**RTOS Questions**

**1. Define RTOS?**

RTOS are those which are used to accomplish a task within a specific amount of time and they are time deterministic and functionally correct

**2. Name of few RTOS?**

Vx-works, FreeRTOS, Xenomai, UC-OS,Qnix

**3. What is interrupt latency and context switching?**

Switching from normal execution of the program to ISR state is called a switching **or** it is a process of storing and restoring the state of process and there is no useful work done during switching that’s why it should b minimum and **context switching is done by scheduler and inside is done by dispatcher**

And the time elapsed between interrupt occur and serviced is called as interrupt latency and this should also be minimum

**4. Name the kernel objects?**

Mutex, semaphore, spinlock, MQ, pipe,

**5. Why RTOS is time deterministic?**

**The scheduler in a Real Time Operating System (RTOS) is designed to provide a predictable (normally described as *deterministic*) execution pattern. This is particularly of interest to embedded systems as embedded systems often have real time requirements. “**A real time requirement is one that specifies that the embedded system must respond to a certain event within a strictly defined time (the *deadline*)**”**.A guarantee to meet real time requirements can only be made if the behaviour of the operating system's scheduler can be predicted (and is therefore deterministic).

So GPOS is not time deterministic

GPOS   
i. Non deterministic   
ii. Time insensitive   
iii. Can use virtual memory concept

**6. Kernal is written in c so kernel contains main function or not?**

**No**

Switching->BIOS->BL->kernel (using memory address called by the bootloader) so kernel is in 2ndry memory.

**7. How you can prove that RTOS is periodic or non periodic?**

**Periodic** Tasks: Time-driven and recurring at regular intervals.

A car checking for a wall every 0.1 seconds;

An air monitoring system grabbing an air sample every 10 seconds.

**Aperiodic**: event-driven

That car having to react to a wall it found

The loss of network connectivity.

**8. What you mean by AMP and SMP in RTOS?**

AMP- Asymmetric Multi-Processing -**same kernel image for diff CPU**

SMP- Symmetric Multi-Processing -**diff. Kernel image for diff CPU**

**An AMP system:**

multiple CPUs

each of which may be a different architecture [but can be the same]

each has its own address space

each may or may not run an OS [and the OSes need not be the same]

**An SMP system:**

multiple CPUs

each of which has the same architecture

CPUs share memory space [or, at least, some of it]

Normally an OS is used and this is a single instance that runs on all the CPUs, dividing work between them

**9. What is difference between modularity and scalability?**

Modularity means which module you are going to use and scalability means you can increase or decrease the length of code as per your convenience

**10. Can we shift the ISR table and address?**

**Yes** , we can shift the ISR table but we have to justify the address by using some registers

**11. What you mean by fixed and static system in RTOS?**

In computer terminology, dynamic usually means capable of action and/or change, while static means fixed

The simplest approach to the scheduling problem is to assign (static/dynamic) priorities to all tasks. It's quite obvious as an example that the creation and deletion of tasks shouldn't be done during RT-tasks.

**12. What is goal of RTOS? And RTOS is monolithic or microkernel?**

Event driven, time deterministic and functionally correct and **RTOS is monolithic**

**14. Which file can be modified in the UC-OS2? And name the file name for each one?**

There are 2 files when we are going to download the UC-OS2 named **port and source**

There are .c .asm and .h file in port folder and they are highly portable and we can’t modify them while in source folder OS\_core.c(initiation,scheduler,highest priority task),OS\_flag(event flag,status register),OS\_Mbox(message to transfer ),OS\_mem,OS\_sem q.c,task.c,timer.c are difined and we can modified them.

**OS\_TIME is much faster them HMSM.**

**15. Which one is better language c or assembly?**

C is better language bcoz this is portable (if and else are not hardware dependent)

And one more thing that the machine efficiency of the c is not upto the mark as per assembley but programming efficiency is good

**16. How many task state are there in the RTOS? can we go from ready to block state?**

**Ready ,running,**waiting state,dormant,ISR**,blocked state**

No, to avoid the starvation of lower priority state we can’t go

**17. Why thread is disadvantage in RTOS for Security purpose?**

**Blocking:** The major disadvantage is that if the kernel is single threaded, a system call of one thread will block the whole process and CPU may be idle during the blocking period.

**Security:** Since there is, an extensive sharing among threads there is a potential problem of security. It is quite possible that one thread over writes the stack of another thread (or damaged shared data) although it is very unlikely since threads are meant to cooperate on a single task.

**18. Which scheduling algorithm is used in RTOS and explain them?which of them dyanamic in nature?**

4th and 5th are **dynamic** priority scheduling

1.Round robin-use of the time quantum,if process is completed within that time,then good another wise it will be done later.

2.Priority based scheduling alorithum-higher the priority sooner will be execution

3.Rate monotonus-**higher the period higher the priority-periodic-static priority and uniprocessor**

4.Earliest dedline first-earlist dedline highest priority-uni processor and dynamic priority

5.Least laxity-least the laxity higher the priority

**6.SJF-shorter the period higher the priority—sbse jyada better**

7.FCFS

8.Fare share

9. Multilevel queue

**19. What is standard of BT and WIFI and working frequency?**

BT- 802.15.1 to 4 ---- 2.4GH

Wifi- 802.11.x (x=a, b, g, n and so on) ------- 2.4, 3.6 and 5 GHz

**20. What is high level view of RTOS?**

Application layer-RTOS layer-BSP-Target hardware

**21. Explain the features of RTOS?**

Reliable, predictable, scalable, functionally and timely correct, performance, diskless, portable

**22. Which are the kernel services? Define services?**

Services are those which provide the set of api calls to operate on the object to develope user application.

Kernal services are semaphore, event flag, mutex, queue, task,time

**23. What is footprint and how you will judge the code length?**

Size occupied by the code in ram is called as FP and we will judge the length by data type of the variable.

**24. What is difference between task, process and thread? And what do you mean by static and dynamic task?**

**Task-** An execution of thread results in a task......A task is simply a set of instructions loaded into the memory. Threads can themselves split themselves into two or more simultaneously running tasks.

**Looks like normal function but it never returns anything and argument is always (void \*) type**

**Process-** **A program in execution is known as ‘Process’**. A program can have any number of processes.......

A process invokes or initiates a program....... A process can have multiple threads running

**Thread**- A thread is the smallest unit of execution that lies within the process and it is a single flow of control

There are 13 task in RTOS in which **11 are application task** (10 create task and 1 context switching task) and **2** **are system task** (1 ideal and 1 statistic task)

Static means we have to make one task statically whose priority will be 0 and those which we are going to create during run time is call dynamic task

**25. What is difference between mutex and semaphore?**

Mutex properties-

Ownership, task deletion safety, priority inversion, recursive locking

**26. What is functionality of the hooks?**

**Hooks are used to extend the functionality of UC-OS2** means printf(“”)

**27. What is difference between if and #if?**

In #if part if that part is not satisfied then that code will be deleted from the processor

#if will work at the pre-processor and if will work at the compilation time

**28. What will happen if we are not going to provide the delay?**

If we are not going to provide the delay then another task will not get a chance to scheduled

**29. What is the different method to enter into a critical section?**

1-disabling the interrupt without taking the backup of the interrupt

Disadvantage-**nesting job because interrupt will be enable in the 2nd half loop**

2-taking backup of the interrupt before disabling

3-taking backup of interrupt and CPU register then disabling the interrupt

**30. What do you mean by priority inversion? And what the method to prevent inversion?**

**Inversion-**if any one locked the semaphore then only that can use and as he will free then other can use

**Unbound and bounded priority** inversion will come when middle order priority will come before the completion of the low priority task execution so after completion, middle will be executed not the higher.

**Inheritance-**lower is running then if higher priority task will come **it will boost till his level**.

**Ceiling-** lower is running then if higher priority task will come **it will boost to the highest priority level.**

**31. How you will determine the highest priority task? And who will decide the priority?**

Using **unmap table and group** **table** we will decide the highest priority.

Y=osunmaptable(osrdygrp)

X= osunmaptable(osrdytbl[y])

Priority=(y<<3)+x

**32.** **What is the Difference between RTOS and GPOS ?**

GPOS are Unpredictable while RTOs are predictable & Deterministic.

**GPOS uses a Monolithic** architecture while rtos uses a **flat memory architecture.**

**Gpos are non scalable and has larger footprint and higher context switch latency.**  
**Rtos uses priority preempitve scheduling** while most gpos uses round robbin way of scheduling.

RTOS preventing simultaneous access of shared resource and multi-tasKing

**33. What do you mean by atomic operation, re-entraned code and volatile variable?**

**AO-**without any interrupt get executed

**Re-enteraned code**-bad for us bcz value changes due to this, we can avoid re-entrant by declare them as locally

**Volatile variable-**which do doesn’t go into the cache memory and complier doesn’t do any operation on that and his value changes frequently

**34. Which is the component in the architecture of RTOS?**

Aplliction software,OS-2,OS-configuration,OS-2 port, CPU and timer

**35. What is cape in RTOS?**

It is just like a size in which another thing is going to be fit

**36. What is porting and what is functionality of the patching?**

Porting is the process of adapting software in an environment for which it was not originally written or intended to execute in. **Adapting A Kernel To Microprocessor Or Microcontroller Is Called Porting**.

We are going to add more functionality with the help of **patch** and we can say that to register the software we need **patch…...IDM**

**37. Which are the basics of the porting?**

Cross compliar, bootloadar, kernel, Root file system.

**38. What do you mean by beagle bone?**

**BeagleBone** is an $89 MSRP, credit-card-sized Linux computer that connects to the Internet and runs software such as Android 4.0 and Ubuntu

The Beagle Board is a low-power open-source hardware single-board computer produced by Texas Instruments in association with Digi-Key and Newark element14

**39.** **What are the similarities between RTOS and GPOS?**

•multi-tasKing

•resource management

•abstraction

**40. What are the task operatons?**

Creation, deletion, mutex, message queue

**41. What do you mean by semaphore, MQ, mutex, event registers?**

These are the synchronizing technique which is used to prevent the simultaneous access of shared memory.

Pend and post, lock and unlock, acquire and release are used for them.

**42. What is signal and what will happen when an interrupt occurs during run time?**

Signals are those which contain some meaning full information and **when an interrupt is going to occur then it passes to signal handler from where it will handle in the kernel space**. And after execution it will come back to the same position

**43. What is audio frequency range? And speech frequency range?**

A voice frequency (VF) or voice band is one of the frequencies, within part of the audio range that is used for the transmission of speech. In telephony, the usable voice frequency band ranges from approximately 300 Hz to 3400 Hz

44**.** **What is the role of scheduler, scheduler entity; give example of scheduler entities?**

**A Maintenance Scheduler is responsible to take a maintenance plan and bring together all the resources needed to complete it. The Scheduling role involves assembling and coordinating the information, people, materials, equipment, along with all the other necessary resources to get the job done.**

Kernel Scheduled Entities, or KSE, is a kernel-supported threading system found in FreeBSD, which allows a single process to have multiple kernel-level threads.

KSEs were mandatory at introduction; made optional at kernel build time in the 7.0 release and removed from the 8.0 release with a compatibility library.

45.**If an ISR makes System calls, the dispatcher is \_\_neutral\_\_\_\_\_ until the ISR fully completes its execution.**

46.**Explain the structure of TASK CONTROL BLOCK**

A Process Control Block (PCB, also called Task Control Block or Task Struct) is **a data structure in the operating system** **kernel** containing the information needed to manage a particular process. **The PCB is "the manifestation of a process in an operating system".**

Task Control Block - The Task Control Block (TCB) specifies all the parameters necessary to schedule and execute a routine. Typically, a TCB is a 6-10 words long and is logically divided into two parts:

**• Task-Independent Parameters -** The first four words (32-bit) of the TCB are task-independent and simply specify the scheduling parameters to the DSP scheduler.

