RTS Assignment.

Shreeja
161640

1. What are the common misconceptions about Real Time Systems?

→ Real Time Systems are systems intended to solve real time ~~apptying~~ application that process data as it comes in, mostly without buffer delay.

The common misconceptions about RTS are as follows:

i) Real-time systems are synonymous with "fast" systems.

ii) Rate-monotonic analysis has solved the real-time problem.

iii) There are universal, widely accepted methodologies for real-time systems specification and design.

iv) There is no more a need to build a real-time operating system because many commercial products exists.

v) The study of real-time systems is mostly about scheduling theory.

The first misconception that real time system must be fast, arises from the fact that many hard-real time systems indeed deal with deadlines. But deadlines are not particularly fast in some real time systems, but satisfying them determines the success or failure of the system.

The second misconception is that rate monotonic systems provide recipe for building real-time systems. While they do provide valuable guidance in the design of real-time system, and there is abundant theory

surrounding them, they are not panacea.

About third misconception, there are no universally accepted and infallible methods for specification and design of real-time systems. This is not a failure of researchers or the software industry, but is because of the difficulty of discovering universal solution for this demanding field.

The fourth misconception is that there is no need to build a real-time os from scratch. While there is a number of ~~ways to build~~ cost-efficient, popular and viable commercial real-time os, these are not panacea.

Finally, while it is scholarly to study scheduling theory, from an engineering standpoint, most published result require impractical simplification and clairvoyance in order to make theory work.

---

2. Discuss whether the following are hard, soft, or firm real time systems.

→

a) ~~A library of congress print manuscript database system~~

→ A library of congress printing manuscript database is a soft real time system. It is because the use of this system as a database for research limits its

urgency. It seems it would be acceptable for missed deadlines to degrade the system performance

b) A police database that provides information on stolen automobiles.

→ A police database that provides information on stolen automobiles is a firm real time system. Though, being a database limits its urgency but it ~~is~~ ~~be~~ system it is being used for is a more time critical system and only missing occasional deadline would be tolerable.

c) Automatic teller machine

→ An automatic teller machine is a firm real time system since money, both the banks and the users is involved and it would be used by the user standing at the machine. Anything more than an occasional missed deadline would be unacceptable.

d) A coin operated video game.

→ A coin operated video game can be considered as soft real time system since missing deadlines only degrade the performance of the system.

Shreeja
161640

3. How are real time systems different from non-real time systems? Explain with examples.

→ A real time system is one whose logical correctness is based on both the correctness of the outputs and their timeliness. It is a computer system that must satisfy ~~one or more~~ bounded response time constraints or risk severe consequences including failure. Eg: flight control system, real time monitors etc.

The realtime system can be soft, hard or firm in nature. Soft realtime system is one in which performance is degraded but not destroyed by failure to meet response-time constraint. A hard realtime is one in which failure to meet even a single deadline may lead to catastrophic system failure and a firm real time system is one in which few missed deadlines will not lead to total failure, but missing more than few may lead to complete or catastrophic system failure.

A non-realtime system is one in which ~~the~~ we cannot guarantee the response time of a task. They are non-deterministic i.e the arrival of event can't be determined. Thus, they are only used for general purpose. For eg: personal computer, work station, etc.

Shreeja
161640

The major difference between real time and non real time system are :-

Real time system.	Non-real time system
→ It is deterministic.	→ It is not deterministic.
→ It is time sensitive	→ It is time insensitive
→ It can't use virtual memory.	→ It can use concept of virtual memory
→ It is dedicated to single work.	→ It is used in multi-user environment
→ It has flat memory model.	→ It has protected memory model
→ It has low interrupt latency.	→ It has high interrupt latency.

4. What is the working mechanism of Petri Net.

→ Petri Nets are class of formal methods used in specifying and analyzing concurrent operation in real-time systems. It can produce executable specifications and are particularly suitable for modeling synchronization among tasks.

Petri Nets can be described graphically as interconnection of only two basic entities.

