Istanbul Kemerburgaz University

MAC OS X PROCESS AND THREAD MANAGEMENT

Presenter: **ABDELHAKIM ALRJAIBI**

STUDENT NUMBER: **163110447**

Supervisor: **Prof .Dr. Hasan H. Balik**

Outline

- *Introduction to Max OS X*
- *Mac OS X Features*
- *Mac OS X Architecture*
- *Process in Mac OS X*
- *Thread Management in Mac OS X*
- *Thread creation costs*

Introduction to Max OS X

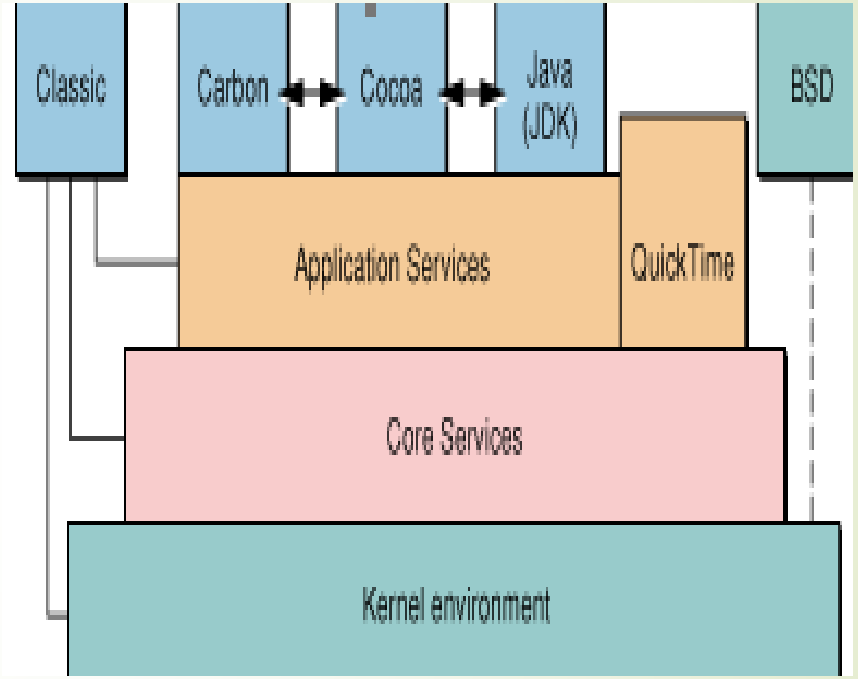
- Mac OS X Panther is a UNIX-based Operating System with the intuitive user interface called Aqua.
- The Mac OS X Panther operation system has met with both technical and commercial success.
- Since the debut of Mac OS X in 2001, its features have continued to improve.

Introduction to Max OS X

- The modern core UNIX-based Operating System brings benefits such as protected memory and preemptive multitasking to Macintosh computing.
- Mac OS X Panther also has a sparkling user interface capable of visual effects such as translucence and drop shadows.