**• Task-Dependent Parameters -** These parameters specify the routine to be executed and the parameters of execution. The number and format of these parameters is routine dependent.

47. **What are the different types of semaphores in RTOS? Which is the fastest and why ?**

Binary semaphores are used for both mutual exclusion and synchronisation purposes.

Binary semaphores and mutexes are very similar but have some **subtle** differences:

Mutexes include a priority inheritance mechanism, binary semaphores do not. This makes binary semaphores the better choice for implementing synchronisation (between tasks or between tasks and an interrupt),

**and mutexes the better choice for implementing simple mutual exclusion.** The description of how a mutex can be used as a mutual exclusion mechanism holds equally for binary semaphores

48. **Explain Hard and Soft real time system .**

* + If the result has utility even after the deadline has passed, the deadline is classified as **soft**, otherwise it is **firm**.
  + If a catastrophe ***could*** result if a firm deadline is missed, the deadline is **hard**.

49. **What is interface? What do you mean by API?**

**Interface-** a point where two systems, subjects, organizations, etc. meet and interact

**API-**Application program interface (API) is a set of routines, protocols, and tools for building software applications. An API specifies how software components should interact and APIs are used when programming graphical user interface (GUI) components

50. **What is difference between message queue and mail box?**

Message queue is used to transfer a single byte of data while mbox is used to transfer more than one byte

**51.Why a task in the ready list cannot move directly to the blocked state?**

Because , A task first need to run so, it can make a blocking call

**52.How can a  running task moves to blocked state ?**

By making a call that request an unavailable resource

By making a call that request to wait for an event to occur

By making a call to delay the task for some duration

**53.Why blocked state is important in real time system ?**

Because without blocked state, lower priority tasks could not run, if higher priority tasks are not designed to block , CPU starvation can result .

**54.What is CPU starvation ?**

CPU starvation occurs when higher priority task use all of the CPU execution time and lower priority task do not get to run

====================================================================================

**1. what is a non re­entrant code?**

Re entrant code is code which does not rely on being executed without interruption before completion.Reentrant code can be used by multiple, simultaneous tasks. Reentrant code generally does not access globaldata. Variables within a reentrant function are allocated on the stack, so each instance of the function hasits own private data. Non­reentrant code, to be used safely by multiple processes, should have accesscontrolled via some synchronization method such as a semaphore.

**2. how  is RTOS different from  other OS?**

A RTOS offers services that allow tasks to be performed within predictable timing constraints

**3. Is unix a multitasking or multiprocessing operating system? whats the difference between the two?**

unix is a multitasking operating system, multiprocessing means it can run on multiple processors, themultiproceesing os coordinates with multiple processors running in parallel.

**4. what is a core dump?**

A core dump is the recorded state of the working memory of a computer program at a specific time, generallywhen the program has terminated abnormally includes the program counter and stack pointer, memorymanagement information, and other processor and operating system flags and information a fatal error usuallytriggers the core dump, often buffer overflows, where a programmer allocates too little memory for incoming or computeddata, or access to null pointers, a common coding error when an unassigned memory reference variable is accessed

5. what is stack  overflow and heap overflow?

stack overflow occurs when when the program tries to access memory that is outside the region reservedfor the call stack

call stack contains the subroutines called, the local variables

overflow occurs when too many functions are called,huge amount of local variables are allocated

6. windows also has multiple processes has process priotities switches between multiple process, how  RTOSis different from  that?

RTOS has predictable timing constranints

**7. how  will u create a process in UNIX or our OS OSE?**

We can use thr fork system call to create a process in UNIX and in OSE the system callcreate\_process is used.

8. what is a flat memory model and a shared memory model?

in a flatmemory model the code and data segment occupies single address space.

in a shared model the large memory is divided into different segments and needs a qualifier to identifyeach segment

in a flat memory model the programmer doesnt need to switch for data and code

**9. what is paging, segmentation y do we need it?**

Paging

Paging is a techinque where in the OS makes available the data required as quickly as possible. It storessome pages from the aux device to main memory and when a prog needs a page that is not on the mainmemory it fetches it from aux memory and replaces it in main memory. It uses specialised algorithms tochoose which page to replace from in main memory.

Caching

It deals with a concept where the data is temperorarily stored in a high speed memory for faster access.This data is duplicated in cache and the original data is stored in some aux memory. This concepts bringsthe average access time lower.

Segmentation

Segmentation is a memory management scheme. This is the technique used for memory protection. Anyaccesses outside premitted area would result in segmentation fault.

Virtual Memory

This technique enables non­contiguous memory to be accessed as if it were contiguous. Same as paging.

**10. write a code  to check whether a stack  grows upwards or downwards?**

void checkStack()

{

int i=2;

int j=3;

if(&i > &j) printf("stack grown downwards");

else printf("stack grows upwards");

}

define 2 local variables one after other and try to print the address

**11. why do we require semaphore?**

For process synchronization, it is a mechanism to invoke the sleeping process to become ready forexecution. Its mechanism where a process can wait for resources to be available.typical example is producer consumer process. The producer process creates resources and signals the semaphore sayingresource is available. Consumer process waiting on the semaphore gets the signal that resource is available.

**12. write a small piece of code  protecting a shared memory variable with a semaphore?**

int global\_i;

void increment\_shared\_memory

{

       wait(semaphore);

       global\_i++;

      signal(semaphore);

}

**13. what are the different types  of semaphores and where they are used?**

Binary semaphore and counting semaphore. Binary semaphore is same as mutex. Binary semaphore tries toprotect only one resource.

Counting semaphore is used in case of multiple resource. For ex: we have 4 printers then the countingsemaphore value will be init to 4. When it reaches 0, the task waiting on the semaphore is suspended.

**14. what are the different inter process communications?**

semaphore, mutex, message passing, shared memory, socket connections

**15. what is present in .bss**

The bss section contains uninitialized data, and is allocated at run­time.  Until it is written to, it remainszeroed

**16. what are the different segments of a program (code, data,stack,heap etc)**

**memory Segments**: Code segment

This phrase used to refer to a portion of memory or of an object file that contains executable

computer instructions. It is generally read­only segment.

data segment

This is one of the sections of a program in an object file or in memory, which contains the global variablesthat are initialized by the programmer. It has a fixed size, since all of the data in this section is set by theprogrammer before the program is loaded. However, it is not read­only, since the values of the variablescan be altered at runtime.

.bss

This segment of memory is part of data segment that contains uninitialized data (static variables). Theseare initialized to 0 and later assigned values during runtime.

stack segment

This segment of memory is a special stack which stores information about the active subroutines of a task. Itcontains the return address to be branched to after a sub­routine has finished execution. It contains localvariables of the sub­routines and the parameters that are passed to those sub­ routines.

heap segment

The segment of memory which is used for dynamic memory allocation is known as heap. It is theresponsibility of the programmer to deallocate it after its use. Or alternatively it will be garbage collectedby the OS.

**17. what is OS scheduling mechanism in our OSE?**

**18. how and wher the context related information is stored PCB i guess**

**19. what is a make command and what are al its uses?**

**20. what if no process responding how it is handled (watch dog timeout mechanism)**

**21. what is an ELF**

Executable and Linking Format is a common standard file format for executables, object code, sharedlibraries, and core dumps.

**22. what is priority inversion?**

In a scenario where a low priority task holds a shared resource (example semaphore) that is required by ahigh priority task. This causes the execution of the high priority task to be blocked

until the low priority task has released the resource. This scenario is averted by the OS by increasing thepriority of the low­proi process until it completes the task and releases the resources.

**23. Locality of reference**:It deals with a process accessing a resource multiple times. There are three typeof localities namely

temporal ­ A resource is referenced at point in time and will be referenced again in near future. spatial ­The concept that the likelihood of referencing a resource is higher if a resource near it has beenreferenced.

sequencial ­ The concept of accessing the memory sequentially.

**SHORT NOTES ON INTERRUPT HANDING**

1) An interrupt is generated by some HW source or SW source (timer or software event).

2) CPU invokes the kernal interrupt handler.

3) Kernal interrupt handler invokes the vector handler which returns the vector number. (Vector handler has themapping from vector number to interrupt source).

4) Kernal invokes the mask handler which masks all existing equal or low priority interrupts.

5) In interrupt routine associated with the vector number is invoked.

6) Kernal invokes the mask handler again to restore old mask.

**THREAD POOLING**

Thread pool is a collection of managed threads usually organized in a queue, which execute the tasks in thetask queue.

Creating a new thread object every time you need something to be executed asynchronously is expensive. In athread pool you would just add the tasks you wish to be executed asynchronously to the task queue

and the thread pool takes care of assigning an available thread, if any, for the corresponding task. As soon as thetask is completed, the the now available thread requests another task (assuming there is any left).

**DIFFERENT TYPES OF MEMORY FRAGMENTATION**

In computer storage, fragmentation is a phenomenon in which storage space is used inefficiently, reducingstorage capacity and in most cases performance. The term is also used to

**RTOS VS GENERAL PURPOSE OS**

The biggest difference is determinacy. An RTOS will have a deterministic scheduler. For any givenset of tasks your process will always execute every number of microseconds or milliseconds exactly,and the same number from schedule to schedule. operating system services consume only known andexpected amounts of time

In UNIX or Windows the scheduler are usually soft­real­time (as opposed to some hard­real­ time RTOS).Soft­real­time means that the scheduler tries to assure your process runs every X number ofmilliseconds, but may fail to do so on some occasions.

Modern RTOSs simply make sure that a) no interrupt is ever lost, and b) no interrupt can be blocked by a lowerpriority process.  
  
//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////  
1) What is RTOS and why do we go for it?  
       RTOS sheludes execution in a timely manner,manages system resources and provides a consistent foundation for developing application code.  
     Application code designed for RTOS could be ranging from simple digital stopwatch to complex aircraft navigation systems.  
RTOS differs from the GPOS by  
  Task scheduling : RTOS like vxWorks,neclueos, uC/OSII uses a strict scheduling algorithms (like pre-emptive scheduling) that makes the tasks meet their deadline to get the job done.  
  The sheduler decides which task run when by knowing its priorities.Whereas In a GPOS, the scheduler typically uses a "fairness" policy that offers no assurances that high-priority, time-critical tasks will execute in preference to lower-priority tasks.This policy is maintained to obtain the overall output by the desktop pc or server pc.  
  
**2) What are the different types of semaphores in vxworks RTOS? Which is the fastest?**  
       VxWorks supports three types of semaphores.  
 1) Binary 2) mutual exclusion, and 3)counting semaphores.  
 Fastest : Binary semaphore.  
  
  
         **3. What is priority inversion ? why priority inversion occurs?**  
 Tasks in RTOS's communucates between them via IPC source semphores for synchornization.  
     The situation priority inversion occurs when a higher priority taks is ready to run but it is waiting for a shared resource that is to be released by a lower priority task currently running .Now higher-priority task is pending on the resourceand it has to wait until the lower priority task to finish. **4. What are the solutions for priority inversion ?**  
 1) Priority inheritance  
    2) Priority ceiling  
 **5. What is super loop ?**  
Super loop is the infinite loop that runs all the time because , most of the embedded systems has no OS in it to return to application.

//Super loop example

while(true)

{

//all other code

}

//or

for(;;)

{

}

//or

label:

//This is is tricky assembly version

goto label:

**6) What is the difference between Structure and union? Where do we use union?**  
  
        When a structure is defined the compiler allocates memory for all of its member variables with necessary alignment .Whereas for unions the compiler allocates memory for the highest the union is equal to the biggest size of member from the union member variable list.union can only store information in one field at any one time. 

