**Process:**

**-unit resource ownership**

**PCB**: holds the ‘process state’ when the process is not executing(not using the CPU).

-There are two components of the PCB: the state of the CPU and the state of resources associated with the process

**Thread:**

-unit of dispatching

-access to memory and resource of its process.

**Thread Benefit:**

-less time to create  
-less time to terminate  
-less time to switch between  
-efficiency in communications

**Thread States:**

-running, ready, block (no suspend)

**User level thread Advantage:**

- All thread mgmt done by app  
kernel not aware of threads  
runs on any OS

**User level thread Disadvantage:**

- many system calls are blocking  
in pure ULT, a multithreaded app cannot take advantage of multiprocessing

**Kernel Level Thread advantages**

- Kernel can simultaneously schedule mult. threads from same prco on mult processors  
if one thread is blocked, kernel can schedule other threads  
kernel routine can be multithreaded

**Kernel Level Thread disadvantages**

-requires mode switch to kernel slower than ULT

**Resource Sharing**: threads share resources of process, easier than shared memory or message passing

In a multiprogramming system with a single processor, it is necessary to save the state of the running process in secondary storage before loading a new process on the CPU.

-true

Nested interrupt servicing occurs only when a higher priority interrupt is raised during the servicing of a lower priority interrupt.

-true

When a process is in the \_\_\_\_\_\_\_\_\_ state, it is in secondary memory but is available for execution as soon as it is loaded into main memory.

-ready and suspend

A computer platform consists of a collection of hardware resources, such as the processor, main memory, I/O modules, timers, and disk drives.

-true

A process that is not in main memory is immediately available for execution, regardless of whether or not it is awaiting an event.

-false

When one process spawns another, the spawned process is referred to as the \_\_\_\_\_\_\_\_\_\_\_\_\_\_.

-child process

\_\_\_\_\_\_\_\_\_ involves moving part or all of a process from main memory to disk.

-swapping

The collection of program, data, stack, and attributes is referred to as the \_\_\_\_\_\_\_\_\_.

-process image

The \_\_\_\_\_\_\_\_\_ is the less-privileged mode.

-user mode

The portion of the OS that selects the next process to run is called the \_\_\_\_\_\_\_\_\_\_

-dispatcher

A process is in the \_\_\_\_\_\_\_\_ state when it is in secondary memory and awaiting an event.

-block and suspend

The process is said to be operating in a \_\_\_\_\_\_\_\_\_ fashion if each process in the queue is given a certain amount of time, in turn, to execute and then returned to the queue, unless blocked.

-round robin

A process in the \_\_\_\_\_\_\_ state is in main memory and available for execution.

-ready

\_\_\_\_\_\_\_\_\_\_\_ supports more than one task making progress

-concurrency

\_\_\_\_\_\_\_May be provided either as user-level or kernel-level.

-Pthread

\_\_\_\_\_\_\_ are used in UNIX systems to notify a process that a particular event has occurred.

-signal

Every signal has \_\_\_\_\_\_\_\_\_\_\_\_\_ that kernel runs when handling signal

-default signal

\_\_\_\_\_\_\_\_Typically use an intermediate data structure between user and kernel threads

-lightweight process

What does the term priority inversion refer to?

1. A situation where a process has been assigned the wrong priority.
2. A data structure where the priority queue is stored backwards.
3. A situation where a low priority process must wait for a high priority process.
4. **A situation where a high priority process must wait for a low priority process.**

How can a priority inversion be corrected (in the least intrusive way)?

1. Temporarily lowering the priority of a high priority process.
2. **Temporarily raising the priority of a low priority process.**
3. Not allowing a low priority process to perform system calls.
4. Not allowing a high priority process to perform system calls.

Which of the following operations would MOST LIKELY need to be inside a critical section?

1. Fetch the value of a nonshared variables.
2. Fetch the value of a shared variable.
3. Reduce the value of a nonshared variable by 1.
4. **Reduce the value of a shared variable by 1.**

Which of the following events would be LEAST LIKELY to cause the end of a CPU burst by a running thread?

