Sunbeam Institute of Information Technology
Pune and Karad
PreCAT

Module – Operating System Concepts

Trainer - Devendra Dhande

Email – devendra.dhande@sunbeaminfo.com

Operating System – Synchronization

- Multiple processes accessing same resource at the same time, is known as "**race condition**".
- When race condition occurs, resource may get corrupted (unexpected results).
- **Peterson's problem:** If two processes are trying to modify same variable at the same time, it can produce unexpected results.
- Code block to be executed by only one process at a time is referred as **Critical section**. If multiple processes execute the same code concurrently it may produce undesired results.
- To resolve race condition problem, one process can access resource at a time. This can be done using **sync objects/primitives** given by OS.
- OS Synchronization objects are:
  - Semaphore, Mutex, Condition variables

Operating System – Synchronization

- Semaphore is a sync primitive given by OS.
- Internally semaphore is a counter.
- On semaphore two operations are supported:
  - **wait operation: dec op: P operation:**
    - semaphore count is decremented by 1.
    - if cnt < 0, then calling process is blocked.
    - typically wait operation is performed before accessing the resource.
  - **signal operation: inc op: V operation:**
    - semaphore count is incremented by 1.
    - if one or more processes are blocked on the semaphore, then one of the process will be resumed.
    - typically signal operation is performed after releasing the resource.

- **Semaphore types**
  - **Counting Semaphore**
    - Allow "n" number of processes to access resource at a time.
    - Or allow "n" resources to be allocated to the process.
  - **Binary Semaphore**
    - Allows only 1 process to access resource at a time or used as a flag/condition.
- **Mutex**
  - Mutex is used to ensure that only one process can access the resource at a time.
  - Similar to "binary semaphore".
  - Mutex can be unlocked by the same process/thread, which had locked it.

Operating System – Deadlock

- **Deadlock occurs** when four conditions/characteristics hold true at the same time.
  - No preemption: A resource should not be released until task is completed.
  - Mutual exclusion: Resources is not sharable.
  - Hold & Wait: Process holds a resource and wait for another resource.
  - Circular wait: Process P1 holds a resource needed for P2, P2 holds a resource needed for P3 and P3 holds a resource needed for P1.
- **Deadlock Prevention**
  - OS system calls are designed so that at least one deadlock condition does not hold true.
  - In UNIX multiple semaphore operations can be done at the same time.
- **Deadlock Avoidance**
  - Processes declare the required resources in advanced, based on which OS decides whether resource should be given to the process or not.
  - Algorithms used for this are:
    - Resource allocation graph:
    - Banker's algorithm:
- **Deadlock Recovery**
  - Deadlock can be solved by resource preemption or terminating one of the process (involved in deadlock)

Operating System – System Call

- System calls are "functions" exposed by the kernel so that user process can access kernel functionalities (e.g. process, memory, file management, ...).
- Typically library functions/APIs (of any language) are internally calling these syscalls.
- System calls internally use Software interrupt/Trap instruction to change CPU mode.
- System calls are different for different operating systems.
  - UNIX - 64 syscalls
  - Linux - 350+ syscalls
  - Windows - 3000+ syscalls
- UNIX/Linux syscalls
  - Process management : fork(), exec()
  - File manangement : open(), close(), read(), write(), lseek(), chmod(), chown()
  - Memory manangement : brk(), mmap()

Operating System – Memory Management

- Compiler and Linker assign addresses to each variable/instruction assuming that program will execute in VM RAM. These addressed are called as "**virtual address**" or "**logical address**". The set of virtual addresses used by the process is referred "**Virtual address space**".
- However while execution these addresses might be occupied by other processes. Loader relocates all instructions/variables to the address available in RAM. The actual addresses given to the process at runtime are called as "**physical address**" or "**real address**". The set of physical addresses used by the process is referred "**Physical address space**".
- CPU always executes a process in its virtual address space i.e. CPU always request virtual addresses (on address bus).
- These virtual addresses are verified and then converted into corresponding physical addresses by a special hardware unit called as "**Memory Management Unit (MMU)**".

Operating System – Memory Management

- In multi-programming OS, multiple programs are loaded in memory.
- RAM memory should be divided for multiple processes running concurrently.
- Memory Management scheme used by any OS depends on the MMU hardware used in the machine.
- There are three memory management schemes are available (as per MMU hardware).
  - 1. Contiguous Allocation – Fixed/Dynamic partition
  - 2. Segmentation
  - 3. Paging

- Simple MMU holds physical base address and limit (length) of the process. The base & limit of each process is stored in its PCB and then loaded into MMU during context switch.

Thank you!

Devendra Dhande

<devendra.dhande@sunbeaminfo.com>