#include <stdio.h>

struct mStruct

{

char name[10];

int age;

float height;

};

union mUnion

{

char name[15];

int age;

float height;

};

int main(void)

{

union mUnion uTest;

struct mStruct sTest;

strcpy(sTest.name,"sTest");

sTest.age = 20;

sTest.height = 6.1;

strcpy(uTest.name ,"uTest");

uTest.height = 6.0;

uTest.age = 20;

printf("\n");

printf("sizeof(uTest) = %d ,sizeof(sTest) = %d \n",sizeof(uTest),sizeof(sTest));

printf("uTest.age = %d , uTest.height = %f , uTest.name = %s\n",uTest.age, uTest.height, uTest.name);

printf("sTest.age = %d , sTest.height = %f , sTest.name = %s\n",sTest.age, sTest.height, sTest.name);

printf("\n");

}

**7) Convert number 2 --> 37 without using any other operator but bitwise ?** 

Answer:

int x = 2;

printf("X before :%d\n",x);

x = (x<<x<<x) | (x<<x<<x) | x<<!!x | !!x ;

printf("X After :%d\n",x);

**8) Set , Get , Clear ,Toggle , Display Bits ?** 

// Display Every bits in a int number

void displayBits(int data)

{

int dataSize = 1<<sizeof(data);

int count = 0;

for (count = dataSize;count >= 0; count--)

{

printf("%d",(testBit(data,count)));

}

}

// test a bit if it is zero or one

int8\_t testBit(int8\_t value,int whichBit)

{

int mask = 1 << whichBit;

if (value&mask)

{

return TRUE;

}

else return FALSE;

}

// Set a bit to one

int8\_t setBit(int8\_t result, int whichBit)

{

return (result |= (1<<whichBit));

}

// toggle a bit

int8\_t toggleBit(int8\_t result, int whichBit)

{

return (result ^= (1<<whichBit));

}

/\* Clear a bit to zero \*/

int8\_t clearBit(int8\_t result, int whichBit)

{

return (result &=~ (1<<whichBit));

}

/////////////////////////////////////////////////////////////////////////////////////////////

 this link provides you to learn more

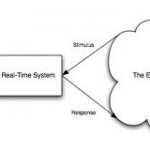
**http://interviewquestionsanswers.org/\_RTOS**

**http://www.allinterview.com/interview-questions/217/rtos.html**

**http://www.globalguideline.com/interview\_questions/Questions.php?sc=Real-Time\_Operating\_System\_RTOS**

**/////////////////////////////////////////////////////////////////////////////////////////////**

In this post you will get know about [Real-time system](http://blog.oureducation.in/real-time-systems/) and its entire scenario. This will prove helpful to you for your Interview purpose as well as for your academic preparation. You can also download the Real-Time systems Interview question in PDF format by clicking the link provided at the end of the Article. we always look for serve better. Please put your suggestions and comments in the box provided below the Article. Wish you All the Best for your Preparation!!

[](http://blog.oureducation.in/wp-content/uploads/2013/01/images-33.jpg)

Real Time systems

**Q1:  What do you mean by a real-time system?**  
**Ans:** A real-time system is one that must process information and produce a response within a specified time, else risk serve consequences, including failure. That is, in a system with a real-time constraint it is no good to have the correct action or the correct answer after a certain deadline: it is either by the deadline or it is useless.

We can also say that any information processing activity or system which has to respond to externally generated input stimuli within a finite and specified period.

**Q2: Discuss issues in real-time system scenario.**  
**Ans:** Most important issues regarding real-time systems are:

* Recovering from Failures.
* Working with distributed Architectures.
* Asynchronous Communication
* Race Conditions and Timing.
* Real Time response.

**Q3: What is an Embedded system? Differentiate between embedded system and real-time system.**

**Ans:** An [embedded system](http://blog.oureducation.in/embedded-systems-based-questions-and-answers/) is some combination of computer hardware and software, either fixed in capability or programmable, that is specifically designed for a particular function.

Embedded System Example

OR

It can be defined as “A specialized computer system that is part of a larger system or machine”.

Embedded systems are the ones found in generally immutable machines, such as ATMs, internet kiosks, airport terminal displays, cellphones,or at the screen at McD that displays your order are all examples of embedded systems. Real-time systems are the ones that are designed to provide a result within a specific time-frame. If you are using a touch-screen to order a sandwich from a gas store chain, you don’t want to have to wait 20 seconds for it to display pictures of each of the ingredients. You want it “now”, or at least within a second or two of pushing the button.

**Q4: Explain real-time communications.**  
**Ans:** Real-time communications (RTC) is any mode of telecommunications in which all users can exchange information instantly or with negligible latency. In this context, the term “real-time” is synonymous with “live.”

RTC can take place in half-duplex or full duplex modes.In half-duplex RTC, data can be transmitted in both directions on a single carrier or circuit but not at the same time. In full-duplex RTC, data can be transmitted in both directions simultaneously on a single carrier or circuit. RTC generally refers to peer-to-peer communications, not broadcast or multicast.

Real-time communications can include:

* Telephony in the conventional sense
* Mobile and cellular telephone.
* IM (instant messaging)
* VoIP (Voice over IP, also called Internet telephone).
* Live videoconference communications.

**Q5: Define Hard and Soft real-time system.**  
**Ans:** A hard real-time system (also known as an immediate real-time system) is hardware or software that must operate within the confines of a stringent deadline. The application may be considered to be failed if it does not complete its function within the allotted time span. Examples of hard real-time systems include components of pacemakers, anti-lock brakes and aircraft control systems.

A soft real-time system is a system where a critical real-time task gets priority over other tasks and retains that priority until it completes. As in hard real time systems, kernel delays need to be bounded.

**Q6:** **Draw structure or block diagram of Real time system OR the components of the RTS.**  
**Ans:** Schematic block diagram of a Real time system.

Real-Time Systems

**Q7. Describe Trigger Generator?**  
**Ans:** The “Trigger generator” is a representation at the mechanism used to trigger the execution of individual jobs. It is not really a separate hardware unit, typically it is a part of an executive software. Many of the jobs are periodic i.e. they execute regularly. The schedule for these jobs can be obtained offline and loaded as a look up table to be used by the scheduler.

**Q8. Define real time database and give its types.**  
**Ans:** A real-time database system is a database system in which a timely response to a user request is needed.

**Types of Real-Time Database Systems:**

• Hard real-time database systems, e.g., safety-critical system such as an early warning system, etc.  
• Soft real-time database systems, e.g., banking system,  
airline reservation system, digital library, stock market  
system, etc.  
• Mixed real-time database systems, e.g., air traffic  
control system, etc.

**Q.9. Explain all types of task classes in real time system?**  
**Ans:** There are five types of task classes:

**(i)** Periodic and aperiodic tasks  
**(ii)** Sporadic task  
**(iii)**Critical task  
**(iv)**Noncritical task

**(1) Periodic task:** There are many tasks in real — time systems that are done repeatitively. For example one may wish to monitor the speed altitude and attitude of an aircraft every 100 ms. This sensor information will be used by periodic tasks that control surfaces of the aircraft  in order to maintain stability and other desired characteristics. The periodicity of these tasks is known to the designer, and much tasks can be pre-scheduled.

**(2) Aperiodic task:** There are many other tasks that are aperiodic, that occur occasionally. For instance, when the pilot wishes to execute a turn a large number of subtasks. Associated with that action are self off aperiodic tasks cannot be predicted and sufficient completing power must be held in a reserve to execute them in a timely fashion.

**(3) Critical tasks:** Critical tasks are those whose timely executions is critical; if deadlines are missed, catastrophes occur. Example include life – support systems and the stability control of air craft. If critical tasks are executed at a higher frequency then it is absolutely necessary.

**(4) Non critical tasks:** Non critical tasks are real times tasks. As the name implies, they are not critical to the application. However they do deal with time varying data and hence they are useless if not completed within a deadline. The goal in scheduling these tasks is to maximize the percentage of jobs successfully executed within their deadlines.

**Q10: Define TargetOS.**  
**Ans:** TargetOS is a full-featured real-time operating system (RTOS) from Blunk Microsystems designed specifically for embedded applications. TargetOS is fast, small, and preemptive. To help reduce your time to market, TargetOS is integrated with development tools and off-the-shelf board support packages. Custom board support packages and drivers are also available.  
Benefits:

* Royalty Free
* Source Code
* Integrated with TargetTools
* Integrated Event Trace Tool
* Board Support Packages and Device Drivers.

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////  
RTOS has become very important as it is concerned with time critical events. Okay, let me list down few of the famous RTOS questions that can be discussed here.   
  
1. What is priority inversion ?  
2. What are the solutions for priority inversion ?  
3. What is priority inheritance ?  
4. What is priority ceiling ?  
5. What is deadlock ?  
6. What is the famous diners problem ?  
7. What is mutex ?  
8. What is spinlock ?  
9. Where are spinlocks used ?  
10. What do you mean by atomic operations ?  
11. what is a semaphore ?  
12. What are the types of semaphore ?  
13. What is binary semaphore ?  
14. What is a counting semaphore ?  
15. What is message queue ?  
16. What is the role of a scheduler ? How does it function ?  
17. What is the difference between a normal OS and RTOS ?  
18. What is preemption ?  
19. What is preemptive multi-tasking/time-sharing ? What is its difference with co-operative multi-tasking/time-sharing ?  
20. Threads are described as lightweight because switching between threads does not involve changing the memory context - True/False ?  
21.What are the factors considered for a RTOS selection ?  
22. What is the use of the method of temporarily masking / disabling interrupts ? When is it used ? What should be taken care while doing this method ?  
23. Since, disabling/masking of interrupts can be done whent the critical section is ONLY SHORT,What method can be used if the critical section is longer than few source lines or if it involves few lengthy loopings ?  
24. Difference between semaphores and disabling/masking of interrupts method ?  
25. Binary semaphore is equivalent to Mutex - True/False. How ?  
26. How can you avoid deadlocks incase of semaphore based designs ?  
27. What is Message passing method ? What is its advantages ?  
28. Tell about the design of Interrupt Handler and Scheduler in RTOS ?  
29. What is interrupt latency ?  
30. Even if we never enables interrupts in the code, the processor automatically disables them often during hardware access - True/False ? In this case how to reduce interrupt latency ?  
31. When should we re-enable the interrupts in an ISR and why ?  
32. How do you measure the latency of your system ?  
33. What are First Level Interrupt handlers and Second level Interrupt handlers ?  
34. What could be the typical design/implementation of FLIH and SLIH ?  
35. Reentrant interrupt handlers might cause a stack overflow from multiple preemptions by the same interrupt vector - True / False ?  
36. What kind of memory allocation procedure is good for embedded systems ?  
37. Is there any RTOS that has non-preemptive scheduling ?  
38. What is reentrant code ?  
39. What is preemptive multitasking ?  
40. What does timeslice refer to ?  
41. If the time slice is too short then the scheduler will consume too much of processing time - True / False  
42. What is I/O bound ? What is CPU bound ?  
43. What is non-preemptive multitasking ?  
44. CFS uses 'nanosecond' granularity accounting, the atomic units by which individual process share the CPU instead of previous notion of 'timeslice' - True/False .  
45. When will you use binary semaphore ?  
46. When will you choose busy-wait instead of context switch ?  
47. What are the possible scenarios in which context switching of threads can occur ?  
48. Can you use mutex/semaphore inside an ISR ?  
49. Explain a scenari that could cause deadlock ? What is the best solution for a deadlock ?  
50. Will the performance of your application improve if it has only a single thread and it is running on multiple cores of a processor ?  
51. What will happen if there are more threads requesting for CPU resource such as time ?  
52. What is Gang Scheduling and how is it useful ?  
53. Can you sleep in interrupt handler ?  
54. What is the main drawback for not considering Linux as realtime / RTOS ?  
55. What is the drawback in using semaphore for synchronization ? How does spinlock help in overcoming it ?  
56. What does a semaphore consist of ? and What does a spinlock consist of ?  
57. Why spinlocks are useless in uniprocessor systems ?  
58. What is timeslice ?  
59. What is the difference between multiprogramming and multiprocessing ?  
60. What is parallel programming ?  
61. What are the types of IPC mechanisms ?  
62. What are the types of synchronization problems and what are the resources that can cause such problems ?  
63. What is data race ?  
64. What is Indefinite Postponement / Indefinite blocking or starvation ?  
65. What are the synchronization relationships that are present in a multithreaded or mulitprogramming applications ?  
66. How Many Processes or Threads Are Enough for an application ?  
67. Tell the advantages and disadvantages of Co-operative multitasking.  
67. When should we use mutex and when should we use semaphore ?  
  