1. Performing an input/output operation.
2. **Performing a wait on an unblocked semaphore.**
3. Performing a wait on a blocked semaphore.
4. Waiting to receive a message from another thread.

A synchronization algorithm is wait-free if, when it is used, no thread

1. **ever needs to wait for another thread to do anything, under any circumstances.**
2. needs to wait for any other thread, as long as no thread spends long inside critical sections.
3. needs to wait for any other thread that does not want to enter a critical section.
4. can be made to wait to enter a critical section while more than some bounded number of other threads enter the critical section ahead of it.

An advantage of a preemptive scheduler over a nonpreemptive scheduler is

1. If a fixed collection of bursts is to be done, the total time to complete the bursts is smaller with the preemptive scheduler.
2. **A preemptive scheduler prevents a process from being locked out of the CPU.**
3. A preemptive scheduler makes solution of the critical section problem easier.
4. A preemptive scheduler is easier to implement than a nonpreemptive sheduler.

What is a test-and-set instruction, and what is it useful for?

1. **A test-and-set instruction *atomically* fetches the value of a variable and sets the variable to hold 1. It is useful for solving the critical section problem.**

Which of the following instructions should be allowed only in kernel mode?

1. disable all interrupts
2. read the time of day clock
3. set the time of day clock
4. change the memory map

-a Kernel mode only. Obvious, c: Needs to be done only in kernel mode otherwise, a job could set the clock back to increase its processor time slice (among other things) , d: Kernel mode only. b: but the reading of the time-of-day clock should be allowed in both the user and kernel mode.

-**threads share** : Memory Regions, address space, resource of process, global variable, heap, static data

-**Thread don’t share**: instruction pointer, stack pointer, register, program counter.

-Each thread has its separate set of register values and a separate stack.

Process API:

\_\_\_\_\_system call is used to create a new process

-fork()

- **fork**

A parent process uses fork() to create a new child process. The child process is a copy of the parent. After the fork, both parent and child executes the same program but in separate processes

The \_\_\_\_\_\_\_\_\_ determines which process runs in a given moment in time

-CPU Scheduler

\_\_\_\_Run a program that is different from a calling program

-exec()

The \_\_\_\_\_\_ family of system calls allows a child to break free from its similarity to its parent and execute an entirely new program.

-exec()

- **exec**

Replaces the program executed by a process. The child may use exec after a fork to replace the process’ memory space with a new program executable making the child execute a different program than the parent.

\_\_\_\_\_\_Concurrent access to shared data may result in

-data inconsistency

**\_\_\_\_\_\_\_\_,** is an instruction that locks the memory bus to prohibit other CPUs from accessing memory until it is done. It is atomic because it also needs to prevent two processes from accessing memory at the same time to do this it must be seperate of process switching.

-TSL, or Test and Set Lock

\_\_\_\_\_\_\_is a module that is used to help prevent deadlock by using mutual exclusion. They are used to ensure that only one thread can be open at a time.

-monitor

Suppose that a 10-MB file is stored on a disk on the same track (track #: 50) in consecutive sectors. The disk arm is currently situated over track number 100. How long will it take to retrieve this file from the disk? Assume that moving the arm from one cylinder to the next takes about 1 ms and it takes about 5 ms for the sector where the beginning of the file is stored to rotate under the head. Also, assume that reading occures at a rate of 100 MB/s.

- 50 ms to move to the correct cylinder + 5 ms to move to the right sector + 100 ms to read the file = 155 ms, or .155 seconds.

Usually, user-level threads, cannot take advantage of multiprocessing whereas, kernel-level threads can take advantage of it.

It simply means that we can run several kernel-level threads, in parallel on a multi-core computer system. But the same cannot be done for user-level threads

Give 3 examples of interrupts

- timer, I/O, segfault

We trick programs into thinking they own all of \_\_\_\_\_\_.

