Quiz 1---
-None of the above are not a component of a computer system.
-In SMP type **each processor performs all tasks within the operating system.**
-Spatial locality is the tendency of execution to involve a number of memory locations that are clustered
-Which of the following storage medium is faster in speed? **Register**
-In which mode is a system call executed? **Kernel mode**
-When an external device detects an event that requires the attention of the operating system, **it sends an interrupt to the processor**
-Which of the following is one type of user interface? **All of the above**
-Which of the following runs inside the kernel? **OS services**
-Which of the following is true regarding system calls? **It can replace the system programs as using it….**
-Which of the following is used to contain temporary data in a process's memory layout? **Stack**
-Which of the following is true about process? **Context of a process is represented in the PCB**
-Which of the following system calls is used to let the parent rpocess wait for termination of child process? **wait()**
-Which of the following system calls is used to let the parent process creates a child process? fork()
-**The buffer is empty when in==out; the buffer is full when ((in+1) % BUFFER SIZE) == out**
-Which of the following is true about threads? **Each thread has a program counter**
-A message-passing model is **easier to implement than a shared memory model for inter-computer communication**
-A microkernel is a kernel **that is compressed before laoding in order to reduce..**
-In a **zero capacity** temporary qeue, the sender must always block until the recipient receives the message
-A process that has terminated, but whose parent has not yet called wait(), is known as a **zombie** process
-A **thread pool** uses an existing hread - rather than creating a new one - to complete a task.
-Is indirect communication between process P and Q **there is a mailbox to help communication between P and Q**
-In UNIX, the return value for the fork system call is **zero for the child** and a **nonzero integer for the parent process**
-F A system call is triggered by hardware
-T Many operating system merge I/O devices and files into a combined file because of the similarity of system calls for each
-**F** The single benefit of a thread pool is to contorl the number of threads
-**T** Application programmers typically use an API rather than directory invoking system calls
-T It is possible to create a thread library without any kernel-level support
-T It is possible to have concurrency without parall

Quiz 2---
Mutual exclusion can be done on?
Hardware/software/OS à **all of them**
A semaphore is a shared integer variable
That cannot drop below zero
A deadlocked state occurs whenever
Every proc in a set is waiting for an event that can only be caused by another proc in the set
A __ type presents a set of programmer-defined operations that are provided mutual exclusion within it **Monitor**
A(n) __ refers to where a proc is accessing/ updating shared data **Critical section**
Whats the purpose of the mutex semaphore in the implementation of the bounded buffer prob
It ensures mutual exclusion
When a semaphore is used to implement mutex lock, what is its value initialized to be **1**
The following program consists of 3 concurrent procs and 3binary semaphores….how many times will P0 print **At least twice**
Which of the following cannot be scheduled by the kernel **User level thread**
When using semaphores………. What would be a possible outcome of this?
Several process could be active in the critical sections at the same time.
Which of the following condition is required for deadlock to be possible
All of the above à **mutual exclusion, proc had to be allocated, no resource ca be forcibly removed**
The circular wait condition can be prevented by
Define a linear ordering of resource types
The proc that are residing in main mem and are ready and waiting to exe
Ready queue
Which of the following scheduling algorithms must be nonpreemptive
FCFS
A cycle in a resource allocation graph is
A necessary and sufficient condition for a deadlock in the case that each resource has exactly one instance
The first reader-writers problem
Req that no reader will be kept waiting unless a writer has already obtained permission to use the shared database
A significant problem with priority scheduling algorithm is ___
Starvation
In multilevel feedback scheduling algorithm
Process can move to a different classified ready queue
TRUE/FALSE
__**F**__: protocols to prevent hold-and-wait conditions typically also prevent starvation
__F__: /a system in an unsafe state will deadlock
__T__: the monitor construct ensures that only one proc can be active RR
__T__: scheduling degenerates to FCFS if the time quantum is too long
__T__: in the Linux CFS scheduler the task with the smallest vruntime is considered to have the highest priority