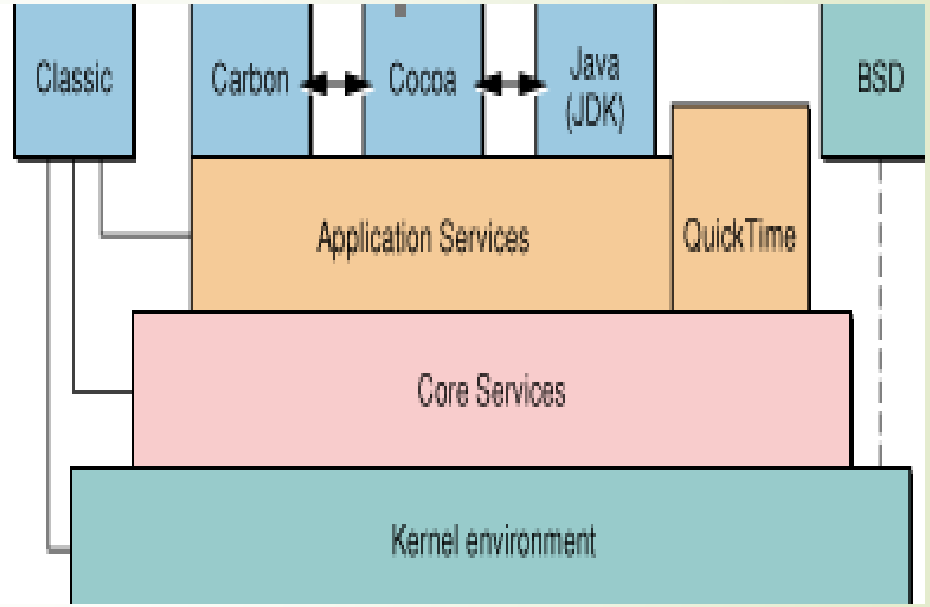
Mac OS X features

- Mac OS X architecture is the layering of system software.
- with one layer having dependencies on, and interfaces with, the layer beneath it.
- Mac OS X has four distinct layers of system software (in order of dependency)



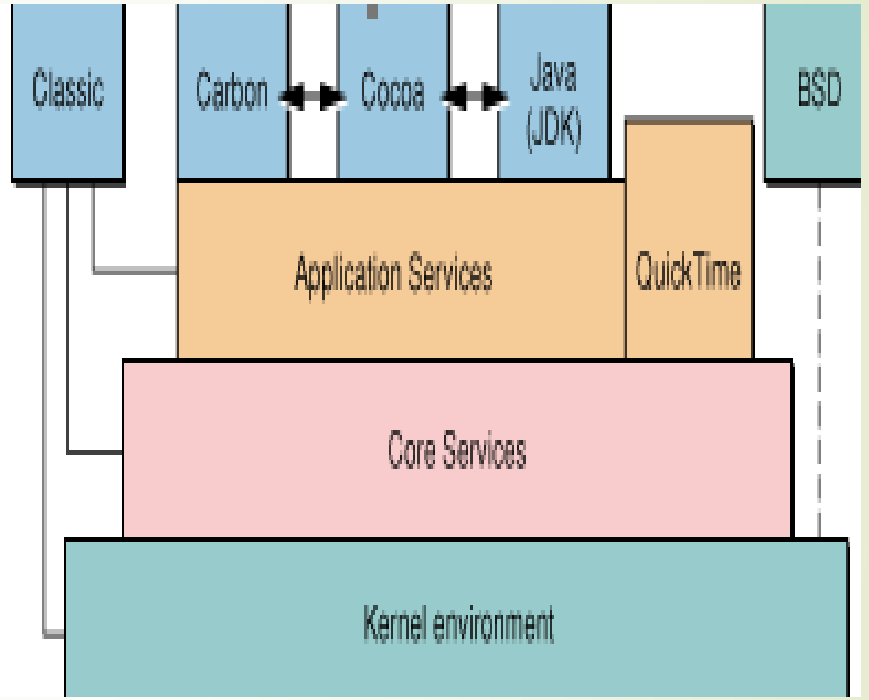
Mac OS X architecture

- **Application Environments**
  - consists of the frameworks, libraries, and services necessary for the runtime execution of programs developed with those API.
  - Mac OS X currently provides five application (or execution) environments: Carbon, Cocoa, Java, Classic, and BSD Commands.