-ram

Name the 2 modes of operation

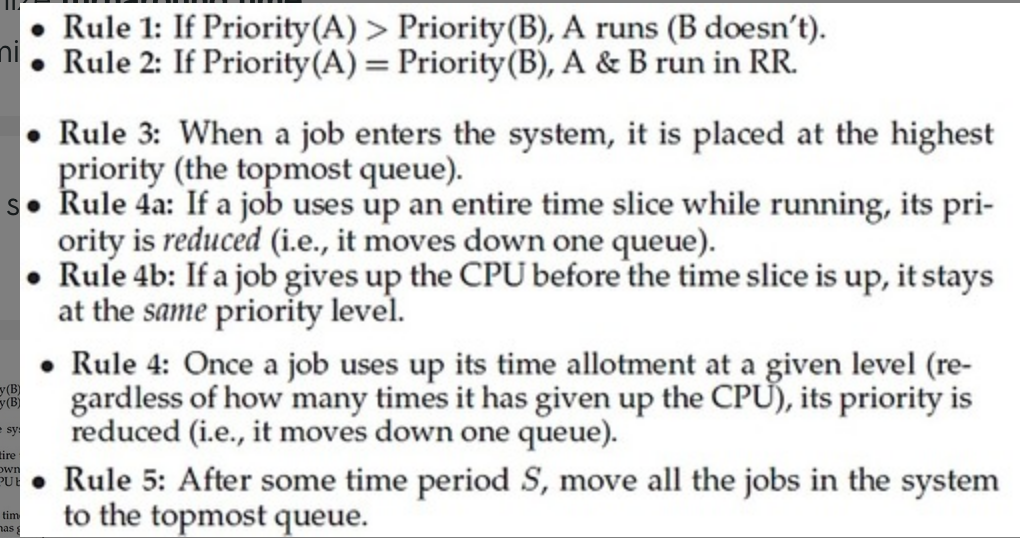
-user level mode , kernel level mode

True or false: An interrupt can be interrupted by another interrupt?

-true

**Multilevel Feedback Queue Scheduling (MLFQ)**

\_\_\_\_\_\_\_\_\_keep analyzing the behavior (time of execution) of processes and according to which it changes its priority

-

An un-interruptible unit is known as :   
a) single   
b) atomic   
c) static   
d) None of these

-atmoic

The TestAndSet instruction is executed:   
a) after a particular process   
b) periodically   
c) atomically   
d) None of these

-atimically

The main disadvantage of spinlocks is that :

-they required busy waiting

Is it faster to parse through an array row-by-row, or column-by-column

-row by row

What is a process?

-An instance of a program

True or false: PID changes on a call to fork

-true

True or false: PID changes on a call to exec

-False

In a call to dup2(A, B) what is A? B?

A is the old file descriptor  
B is the new file descriptor

True or false: A pipe is bidirectional

False (unidirectional)

P(mutex) locks or unlocks? ++ or --?

Locks, --

V(mutex) locks or unlocks? ++ or --?

Unlocks, ++

How do servers and clients communicate?

Sockets

True or false: Threads are an efficient way to communicate between different programs.

-true

When does a process leave the terminated state?

-when wait()

quiz 2

A preempted process is placed in the blocked state after the CPU is reallocated to another process.

-false

The exec() function overlays the existing code, data, and PCB of the calling process.

-False,the PCB is not overlaid; it is updated with the new memory information for the loaded code and data.

The \_\_\_\_ of a process contains temporary data such as function parameters, return addresses, and local variables.

-stack

Child processes inherit UNIX ordinary pipes from their parent process because

-A pipe is treated as a file descriptor and child processes inherit open file descriptors from their parents.

In RR scheduling, the time quantum (time-slice) should be small with respect to the context-switch time.