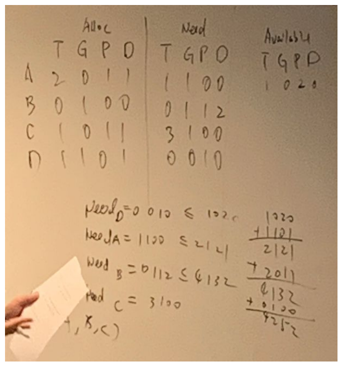
Quiz 3-------------------------------
-Increasing the RAM of a computer typically improves performances because f**ewer page faults occur**
-On media that uses constant linear velocity, **the density of bits per track is uniform**
With segmentation, a logical address consists of **segment number and offset**
-Which of the following data structures is appropriate for placing into its own segment? **Heap, kernel code and data, user code and data**
-Assume the value of the base and limit registers are 1200 and 350 respectively. Which of the following addresses Is legal? **1200**
-Which of the following statements are true with respect to

7. What is the maximum number of processes/threads that can be active in a monitor at the same time? – One.
8. What is the difference between and I/O-bound process and a CPU-bound process? -
9. What are the limitations of Amdahl's Law? – BOTH: It assumes that the parallelizable code can be divided equally among
the CPUs. – It does not account for communication and synchronization overhead.
10. Which of the following is true about threads? –
11. Which of the following is true about process states? –
12. Which of the following is not true about an Orphan process? – Its parent has not called wait(), but the parent has not
terminated yet.
13. How does protecting a critical section (CS) with a semaphore ensure mutual exclusion? - When a process is in its CS, any
process that tries to access a CS that is protected by the same semaphore is placed in the waiting state.
14. Which of the following is not true about shared memory and message passing? - Shared memory is slower than message passing on all multiprocessor systems.
15. Which of the following is not true about message naming? - In direct communication, multiple links may exist between a pair of processes.
16. Which of the following is not true about Remote Procedure Calls (RPCs)?- The user program communicates directly with the matchmaker without kernel intervention.
17. Which of the following is not true about concurrency and parallelization? - With concurrency, it is impossible achieve any speedup relative to sequential execution.
18. In a given program, only one fifth of the code is parallelizable. What's the maximum speedup factor that can be achieved on
a quad-core system under ideal conditions? - 20/17.
19. which of the following is true about shared memory and message passing? - Shared memory may be slower than message
passing on multiprocessor systems.

```
void Producer(int bufSize, int itemCnt, int randSeed){
21. int i, in = 0, out = 0, val;
22.
23. 1 for (i=0; i&lt;itemCnt; i++) {
24. 2 while((GetIn()+1)%bufSize == GetOut());
25. 3 val = GetRand(0, 1000);
26. 4 WriteAtBufIndex(in, val);
27. 5 X1 in = (in + 1)%bufSize;
28. 6 SetIn(in);
30.31. void Consumer(){
32. //Code to open shared memory block and map it to
gShmPtr
33. 1 int bufSize = GetBufSize();
34. 2 int itemCnt = GetItemCnt();
35. 3 int in = GetIn();
36. 4 int out = GetOut();
37. 5 for(i=0; i&lt;itemCnt; i++){
38. 6 X2 while(GetIn() == GetOut());
39. 7 val = ReadAtBufIndex(out);
40. 8 X3 out = (out + 1)%bufSize;
41. 9 SetOut(out);
```

-Which line in the above code does each of the following? Make sure you specify whether that line is in the Producer or in the Consumer
Waiting if the buffer is empty- Consumer wait when buffer is empty.
Updating the value of in in the shared memory buffer- line 6 in producer update.
The length of the time quantum (slice) given by the operating system scheduler to each process must be selected carefully. What's the negative consequence of making this time slice too long and what's the negative consequence of making it too short?
Problem with too long time quantum- the system is less responses.
Problem with too short time quantum- a lot of time will be waist in context switching.
1. Give two clear advantages of using multiple threads compared to using multiple processes.
1. Faster, because there is less system overhead during creation and context switching
2. Resource sharing. Since threads share resources, that better utilizes system resources.
3. Faster communication using global variables
2. Name three different reasons that may cause a process to transition from the running state into the waiting state.
1. Requesting I/O
2. Waiting for a child to terminate
3. Waiting for a semaphore
3. Amdahl's Law gives the potential performance gain that may be achieved by using multiple processors under ideal conditions. What are the ideal conditions that must be satisfied in order to actually achieve the performance gain computed by Amdahl's Law?
1. There is zero overhead. So, we are ignoring thread (process) creation and context switching overhead as well as communication and synchronization overhead.
2. We are assuming perfect load balancing (we can perfectly divide the work among multiple threads or processes).

