**Slide 1**

**WEEK 2 — OVERVIEW**
- Operating Systems Overview, continued
- A Closer Look at System Calls
  - User’s perspective
  - Implementation of System Calls
- Threads and Processes, Part I

**Slide 2**

**PROCESSES**
- Problems occurring in multiprogramming batch systems, time-sharing systems required a closer look at “jobs”.
- What exactly is a Process?

**Exact definition is differs from to textbook to textbook:**
- A program in execution
- An instance of a program running on a computer
- A unit of execution characterised by
  - a single, sequential thread of execution
  - a current state
  - an associated set of system resources (memory, devices, files)

We define a Process to be an unit of resource ownership.

**Slide 3**

The OS has to
- Load the executable from hard disk to main memory
- Keep track of the states of every process currently executed
- Make sure
  - no process monopolises the CPU
  - no process starves
  - interactive processes are responsive

**Slide 4**

**PROCESS Characterised by:**
1. An executable program (code)
2. Associated data needed by the program (global data, stack)
3. Execution context (or state) of the program, e.g.:
   - contents of data registers
   - program counter, stack pointer
   - state (waiting on an event?)
   - memory allocation
   - status of open files
Slide 5
- Process table keeps track of processes.
- Context information stored in Process Control Block (PCB).
- Process suspended: register contents etc stored in PCB.
- Process resumed: PCB contents loaded into registers.

Slide 6
- Dealing with multiple processes is difficult!
  - Synchronization
    - Ensure a process waiting for an I/O device receives the signal.
    - Signals may be lost or duplicated.
  - Failed mutual exclusion
    - Attempt to use a shared resource at the same time.
  - Non-deterministic program operation
    - Program should only depend on input to it, not relying on common memory areas.
  - Deadlocks

Slide 7
- Memory Management
  - Automatic allocation and management:
    - Memory hierarchy should be transparent to programmer.
    - Programmer should not be able to access physical memory directly.
  - Process isolation:
    - Protect data and memory from other processes.
  - Support for modular programming.
  - Protection and access control.

Slide 8
- Virtual Memory
  - Paging and Dynamic Mapping:
    - Process memory is split into equally sized blocks called pages.
    - Main memory is also split into blocks of the same size, called frames.
    - Pages of a process are dynamically loaded into main memory whenever required.
Slide 9

Main Memory

<table>
<thead>
<tr>
<th>Process Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stack</td>
</tr>
<tr>
<td>Data</td>
</tr>
<tr>
<td>Code</td>
</tr>
</tbody>
</table>

Slide 10

Advantages:

- Reduces start up time of processes
- Reduces fragmentation of main memory
- Possible overlap of execution and loading time of different processes

Virtual Address:

- Virtual address: page number plus offset
- OS maps virtual address to physical address
- From user point of view, every process has its own address space

Disadvantages:

- Extra hardware (MMU) is necessary
- Mapping of virtual address to physical address is complicated

Slide 11

Virtual Address:

- Gives applications the illusion to have all RAM to themselves
- Provides an address space for each process which is much larger than actual RAM
- Provides complete isolation of processes from each other

Advantages:

- Gives applications the illusion to have all RAM to themselves
- Provides an address space for each process which is much larger than actual RAM
- Provides complete isolation of processes from each other

Disadvantages:

- Extra hardware (MMU) is necessary
- Mapping of virtual address to physical address is complicated

Slide 12

Translation of Virtual Addresses:

1. Virtual address goes to Memory Management Unit (MMU)
2. MMU translates virtual address to physical address
3. Causes exception (page fault) if page is not mapped
4. OS (exception handler) fetches page and restarts operation
Virtual Memory Addressing

Processor -> Memory Management Unit -> Main Memory -> Real Address

Main Memory

Secondary Memory

Disk Address

Virtual Memory Addressing

Slide 13

All requests of user level programs for OS services go via system calls:

Application

Procedure Calls

System Calls

System Libraries

User Mode

User Mode

Kernel Mode

OS Kernel

Hardware

Slide 15

Slide 16
FILE SYSTEM

Files and directories (used to group files) provided by the OS to implement a uniform interface to
- disks
- I/O devices

Provide
- human-readable namespace for data
- support for exchange of data between systems

EXAMPLE

Root directory

Mounting File System
- In Unix-like OS’s to provide clean interface to removable I/O devices

Mounting File System

Information Protection and Security
- Access control
  - regulate user access to the system, e.g.: password protected access
- Information flow control
  - regulate flow of data within the system and its delivery to users: e.g. Unix file access permissions
- Certification
  - proving that access and flow control perform according to specifications
SCHEDULING AND RESOURCE MANAGEMENT

- Fairness
  - give fair access to all processes
- Differential responsiveness
  - discriminate between different classes of jobs (interactive, CPU bound)
- Efficiency
  - maximize throughput, minimize response time, and accommodate as many uses as possible

SYSTEM STRUCTURE

Struggle to cope with the increasing complexity of OS
- Software Engineering solutions (modular design, clean & simple interfaces) were not sufficient

Hierarchical Layers and Information Abstraction:
- View the system as a series of levels (lowest may be hardware)
- Each level performs a related subset of functions
- Each level relies on the next lower level to perform more primitive functions
- This decomposes a problem into a number of more manageable subproblems

Monolithic Systems:
- usually evolved from simpler to more complex systems:
  - MS-DOS
  - traditional Unix
- little internal structure

Examples:
- THE system, Dijkstra, 1968
- MULTICS (M.I.T, Bell, GE)
**Microkernel Architecture**

- assigns only a few essential functions to the kernel
  - address space
  - interprocess communication (IPC)
  - basic scheduling
- other services implemented by user-level servers

**Characteristics of Modern Operating Systems**

- Symmetric multiprocessing
  - multiple processors are available
  - these processors share same main memory and I/O facilities
  - All processors can perform the same functions
  - Potential benefits:
    - availability
    - incremental growth
    - performance & scaling

**Microkernel Architecture**

- Mach, developed mid 80’s at CMU
- MacOS X based on Mach, many services moved back to kernel
- Windows NT partially based on Microkernel architecture
  - “modified microkernel architecture”
  - OS environments (DOS, Win16, Win32, OS/2, POSIX) run in user mode
  - Other services (process manager, vm manager) run in kernel mode
- L4 Microkernel Architecture (GMD, IBM)