→ A set of circles called 'places' to represent data stores or conditions.

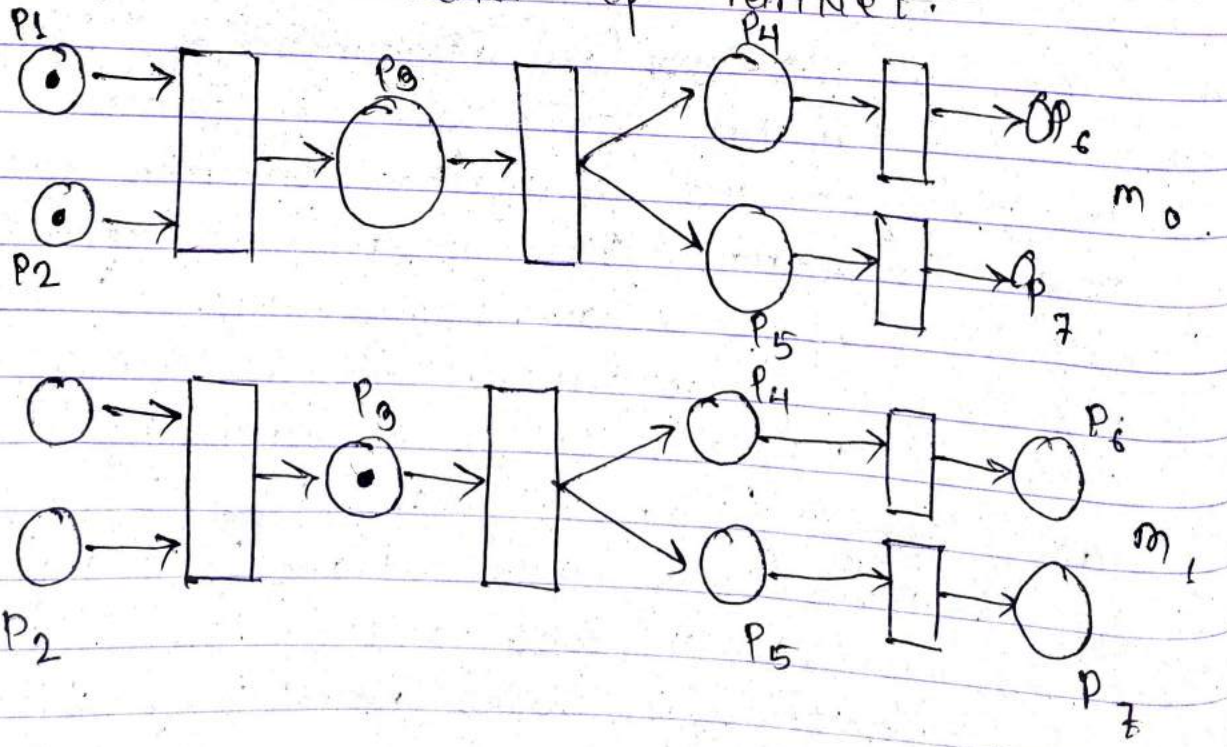
→ Rectangular boxes representing transitions or events.

Shreeja
161640

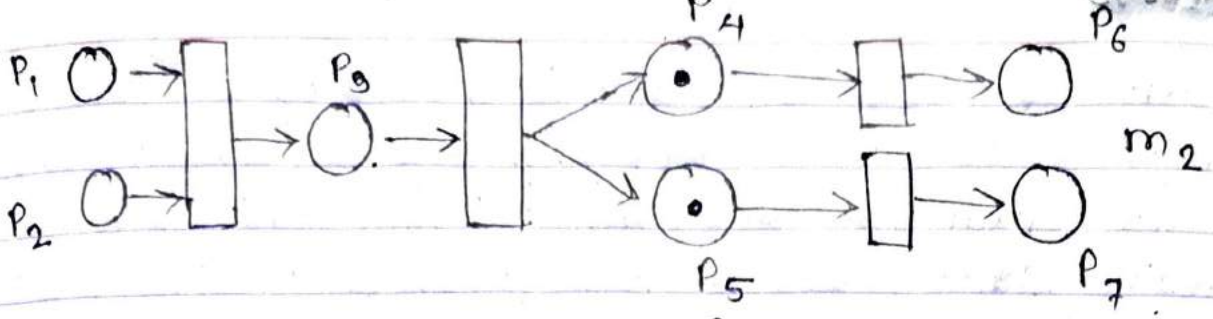
Each place (P) and transition (T) is labeled with data count and transition function. respectively and are connected by unidirectional arrows.