Midterm 2--
-A race condition is when **the correctness of the code depends upon the timing of the execution**
-The producer-consumer problem **is related to the allocation of resources to process states**
-Bounded waiting implies ... number of times a process is allowed to enter its critical section **after a process has made a request …**
-A system has 3 processes sharing 4 resources. If each process needs a maximum of 2 units, then deadlock **can never occur**.
-Which of the following is NOT true about segment based memory management. **Segment length must be a multiple of the page size**
-A multi-level page table is preferred in comparison to a single-level … **It helps to reduce the size of page**
-Which of the following is not true of virtual memory? **It requires the use of disk or other secondary storage**
-External fragmentation will not occur when **no matter which algorithm is used, it will always occur**.
-The purpose of a TLB is **to cache page translation information**
-Which page replacement algorithm suffers from Belady's anomaly? **FIFO**
-One of the disadvantages of the priority scheduling algorithm is that **it can lead to some low priority…**
-Which of the following is false with regards to Linux CFS scheduler? **There is a single, system-wide value or vruntime**
-Which of the following is not usually stored in a two-level page

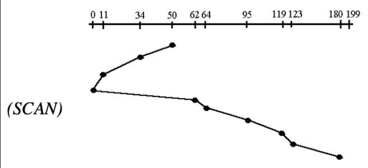


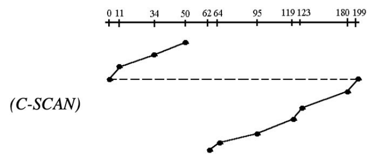
->Still needs=Max-Current
->Compare need<= available and if True, add current alloc + available

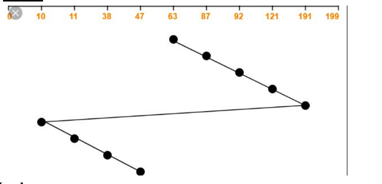
(5 points) Consider a memory system with a cache access time of 10ns and a memory access time of 110ns – assume the memory access time includes th to check the cache. If the effective access time is 10% greater than the cach access time, what is the hit ratio *H*? **(fractional answers are OK)**

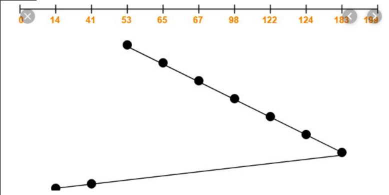
$$Effective\ Access\ Time = H*T_{cache} + (1\text{-}H) * T_{memory}$$
$$1.1 * T_{cache} = H*T_{cache} + (1\text{-}H) * T_{memory}$$
$$1.1 \times 10 = H*10 + (1\text{-}H)110$$
$$11 = H*10 + 110 - 110*H$$
$$-99 = -100*H$$
$$H = 99/100$$

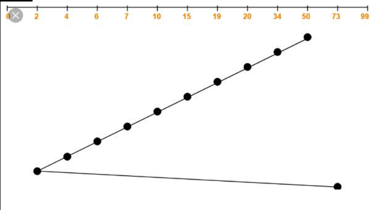
```
1. Given five memory partitions of 100Kb, 500Kb, 200Kb, 300Kb, 600Kb (in c
the first-fit, best-fit, and worst-fit algorithms place
   processes of 212 Kb, 417 Kb, 112 Kb, and 426 Kb (in order)? Which algor
most efficient use of memory?

First-fit:
212K is put in 500K partition
417K is put in 600K partition
112K is put in 288K partition (1
426K must wait

Best-fit:
212K is put in 300K partition
417K is put in 500K partition
112K is put in 200K partition
426K is put in 600K partition

Worst-fit:
212K is put in 600K partition
417K is put in 500K partition
112K is put in 388K partition
426K must wait

In this example, best-fit turns
```





CLook


Look


SSTF


-FCFS (First Come, First Served)
perform operations in order requested no reordering of work queue
no starvation: every request is serviced

Final---
-In the process state transition diagram, the transition from the READY state to the RUNNING state indicates that A. **a process was preempted by another process**
-A critical section is **the part of a program in which shared data is accessed.**
-The Banker's Algorithm is an example of a technique for **deadlock avoidance**
-which of the following is NOT true of virtual memory, **it allows a more efficient use of memory**
-which is NOT usually stored in a two-level page table? **Virtual page number**
-The copy-on-write mechanism provides **A clever way to share virtual memory page**
-System calls **Protect kernel data structures from user code**
-Buffering is useful because **It allows devices and the CPU to operate asynchronously**
-Which of following file sys types requires fewest 4KB file block to store 16KB file **Contiguous allocation**
-A piece of code which lies dormant until triggered by some event causing system damage is called **Logic Bomb**
-True The CPU utilization is always increased as deg. Of multiprogram is increased
-False 32bit logical addr with 8bit page size will have 1,000,000 entries in a conventional page table
-In the process state transition diagram, the transition from the READY state to the RUNNING state indicates that A. **a process was preempted by another process**
-A critical section is **the part of a program in which shared data is accessed.**

