Operating systems Interview Questions

1. What is Operating System?
   a. Operating system is an interface between user and computer system
   b. It basically is a computer program that manages computer system
   c. A control program that manages all the activities that has been performed inside a computer system.
2. Main purpose of OS?

. To make sure that the computer system performs well by managing all its computational activities.

a. To provide an environment for the development and execution of programs.

i. User's view

   1. High performance = convenience (Primary)
   2. Maximum resource utilization = efficiency (Secondary)

ii. System's view

   1. Resource allocation
   2. Resource management

3. Types of OS?

. Batched OS

.- too much response time

i.- cpu idle

ii.- too much TAT

iii.- bad resource utilization

a. Multiprogrammed OS

.+ better cpu utilization

i. Process = cpu + i/o

ii. Cpu not idle

iii. Better thruput

iv. Better resource utilization

b. Timesharing OS / multitasking

.Fixed cpu time for every process

i.+ many users

ii.+ less response time

iii.+ switching more frequently

iv.- switching

c. Distributed

- Objective = transparency

e. Real-time Os

- Deadline
- reprecotions

f. Multiprocessing

i. + reliable
ii. + faster
iii. + multiple processes
iv. - cost

-> SMP = Symmetric Multi-Processing
-> Asymmetric Multi-Processing

4. What is spooling?
    - Spooling is simultaneous peripheral operations online
    - Spooling is a process in which data is temporarily gathered to be used and executed by a device, program or the system.
    - It is associated with printing.
5. Responsibilities / work of OS?

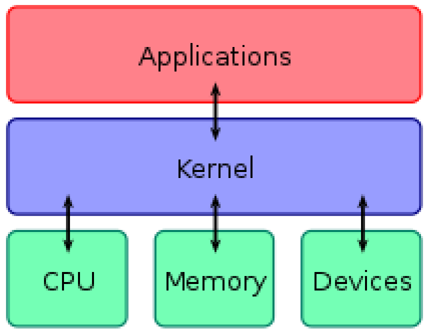
a. Memory management
b. Primary m/m
c. Secondary m/m
d. Resource
e. Device
f. security

6. What is a system call?

    System call is a programming interface to the services provided by OS. e.g. create() , fork(),etc.
    System call is a mediator between user program and services provides ny OS.
    Types:
    - Process control system call = Create process, terminate process,end,allocate and free memory etc
    - File manipulation = Create file, delete file, open file, close file, read, write.
    - Communication = Send, receive messages, transfer status information
    - DEvice manipulation = request device, release device, read, write, reposition, get device attributes, set device attributes etc.
7. What is kernel?
    - Core and essential part of computer OS that provides basic services for all parts of OS.
    - It manages communication between hardware and software components to ensure reliability and usability.
    - Monolithic kernel - a kernel which includes all operating system code is in single executable image.
    - Micro kernel: micro kernel is the kernel which runs minimal performance affecting services for operating system. In micro kernel operating system all other operations are performed by processor.

- Macro Kernel: Macro Kernel is a combination of micro and monolithic kernel.
- Kernel is the part of OS which handles all details of sharing resources and device handling.
- - It can be considered as the core of OS which manages the core features of an OS.
- - Its purpose is to handle the communication between software and hardware
- - Its services are used through system calls.
- - A layer of software called shell wraps around the Kernel.

- Core component of an operating system
- Using process communication, it acts as a bridge between applications and data processing performed at the hardware level.



-
- When an operating system is loaded into memory, the kernel loads first and remains in memory until the operating system is shut down again.

The kernel is responsible for:

- Process management for application execution
- Memory management, allocation and I/O
- Device management through the use of device drivers
- System call control, which is essential for the execution of kernel services

- Monolithic Kernels: All operating system services run along the main kernel thread in a monolithic kernel, which also resides in the same memory area, thereby providing powerful and rich hardware access.
- Microkernels: Define a simple abstraction over hardware that use primitives or system calls to implement minimum OS services such as multitasking, memory management and interprocess communication.

8. Difference between multitasking, multithreading and multiprocessing
9. What is a process?
    - An instance of a pr0gram in execution
    - States of a process
    - Attributes of a process
    - Process = dynamic in nature(in execution undergoes all states)
    - Program = static in nature

10. PCB?
    A DS containing all the information about a process.
    Also called as context.
11. Difference between context switching and mode switching.
    Context switching of one process with another. Its like loading and unloading.
    Mode switching = kernel vs user mode.
12. Process Scheduling
    Set of instructions and mechanism that gives the order in which the process is to be executed.
    Scheduler = an OS module
13. Different shedulars?

a. Long term

i. Job scheduler

ii. Select process from job queue and uploads it in main memory for execution.

b. Medium term

. Swapping

i. Suspension - swaps in and out a process

c. Short term

. Context switching

i. Allocates CPU to those who are ready to execute

14. What is context switching?
    - Pull out one context and replace it with another.
    - Replace completely the status of cpu with the status of new process
    - It is an overhead.
15. Dispatcher vs scheduler.
    - Scheduler = main task is to select a job to be submitted to the system and decide which process to run
    - Dispatcher = is a special program which takes the selected process to the desired state / queue.
16. Different time parameters.