-false, be large

The following code is an example of poor \_\_\_\_\_\_\_ locality

for( int i = 0; i < NUM; i++ ) {

    for( int j = 0; j < NUM; j++ ) {

            float f = Array[ j][ i ] ; ;

            sum += f;

         }

}

-spatial

**Spatial** –adjacent cells, or cells in relatively close memory regions

**Temporal** –reuse of cells or resources within a relatively short time duration

The redirection operation in a shell is effectively associating a designated file descriptor value with a different entry in the process file table.

-true

The number of processes completed per unit time is known as \_\_\_\_\_\_\_\_\_\_  
b) Throughput

The state of a process is defined by:  
a) the final activity of the process  
b) the activity just executed by the process  
c) the activity to next be executed by the process  
d) the current activity of the process

-d

The Process Control Block is:  
a) Process type variable  
b) Data Structure  
c) A secondary storage section  
d) A Block in memory

-b

The entry of all the PCBs of the current processes is in:  
a) Process Register  
b) Program Counter  
c) Process Table  
d) Process Unit

-c

The degree of multiprogramming is:  
a) the number of processes executed per unit time  
b) the number of processes in the ready queue  
c) the number of processes in the I/O queue  
d) the number of processes in memory

-d

A single thread of control allows the process to perform:  
a) only one task at a time  
b) multiple tasks at a time  
c) only two tasks at a time  
c) all of the mentioned

-a

The objective of multiprogramming is to :  
a) Have some process running at all times  
b) Have multiple programs waiting in a queue ready to run  
c) To minimize CPU utilization  
d) None of the mentioned

-a

The systems which allows only one process execution at a time, are called   
a) uniprogramming systems  
b) uniprocessing systems   
c) unitasking systems   
d) none of the mentioned

-a

What is the ready state of a process?   
a) when process is scheduled to run after some execution   
b) when process is unable to run until some task has been completed  
c) when process is using the CPU   
d) none of the mentioned

-a

The objective of multi-programming is to : (choose two)   
a) Have some process running at all times   
b) Have multiple programs waiting in a queue ready to run   
c) To minimize CPU utilization   
d) To maximize CPU utilization → D

-a,d

True or false: A context switch from one process to another can be accomplished without executing OS code in kernel mode

-false

A process that just called P() on a semaphore with value 0 is likely in the \_\_\_\_\_\_\_\_\_\_\_\_\_\_ state.

-waiting state

A process that has just completed I/O and is waiting to regain access to the CPU is likely in the \_\_\_\_\_\_\_\_\_\_\_\_ state.

-ready state

**Race condition:**

-Can occur when multiple processes or threads read and write shared data items

-the “loser” of the race is the process that updates last and will determine the final value of the variable

-Coarse-grained Atomic Actions :

-A sequence of fine-grained atomic actions that appear to execute indivisibly. Need a synchronization mechanism to construct a “coarse- grained” atomic action

T / F A program containing a race condition will always result in data corruption or some other Incorrect behavior when executing.

-false

T / F Although the spin­lock solution and the Peterson algorithm provide busy-­waiting solutions to the critical section problem, only the spin­lock solution controls the order of entry for waiting processes.

-false

T/F Condition Synchronization is concerned with delaying a process until the next state of the process is one that will be safe or valid.

-true

T/ F When implementing semaphores in user mode in a system with multiple processors, we can use a spinlock to create the necessary atomicity of the semaphore functions.

-true

T / F In the solution to the multi-slot buffer producer-consumer problem, we can use semaphores to provide bounded concurrency.

-true

T / F When semaphores and semaphore operations are implemented in a non-preemptable kernel, we will not require busy waiting in the implementation.

-true

A relocation register is used to check for invalid memory addresses generated by a CPU.

-false

Absolute code can be generated for \_\_\_\_.

A) compile time binding

B) load time binding

C) execution time binding

D) interrupt binding

-A

\_\_\_\_\_ is the method of binding instructions and data to memory performed by most general-purpose operating systems.

A) Interrupt binding

B) Compile time binding

C) Execution time binding

D) Load time binding

-c

An address generated by a CPU is referred to as a \_\_\_\_.