Final----------------------
-**How do paging and segmentation affect external fragmentation?** Paging eliminates external fragmentation and segmentation only reduces external fragmentation.
-**Which of the following is true about paging and segmentation?** In both segmentation and paging the address space of a process does not have to be contiguous.
-**Which of the following is the most time consuming step in handling a page fault?** Transferring data between disk and memory.
-**What is the difference between the C-Scan and the C-Look disk scheduling algorithms?** C-Scan always reaches the end of the disk, while C-Look does not reach a disk end unless there is a request at that end.
-**Which disk scheduling algorithm is the most likely algorithm to cause starvation**? -SSTF, because if requests to locations that are adjacent to the current location keep arriving, an earlier request to a far location may wait indefinitely.
-**What's the advantage of a File Allocation Table (FAT) over basic linked allocation?** Traversing the links in FAT is faster, because they are more localized. AND - The links in FAT are more cacheable.
-**How does indexed disk allocation compare with contiguous disk allocation?** Contiguous gives faster sequential access but indexed causes less external fragmentation.
-**How does indexed disk allocation compare with linked disk allocation?** - Indexed gives faster random access but linked is less complex from implementation point of view.
-**Which of the following is (are) true about internal fragmentation in linked disk allocation?** - Increasing the block size increases internal fragmentation. AND - Clustering increases internal fragmentation.
-**Which of the following is true about multilevel-feedback-queue scheduling?** If a process uses its entire time quantum, it is moved to a lower priority level. AND - If a process spends a lot of time in a low-priority level without getting the CPU, it is moved to a higher priority level.
-**Which of the following is not true about threads?** Parallel programming using threads can utilize more cores than parallel prog. using processes.
-**How does the page-fault frequency (PFF) technique prevent thrashing?** It takes frames from a process if its page fault rate falls below a certain lower bound. AND - It gives more frames to a process if its page fault rate exceeds a certain upper bound.
-**Which of the following is true about memory frame allocation?** With global frame allocation, the execution time of a process depends on other processes. AND - Global frame allocation does a better job at utilizing memory than local frame allocation.
-**Which of the following is not true about open-file tables?** A pointer to the last read location is stored in the system-wide table.
-**Which of the following is not true about virtual memory (VM) and physical memory (PM)?** VM does not need any hardware support.
-**What's the relationship between page size and fragmentation?** A larger page size increases internal fragmentation.
-**A logical address space of a process has 512 pages with an 8-KB page size.** How many bits are required in the logical address? - 22
-**Which of the following is not true about paging and segmentation?** FALSE: A single operand can never access more than one page. TRUE: Paging divides memory into equal blocks. Segmentation may divide memory into unequal blocks. A single instruction may access multiple pages. In both segmentation and paging the address space of a process does not have to be contiguous. A process can run even if only a small subset of its pages are leaded in physical memory.
-**Which of the following is true about a page fault?** - Handling a page fault always involves transferring data from disk to memory. AND - Page faults occur infrequently due to locality of reference.
-**Which of the following is true about page replacement policies?** - Optimal page replacement is theoretically the best but is impossible to implement in practice. AND - With FIFO page replacement, increasing the number of frames may increase page faults.
-**What is the difference between the Scan and the C-Scan disk scheduling algorithms?** Scan services requests when it is moving in both directions, while C-Scan services requests only when it is moving in one direction.
-**Which of the following is true about starvation in disk scheduling?** FCFS can never cause starvation. AND - SSTF is the only algorithm (among the ones we studied) that may cause starvation. AND - Scan doesn't cause starvation, because the waiting time can be long but will always be bounded.
-**Which of the following is true about the difference between the Round Robin (RR) and the Shortest-Job-First (SJF) scheduling algorithms?** SJF minimizes the average waiting time, but RR does not.
-**What's the difference between Rate-Monotonic Scheduling (RM) and Earliest-Deadline-First (EDF) Scheduling?** RM uses fixed priorities while EDF dynamically adjusts priorities.
-**How does contiguous disk allocation compare with linked disk allocation?** Contiguous provides faster sequential and