- Arrival time
- Response time
- Completion time
- Burst time = CPU time
- Waiting time
- Turn around time = CT-AT or WT + BT4
- Throughput
  - = amount of work completed in unit time
  - = number of processes that complete their execution per unit time
  - = number of processes / total time

17. Different schedulings?

a. FCFS
   - Convoy effect
   - Solution = Shortest job first
   - Using queue

b. SJF
   - Starvation

c. SJRF
   - Preemptive
   - Burst time

d. Priority scheduling
   - Starvation
   - aging

e. Round Robin

19. Explain convoy effect
    Convoy effect is associated with FCFS algorithm, in which the whole OS slows down due to few slow processes.

19. What is starvation?
    Starvation occurs when a process waits infinite period of time to get the resource it requires.(usually in priority queue).

20. Explain aging in operating system.
    Aging is a scheduling technique used to avoid starvation. Priority of jobs gradually increases with time.

21. What is a deadlock?
    Starvation = long waiting
    Deadlock = infinite waiting
    When a waiting process never changes its state because it never acquires the resource it needs.

22. Necessary conditions for a deadlock to occur.

- Mutual exclusion = no two process can share a resource (i.e. no shareable resource)
- Hold and wait
- Non - preemptive
- Circular wait = each process is waiting for the resource held by another process

23. Methods for handling a deadlock.

a. Deadlock prevention

i. Mutual exclusion allowed = use shared resources e.g. read only files instead of non sharable resources e.g. printer

ii. Preemption allowed = if requests cannot be met, preempt all the resources it is holding

iii. No hold and wait

1. Allocate all resources before it begins execution
2. Request only when it has no resources

iv. No circular wait

Give increasing order to resources and devices can only request a resource in increasing order

b. Deadlock avoidance

- Banker's algorithm
  - Safe algorithm
  - Resource request algorithm

- requires extra information

- safe state = no deadlock state / unsafe state

c. Deadlock ignorance

i. Ostrich approach

d. Deadlock detection and recovery

.Detection

1. Single instance of resources = if cycle in system resource allocation graph
2. Multiple instance resource type = banker's algo

i. Recovery

1. Inform the operator / user
2. Let the system recovery(automatically)

a. Options for breaking a deadlock cycle

i. Abort one or more process to break circular wait (process termination)

ii. Resource termination = preempt some resources from one or more deadlock processes.

25. What is a thread in OS?

- A thread is an execution unit which has its own program counter, stack and set of registers but shared code section, data sections and OS resources.
- Popular way to improve application through parallelism .
- Dissimilarity
    - thread within the same process run in a shared memory space, while process run in separate memory
    - Threads are not independent like processes
    - as a result threads share with other threads their code section, data section, and OS resources
- Similarity
    - But, like process, a thread has its own program counter (PC), register set, and stack space.
    - Have almost same states as process
    - Share CPU
    - Has its own PC and stack
    - Can create child thread

25. Types of threads?

a. User level

i. Faster to create
ii. Created by user
iii. Easy to implement
iv. Context switching takes less time

b. Kernel level

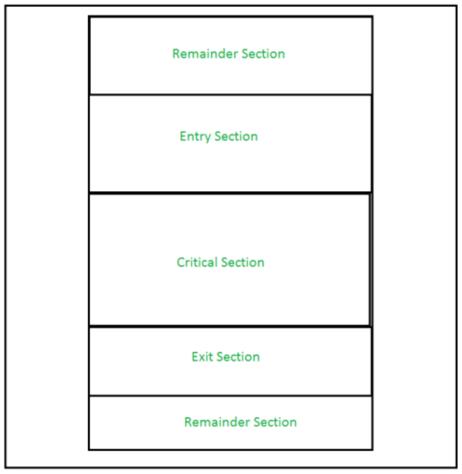
. Slower to create
i. Created by kernel
ii. Complex to implement
iii. More context switching time

26. What is race condition?
    - Race condition is an undesirable situation where the final result depends upon the order of occurrence of operations/processes .
    - Solution = synchronization
    - Reason = shared data
27. What is critical section?