 for answers see this link  
  
**http://embedded-telecom-interview.blogspot.in/2010/06/rtos-interview-questions.html**  
  
**///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////**

**Major Concerns of selecting RTOS**

1) Interuppt latency  
2) Footprint (size of the executable which is generated  
after compiling)  
3) Context switching time is also considered as vital  
element in selection

**Linux and Real Time**

Linux is built as a general-purpose multiuser operating system. General-purpose operating systems are tuned to maximize average throughput even at the expense of latency, while real-time operating systems attempt to minimize, and place an upper bound on, latency, sometimes at the expense of average throughput. There are several reasons why standard Linux is not suitable for real-time  
use:

* **Non-preemptive kernel procedure** – This is a fancy way of saying that kernel system calls are not preemptible. Once a process enters the kernel, it can’t be preempted until it’s ready to exit the kernel. If an event occurs while the kernel is executing, the process waiting for that event **can’t be scheduled until the currently executing process exits the kernel**. Some kernel calls, fork() for example, can hold off preemption for tens of milliseconds.
* **Paging** – The process of swapping pages in and out of virtual memory is, for all practical purposes, **unbounded**. We have no way of knowing how long it will take to get a page off a disk drive and so we simply can’t place an upper bound on the time a process may be delayed due to a page fault.
* **Fairness in Scheduling** – Reflecting its Unix heritage as a multi-user time-sharing system, the conventional Linux scheduler does its best to be fair to all processes. Thus, the scheduler may give the processor to a low-priority process that has been waiting a long time even though a higher-priority process is ready to run.
* **Request Reordering** – Linux reorders I/O requests from multiple processes to make more efficient use of hardware. For example, hard disk block reads from a lower priority process may be given precedence over read requests from a higher priority process in order to minimize disk head movement or improve chances of error recovery.
* **Batching** – Linux will batch operations to make more efficient use of resources. For example, instead of freeing one page at a time when memory gets tight, Linux will run through the list of pages, clearing out as many as possible, delaying the execution of all processes.

**Major Concerns of selecting RTOS**

1) Interuppt latency  
2) Footprint (size of the executable which is generated  
after compiling)  
3) Context switching time is also considered as vital  
element in selection

**Linux and Real Time**

Linux is built as a general-purpose multiuser operating system. General-purpose operating systems are tuned to maximize average throughput even at the expense of latency, while real-time operating systems attempt to minimize, and place an upper bound on, latency, sometimes at the expense of average throughput. There are several reasons why standard Linux is not suitable for real-time  
use:

* **Non-preemptive kernel procedure** – This is a fancy way of saying that kernel system calls are not preemptible. Once a process enters the kernel, it can’t be preempted until it’s ready to exit the kernel. If an event occurs while the kernel is executing, the process waiting for that event **can’t be scheduled until the currently executing process exits the kernel**. Some kernel calls, fork() for example, can hold off preemption for tens of milliseconds.
* **Paging** – The process of swapping pages in and out of virtual memory is, for all practical purposes, **unbounded**. We have no way of knowing how long it will take to get a page off a disk drive and so we simply can’t place an upper bound on the time a process may be delayed due to a page fault.
* **Fairness in Scheduling** – Reflecting its Unix heritage as a multi-user time-sharing system, the conventional Linux scheduler does its best to be fair to all processes. Thus, the scheduler may give the processor to a low-priority process that has been waiting a long time even though a higher-priority process is ready to run.
* **Request Reordering** – Linux reorders I/O requests from multiple processes to make more efficient use of hardware. For example, hard disk block reads from a lower priority process may be given precedence over read requests from a higher priority process in order to minimize disk head movement or improve chances of error recovery.
* **Batching** – Linux will batch operations to make more efficient use of resources. For example, instead of freeing one page at a time when memory gets tight, Linux will run through the list of pages, clearing out as many as possible, delaying the execution of all processes.

**Major Concerns of selecting RTOS**

1) Interuppt latency  
2) Footprint (size of the executable which is generated  
after compiling)  
3) Context switching time is also considered as vital  
element in selection

**Linux and Real Time**

Linux is built as a general-purpose multiuser operating system. General-purpose operating systems are tuned to maximize average throughput even at the expense of latency, while real-time operating systems attempt to minimize, and place an upper bound on, latency, sometimes at the expense of average throughput. There are several reasons why standard Linux is not suitable for real-time  
use:

* **Non-preemptive kernel procedure** – This is a fancy way of saying that kernel system calls are not preemptible. Once a process enters the kernel, it can’t be preempted until it’s ready to exit the kernel. If an event occurs while the kernel is executing, the process waiting for that event **can’t be scheduled until the currently executing process exits the kernel**. Some kernel calls, fork() for example, can hold off preemption for tens of milliseconds.
* **Paging** – The process of swapping pages in and out of virtual memory is, for all practical purposes, **unbounded**. We have no way of knowing how long it will take to get a page off a disk drive and so we simply can’t place an upper bound on the time a process may be delayed due to a page fault.
* **Fairness in Scheduling** – Reflecting its Unix heritage as a multi-user time-sharing system, the conventional Linux scheduler does its best to be fair to all processes. Thus, the scheduler may give the processor to a low-priority process that has been waiting a long time even though a higher-priority process is ready to run.
* **Request Reordering** – Linux reorders I/O requests from multiple processes to make more efficient use of hardware. For example, hard disk block reads from a lower priority process may be given precedence over read requests from a higher priority process in order to minimize disk head movement or improve chances of error recovery.
* **Batching** – Linux will batch operations to make more efficient use of resources. For example, instead of freeing one page at a time when memory gets tight, Linux will run through the list of pages, clearing out as many as possible, delaying the execution of all processes.

===========================================================================

**Q. In RTOS having 1okbytes of memory and your program for asking for 5kbytes and you got null error? why does it behaves like this?**  
RTOS might be having 10k memory, might not fragmentation problem, and the largest hole available to user might be lesser than 5k, this condition might be one of the reason for NULL error.

**Q. What is the need of creating 4GB of pages in Linux?**  
Well, the simple answer is it can address so much of memory with available 32 address lines. 2 ^ 32 = 4 GB.

**Q. What are the rules you follow when you are writing critical section of code?**  
a)Use Atomic Instructions  
b) Remember to enable interrupts  
c) Make the critical section code as small as possible. (Prefer not more than 20 instructions)  
d) Prefer not to call other functions from the critical   
if u r calling, see that there is no critical  section in the other function too. Critical section is  bounded by Disable Interrupt and Enable Interrupt.  
Check the example below.  
fnA()  
{  
/\* Critical Section Start \*/  
Disable\_Interrupt();  
Some Instructions A ….  
Call FnB();  
/\* do Something B \*/  
Some Instructions B ….  
/\* Critical Section End \*/  
}

fnB()  
{  
/\* Critical Section Start \*/  
Disable\_Interrupt();  
Some Instructions ..   
Enable\_Interrupts();  
/\* Critical Section End \*/  
}

Now the Enable\_Interrupts in fnB() will enable the  interrupts and hence “Some Instructions B ..” in fnA()  which should have been in critical section will no more be  in critical section because the interrupts are already  enabled!!  
Please check if this condition is handled by the Enable and  Disable functions. If you want suggestions on how to solve  this problem, do revert back

[**Checkout Kernel RTOS Development Tutorials**](https://mindmajix.com/kernel-rtos-development)

**Q. What are the advantages and disadvantages of winCE compared to GPOS?  
Advantages**:  
1) Supports various types of processor platforms  
2) Possible to Customize the Kernel and can reduce its size (upto ~300KB kernel size)  
3) Especially for the embedded device platforms

**Disadvantages:**  
1) Process limits, max. 32 process at a time in WinCE 5.0  
2) Limited space allocation for each processes, 32MB/each process

**Q. Why MFC is not Supporting in Smartphones,This is also winCE mobile then why?**  
Well there is nothing to related with limited H/w I believe, Because MFC is just a collection classes designed using the Windows APIs , to provide ease of use.  So the proper answer may Be… To design any application on mobile.. We have very limited APIs which user can directly used. Generally in Embedded system (even in Soft RTOS also ) , if application design using less ( comfort )layer like MFC ( and other if it is there :) ),then your aplication would be more efficient. That’s why even today’s era… The application written in Assembly is the most efficient application then others.

**Q. What is the difference between winCE4.0 and winCE5.0 Is there any specialty?**  
Speciality in the sense …  
1) number of processes supported by wince 5.0 is more  than wince 4.0  
2) memory mapping might be varying.

**Q. What are the levels in winCE architecture?**  
There are four levels in Win CE architecture.  
They are   
1)Hardware layer  
2)OEM layer  
3)OS layer  
4)Application layer

**Q. What are the SDKs developed by winCE?**  
MSFT standard SDK’s are,  
1. Windows Mobile for Smartphone = Windows Mobile Standard.  
2. Windows Mobile for Pocket PC = Windows Mobile Classic.  
3. Windows Mobile for Pocket PC Phone Edition = Windows Mobile Professional.

**Q. What is the difference between normal OS and winCE OS?**  
Normal OS wince   
drivers runs under User space(part of device manager)  
kernel space  
cannot be customized customized for embedded application  
for embedded application like mobile phones,thin client…  
(exception linux)  
kernel – monolithic kernel- micro kernel

Explore Kernel RTOS Development Sample Resumes! Download & Edit for Free..! [**Download Now!**](https://mindmajix.com/kernel-rtos-development-sample-resumes)

**Q. What is CE Stands in winCCE?**  
In the name “Windows CE,” the letters “CE” are not an  abbreviation for anything, but rather they imply a number  of the precepts around which Windows CE is designed,  including “Compact,” Connectable,”  Compatible,” “Companion,” and “Efficient.”

====================================================================

**Q. First of all, tell us a little bit about yourself and your responsibilities at On Time Software**.

A. My name is Peter Petersen and I am the managing director of On Time. I started this company back in 1989, which makes On Time one of the more persistent players in the embedded systems market. After having worked for several years as a software engineer and consultant, I started On Time 20 years with RTKernel, a real-time kernel for MSDOS. In 1996, our current principal product On Time RTOS-32 was launched.

**Q. If you would, please give us a very brief, bulleted outline of your products. What sorts of real-time operating systems (RTOSes), tools, and/or services does your company offer?**

A. On Time’s main product is On Time RTOS-32, a real-time OS for 32-bit x86 embedded systems. On Time RTOS-32 is a modular OS with 6 main components. Two of these (RTUSB-32, a USB host stack, and RTFiles-32, a file system) are also sold separately as they can easily be ported to other platforms.

**Q. What is On Time Software’s “unique value proposition” for the embedded systems engineer or programmer who is considering an embedded RTOS? What do you and your products do to help him get his product to market faster, cheaper, better?**

A. Ease of use is the main feature, which makes our customers get their products out the door faster than with other systems. Developers can work with one of the best available IDEs: Microsoft Visual Studio. On Time RTOS-32 can be very small (16k RAM/ROM), but it can also be very comprehensive and contains everything a developer needs from an OS. Nevertheless, it can boot in less than 1 second.

**Q. How are you different as a company from competitors? What sets your products apart from those of other RTOS companies?**

A. A combination of many factors distinguishes us from others. “Ease of use”, as mentioned above, makes developers more productive and less time is lost with learning yet another development environment. Our support for the Win32 API allows many software components originally developed for Windows to be reused on embedded systems. The INSIDERS’ GUIDE EMBEDDED RTOS: PRODUCT INTERVIEWS full source code of On Time RTOS-32 is available and technical support is free with less than 24-hour turn-around times guaranteed.