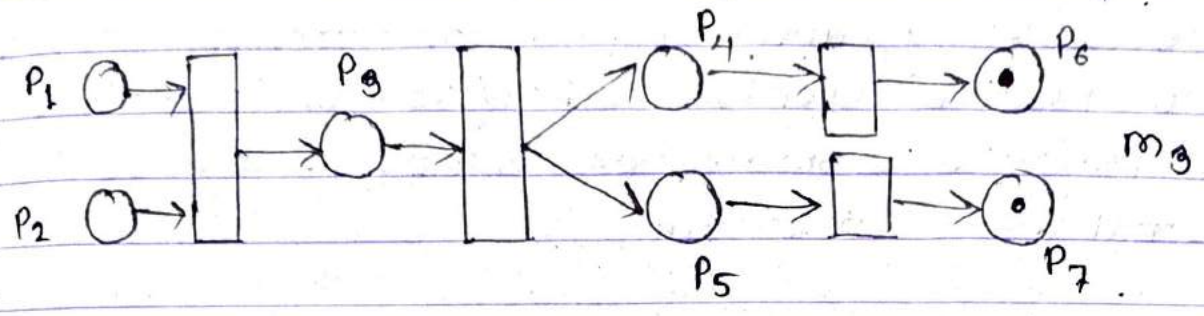
The initial Petri net graph is marked as m_0 which represents initial data count in all the places. Subsequent marking (m_i), $i \in \{1, 2, 3\}$ are results of firing transaction. Transaction fires if it has as many input data as required for producing associated output. In case of Petri net with subset that has two transactions with possibility to fire, only one of them will fire. PNs helps to identify and analyze race conditions and deadlocks and also in avoiding them.

Eg: sequential behavior of PetriNet.



Shreeja
161640

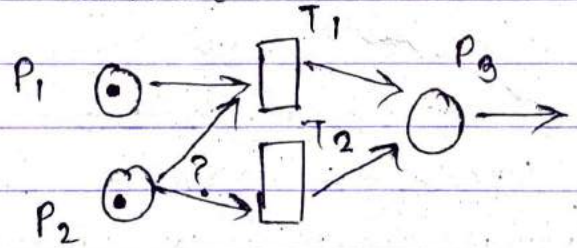


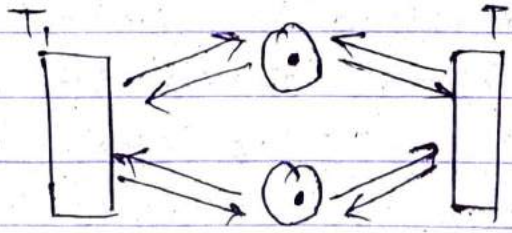


Firing table of above PetriNet.

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
m_0	1	1	0	0	0	0	0
m_1	0	0	1	0	0	0	0
m_2	0	0	0	1	1	0	0
m_3	0	0	0	0	0	1	1

Eg 2: Identification of race condition and deadlock with PetriNet.





Shreeja
161640

5. Differentiate between aperiodic, periodic and sporadic tasks of a real time system with examples for the example systems inertial measurement, nuclear plant monitoring, airline reservation, pasta bottling, and traffic light control, enumerate some possible events. and note whether they are periodic, sporadic or episodic. Discuss reasonable response time.