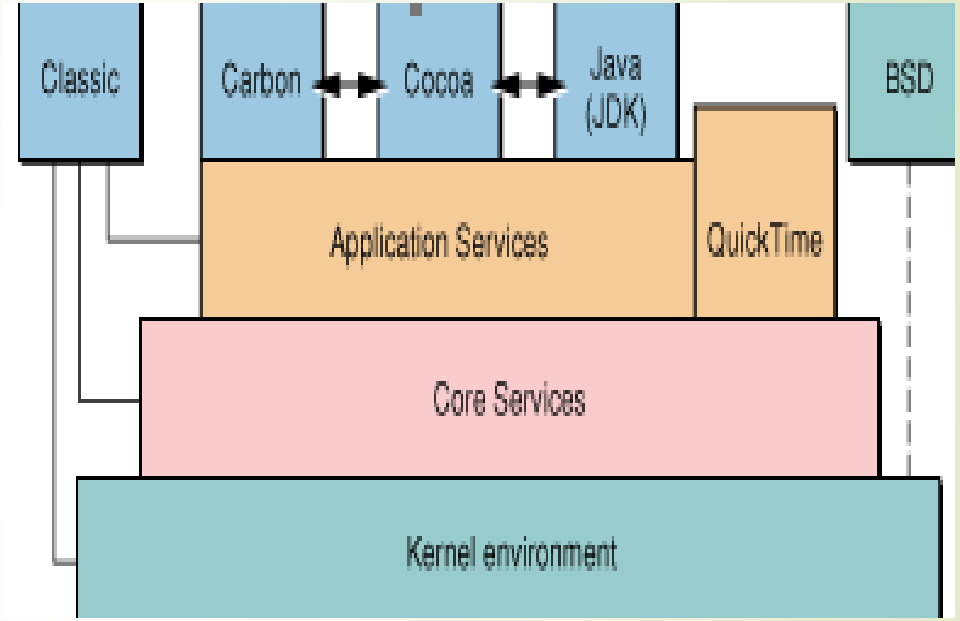
Mac OS X Architecture

- **Application Services**
- Incorporates the system services available to all application environments that have some impact on the graphical user interface.
- It includes Quartz, QuickDraw, and OpenGL as well as essential system managers.



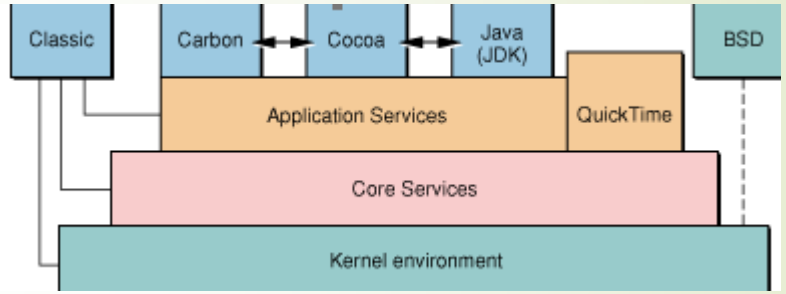
Mac OS X Architecture

- **Core Services**
- Incorporates those system services that have no effect on the graphical user interface.
- It includes Core Foundation, Open Transport, and certain core portions of Carbon.



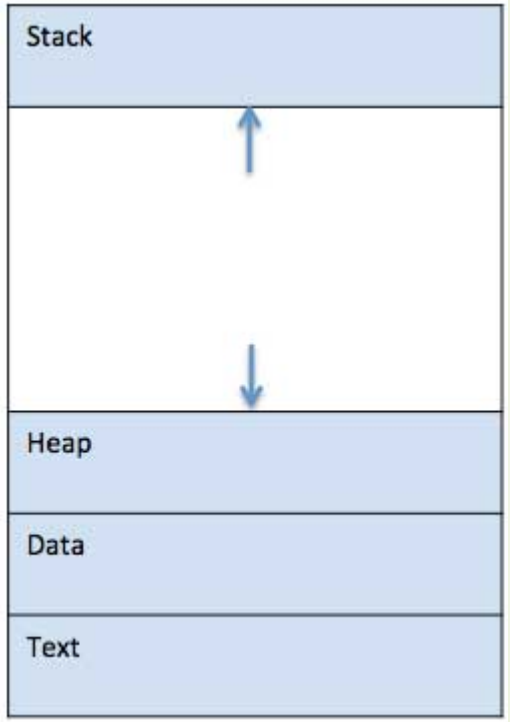
Mac OS X Architecture

- **Kernel Environment**
- provides the foundation layer of Mac OS X.
- The kernel environment offers facilities for developing device drivers (the I/O Kit) and loadable kernel extensions, including Network Kernel Extensions (NKEs).



Process in Mac OS X

- A process is basically a program in execution.
- The execution of a process must progress in a sequential fashion.
- When a program is loaded into the memory and it becomes a process.
- It can be divided into four sections stack, heap, text and data.



