling

SJF is optimal – gives minimum average waiting time for a given set of processes

The difficulty is knowing the length of the next CPU request

The CPU is allocated to the process with the highest priority

- Preemptive
- Nonpreemptive

CPU scheduling more complex when multiple CPUs are available Homogeneous processors within a multiprocessor

Asymmetric multiprocessing system activities handled by a single processor other processors execute user-code only

Only one processor accesses the system data structures, alleviating the need for data sharing

Symmetric multiprocessing (SMP)

- Each processor is self-scheduling
- All processes in a common ready queue; or each processor has its own private ready queue
- Currently, most common ν Processor affinity – process has affinity for processor on which it is currently running
- Soft affinity: OS attempts to keep the process running on the same processor
- Hard affinity: systems calls used to specify the processor (or subset of processors) affinity

Completely Fair Scheduler (CFS) ν Based on scheduling classes

- Each class has specific priority
- Scheduler picks highest priority task in highest scheduling class
- Rather than quantum based on fixed time allotments, based on proportion of CPU time
- 2 scheduling classes included, others can be added
    1. default 2. real-time

Dispatcher is part of Windows kernel responsible for scheduling

Windows uses priority-based preemptive scheduling

Open-file table: tracks all open files

- File pointer: pointer to last read/write location, per process that has the file open
- File-open count: counter of number of times a file is open – to allow removal of data from open-file table when last process closes it

Shared lock similar to reader lock – several processes can acquire concurrently

- Exclusive lock similar to writer lock, only one process acquires the lock
- Mandatory – operating system prevents other processes to access the locked file
- Advisory – processes are responsible for appropriate lock acquire and release

Entity containing file system is known as a volume

Chapter 6: Synchronization

Each process has a critical section segment of code

Process may be changing common variables, updating table, writing file, etc

When one process is in critical section, no other may be in its critical section

Mutual Exclusion - If process Pi is executing in its critical section, then no other processes can be executing in their critical sections

Progress

- If no process is executing in its critical section and there exist some processes that wish to enter their critical section, then the selection of the processes that will enter the critical section next cannot be postponed indefinitely

Kernel

- Preemptive kernel – allows preemption of process when running in kernel mode
- Non-preemptive kernel – does not allows preemption of process when running in kernel mode

Mutex: mutual exclusion

Semaphore S is an integer variable

Counting semaphore – integer value can range over an unrestricted domain

Binary semaphore – integer value can range only between 0 and 1

Deadlock – two or more processes are waiting indefinitely for an event that can be caused by only one of the waiting processes

Chapter 10

Objectives

- To explain the function of file systems
- To describe the interfaces to file systems
- To discuss file-system design tradeoffs, including access methods, file sharing, file locking, and directory structures
- To explore file-system protection

File

- Information about files are kept in the directory structure,
- File is an abstract data type

Open File Locking

- Shared lock similar to reader lock – several processes can acquire concurrently
- Exclusive lock similar to writer lock, only one process acquires the lock

Access Methods

- Sequential Access the file is processed one record after another
- Direct Access file consists of fixed length logical records or blocks. n: block number, read(n, write(n)

Disk Structure

- Disks or partitions can be **RAID** protected against failure **R**edundant **A**rray of **I**ndependent (or Inexpensive) **D**isks: **RAID**
- Single-level Directory, each file must have a unique name
- Two-Level Directory, can have the same name for different user
- Tree-Structured Directories. Grouping capability. Absolute or relative path name.

File Sharing

- Sharing may be done through a protection scheme
- On distributed systems, files may be shared across a network
- Network File System (NFS) is a common distributed file-sharing method
- If multi-user system
  User IDs identify users, *allowing permissions and protections to be per-user*
  Group IDs allow users to be in groups, *permitting group access rights*
- Owner of a file / directory: the user able to change the attributes.
- Client-Server : Network File System **(**NFS**)** is used in UNIX
  Common Internet File System **(**CIFS**)** is used by Microsoft