→

i) Aperiodic task.

→ Real time systems are required to respond to external events which occur at random instants of time, the system executes a set of job in response to these events. The release time of these jobs are not known until the event triggering them occurs.

Eg: ~~cyclic code~~, garbage collection, typical branch instruction

ii) Periodic task.

→ Periodic tasks are time driven i.e, the characteristics are known in advance and they execute at periodical instants of time. Every task T_i is characterized by (p_i, e_i) where p_i is task period and e_i is the worst case execution time, P_i is the minimum time period between job release.

→ Eg: cyclic code, processed schedule by internal clock.

Shreeja
161640

iii) Sporadic tasks

→ Sporadic tasks are the jobs with hard deadlines. These jobs execute to accomplish the mode change. Every task is characterized by arrival time and deadline.

- Eg: pilot disengaging the autopilot system arbitrarily making system changes ~~for~~ from cruise mode to standby mode in a transient fault tolerance.

In inertial measurement system of aircraft. The software will receive x, y and z accelerometer pulses at 10 ms rate. The software will determine the acceleration components in each direction, and corresponding roll, pitch and yaw. It is periodic in nature.

In monitoring system for a nuclear power plant that will be handling 3 event signals. First event is triggered by any of several signals at various security points which will indicate a security breach. System must respond within a ~~m~~ second. Second important ~~one~~ task is to check if reactors core temperature has reaches an over-temperature. It must be dealt with within 1 milliseconds (1 ms). Finally, an operator's displayed to be ~~at~~ at 30 m/s. This is also periodic system. But dealing with over-temperature is sporadic

Shreeja
161640

The airlines reservation system has to prevent long lines and customer dissatisfaction. So, TAT must be less that 15 seconds, and no overbooking will be permitted. At any time, several travel agent may try to access the reservation database and perhaps book the same flight simultaneously.

The bottling of jars of pasta sauce is controlled as they travel over a conveyer belt. The empty jars are disinfected and each jar is filled with precise service of specific sauce if it passes beneath. There are numerous events triggered by exceptional condition. Thus, both synchronous and asynchronous conditions are dealt.

System used to control a set of traffic light at intersection takes input from camera, emergency, vehicles etc. The traffic lights occurs in ~~po~~ synchronized fashion, and yet react to asynchronous events - such as a pedestrian pressing button at crosswalk.

Shreeja
161640

6. What is the role of kernel in ~~real time~~ operating systems? Explain different types of kernels.

→ Kernel is a computer ~~system~~ program that is the heart of an operating system that facilitates interaction between hardware and software components. The kernel has complete control over the system performing tasks like running processes, managing hardware devices such as hard disks and handling interrupts

The kernel in RTOS must provide three specific functions with respects to tasks.

i) Scheduling
   - A scheduler determines which task will run next in a multitasking system.

ii) Dispatching.
   - A dispatcher performs the necessary book keeping to start that task.

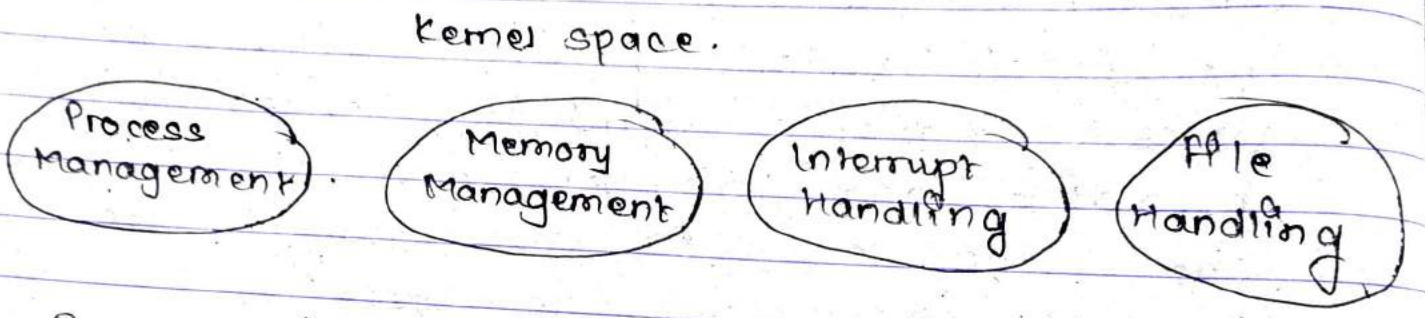
iii) Intercommunication and synchronization.
   - Inter-task communication and synchronization assure that tasks cooperates.