    It is the part of the program where shared resources are accessed by the process.
28. What is process synchronization?

    A technique which is used to coordinate the process that use shared data.
    - Entry section
    - Critical section
    - Exit section

- Remainder section



29. Requirements of synchronization mechanisms?
    - Mutual exclusion (must)
    - Progress (must)
    - Bounded waiting
    - Portability

30. Popular synchronization mechanisms

a. Lock variable
b. TSL (test set variable)
c. Disabled interrupt
d.

31. Lock variable?

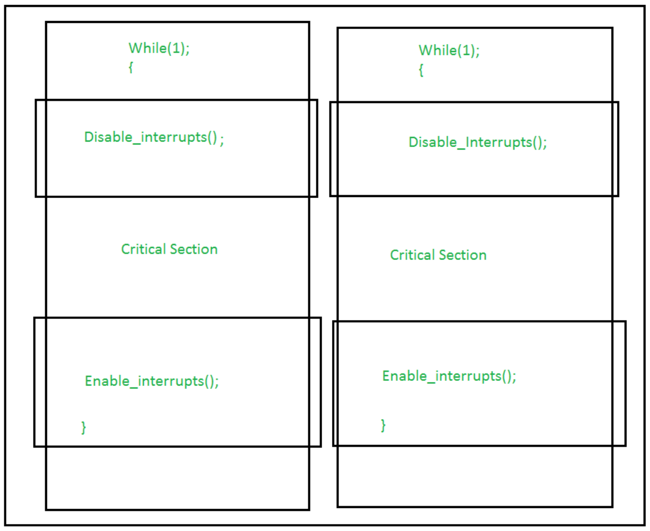
. Software mechanism implemented in user mode
a. >2 processes
b. busy waiting
c. Mutual exclusion (no)

    1. Load Lock, R0 ; (Store the value of Lock in Register R0.)
    2. CMP R0, #0 ; (Compare the value of register R0 with 0.)
    3. JNZ Step 1 ; (Jump to step 1 if value of R0 is not 0.)
    4. Store #1, Lock ; (Set new value of Lock as 1.)
    Enter critical section
    5. Store #0, Lock ; (Set the value of lock as 0 again.)

        - Solution = TSL

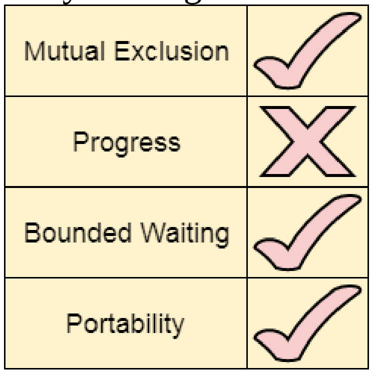
32. Test and set?
    - Make first 3 steps atomic (put lock into your pocket and put busy outside. See later.)
    - Hardware solution to critical section problem
    - Mutual exclusion (yes)

- Progress (yes)
- Bounded wait (no)
- Portability (no)
- Problem = priority inversion (P1 > P2)
    - P1 has CPU
    - P2 has resource
- This is actually a deadlock called spin lock

33. Disabling interrupt?
    - Mutual exclusion (yes)
    - Progress (yes)
    - Bounded wait (no)
    - Portability (no)
    - Problem = cannot give user authority to enable/disable interrupt
    - 

```
While(1);
{
Disable_interrupts() ;

Critical Section

Enable_interrupts();
}
```

```
While(1);
{
Disable_Interrupts();

Critical Section

Enable_interrupts();
}
```

34. Turn variable or strict alteration approach?
    - Software approach implemented at user mode
    - Busy waiting
    - 

Mutual Exclusion	✓
Progress	✗
Bounded Waiting	✓
Portability	✓

    - 2 process solution

- No progress = even if the process is not interested it can keep the CS
- Solution = interested variable = instead of using TURN, use
    - INTERESTED[0] = T
    - INTERESTED[1] = T

35. Interested variable?
    - Mutual exclusion (yes)
    - Progress (yes)
    - Bounded wait (no)
        - deadlock = both show interest and both get interrupted
    - portability(yes)

36. Peterson's solution?
    - Software mechanism at user level
    - Has everything
    - Busy waiting
    - 2 process solution
    - Uses (TURN and FLAG) variables
        - boolean flag[i] :Initialized to FALSE, initially no one is interested in entering the critical section
        - int turn : The process whose turn is to enter the critical section.
    - No deadlock

```
do {

     flag[i] = TRUE ;
     turn = j ;
     while (flag[j]  &&  turn == j) ;

          critial section

     flag[i] = FALSE ;

          remainder section

  }  while (TRUE) ;
```