A) physical address

B) logical address

C) post relocation register address

D) memory management unit (MMU) generated address

-b

Suppose a program is operating with execution-time binding and the physical address generated is 300. The relocation register is set to 100. What is the corresponding logical address?

A) 199

B) 201

C) 200

D) 300

-c

The mapping of a logical address to a physical address is done in hardware by the \_\_\_\_\_\_\_\_.

A) memory management unit (MMU)

B) memory address register

C) relocation register

D) dynamic loading register

-a

The \_\_\_\_\_ binding scheme facilitates swapping.

A) interrupt time

B) load time

C) assembly time

D) execution time

-d

The roll out, roll in variant of swapping is used \_\_\_\_.

A) when a backing store is not necessary

B) for the round-robin scheduling algorithm

C) for priority-based scheduling algorithms

D) when the load on the system has temporarily been reduced

-c

\_\_\_\_\_ is the dynamic storage allocation algorithm that results in the smallest leftover hole in memory. /largest hole: worst fit

A) First-fit

B) Best-fit

C) Worst-fit

D) None of the above

-b

A(n) \_\_\_\_ page table has one page entry for each real page (or frame) of memory.

A) inverted

B) clustered

C) forward-mapped

D) virtual

-a

Consider a logical address with a page size of 8 KB. How many bits must be used to represent the page offset in the logical address?

A) 10

B) 8

C) 13

D) 12

-c,

2^13 = 8192

8kb = 8192 byte

Given the logical address 0xAEF9 (in hexadecimal) with a page size of 256 bytes- what is the page number /page offset?

-0xAE

-0xF9

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?

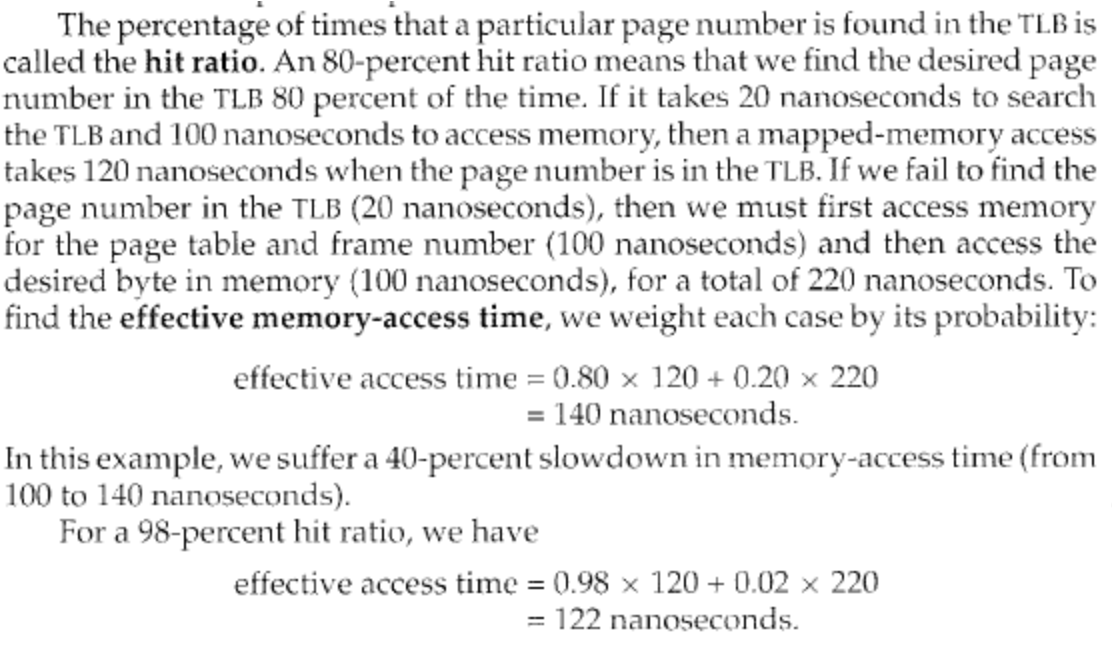
A) 108.5

B) 100

C) 22

D) 176.5

-a



15 + 85 = 100 to map memory

15 + 85 +85 =185

(.90) \* 100 + (.10) \*185 =108.5

. Consider a logical address with 18 bits used to represent an entry in a conventional page table. How many entries are in the conventional page table?