The kernels can mainly be classified into ~~two~~ three types: Monolithic kernel
: Micro kernel
: hybrid kernel.

Shreeja
161640

i) Monolithic kernel.

→ In this type of kernel architecture, all the functions like process management, memory management, interrupt handling etc. are performed in kernel space.

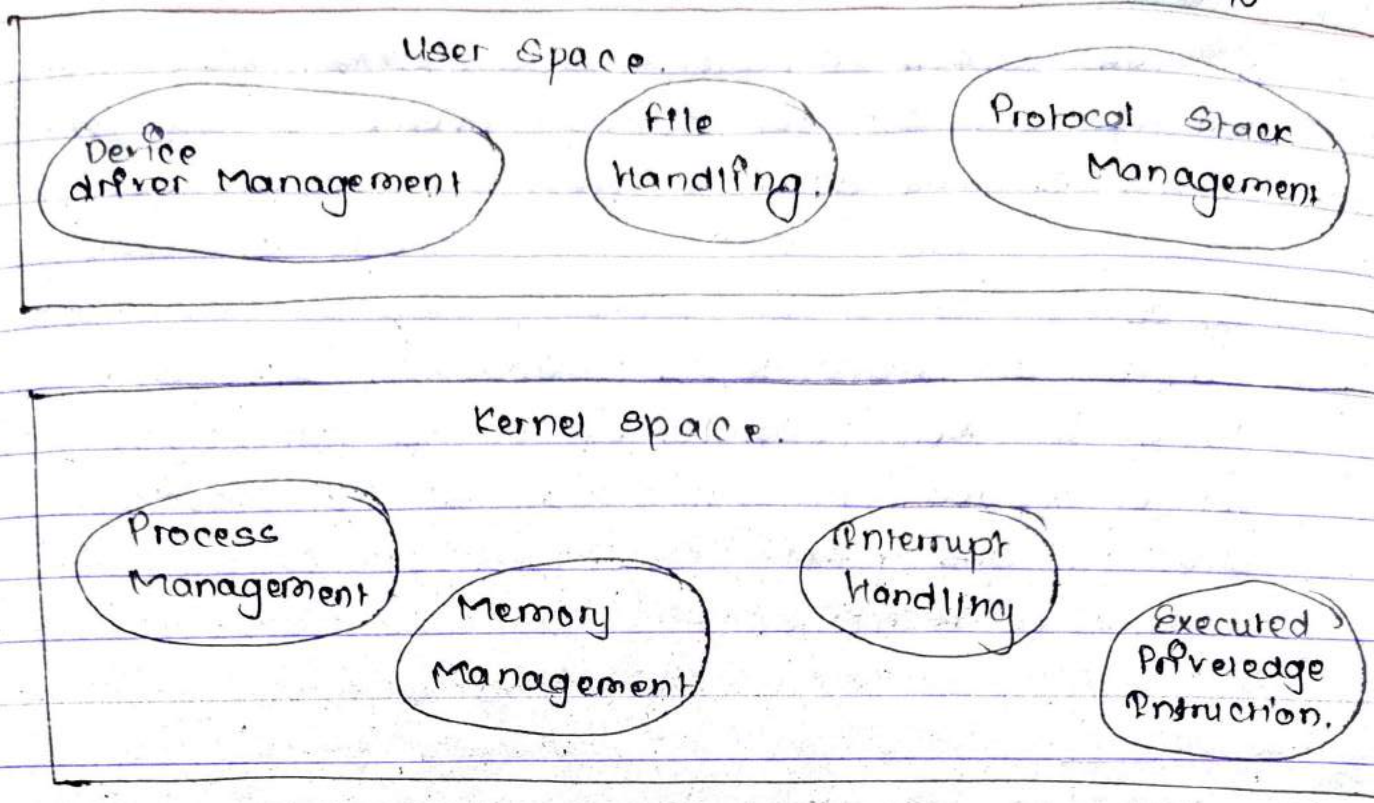


The monolithic kernel initially consisted of single module, but with increasing performance of OS, the kernel were large sized and have ~~more~~ very low reliability. So to increase the performance of the system, modular approach was made in the monolithic kernels in which each function was present in a different module inside kernel space.

ii) Microkernels.

→ In this type of kernel architecture, basic services like device driver management, protocol stacks management, file system management and graphic control are present in the user space and the rest like memory management like in kernel space.

Shreeja
161640



iii) Hybrid Kernel

→ It is combination of monolithic & microkernel.

---

8. 7. Explain different types of real time programming language with examples.

→ various programming language used in real time systems are as follows.

i) Ada

→ Ada was originally planned to be mandatory language for all US DOD projects. The initial version Ada 83 had shortcomings with scheduling, resource contention etc. So Ada 95 was brought which was object oriented. Ada 2005 is latest version.

Shreeja
161640

of Ada which contains additional dispatching policies, support for timing events & support for control of CPU-time utilization.

ii) C

→ C is a programming language developed around 1972 in Bell Laboratories. It is particularly good for embedded programming as it provides for structure and flexibility without complex language restriction.

iii) C++

→ C++ is a hybrid object oriented programming language that was originally implemented as a macro extension of C in 1980s.

iv) C #

→ C# is a C like language that has similarities to Java and the Java virtual machine. C# is ability to interact effectively with operating system APIs shield developers from complex memory management logic together with C#'s good floating point performance. These attributes make it a programming language that is highly potential for soft and firm real time application.

Shreeja
161640.

v) Real-time Java