hashed page table? **A common approach for handling addresses larger than 32 bits**
-Which of the following is a benefit of allowing a program that is only partially in memory to execute? **Programs can be written to user more memory than is available in physical memory;** CPU utilization and throughout in increased; Less I/O is needed to load or swap each user program into memory
-Suppose we have the following page access 1 2 3 4 2 3 4 1 2 1 1 3 4 and that there are three frames within our system. Using the LRU replacement algorithm, what is the number of page faults? **8**
-What size segment will be allocated for a 39KB request on a system using the Buddy system..? **64 KB**
-Which of the following is the simplest method for implementing a directory? **Linear List**
-Which on the following statements is false? **Virtual memory reduces the context switching overhead**
-DMA controllers **can steal memory access cycles from the main CPU**
-Consider a disk queue holding requests to the following cylinders in the listed order: 116, 22, 3, 11, 75, 185, 100, 87. Using the SCAN at head 88? **100 – 116- 185 – 87 – 75 – 22 -11 – 3**
-Which of the following disk head scheduling algorithms does not take into account the current position of the disk head? **FCFS**
-Which of the following is not considered a classification of user in connection with each file? **Current user**
-Contiguous allocation
-A disk with free blocks 0, 1, 5, 9, 15 would be represented with what bit map? **1100010001000001**
-Which algorithm is considered reasonable for managing a buffer cache? **Least-frequently-user LRU**
-True In general, LOOK disk head scheduling will involve less movement of the disk heads than SCAN disk head scheduling.
-False A relative path name begins at the root
-False Inverted page table require each process to have its own page table.
-False Linked allocation suffers from external fragmentation
-False Indexed allocation may require substantial overhead for its index

1) Indexed Files - PROS + not much wasted + Both Sequential and random accesses are easy. CONS - wasted space in file descriptors - sets a maximum file size - lots of seeks because data is not contiguous
2) Linked Files - PROS + fragmentation + file size changes + efficiently supports which type of access. CONS - Does not support which type of access - Number of seeks.
3) Contiguous - PROS + simple(only starting location and length are required). + Access time (number of seeks CONS - Changing file - Fragmentation (External fragmentation, need for compaction)
Multilevel Indexed Files. PROS + Supports incremental file growth + small files. CONS -Indirect access is inefficient for random access to very large files. -Lots of seeks because data is not contiguousU

True about shared memory &amp; message passing?- shared memory may be slower than message passing on multi-processor system.
Not true about message passing?- in indirect communication, multiple links may exist between pair of process.
Spinlock – are good for multiprocess b/c the waiting can be on one &amp;rest can be on other.
Mutex lock- waiting process is replaced in queue. State: wait lock() checks on value of lock, if available enters oth erunise waits unlock().
Spinlock- waiting process is busy waiting (still gets cpu cycles). State: running/ready this makes more sense while wait time its smaller.
Dual model - when process does something wrong the hardware will take control over the operating system.
Monitor - are by themselves not enough powerful to solve synchronization problems condition x,y.
Data parallelism- distribute subset of same data across multiple cores, same operation on each.
Task parallelism- distributing thread across cores each thread performing unique operations.
Starvation- process in not guaranteed to be remove from semaphore wait list in which it suspended. Solution: bounded waiting. I
turn per process not turns. It eventually get turn in starvation where as in deadlock no change.
Priority inversion- lower priority process holds a lock needed by higher priority process. Solved by priority: inheritance protocol.
Diners problem: (i+1)%5 or (i-1)%5 1. Allows only 4 philosophers to be sitting simultaneously on the table. 2. Allow a philosopher to pick up only if both are available (use critical section) 3. Use an asymmetric sol, odd: pick left before right. Even:
pick right before left.
Idealization assumptions for AMDAHL'S LAW: 1. Ignore OS overhead creation, context switching maintenance.
2. ignore communication &amp; synchronization. 3. Perfect balance is achieved in division of programs into n.
3 reasons that causes a process to transition from running to wait: 1. Process need to perform I/O operations. 2. Waiting for some event to occur such as a non busy, waiting semaphore. 3. Time quantum expires.
Different reasons that causes process to transition from running to waiting: 1. Requesting I/O. 2. Waiting for a children to terminate. 3. Waiting for a semaphore.
1. What is (are) the advantage(s) of dividing an application into multiple threads relative to dividing it into multiple processes? –BOTH: Using less resources. - Easier communication using global variables
2. In a given program, only one sixth of the code is parallelizable. What's the maximum speedup factor that can be achieved on
a dual-core system under ideal conditions? – 1/(5/6+1/6) = 12/11
3. How does protecting a critical section (CS) with a semaphore ensure mutual exclusion? - When a process is in its CS, any process that tries to access a CS that is protected by the same semaphore is placed in the waiting state.
4. Which of the following is true about spin locks and mutex locks? –BOTH: A process waiting on a spin lock uses CPU cycles but a process waiting on a mutex does not. - Using mutex locks involves more context switching.
5. Which of the following is not true about shared memory and message passing? - Shared memory is slower than message passing on all multiprocessor systems.
6. Which of the following is not true about Remote Procedure Calls (RPCs)? - The user program communicates directly with the matchmaker without kernel intervention.

