Embedded Operating Systems

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 syscalls 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.

    ```
    P(s1);
    P(s2);
    ```

    is same as

    ```
    P(s1, s2);
    ```

- P(s1, s2) is atomic operation.

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: OS maintains graph of resources and processes. A cycle in graph indicate circular wait will occur. In this case OS can deny a resource to a process.
- Banker's algorithm: A bank always manage its cash so that they can satisfy all customers.
- Safe state algorithm: OS maintains statistics of number of resources and number processes. Based on stats it decides whether giving resource to a process is safe or not (using a formula):
  - Max num of resources required < Num of resources + Num of processes
    - If condition is true, deadlock will never occur.
    - If condition is false, deadlock may occur.

Starvation vs Deadlock

- Deadlock
  - Deadlock happens when all four conditions hold true at same time i.e. No preemption, Mutual exclusion, Hold and wait, and Circular wait.
  - The processes involved in deadlock are in blocked state. They are in waiting queue (not in ready queue).
  - Deadlock can be prevented by designing systems properly and/or avoided using some algorithms like safe state, resource allocation graph, or banker algorithm.
- Starvation
  - Due to other high priority processes some low priority process is not getting CPU time for the execution.
  - The starved process is in ready state. They are in ready queue.
  - The starved process's priority can be increased dynamically, so that it will be scheduled (later). This technique is called as "aging".

Agenda

- Memory Management
  - Contiguous allocation
  - Segmentation
  - Paging

Contiguous Allocation

Fixed Partition

- RAM is divided into fixed sized partitions.

- This method is easy to implement.
- Number of processes are limited to number of partitions.
- Size of process is limited to size of partition.
- If process is not utilizing entire partition allocated to it, the remaining memory is wasted. This is called as "internal fragmentation".

Dynamic/Variable Partition

- Memory is allocated to each process as per its availability in the RAM. After allocation and deallocation of few processes, RAM will have few used slots and few free slots.
- OS keep track of free slots in form of a table.
- For any new process, OS use one of the following mechanism to allocate the free slot.
    - First Fit: Allocate first free slot which can accommodate the process.
    - Best Fit: Allocate that free slot to the process in which minimum free space will remain.
    - Worst Fit: Allocate that free slot to the process in whic maximum free space will remain.
- Statistically it is proven that First fit is faster algo; while best fit provides better memory utilization.
- Memory info (physical base address and size) of each process is stored in its PCB and will be loaded into MMU registers (base & limit) during context switch.
- CPU request virtual address (address of the process) and is converted into physical address by MMU as shown in diag.
- If invalid virtual address is requested by the CPU, process will be terminated.
- If amount of memory required for a process is available but not contiguous, then it is called as "external fragmentation".
- To resolve this problem, processes in memory can be shifted/moved so that max contiguous free space will be available. This is called as "compaction".

Virtual Memory

- The portion of the hard disk which is used by OS as an extension of RAM, is called as "virtual memory".
- If sufficient RAM is not available to execute a new program or grow existing process, then some of the inactive process is shifted from main memory (RAM), so that new program can execute in RAM (or existing process can grow). It is also called as "swap area" or "swap space".
- Shifting a process from RAM to swap area is called as "swap out" and shifting a process from swap to RAM is called as "swap in".
- In few OS, swap area is created in form of a partition. E.g. UNIX, Linux, ...
- In few OS, swap area is created in form of a file E.g. Windows (pagefile.sys), ...
- Virtual memory advantages:
    - Can execute more number of programs.

  - Can execute bigger sized programs.

Segmentation

- Instead of allocating contiguous memory for the whole process, contiguous memory for each segment can be allocated. This scheme is known as "segmentation".
- Since process does not need contiguous memory for entire process, external fragmentation will be reduced.
- In this scheme, PCB is associated with a segment table which contains base and limit (size) of each segment of the process.
- During context switch these values will be loaded into MMU segment table.
- CPU request virtual address in form of segment address and offset address.
- Based on segment address appripriate base-limit pair from MMU is used to calculate physical address as shown in diag.
- MMU also contains STBR register which contains address of current process's segment table in the RAM.

Demand Segmentation

- If virtual memory concept is used along with segmentation scheme, in case low memory, OS may swap out a segment of inactive process.
- When that process again start executing and ask for same segment (swapped out), the segment will be loaded back in the RAM. This is called as "demand segmentation".
- Each entry of the segment table contains base & limit of a segment. It also contains additional bits like segment permissions, valid bit, dirty bit, etc.
- If segment is present in main memory, its entry in seg table is said to be valid (v=1). If segment is swapped out/on disk, its entry in segment table is said to be invalid (v=0).