→ RTSJ is extension of Java with unpredictable performance of garbage collection and broad guidance for schedule. It is used for soft, firm & even hard real time systems.

vi) Special Real time languages

→ PEARL

→ The process and ~~deretop~~ experiment automation RT language was developed in early 1970s by German researches. It uses the augumentati on strategy and has fairly wide application in Germany, in industrial control settings.

↳ Real-Time Euclid

→ It is experimental language from 1970s that enjoys distinction of being only language suited for schedwability.

↳ occam2

↳ It is language based on communicating sequential-processes formalism that was design ed to support concurrency on transputers.

↳ Real-time C

↳ It is generic name for variety of C macro-exte nsion packages which typically provided timing & control constructs.

↳ Neuron C

↳ It is enhancement to standard C with extension for event handling, network

Shreeja
161640

communication & hardware I/O.

→ Real-time C++.

It is generic name for one of several object class library specifically developed for C++. These libraries augumented standard C++ to provide increased level of timing & control.

---

8. What are different extensions being done in Structured System Analysis and Design for gaining requirement specification in RTS.

→ SASD or Structural Analysis and Structure Design is a technique that grew alongside language such as C & fortran. Structured Analysis for real time system is based on fundamental notion of flow of data between successive data transformation and hence provides little support for concurrency. Hence, various contributions has been made to extend SASD for gaining requirement as per the specification of RTOS.

The various extensions made to SASD are:

i) Gomaa's darts was introduced to show concurrency. DARTS solves this problem by providing decomposition principles. ~~or~~ DARTS was

later extended to ADARTS. (ADA-based design approach for RTS) and CODARTS (Concurrent Design Approach for RTS).

ii) Ward and Mellor extended data flow diagram of SASD by adding edges to represent control & state machines for representing behavior. The extensions were made to accomodate following demands:

- Information flow is gathered or produced on a time continuous basis.
- Control information is passed throughout the system and associated control processing.