table? **Virtual page number**
-To handle deadlocks, operating system most often pretend that deadlocks **never occur**
-The monitor construct ensures that **only one process.**
-Thrashing occurs when **processes frequently access pages not in memory**
-A system uses FIFO policy for page replace
ment... **196**
T-The circular-wait conditions for a deadlock implies the hold and wait condition.
T-Deadlock can never occur if no process is allowed to hold a resource while requesting another resource
T-Even if a system is in an unsafe state, it is possible for the process to complete their execution without entering a deadlock state
F-In a virtual memory system, a virtual address and a physical address must be the same size.
T-Hashed page tables are particularly useful for processes with sparse address spaces.
F-Segmentation avoids external fragmentation
T-The buddy system for allocation kernel memory is very likely to cause fragmentation within the allocated segments.
F-A 32-bit logical address with 8KB page size will have 1,000,000 entries in a conventional page table.
T-A page fault must be preceded by a TLB miss
T-Stack algorithms can never exhibit Belady's anomaly
F-Protocols to prevent hold-and-wait conditions typically also prevent starvation
F-Non-preemptive scheduling algorithms are better for interactive jobs since they tend to favor...
F- In round robin scheduling, it is advantageous to give each i/o bound process a longer quantum...
F-fragmentation does not occur in a paging system.
T- A TLB miss could occur even though the requested page was
F- A deadlock-free solution eliminates...
F - Paging may suffer from internal frag…
Midterm 1---
-For a single-processor system **there will never be more than one running process**
-What is a trap/exception. **Software generated interrupt**
-The major difficulty in designing a layered operating system approach is **appropriately defining the various layers**
-The text segment of a process address space contains **the executable code associated with the process**
-Which of the following is not true about message passing **in direct communication multiple links may exist between a pair of processes**
-What is the READY state of a process **when process is scheduled to run after some execution**
-Which is true about processes and threads **threads in a process share the same file descriptors**
-The multithreading model supported by the linux OS is **One to One**
-When a process is accessing its heap space, it exists in the **running state**
-Which of the following is true about multilevel queue scheduling? **Each queue has its own scheduling algorithm**
-Which of the following statements is false with regards to the linux cfs scheduler? **There is a single, system wide value or vruntime**
-F Non-preemptive scheduling algorithms are better for interactive jobs since they tend to favor jobs that require quick responses
-T An interrupt vector contains the addresses of the handlers for the various interrupts
-F System calls can be run in either user mode or kernel mode
-F Each thread of a process has its own virtual address space
-T All processes in UNIX first translate to a zombie process upon termination
-F In round robin scheduling it is advantageous to give each I/O bound process a longer quantum than each CUP-bound process (since this has the effect of giving the I/O bound process a higher priority)
-F Processes in a microkernel architecture operating system usually communicate using shared memory
-An interrupt is **a signal that causes the control unit to branch to a specific location**
-When a process is created using the classical fork() system call, which of the following is not inherited by the child process? **Process ID**
-A race condition is when **the correctness of the code depends upon the timing of the execution**
-The text segment of a process address space contains **the executable code associated with the process**
-Which of the following would lead you to believe that a given system is an SMP-type system? **Each processor performs all tasks within the operating system**
-Embedded computers typically run on a **real time** operating system
-A message-passing model is **easier to implement than a shared memory model for inter-computer communication**
-The major difficulty in designing a layered operating system approach is **Appropriately defining the various layers**
-Most often, application programs access system resources using **application program interfaces**
-For single-processor system **there will never be more than one running process.**
-A thread control block **does not include information about the parent process resource allocation**
-The Producer-Consumer Problem is related to **the allocation of resources to process states**
-T The two primary purposes of an OS are to manage the resources of the computer and to provide a convenient interface to the hardware for programmers
-F A deadlock-free solution eliminates the possibility of starvation
-F The code that changes the system clock runs in user mode
-F A thread can be blocked on multiple condition variables simultaneously
-T It is possible to have concurrency without parallelism
-F The main difference between the use of test and set and the use of semaphores is that semaphores require the OS to do the busy waiting rather than the user program
-T Aging can alleviate the starvation problem of a low priority job
-F In a monolithic kernel, most OS components. E.g., memory management, inter-process communication, and basic synchronization modules, execute outside the kernel
-**Does unsafe state always lead to deadlock?** Deadlock means something specific: there are two (or more) processes that are currently blocked waiting for each other. In an unsafe state you can also be in a situation where there might be a deadlock sometime in the future, but it hasn't happened yet because one or both of the processes haven't actually started waiting.