**Q. What embedded architectures do you support - e.g., Intel architecture, MIPS, ARM, PowerPC, etc.?**

A. On Time RTOS-32 runs on any 32/64-bit x86 CPU, including the Intel Atom CPU family. Being specialized on the x86 architecture, we take advantage of all of its unique features. For example, we support memory protection, running at different privilege levels, SSE, multi-processor/multi-core, etc. For targets with a BIOS (anything which can boot DOS or Windows), On Time RTOS-32 runs out-of-the-box with no customization and contains drivers for all commonly used PC peripherals. We also support targets without a BIOS and ship several examples of target-specific chip-set initializations required for such targets.

**Q. What additional software do you offer such as networking, file systems, TCP/IP, security, IDE, GUIs etc.? What about development tools? Are there particular partnerships with other software companies that are especially helpful?**

A. On Time RTOS-32 includes a FAT/CD-ROM file system, TCP/IP stack, USB host stack, and an object-oriented, Windows look-and-feel GUI. For development tools, we support Microsoft Visual Studio (including its integrated debugger), Borland Delphi, and Borland C/C++. For many years now, we have been working together with EBSnet Inc. (http://www.ebsnetinc.com/) and Swell Software (http://www.swellsoftware.com/) for our network and GUI components. We have recently entered new partnership agreements with acontis (http://www.acontis.com/) and Real-Time Systems (http://www.real-time-systems.com/) to support their respective software solutions to run an RTOS in parallel with Windows on the same computer.

**=====================================**

**1) Explain the main purpose of an operating system?**

Operating systems exist for two main purposes. One is that it is designed to make sure a computer system performs well by managing its computational activities. Another is that it provides an environment for the development and execution of programs.

**2) What is demand paging?**

Demand paging is referred when not all of a process’s pages are in the RAM, then the OS brings the missing(and required) pages from the disk into the RAM.

**3) What are the advantages of a multiprocessor system?**

With an increased number of processors, there is a considerable increase in throughput. It can also save more money because they can share resources. Finally, overall reliability is increased as well.

**4) What is kernel?**

A kernel is the core of every operating system. It connects applications to the actual processing of data. It also manages all communications between software and hardware components to ensure usability and reliability.

**5) What are real-time systems?**

Real-time systems are used when rigid time requirements have been placed on the operation of a processor. It has well defined and fixed time constraints.

**6) What is a virtual memory?**

Virtual memory is a memory management technique for letting processes execute outside of memory. This is very useful especially is an executing program cannot fit in the physical memory.

**7) Describe the objective of multiprogramming.**

The main objective of multiprogramming is to have a process running at all times. With this design, CPU utilization is said to be maximized.

**8 ) What is time- sharing system?**

In a Time-sharing system, the CPU executes multiple jobs by switching among them, also known as multitasking. This process happens so fast that users can interact with each program while it is running.

**9) What is SMP?**

SMP is a short form of Symmetric Multi-Processing. It is the most common type of multiple-processor systems. In this system, each processor runs an identical copy of the operating system, and these copies communicate with one another as needed.

**10) How are server systems classified?**

Server systems can be classified as either computer-server systems or file server systems. In the first case, an interface is made available for clients to send requests to perform an action. In the second case, provisions are available for clients to create, access and update files.

**11) What is asymmetric clustering?**

In asymmetric clustering, a machine is in a state known as hot standby mode where it does nothing but to monitor the active server. That machine takes the active server’s role should the server fails.

**12) What is a thread?**

A thread is a basic unit of CPU utilization. In general, a thread is composed of a thread ID, program counter, register set, and the stack.

**13) Give some benefits of multithreaded programming.**

– there is increased responsiveness to the user  
– resource sharing within the process  
– economy  
– utilization of multiprocessing architecture

**14) Briefly explain FCFS.**

FCFS stands for First-come, first-served. It is one type of scheduling algorithm. In this scheme, the process that requests the CPU first is allocated the CPU first. Implementation is managed by a FIFO queue.

**15) What is RR scheduling algorithm?**

RR (round-robin) scheduling algorithm is primarily aimed for time-sharing systems. A circular queue is a setup in such a way that the CPU scheduler goes around that queue, allocating CPU to each process for a time interval of up to around 10 to 100 milliseconds.

**16) What are necessary conditions which can lead to a deadlock situation in a system?**

Deadlock situations occur when four conditions occur simultaneously in a system: Mutual exclusion; Hold and Wait; No preemption; and Circular wait.

**17) Enumerate the different RAID levels.**

RAID 0 – Non-redundant striping  
RAID 1 – Mirrored Disks  
RAID 2 – Memory-style error-correcting codes  
RAID 3 – Bit-interleaved Parity  
RAID 4 – Block-interleaved Parity  
RAID 5 – Block-interleaved distributed Parity  
RAID 6 – P+Q Redundancy

**18) Describe Banker’s algorithm**

*[](https://career.guru99.com/wp-content/uploads/2012/05/Bankers_Algorithm.gif)*

*Bankers Algorithm*

Banker’s algorithm is one form of deadlock-avoidance in a system. It gets its name from a banking system wherein the bank never allocates available cash in such a way that it can no longer satisfy the needs of all of its customers.

**19) What factors determine whether a detection-algorithm must be utilized in a deadlock avoidance system?**

One is that it depends on how often a deadlock is likely to occur under the implementation of this algorithm. The other has to do with how many processes will be affected by deadlock when this algorithm is applied.

**20) State the main difference between logical from physical address space.**

Logical address refers to the address that is generated by the CPU. On the other hand, physical address refers to the address that is seen by the memory unit.

**21) How does dynamic loading aid in better memory space utilization?**

With dynamic loading, a routine is not loaded until it is called. This method is especially useful when large amounts of code are needed in order to handle infrequently occurring cases such as error routines.

**22) What are overlays?**

Overlays are used to enable a process to be larger than the amount of memory allocated to it. The basic idea of this is that only instructions and data that are needed at any given time are kept in memory.

**23) What is the basic function of paging?**

Paging is a memory management scheme that permits the physical address space of a process to be noncontiguous. It avoids the considerable problem of having to fit varied sized memory chunks onto the backing store.

**24) What is fragmentation?**

Fragmentation is memory wasted. It can be internal if we are dealing with systems that have fixed-sized allocation units, or external if we are dealing with systems that have variable-sized allocation units.

**25) How does swapping result in better memory management?**

During regular intervals that are set by the operating system, processes can be copied from main memory to a backing store, and then copied back later. Swapping allows more operations to be run that can fit into memory at one time.

**26) Give an example of a Process State.**

– New State – means a process is being created  
– Running – means instructions are being executed  
– Waiting – means a process is waiting for certain conditions or events to occur  
– Ready – means a process is waiting for an instruction from the main processor  
– Terminate – means a process is stopped abruptly

**27) What is a socket?**

A socket provides a connection between two applications. Each endpoint of a communication is a socket.

**28) What is Direct Access Method?**

Direct Access method is based on a disk model of a file, such that it is viewed as a numbered sequence of blocks or records. It allows arbitrary blocks to be read or written. Direct access is advantageous when accessing large amounts of information.

**29) When does thrashing occur?**

Thrashing refers to an instance of high paging activity. This happens when it is spending more time paging instead of executing.

**30) What is the best page size when designing an operating system?**

The best paging size varies from system to system, so there is no single best when it comes to page size. There are different factors to consider in order to come up with a suitable page size, such as page table, paging time, and its effect on the overall efficiency of the operating system.

**31) When designing the file structure for an operating system, what attributes are considered?**

Typically, the different attributes for a file structure are naming, identifier, supported file types, and location for the files, size, and level of protection.

**32) What is root partition?**

Root partition is where the operating system kernel is located. It also contains other potentially important system files that are mounted during boot time.

**33) What are device drivers?**

Device drivers provide a standard means of representing I/O devices that maybe manufactured by different companies. This prevents conflicts whenever such devices are incorporated in a systems unit.

**34) What are the primary functions of VFS?**

VFS, or Virtual File System, separate file system generic operations from their implementation by defining a clean VFS interface. It is based on a file-representation structure known as vnode, which contains a numerical designator needed to support network file systems.

**35) What are the different types of CPU registers in a typical operating system design?**

– Accumulators  
– Index Registers  
– Stack Pointer  
– General Purpose Registers

**36) What is the purpose of an I/O status information?**

I/O status information provides information about which I/O devices are to be allocated for a particular process. It also shows which files are opened, and other I/O device state.

**37) What is multitasking?**

Multitasking is the process within an operating system that allows the user to run several applications at the same time. However, only one application is active at a time for user interaction, although some applications can run “behind the scene”.

**38) Explain pros and cons of a command line interface?**

A command line interface allows the user to type in commands that can immediately provide results. Many seasoned computer users are well accustomed to using the command line because they find it quicker and simpler.

However, the main problem with a command line interface is that users have to be familiar with the commands, including the switches and parameters that come with it. This is a downside for people who are not fond of memorizing commands.

**39) What is caching?**

Caching is the processing of utilizing a region of fast memory for a limited data and process. A cache memory is usually much efficient because of its high access speed.

**40) What is spooling?**

Spooling is normally associated with printing. When different applications want to send an output to the printer at the same time, spooling takes all of these print jobs into a disk file and queues them accordingly to the printer.

**41) What is an Assembler?**

An assembler acts as a translator for low-level language. Assembly codes written using mnemonic commands are translated by the Assembler into machine language.

**42) What are interrupts?**

Interrupts are part of a hardware mechanism that sends a notification to the CPU when it wants to gain access to a particular resource. An interrupt handler receives this interrupt signal and “tells” the processor to take action based on the interrupt request.

**43) What is GUI?**

GUI is short for Graphical User Interface. It provides users with an interface wherein actions can be performed by interacting with icons and graphical symbols. People find it easier to interact with the computer when in a GUI especially when using the mouse. Instead of having to remember and type commands, users click on buttons to perform a process.

**44) What is preemptive multitasking?**

Preemptive multitasking allows an operating system to switch between software programs. This, in turn, allows multiple programs to run without necessarily taking complete control over the processor and resulting in system crashes.

**45) Why partitioning and formatting is a prerequisite to installing an operating system?**

Partitioning and formatting create a preparatory environment on the drive so that the operating system can be copied and installed properly. This includes allocating space on the drive, designating a drive name, determining and creating the appropriate file system and structure.

**46) What is plumbing/piping?**

It is the process of using the output of one program as an input to another. For example, instead of sending the listing of a folder or drive to the main screen, it can be piped and sent to a file, or sent to the printer to produce a hard copy.

**47) What is NOS?**

NOS is short for Network Operating System. It is a specialized software that will allow a computer to communicate with other devices over the network, including file/folder sharing.

**48) Differentiate internal commands from external commands.**

Internal commands are built-in commands that are already part of the operating system. External commands are separate file programs that are stored in a separate folder or directory.

**49) Under DOS, what command will you type when you want to list down the files in a directory, and at the same time pause after every screen output?**  
**a) dir /w  
b) dir /p  
c) dir /s  
d) dir /w /p**

Answer: d) dir /w /p

**50) How would a file name EXAMPLEFILE.TXT appear when viewed under the DOS command console operating in Windows 98?**

The filename would appear as EXAMPL~1.TXT . The reason behind this is that filenames under this operating system are limited to 8 characters when working under DOS environment.

**51) What is a folder in Ubuntu?**

There is no concept of Folder in Ubuntu. Everything included in your hardware is a FILE.

**52) Explain why Ubuntu is safe and not affected by viruses?**

* It does not support malicious e-mails and contents, and before any e-mail is opened by users it will go through many security checks
* Ubuntu uses Linux, which is a super secure O.S system
* Unlike other O.S, countless Linux users can see the code at any time and can fix the problem if there is any
* Malware and viruses are coded to take advantage of the weakness in Windows

**53) Explain what is Unity in Ubuntu? How can you add new entries to the launcher?**

In Ubuntu, Unity is the default graphical shell.  On the left side of the Ubuntu, it introduces the launcher and Dash to start programs.