A) 262144

B) 1024

C) 1048576

D) 18

-a

2^18 = 262144

. Consider a 32-bit address for a two-level paging system with an 8 KB page size. The outer page table has 1024 entries. How many bits are used to represent the second-level page table?

A) 10

B) 8

C) 12

D) 9

-d

-1024 = 2^10 = 10 bit

32 = 10 +13 +x

x = 9

2^9 = 512 entries

The Operating System maintains the page table for

-each process

In paging, every address generated by the CPU is divided into two parts :

-page number , page offset

If the size of logical address space is 2 to the power of m, and a page size is 2 to the power of n addressing units, then the high order \_\_\_\_\_ bits of a logical address designate the page number, and the \_\_\_\_ low order bits designate the page offset.

-m-n , n

After return from a call to pthread\_cond\_wait(&C, &M) the calling thread can be sure that the mutex M is locked

-true

If one or more threads are sleeping on a call to pthread\_cond\_wait() and another thread calls pthread\_cond\_signal() for the same CV, one of the sleeping threads will wake up and try to relock the mutex.

-true

A page table entry (PTE) contains the number of the frame the page maps to.

-true

In order to reduce fragmentation with contiguous programs, memory compaction is used. If DMA I/O is active at the time of compaction, the compaction process may have to be suspended

-true

With segmentation, a logical address consists of \_\_\_\_\_.

A) segment number and offset

B) segment name and offset

C) segment number and page number

D) segment table and segment number

-a

Which of the following data structures is appropriate for placing into its own segment?

A) heap

B) kernel code and data

C) user code and data

D) all of the above

-d

Assume the value of the base and limit registers are 1200 and 350 respectively. Which of the following addresses is legal?

A) 355

B) 1200

C) 1551

D) all of the above

-b

A(n) \_\_\_\_\_\_ matches the process with each entry in the TLB.

A) address space identifier

B) process id

C) stack

D) page number

-a

Hashed page tables are commonly used when handling addresses larger than 32 bits.

-true

The \_\_\_\_ is the number of entries in the TLB multiplied by the page size. A) TLB cache B) page resolution C) TLB reach D) hit ratio

-c

\_\_\_\_\_ occurs when a process spends more time paging than executing. A) Thrashing B) Memory-mapping C) Demand paging D) Swapping

-a

The \_\_\_\_\_ is an approximation of a program's locality. A) locality model B) working set C) page fault frequency D) page replacement algorithm

-working set

On a system with demand-paging, a process will experience a high page fault rate when the process begins execution

-true

Mobile operating systems typically support swapping.

-false

The TLB only contains a fraction of the entries in a page table for a process. It may contain some page table entries for many processes.

-true

In paging, when the valid – invalid bit is set to valid, it means that the associated page :

-logical address

In a paged memory, the page hit ratio is 0.35. The time required to access a page in main memory is equal to 100 ns. The time required to access a page in the TLB is 10 ns. The average time required to access a page is :

- Page access time=(1-page fault)\*(Time required to access a page in primary memory)

+(page fault)Miss penalty

=(1-.65) \* 10 + (.65) \* 100 =68.5

Average access time = hit ratio \* primary memory access time + (1 – hit ratio) \* secondary memory access time

=.35\*10 + (1-.35)\*100 = 68.5

In a pure segmentation system, if there are 32 segments, each of size 1Kb, then the logical address should have

-1kb = 1024 byte

2^10 = 1024

32 = 2^5

10 bit + 5 bit = 15