- Access time has two major components
  - Seek time is time to move the heads to the cylinder containing the desired sector
  - Rotational latency is additional time waiting to rotate the desired sector to the disk head

-SSTF (Shortest Seek Time First)
after a request, go to the closest request in the work queue, regardless of direction reduces total seek time compared to FCFS
Disadvantages: starvation is possible; stay in one area of the disk if very busy
switching directions slows things down
-SCAN: go from the outside to the inside servicing requests and then back from the outside to the inside servicing requests. repeats this over and over.
-LOOK: like SCAN but stops moving inwards (or outwards) when no more requests in that direction exist.
-C-SCAN : moves inwards servicing requests until it reaches the innermost cylinder; then jumps to the outside cylinder of the disk without servicing any requests.
repeats this over and over.
-C-LOOK: moves inwards servicing requests until there are no more requests in that direction, then it jumps to the outermost outstanding requests. repeat this over and over.

CPU SCHEDULING
-Non-preemptive SJF: Avg wait time= (Time the process starts - arrival time) / # of proceesses
-Preemptive SJF (SRTF): Avg wait time= **if only one, then (Time the process starts - arrival time) **if it repeats, then add (Time the process starts - arrival time) of the 1st one + (finishing time - arrival time) of 2nd
Wait = Tturnaround - Tburst
Response = TfirstRun - Tarrival
Turnaround = Tcompletion - Tarrival

Assuming a page size of 4 Kbytes and that a page table entry takes 4 bytes, how many le
page tables would be required to map a 64-bit address space, if the top-level page table fi
a single page?

Data from the question:

Page table entry size = 4 bytes

Page size = 4 Kbytes

The number pages = $4 \times 1024 = 2^2 \times 2^{10} = 2^{12}$

Total address = $2^{10} \times 2^{12} = 2^{22}$ bytes

The address space is 2^{64} bytes. By adding a second layer of page tables, the top page table would point to 2^{10} page tables, addressing a total of 2^{32} bytes. Repeat this process,

Depth	Address Space
1	2^{22} bytes
2	2^{32} bytes
3	2^{42} bytes
4	2^{52} bytes
5	2^{62} bytes
6	2^{72} bytes $(> 2^{64}$ bytes$)$

Therefore, **6 levels of page table** required to map the 64-bit address.

1. Assume a system has a TLB hit ratio of 90%. It requires 15 nanoseconds to access the TLB, and 85 nanoseconds to access main memory. What is the effective memory access time in nanoseconds for this system?

Given
TLB Hit Ratio = 90%
Time to access TLB = 15 nanoseconds.
Memory access time = 85 nanoseconds.
Therefore EAT = [Hit ratio (TLB Access Time + Memory Access Time) + ((2 * Memory Access Time) + TLB Access Time) * (1-hit ratio)]
EAT = (85 + 15) *0.9 + (2*85 + 15)(1 – 0.9) = 90+18.5=108.5 nanoseconds