In order to add new entries to the launcher, you can create a file name like **.desktop** and then drag the file on the launcher.

**54) Explain the purpose of using a libaio package in Ubuntu?**

Libaio is Linux Kernel Asynchronous I/O (A/O).  A/O allows even a single application thread to overlap I/O operations with other processing, by providing an interface for submitting one or more I/O requests in one system call without waiting for completion.  And a separate interface to reap completed I/O operations associated with a given completion group.

**55) What is the use of behavior tab in Ubuntu?**

Through behaviors tab, you can make many changes on the appearance of the desktop

* Auto-hide the launcher: You can use this option to reveal the launcher when moving the pointer to the defined hot spot.
* Enable workspaces:  By checking this option, you can enable workspace
* Add show desktop icon to the launcher: This option is used to display the desktop icon on the launcher

**56) What is the meaning of “export” command in Ubuntu?**

Export is a command in Bash shell language. When you try to set a variable, it is visible or exported to any subprocess started from that instance of bash.  The variable will not exist in the sub-process without the export command.

**57) Explain how you can reset Unity Configuration?**

To reset the unity configuration the simplest way to do is to hit open a Terminal or hit Atl-F2  and run the command # unity –reset

**What is a process and process table? What are different states of process**  
A *process*is an instance of program in execution. For example a Web Browser is a process, a shell (or command prompt) is a process.  
The operating system is responsible for managing all the processes that are running on a computer and allocated each process a certain amount of time to use the processor. In addition, the operating system also allocates various other resources that processes will need such as computer memory or disks. To keep track of the state of all the processes, the operating system maintains a table known as the *process table*. Inside this table, every process is listed along with the resources the processes is using and the current state of the process.  
*Processes can be in one of three states: running, ready, or waiting*. The running state means that the process has all the resources it need for execution and it has been given permission by the operating system to use the processor. Only one process can be in the running state at any given time. The remaining processes are either in a waiting state (i.e., waiting for some external event to occur such as user input or a disk access) or a ready state (i.e., waiting for permission to use the processor). In a real operating system, the waiting and ready states are implemented as queues which hold the processes in these states. The animation below shows a simple representation of the life cycle of a process (Source: <http://courses.cs.vt.edu/csonline/OS/Lessons/Processes/index.html>)

**What is a Thread? What are the differences between process and thread?**  
A thread is a single sequence stream within in a process. Because threads have some of the properties of processes, they are sometimes called *lightweight processes*. Threads are popular way to improve application through parallelism. For example, in a browser, multiple tabs can be different threads. MS word uses multiple threads, one thread to format the text, other thread to process inputs, etc.  
A thread has its own program counter (PC), a register set, and a stack space. Threads are not independent of one other like processes as a result threads shares with other threads their code section, data section and OS resources like open files and signals. See <http://www.personal.kent.edu/~rmuhamma/OpSystems/Myos/threads.htm> for more details.

**What is deadlock?**  
Deadlock is a situation when two or more processes wait for each other to finish and none of them ever finish.  Consider an example when two trains are coming toward each other on same track and there is only one track, none of the trains can move once they are in front of each other.  Similar situation occurs in operating systems when there are two or more processes hold some resources and wait for resources held by other(s).

**What are the necessary conditions for deadlock?**  
*Mutual Exclusion:* There is a resource that cannot be shared.  
*Hold and Wait:*A process is holding at least one resource and waiting for another resource which is with some other process.  
*No Preemption:* The operating system is not allowed to take a resource back from a process until process gives it back.  
*Circular Wait:*A set of processes are waiting for each other in circular form.

**What is Virtual Memory? How is it implemented?**  
Virtual memory creates an illusion that each user has one or more contiguous address spaces, each beginning at address zero. The sizes of such virtual address spaces is generally very high.  
The idea of virtual memory is to use disk space to extend the RAM. Running processes don’t need to care whether the memory is from RAM or disk. The illusion of such a large amount of memory is created by subdividing the virtual memory into smaller pieces, which can be loaded into physical memory whenever they are needed by a process.

**What is Thrashing?**  
Thrashing is a situation when the performance of a computer degrades or collapses. Thrashing occurs when a system spends more time processing page faults than executing transactions. While processing page faults is necessary to in order to appreciate the benefits of virtual memory, thrashing has a negative affect on the system. As the page fault rate increases, more transactions need processing from the paging device. The queue at the paging device increases, resulting in increased service time for a page fault (Source: h[ttp://cs.gmu.edu/cne/modules/vm/blue/thrash.html](http://cs.gmu.edu/cne/modules/vm/blue/thrash.html))

**What is Belady’s Anomaly?**  
Bélády’s anomaly is an anomaly with some page replacement policies where increasing the number of page frames results in an increase in the number of page faults. It occurs with First in First Out page replacement is used. See [the wiki page](http://en.wikipedia.org/wiki/B%C3%A9l%C3%A1dy's_anomaly)for an example and more details.

**Differences between mutex and semphore?**  
See <https://www.geeksforgeeks.org/mutex-vs-semaphore/>

* Practice [Quizzes](http://quiz.geeksforgeeks.org/quiz-corner/#Operating%20Systems%20Mock%20Tests) on Operating System topics
* [Last Minute Notes](http://quiz.geeksforgeeks.org/last-minute-notes-operating-systems/) – OS
* OS [articles](http://quiz.geeksforgeeks.org/gate-cs-notes/)

# **Operating Systems | Set 2**

Following questions have been asked in GATE CS exam.  
  
**1. Consider a machine with 64 MB physical memory and a 32-bit virtual address space. If the page size is 4KB, what is the approximate size of the page table? (GATE 2001)**(a) 16 MB  
(b) 8 MB  
(c) 2 MB  
(d) 24 MB

**Answer:** (c)  
**Explanation:**  
A page entry is used to get address of physical memory. Here we assume that single level of Paging is happening. So the resulting page table will contain entries for all the pages of the Virtual address space.

Number of entries in page table =

(virtual address space size)/(page size)

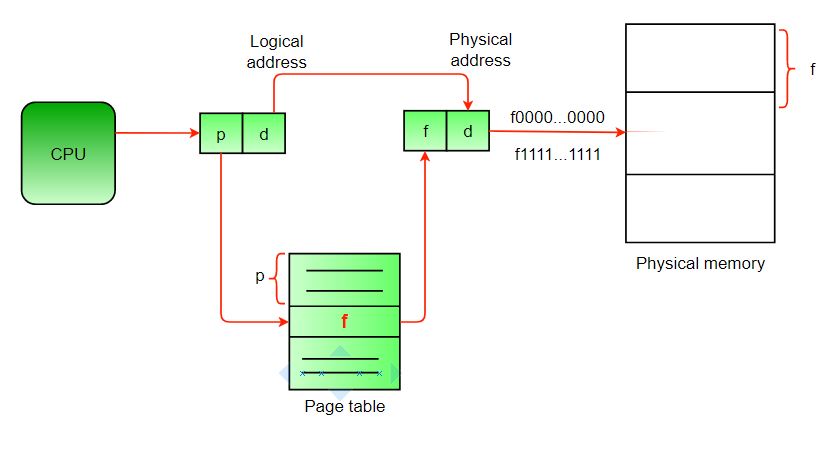
Using above formula we can say that there will be 2^(32-12) = 2^20 entries in page table.  
No. of bits required to address the 64MB Physical memory = 26.  
So there will be 2^(26-12) = 2^14 page frames in the physical memory. And page table needs to store the address of all these 2^14 page frames. Therefore, each page table entry will contain 14 bits address of the page frame and 1 bit for valid-invalid bit.  
Since memory is byte addressable. So we take that each page table entry is 16 bits i.e. 2 bytes long.

Size of page table =

(total number of page table entries) \*(size of a page table entry)

= (2^20 \*2) = 2MB

For the clarity of the concept, please see the following figure. As per our question, here p = 20, d = 12 and f = 14.



**2. Consider Peterson’s algorithm for mutual exclusion between two concurrent processes i and j. The program executed by process is shown below.**

**repeat**

**flag [i] = true;**

**turn = j;**

**while ( P ) do no-op;**

**Enter critical section, perform actions, then exit critical**

**section**

**flag [ i ] = false;**

**Perform other non-critical section actions.**

**until false;**

**For the program to guarantee mutual exclusion, the predicate P in the while loop should be (GATE 2001)**  
a) flag [j] = true and turn = i  
b) flag [j] = true and turn = j  
c) flag [i] = true and turn = j  
d) flag [i] = true and turn = i

**Answer:** (b)  
Basically, Peterson’s algorithm provides guaranteed mutual exclusion by using the two following constructs – flag[] and turn. flag[] controls that the willingness of a process to be entered in critical section. While turn controls the process that is allowed to be entered in critical section. So by replacing P with the following,

flag [j] = true and turn = j

process i will not enter critical section if process j wants to enter critical section and it is process j’s turn to enter critical section. The same concept can be extended for more than two processes. For details, refer the following.  
**References:**  
<http://en.wikipedia.org/wiki/Peterson%27s_algorithm>

**3 More than one word are put in one cache block to (GATE 2001)**  
(a) exploit the temporal locality of reference in a program  
(b) exploit the spatial locality of reference in a program  
(c) reduce the miss penalty  
(d) none of the above

**Answer:** (b)  
Temporal locality refers to the reuse of specific data and/or resources within relatively small time durations. Spatial locality refers to the use of data elements within relatively close storage locations.  
To exploit the spatial locality, more than one word are put into cache block.  
**References:**  
<http://en.wikipedia.org/wiki/Locality_of_reference>

**4. Which of the following statements is false? (GATE 2001)**  
a) Virtual memory implements the translation of a program’s address space into physical memory address space  
b) Virtual memory allows each program to exceed the size of the primary memory  
c) Virtual memory increases the degree of multiprogramming  
d) Virtual memory reduces the context switching overhead

**Answer:** (d)  
In a system with virtual memory context switch includes extra overhead in switching of address spaces.  
**References:**  
<http://www.itee.adfa.edu.au/~spike/CSA2/Lectures00/lecture.vm.htm>

**5. Consider a set of n tasks with known runtimes r1, r2, … rn to be run on a uniprocessor machine. Which of the following processor scheduling algorithms will result in the maximum throughput? (GATE 2001)**  
(a) Round-Robin  
(b) Shortest-Job-First  
(c) Highest-Response-Ratio-Next  
(d) First-Come-First-Served

**Answer:** (b)

# **Operating Systems | Set 3**

Following questions have been asked in GATE CS exam.  
  
**1. Suppose the time to service a page fault is on the average 10 milliseconds, while a memory access takes 1 microsecond. Then a 99.99% hit ratio results in average memory access time of (GATE CS 2000)**  
(a) 1.9999 milliseconds  
(b) 1 millisecond  
(c) 9.999 microseconds  
(d) 1.9999 microseconds

**Answer:** (d)  
**Explanation:**

Average memory access time =

[(% of page miss)\*(time to service a page fault) +

(% of page hit)\*(memory access time)]/100

So, average memory access time in microseconds is.  
(99.99\*1 + 0.01\*10\*1000)/100 = (99.99+100)/1000 = 199.99/1000 =1.9999 µs

**2. Which of the following need not necessarily be saved on a context switch between processes? (GATE CS 2000)**  
(a) General purpose registers  
(b) Translation look-aside buffer  
(c) Program counter  
(d) All of the above

**Answer:** (b)  
**Explanation:**  
In a process context switch, the state of the first process must be saved somehow, so that, when the scheduler gets back to the execution of the first process, it can restore this state and continue.

The state of the process includes all the registers that the process may be using, especially the program counter, plus any other operating system specific data that may be necessary.