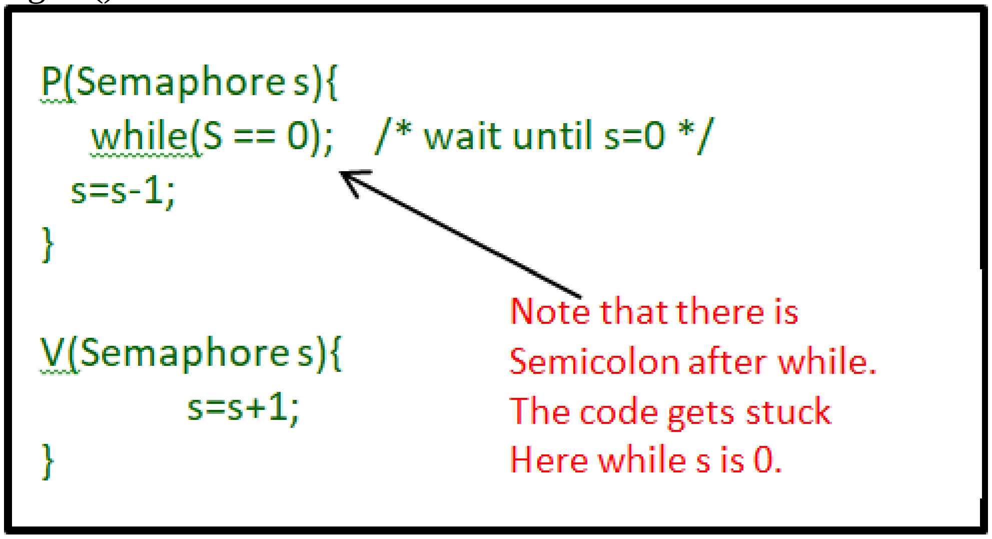
-

37. Problems with TSL and peterson's solution
    Both are best but have two problems:

a. Wastage of time while waiting
b. Priority inversion
    SOLUTION:-
    Sleep and wake

38. What are mutexes?
    -Mutex is the short form for ‘ Mutual Exclusion object’ .

- A mutex allows multiple threads for sharing the same resource.

39. What are semaphores?
    - Simply a non-negative shared variable between threads.
    - 2 types:
        - Binary semaphore
            - Takes only 2 values = 0,1
            - Used to implement mutual exclusion
        - Counting semaphore
            - Non-negative integer for multiple processes
    - Can operate only 2 operators
        - wait()
        - signal()
        - 

```
P(Semaphore s){
    while(S == 0);   /* wait until s=0 */
   s=s-1;
}

V(Semaphore s){
          s=s+1;
}
```

Note that there is Semicolon after while. The code gets stuck Here while s is 0.

40. Difference between mutex and semaphore
41. Producer consumer problem

```
wait(S){
while(S<=0);   // busy waiting
S--;
}

signal(S){
S++;
}
```

- Problem Statement – We have a buffer of fixed size. A producer can produce an item and can place in the buffer. A consumer can pick items and can consume them. We need to ensure that when a producer is placing an item in the buffer, then at the same time consumer should not consume any item. In this problem, buffer is the critical section.
- To solve this problem, we need two counting semaphores – Full and Empty. “Full” keeps track of number of items in the buffer at any given time and “Empty” keeps track of number of unoccupied slots.
- Initialization of semaphores –

  mutex = 1
  Full = 0 // Initially, all slots are empty. Thus full slots are 0
  Empty = n // All slots are empty initially

- Solution for Producer –

```
do{
//produce an item
wait(empty);
wait(mutex);

//place in buffer
signal(mutex);
signal(full);
}while(true)
```

- Solution for Consumer –

```
do{

wait(full);

wait(mutex);

// remove item from buffer

signal(mutex);

signal(empty);

// consumes item

}while(true)
```

42. Dining philosopher problem

```
process P[i]
 while true do
   {  THINK;
      PICKUP(CHOPSTICK[i], CHOPSTICK[i+1 mod 5]);
      EAT;
      PUTDOWN(CHOPSTICK[i], CHOPSTICK[i+1 mod 5])
   }
```

43. What is virtual memory?
    - Virtual memory is a storage allocation scheme in which secondary memory can be addressed as though it was a part of main memory.
44. What is thrashing?
45. What is fragmentation? Different types of fragmentation
46. Paging and its basic function.
47. What is cache memory?
48. Cache coherency?
49. Logical and physical address?
50. Compiler vs interpreter?
51. What is daemon?
52. Difference between secondary and primary storage?
53. What is DRAM?
54.
55. f