Mac OS X

Processor Modes & Privileged Instructions

- **Supervisor Mode** – allows execution of privileged instructions and access to privileged registers

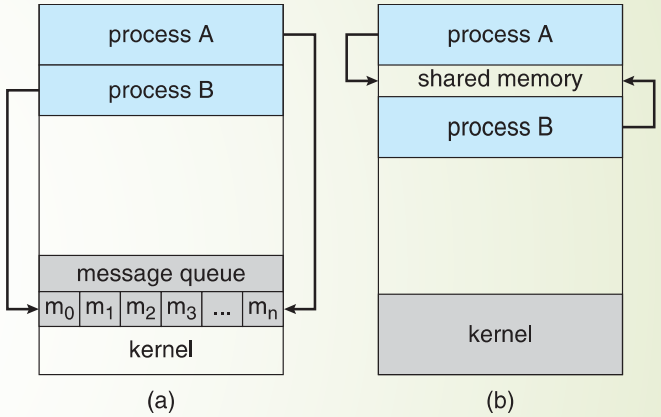
- **User Mode** - The processor mode that forbids execution of privileged instructions and access to privileged registers. Any attempt to do so will result in a privilege violation exception.

Mac OS X
Process States

- SIDL: Process is partially created
- SRUN: Process is runnable
- SSLEEP: Process is awaiting event
- SSTOP: Process is stopped (by signal or parent process)
- SZOMB: Process is partially terminated.(waiting for parent process to collect status)

Interprocess Communication in Mac X

- For fast exchange of information, cooperating processes need some **interposes communication (IPC)** mechanisms.
- Two models of IPC
  - ✓ **Shared memory**
  - ✓ **Message passing**



Grand Central Dispatch (GCD)

- Technology developed by Apple Inc.
- optimize application support for systems with multi-core processors and other symmetric multiprocessing systems.
- It is an implementation of task parallelism based on the thread pool pattern.
- The fundamental idea is to move the management of the thread pool out of the hands of the developer, and closer to the operating system.

Grand Central Dispatch (GCD) Examples:

- Two examples that demonstrate the use of Grand Central Dispatch can be found in John Syracuse's *Ars Technical* Snow Leopard review.

```
- (IBAction)analyzeDocument:(NSButton *)sender {
    NSDictionary *stats = [myDoc analyze];
    [myModel setDict:stats];
    [myStatsView setNeedsDisplay:YES];
}
```

```
- (IBAction)analyzeDocument:(NSButton *)sender {
    dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0), ^{
        NSDictionary *stats = [myDoc analyze];
        dispatch_async(dispatch_get_main_queue(), ^{
            [myModel setDict:stats];
            [myStatsView setNeedsDisplay:YES];
        });
    });
}
```

Thread Management in Mac OS X

- Each process (application) in OS X or iOS is made up of one or more threads.
- Every application starts with a single thread
- Each thread has its own execution stack and is scheduled for runtime separately by the kernel.
- A thread can communicate with other threads and other processes.
- perform I/O operations, and do anything else you might need it to do.

Thread Management in Mac OS X

- All threads in a single application share the same virtual memory space and have the same access rights as the process itself. Because they are inside the same process space.

- **Thread Costs in Mac OS X**

- ✓ Threading has a real cost to your program (and the system) in terms of memory use and performance.

- ✓ Each thread requires the allocation of memory in both the kernel memory space and your program's memory space.

Thread creation costs

Item	Approximate cost	Notes
Kernel data structures	Approximately 1 KB	This memory is used to store the thread data structures and attributes, much of which is allocated as wired memory and therefore cannot be paged to disk.
Stack space	512 KB (secondary threads) 8 MB (OS X main thread) 1 MB (iOS main thread)	The minimum allowed stack size for secondary threads is 16 KB and the stack size must be a multiple of 4 KB. The space for this memory is set aside in your process space at thread creation time, but the actual pages associated with that memory are not created until they are needed.
Creation time	Approximately 90 microseconds	This value reflects the time between the initial call to create the thread and the time at which the thread's entry point routine began executing. The figures were determined by analyzing the mean and median values generated during thread creation on an Intel-based iMac with a 2 GHz Core Duo processor and 1 GB of RAM running OS X v10.5.

User Threads and Kernel Threads in Mac OS X

- **User threads**
  - Support provided at the user-level
  - Management is done by thread library
- **Kernel threads**
  - Supported and managed by OS
  - Virtually all modern general-purpose operating systems support them

End the presentation

Thank you for your lessening