A Translation lookaside buffer (TLB) is a CPU cache that memory management hardware uses to improve virtual address translation speed. A TLB has a fixed number of slots that contain page table entries, which map virtual addresses to physical addresses. On a context switch, some TLB entries can become invalid, since the virtual-to-physical mapping is different. The simplest strategy to deal with this is to completely flush the TLB.  
**References:**  
<http://en.wikipedia.org/wiki/Context_switch>  
<http://en.wikipedia.org/wiki/Translation_lookaside_buffer#Context_switch>

**3. Where does the swap space reside ? (GATE 2001)**  
(a) RAM  
(b) Disk  
(c) ROM  
(d) On-chip cache  
**Answer:** (b)  
**Explanation:**  
Swap space is an area on disk that temporarily holds a process memory image. When physical memory demand is sufficiently low, process memory images are brought back into physical memory from the swap area. Having sufficient swap space enables the system to keep some physical memory free at all times.  
**References:**  
<http://docs.hp.com/en/B2355-90672/ch06s02.html>

**4. Which of the following does not interrupt a running process?**(GATE CS 2001)  
(a) A device  
(b) Timer  
(c) Scheduler process  
(d) Power failure

**Answer:** (c)  
**Explanation:**  
Scheduler process doesn’t interrupt any process, it’s Job is to select the processes for following three purposes.  
Long-term scheduler(or job scheduler) –selects which processes should be brought into the ready queue  
Short-term scheduler(or CPU scheduler) –selects which process should be executed next and allocates CPU.  
Mid-term Scheduler (Swapper)- present in all systems with virtual memory, temporarily removes processes from main memory and places them on secondary memory (such as a disk drive) or vice versa. The mid-term scheduler may decide to swap out a process which has not been active for some time, or a process which has a low priority, or a process which is page faulting frequently, or a process which is taking up a large amount of memory in order to free up main memory for other processes, swapping the process back in later when more memory is available, or when the process has been unblocked and is no longer waiting for a resource.

**5. Which of the following scheduling algorithms is non-preemptive? (GATE CS 2002)**  
a) Round Robin  
b) First-In First-Out  
c) Multilevel Queue Scheduling  
d) Multilevel Queue Scheduling with Feedback

**Answer:**(b)

# **Operating System | Critical Section**

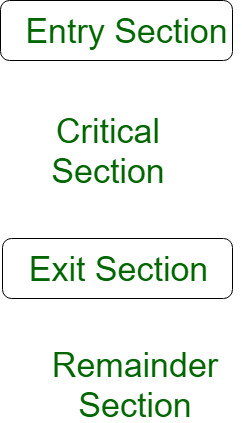
[**Critical Section**](http://en.wikipedia.org/wiki/Critical_section)**:**

In simple terms a critical section is group of instructions/statements or region of code that need to be executed atomically ([read this post](https://www.geeksforgeeks.org/g-fact-57/) for atomicity), such as accessing a resource (file, input or output port, global data, etc.).

In concurrent programming, if one thread tries to change the value of shared data at the same time as another thread tries to read the value (i.e. data race across threads), the result is unpredictable.

The access to such shared variable (shared memory, shared files, shared port, etc…) to be synchronized. Few programming languages have built-in support for synchronization.

It is critical to understand the importance of race condition while writing kernel mode programming (a device driver, kernel thread, etc.). since the programmer can directly access and modifying kernel data structures.



A simple solution to the critical section can be thought as shown below,

acquireLock();

Process Critical Section

releaseLock();

A thread must acquire a lock prior to executing a critical section. The lock can be acquired by only one thread. There are various ways to implement locks in the above pseudo code. Let us discuss them in future articles.

# **Operating Systems | Set 6**

Following questions have been asked in GATE 2011 CS exam.

**1) A thread is usually defined as a ‘light weight process’ because an operating system (OS) maintains smaller data structures for a thread than for a process. In relation to this, which of the followings is TRUE?**  
(A) On per-thread basis, the OS maintains only CPU register state  
(B) The OS does not maintain a separate stack for each thread  
(C) On per-thread basis, the OS does not maintain virtual memory state  
(D) On per thread basis, the OS maintains only scheduling and accounting information.

Answer (C)  
Threads share address space of Process. Virtually memory is concerned with processes not with Threads.

**2) Let the page fault service time be 10ms in a computer with average memory access time being 20ns. If one page fault is generated for every 10^6 memory accesses, what is the effective access time for the memory?**  
(A) 21ns  
(B) 30ns  
(C) 23ns  
(D) 35ns

Answer (B)

Let P be the page fault rate

Effective Memory Access Time = p \* (page fault service time) +

(1 - p) \* (Memory access time)

= ( 1/(10^6) )\* 10 \* (10^6) ns +

(1 - 1/(10^6)) \* 20 ns

= 30 ns (approx)

**3) An application loads 100 libraries at startup. Loading each library requires exactly one disk access. The seek time of the disk to a random location is given as 10ms. Rotational speed of disk is 6000rpm. If all 100 libraries are loaded from random locations on the disk, how long does it take to load all libraries? (The time to transfer data from the disk block once the head has been positioned at the start of the block may be neglected)**  
(A) 0.50s  
(B) 1.50s  
(C) 1.25s  
(D) 1.00s

Answer (B)  
Since transfer time can be neglected, the average access time is sum of average seek time and average rotational latency. Average seek time for a random location time is given as 10 ms. The average rotational latency is half of the time needed for complete rotation. It is given that 6000 rotations need 1 minute. So one rotation will take 60/6000 seconds which is 10 ms. Therefore average rotational latency is half of 10 ms, which is 5ms.

Average disk access time = seek time + rotational latency

= 10 ms + 5 ms

= 15 ms

For 100 libraries, the average disk access time will be 15\*100 ms

**4. Consider the following table of arrival time and burst time for three processes P0, P1 and P2.**

Process Arrival time Burst Time

P0 0 ms 9 ms

P1 1 ms 4 ms

P2 2 ms 9 ms

**The pre-emptive shortest job first scheduling algorithm is used. Scheduling is carried out only at arrival or completion of processes. What is the average waiting time for the three processes?**  
(A) 5.0 ms  
(B) 4.33 ms  
(C) 6.33 ms  
(D) 7.33 ms

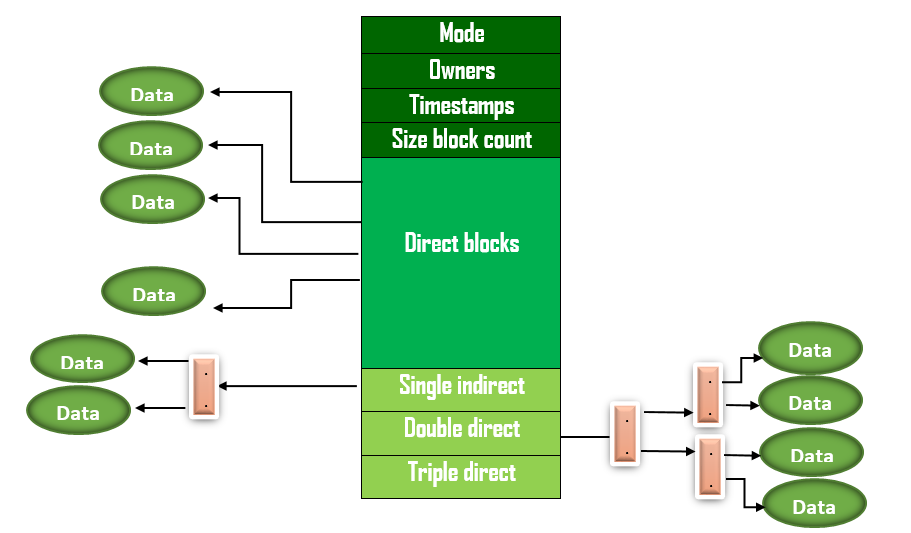
Answer: – (A)  
Process P0 is allocated processor at 0 ms as there is no other process in ready queue. P0 is preempted after 1 ms as P1 arrives at 1 ms and burst time for P1 is less than remaining time of P0. P1 runs for 4ms. P2 arrived at 2 ms but P1 continued as burst time of P2 is longer than P1. After P1 completes, P0 is scheduled again as the remaining time for P0 is less than the burst time of P2.  
P0 waits for 4 ms, P1 waits for 0 ms amd P2 waits for 11 ms. So average waiting time is (0+4+11)/3 = 5.

# **Operating Systems | Set 10**

Following questions have been asked in GATE 2008 CS exam.

**1) The data blocks of a very large file in the Unix file system are allocated using**  
(A) contiguous allocation  
(B) linked allocation  
(C) indexed allocation  
(D) an extension of indexed allocation

Answer (D)  
The Unix file system uses an extension of indexed allocation. It uses direct blocks, single indirect blocks, double indirect blocks and triple indirect blocks. Following diagram shows implementation of Unix file system.

[](https://cdncontribute.geeksforgeeks.org/wp-content/uploads/operatingsystem.png)  
  
  
  
**2) The P and V operations on counting semaphores, where s is a counting semaphore, are defined as follows:**

P(s) : s = s - 1;

if (s < 0) then wait;

V(s) : s = s + 1;

if (s <= 0) then wakeup a process waiting on s;

**Assume that Pb and Vb the wait and signal operations on binary semaphores are provided. Two binary semaphores Xb and Yb are used to implement the semaphore operations P(s) and V(s) as follows:**

P(s) : Pb(Xb);

s = s - 1;

if (s < 0) {

Vb(Xb) ;

Pb(Yb) ;

}

else Vb(Xb);

V(s) : Pb(Xb) ;

s = s + 1;

if (s <= 0) Vb(Yb) ;

Vb(Xb) ;

**The initial values of Xb and Yb are respectively**  
(A) 0 and 0  
(B) 0 and 1  
(C) 1 and 0  
(D) 1 and 1

Answer (C)  
Both P(s) and V(s) operations are perform Pb(xb) as first step. If Xb is 0, then all processes executing these operations will be blocked. Therefore, Xb must be 1.  
If Yb is 1, it may become possible that two processes can execute P(s) one after other (implying 2 processes in critical section). Consider the case when s = 1, y = 1. So Yb must be 0.

**3) Which of the following statements about synchronous and asynchronous I/O is NOT true?**  
(A) An ISR is invoked on completion of I/O in synchronous I/O but not in asynchronous I/O  
(B) In both synchronous and asynchronous I/O, an ISR (Interrupt Service Routine) is invoked after completion of the I/O  
(C) A process making a synchronous I/O call waits until I/O is complete, but a process making an asynchronous I/O call does not wait for completion of the I/O  
(D) In the case of synchronous I/O, the process waiting for the completion of I/O is woken up by the ISR that is invoked after the completion of I/O

Answer (B)  
An interrupt service routine will be invoked after the completion of I/O operation and it will place process from block state to ready state, *because* process performing I/O operation was placed in blocked state till the I/O operation was completed in **Synchronous I/O**.

However, process performing I/O will not be placed in the block state and process continues to execute the remaining instructions in **Asynchronous I/O**, *because* handler function will be registered while performing the I/O operation, when the I/O operation completed signal mechanism is used to notify the process that data is available.

So, option (B) is false.

# **Operating Systems | Set 7**

Following questions have been asked in GATE CS exam.  
 **1) Let the time taken to switch between user and kernel modes of execution be t1 while the time taken to switch between two processes be t2. Which of the following is TRUE? (GATE CS 2011)**  
(A) t1 > t2  
(B) t1 = t2  
(C) t1 < t2 (D) Nothing can be said about the relation between t1 and t2 Answer: - (C) Process switching involves mode switch. Context switching can occur only in kernel mode.   
  
  
**2) A system uses FIFO policy for page replacement. It has 4 page frames with no pages loaded to begin with. The system first accesses 100 distinct pages in some order and then accesses the same 100 pages but now in the reverse order. How many page faults will occur? (GATE CS 2010)**  
(A) 196  
(B) 192  
(C) 197  
(D) 195

Answer (A)  
Access to 100 pages will cause 100 page faults. When these pages are accessed in reverse order, the first four accesses will not cause page fault. All other access to pages will cause page faults. So total number of page faults will be 100 + 96.