- A multi-level page table is preferred in comparison to n. single-level page table for translating virtual address to physical address it helps to reduce the size of page table needed to implement the virtual address space of a process.
- Which Of the following is not true Of virtual memory **It requires hardware support**
- External fragmentation will not occur when **No matter which algorithm is used. it will always occur**
- The purpose of **A TLB is to cache page translation information.**
- Which page replacement algorithm suffers from Belady's anomaly? **First In First out (FIFO)**
- What is the difference between the unit and signal operations of a semaphore and those of a condition variable (of a monitor)? A call to the condition variable operation Wait() will always cause the calling thread to block (wait). **The semaphore P() is a test-and-maybe-wait operation: only if the semaphore is not positive will the thread be forced to wait.**

----

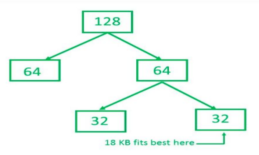
2. We are assuming perfect load balancing (we can perfectly divide the work among multiple threads or processes).

Threads	Process
• Run in a shared memory space. • Allocating space for threads are user space. • Scalability process can take in advantages if multiprocessor architecture.	• Process run in separate memory space. • Kernel space • Hierarchy weight • Less resources sharing • Economy & speed

Change	Will it work correctly?	Items Produced	Items Consumed	Explanation
Delete ***SetIn()*** on Line 6 of the Producer	No			
Replace ***GetIn()*** with ***in*** ***Producer*** ***consumer***	Yes No			
Replace **GetOut()** with ***out*** Producer consumer	No Yes	n	n	Out is update by consumer. Doesn't need memory.

Case	Max useful producers	Max speedup ratio	Brief Explanation
ConsTime = ProdTime	1	1	Time = speeding up prod will not hel
ConsTime = 5 x ProdTime	1	1	
ProdTime = 5 x ConsTime	5	5	Speed up to 5 times of producer

Consider a system having buddy system with physical address space 128 KB.Calculate the size of partition for 18 KB process.
Solution –



So, size of partition for 18 KB process = 32 KB. It divides by 2, till possible to get minimum block to fit 18 KB.

random access but causes external fragmentation.
-If a process uses 1100B of memory in a system with a page size of 512B, what's the size of internal fragmentation? 436B
-How does indexed disk allocation compare with FAT-based linked allocation? - Indexed allocation may waste a substantial amount of disk space if there are many small files.
-**Which of the following is (are) true about the Working Set Size (WSS)? -**A larger WSS increases the page fault rate. AND - If the WSSs of all processes exceeds the number of available frames, the system will thrash.
-**Which of the following is true about storage allocation algorithms?** - First fit and best fit have been found to give better utilization, on average, than worst fit.
-How do processor affinity and load balancing interact in a multiprocessor environment? Improving load balancing may conflict with the processor affinity requirement in some cases.
Q2: (a). **Schedule each of the following inputs using the algorithm shown below it.** First give the output in the form of a Gantt chart, and then compute the average waiting time. Show your work.
Input 1:

Process	CPU Burst
P1	5 6
P2	2 4
P3	7 9
P4	3 3

-Apply Round Robin (with a time quantum of 3 4). Assume that all processes arrive at time 0 but in the order shown in the above table (there are negligibly small differences in arrival times).
Intervals of 4 time quantum.
After First round: P1 needs 2 more, p3 needs 5 more
After Second round: p3 needs 1 more
P1 waited 0 +15 – 4
P2 waited 4
P3 waited 8 + (17-12) + 0
P4 waited 12
Input 2:

Process	Arrival Time	CPU Burst	Priority
P1	0	7	2
P2	2	3	1
P3	4	2	3
P4	6	4	1

Apply Preemptive Priority Scheduling. Assume that a smaller number indicates higher priority. If a tie ever occurs, break it arbitrarily.
Input 3:

Process	Arrival Time	CPU Burst
P1	0	7
P2	2	3
P3	4	2
P4	5	6

Apply shortest remaining time first
P1 (0-2) > p2 (2-4,5) > p3 (5-7) > p1 (7-12) > p4 (12-18)
Average wait time
P1: 0 + (7-2)
P2: 2-2
P3: 5-4
P4: 12-5
Then divide sum by 4
(b) A major problem with priority scheduling is starvation. Why is priority scheduling more likely than other scheduling algorithms to cause starvation? Describe how starvation happens in priority scheduling.
Starvation happens when high priority processes keep arriving and low priority processes never get a chance to execute.
-**Give one good solution that operating systems use to prevent starvation in priority scheduling. Give a brief description of the solution, not just its name.**
Aging is a good solution because the longer the process waits in the queue, the higher the priority of this process becomes so it eventually gets the cpu.