**3) Which of the following statements are true? (GATE CS 2010)  
I. Shortest remaining time first scheduling may cause starvation  
II. Preemptive scheduling may cause starvation  
III. Round robin is better than FCFS in terms of response time**  
(A) I only  
(B) I and III only  
(C) II and III only  
(D) I, II and III

Answer (D)

I) Shortest remaining time first scheduling is a preemptive version of shortest job scheduling. It may cause starvation as shorter processes may keep coming and a long CPU burst process never gets CPU.  
II) Preemption may cause starvation. If priority based scheduling with preemption is used, then a low priority process may never get CPU.  
III) Round Robin Scheduling improves response time as all processes get CPU after a specified time.

**4) Consider the methods used by processes P1 and P2 for accessing their critical sections whenever needed, as given below. The initial values of shared boolean variables S1 and S2 are randomly assigned.**

**Method Used by P1**

while (S1 == S2) ;

Critica1 Section

S1 = S2;

**Method Used by P2**

while (S1 != S2) ;

Critica1 Section

S2 = not (S1);

**Which one of the following statements describes the properties achieved? (GATE CS 2010)**  
(A) Mutual exclusion but not progress  
(B) Progress but not mutual exclusion  
(C) Neither mutual exclusion nor progress  
(D) Both mutual exclusion and progress

Answer (A)  
It can be easily observed that the Mutual Exclusion requirement is satisfied by the above solution, P1 can enter critical section only if S1 is not equal to S2, and P2 can enter critical section only if S1 is equal to S2.  
Progress Requirement is not satisfied. Let us first see definition of Progress Requirement.  
Progress Requirement: If no process is executing in its critical section and there exist some processes that wishes to enter their critical section, then the selection of the processes that will enter the critical section next cannot be postponed indefinitely.  
If P1 or P2 want to re-enter the critical section, then they cannot even if there is other process running in critical section.

# **Operating Systems | Set 8**

Following questions have been asked in GATE 2009 CS exam.

**1) In which one of the following page replacement policies, Belady’s anomaly may occur?**  
(A) FIFO  
(B) Optimal  
(C) LRU  
(D) MRU

Answer (A)  
Belady’s anomaly proves that it is possible to have more page faults when increasing the number of page frames while using the First in First Out (FIFO) page replacement algorithm.  
See the [wiki page](http://en.wikipedia.org/wiki/B%C3%A9l%C3%A1dy's_anomaly) for an example of increasing page faults with number of page frames.

**2) The essential content(s) in each entry of a page table is / are**  
(A) Virtual page number  
(B) Page frame number  
(C) Both virtual page number and page frame number  
(D) Access right information

Answer (B)  
A page table entry must contain Page frame number. Virtual page number is typically used as index in page table to get the corresponding page frame number. See [this](http://codex.cs.yale.edu/avi/os-book/OS8/os8e/slide-dir/PDF-dir/ch8.pdf)for details.

**3) Consider a system with 4 types of resources R1 (3 units), R2 (2 units), R3 (3 units), R4 (2 units). A non-preemptive resource allocation policy is used. At any given instance, a request is not entertained if it cannot be completely satisfied. Three processes P1, P2, P3 request the sources as follows if executed independently.**

**Process P1:**

t=0: requests 2 units of R2

t=1: requests 1 unit of R3

t=3: requests 2 units of R1

t=5: releases 1 unit of R2

and 1 unit of R1.

t=7: releases 1 unit of R3

t=8: requests 2 units of R4

t=10: Finishes

**Process P2:**

t=0: requests 2 units of R3

t=2: requests 1 unit of R4

t=4: requests 1 unit of R1

t=6: releases 1 unit of R3

t=8: Finishes

**Process P3:**

t=0: requests 1 unit of R4

t=2: requests 2 units of R1

t=5: releases 2 units of R1

t=7: requests 1 unit of R2

t=8: requests 1 unit of R3

t=9: Finishes

**Which one of the following statements is TRUE if all three processes run concurrently starting at time t=0?**  
(A) All processes will finish without any deadlock  
(B) Only P1 and P2 will be in deadlock.  
(C) Only P1 and P3 will be in a deadlock.  
(D) All three processes will be in deadlock

Answer (A)  
We can apply the following Deadlock Detection algorithm and see that there is no process waiting indefinitely for a resource. See [this](http://codex.cs.yale.edu/avi/os-book/OS8/os8c/slide-dir/PPTX-dir/ch7.pptx) for deadlock detection algorithm.

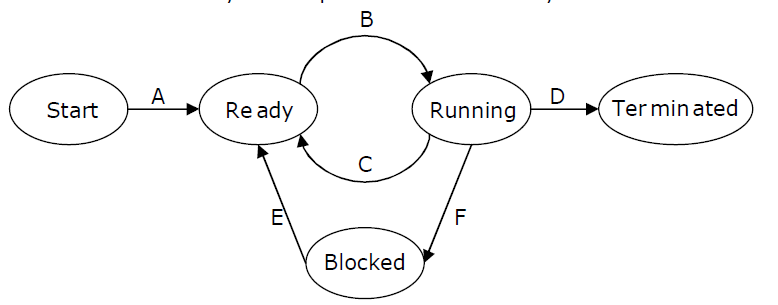
**4) Consider a disk system with 100 cylinders. The requests to access the cylinders occur in following sequence:  
4, 34, 10, 7, 19, 73, 2, 15, 6, 20  
Assuming that the head is currently at cylinder 50, what is the time taken to satisfy all requests if it takes 1ms to move from one cylinder to adjacent one and shortest seek time first policy is used?**  
(A) 95ms  
(B) 119ms  
(C) 233ms  
(D) 276ms

Answer (B)  
4, 34, 10, 7, 19, 73, 2, 15, 6, 20  
Since shortest seek time first policy is used, head will first move to 34. This move will cause 16\*1 ms. After 34, head will move to 20 which will cause 14\*1 ms. And so on. So cylinders are accessed in following order 34, 20, 19, 15, 10, 7, 6, 4, 2, 73 and total time will be (16 + 14 + 1 + 4 + 5 + 3 + 1 + 2 + 2 + 71)\*1 = 119 ms.

**Please see**[**GATE Corner**](http://geeksquiz.com/gate-corner-2/)**for all previous year paper/solutions/explanations, syllabus, important dates, notes, etc.**

# **Operating Systems | Set 9**

Following questions have been asked in GATE 2009 CS exam.

**1) In the following process state transition diagram for a uniprocessor system, assume that there are always some processes in the ready state: Now consider the following statements:  
[](https://www.geeksforgeeks.org/wp-content/uploads/gate2009.png)  
I. If a process makes a transition D, it would result in another process making transition A immediately.  
II. A process P2 in blocked state can make transition E while another process P1 is in running state.  
III. The OS uses preemptive scheduling.  
IV. The OS uses non-preemptive scheduling.  
Which of the above statements are TRUE?**  
(A) I and II  
(B) I and III  
(C) II and III  
(D) II and IV

Answer (C)  
I is false. If a process makes a transition D, it would result in another process making transition B, not A.  
II is true. A process can move to ready state when I/O completes irrespective of other process being in running state or not.  
III is true because there is a transition from running to ready state.  
IV is false as the OS uses preemptive scheduling.

**2) The enter\_CS() and leave\_CS() functions to implement critical section of a process are realized using test-and-set instruction as follows**:

void enter\_CS(X)

{

while test-and-set(X) ;

}

void leave\_CS(X)

{

X = 0;

}

**In the above solution, X is a memory location associated with the CS and is initialized to 0. Now consider the following statements:  
I. The above solution to CS problem is deadlock-free  
II. The solution is starvation free.  
III. The processes enter CS in FIFO order.  
IV More than one process can enter CS at the same time.**  
**Which of the above statements is TRUE?**  
(A) I only  
(B) I and II  
(C) II and III  
(D) IV only

Answer (A)  
The above solution is a simple [test-and-set](http://en.wikipedia.org/wiki/Test-and-set) solution that makes sure that deadlock doesn’t occur, but it doesn’t use any queue to avoid starvation or to have FIFO order.

**3) A multilevel page table is preferred in comparison to a single level page table for translating virtual address to physical address because**  
(A) It reduces the memory access time to read or write a memory location.  
(B) It helps to reduce the size of page table needed to implement the virtual address space of a process.  
(C) It is required by the translation lookaside buffer.  
(D) It helps to reduce the number of page faults in page replacement algorithms.

Answer (B)  
The size of page table may become too big (See [this](http://dysphoria.net/OperatingSystems1/4_multilevel_paging.html)) to fit in contiguous space. That is why page tables are typically divided in levels.

# **Operating Systems | Set 13**

Following questions have been asked in GATE CS 2007 exam.

**1) A virtual memory system uses First In First Out (FIFO) page replacement policy and allocates a fixed number of frames to a process. Consider the following statements:  
P: Increasing the number of page frames allocated to a process sometimes increases the page fault rate.  
Q: Some programs do not exhibit locality of reference. Which one of the following is TRUE?**  
(A) Both P and Q are true, and Q is the reason for P  
(B) Both P and Q are true, but Q is not the reason for P.  
(C) P is false, but Q is true  
(D) Both P and Q are false.

Answer (B)  
P is true. Increasing the number of page frames allocated to process may increases the no. of page faults (See [Belady’s Anomaly](http://en.wikipedia.org/wiki/B%C3%A9l%C3%A1dy's_anomaly)).  
Q is also true, but Q is not the reason for-P as Belady’s Anomaly occurs for some specific patterns of page references.

**2) A single processor system has three resource types X, Y and Z, which are shared by three processes. There are 5 units of each resource type. Consider the following scenario, where the column alloc denotes the number of units of each resource type allocated to each process, and the column request denotes the number of units of each resource type requested by a process in order to complete execution. Which of these processes will finish LAST?**

alloc request

X Y Z X Y Z

P0 1 2 1 1 0 3

P1 2 0 1 0 1 2

P2 2 2 1 1 2 0

(A) P0  
(B) P1  
(C) P2  
(D) None of the above, since the system is in a deadlock

Answer (C)  
Once all resources (5, 4 and 3 instances of X, Y and Z respectively) are allocated, 0, 1 and 2 instances of X, Y and Z are left. Only needs of P1 can be satisfied. So P1 can finish its execution first. Once P1 is done, it releases 2, 1 and 3 units of X, Y and Z respectively. Among P0 and P2, needs of P0 can only be satisfied. So P0 finishes its execution. Finally, P2 finishes its execution.

**3) Two processes, P1 and P2, need to access a critical section of code. Consider the following synchronization construct used by the processes:Here, wants1 and wants2 are shared variables, which are initialized to false. Which one of the following statements is TRUE about the above construct?**

/\* P1 \*/

while (true) {

wants1 = true;

while (wants2 == true);

/\* Critical

Section \*/

wants1=false;

}

/\* Remainder section \*/

/\* P2 \*/

while (true) {

wants2 = true;

while (wants1==true);

/\* Critical

Section \*/

wants2 = false;

}

/\* Remainder section \*/

(A) It does not ensure mutual exclusion.  
(B) It does not ensure bounded waiting.  
(C) It requires that processes enter the critical section in strict alternation.  
(D) It does not prevent deadlocks, but ensures mutual exclusion.

Answer (D)

The above synchronization constructs don’t prevent deadlock. When both wants1 and wants2 become true, both P1 and P2 stuck forever in their while loops waiting for each other to finish.

**4) Consider the following statements about user level threads and kernel level threads. Which one of the following statement is FALSE?**  
(A) Context switch time is longer for kernel level threads than for user level threads.  
(B) User level threads do not need any hardware support.  
(C) Related kernel level threads can be scheduled on different processors in a multi-processor system.  
(D) Blocking one kernel level thread blocks all related threads.

Answer (D)  
Since kernel level threads are managed by kernel, blocking one thread doesn’t cause all related threads to block. It’s a problem with user level threads. See [this](http://www.personal.kent.edu/~rmuhamma/OpSystems/Myos/threads.htm)for more details.