The double headed arrow (⇉) was used to show continuous data and dashed or shaded arrows were used for control flow.

iii) Yourdon's Modern SA uses three viewpoints to describe SASD: an environmental aspect, a behavioral model and implementation model.
Environmental model embodies the analysis aspect of SASD and models the system at a high level of abstraction with the help of context diagram and an event list.
Behavioral model embodies design aspects of SASD as a series of DFDs, ERDs etc and using various combination of these tools, the designer models the processes, functions &

Shreeja
161840

flow of system in details

iv) Hatley and Pirbhai extended the SASD method by allowing for the addition of control flow analysis. Arcs made of dashed line indicates the flow of control information. Solid bars indicating 'stored' control commands, which are left unlabeled.
The addition of new control flows and control stores allowed the creation of control flow diagram (CFD).
These CFDs can be decomposed into C-SPECS and P-SPECS. The control specification C-SPECS represents the behavior of the system in state transition diagram and program activation table. The process specification (PSPEC) is used to describe all flow model processes that appears at the final level of refinement. It includes narrative text, a program design language (PDL) description of the process algorithm, mathematical equations, tables, diagrams or charts.

Shreeja
161640

9. How can requirements for a real-time system be captured using SASD and OOA/D approaches? Explain by mentioning the extensions incorporated.

→ The requirements for a real-time systems can be captured by using SASD and OOA/D approaches with the help of various extensions as follows:

In SASD following extensions are used

→ Ward and Mellor extended data flow diagram of SASD by adding edges to represent control & state machines for representing behaviors. They accomoded demands for continuous output/ output and discrete input/output.

→ The control flow is represented using dashed and ~~dotted~~ shaded arrows. Similarly control process were represented using dashed circles.

→ Yorden's modern structured analysis used environmental, behavioral and implementation model. Environmental model embodies the analysis aspect of SASD and models system at high level of abstraction. Behavioral model embodies design aspect of SASD as a series of DFDs, ERDs etc. & using various combination of these tools, the designer models the processes, functions & flow of system in details. In the implementation model the developer uses a selection of

Shreeja
161640

Structure charts, natural languages and pseudocode to describe the system.

→ The object oriented analysis and design methodologies have provision for real-time requirements:

- ROOM (Real time object oriented modeling)
- HRT-HOOD.

HRT-HOOD provides a method for guaranteeing deterministic timing behavior behavior in hard real time system. This is achieved by:

→ Eliminating inheritance
→ Introducing a small number of class stereotypes.
→ Restricting the synchronization mechanism for class operations.

UPM with UML has been used successfully with a number of extensions, e.g. UML 2.0, with significant extensions for real-time applications.

Behavioral aspects of the design can be represented by a number of different diagrams in the UML, e.g. sequence diagrams. other techniques for modeling time outside of the UML includes the use of $ models and various temporal logics.

Shreeja
161640

Another extension to OOAD for Real time system are UML interaction diagrams used to show interactions when a primary purpose of the diagram is to reason about time. It focuses on conditions changing within and among ~~lifelines~~ lifelines along a linear time axis.

Timing diagram describe behavior of both individual classifiers and interactions of classifiers, focusing attention on time of events causing changes in the modeled condition of the lifelines. Major elements of timing UML diagram :
- lifeline
- timeline
- state or condition
- message
- duration constraint
- timing ruler.

~~Lifeline~~

Timing diagram contains various parts like : States and conditions. It shows duration constraints, time constraint etc.

Shreeja
161540

10. "Real time computing is all about fast computing". Do you agree? Give your justification

→ Real time system is a system that is put through real time which means response is obtained within a specified timing constraint or system meets the specified deadlines. "Real time" here pertains to application in which computers must respond as rapidly as required by user or unnecessited by the process being controlled.

Real time computing is not all about fast computing. Though time is important factor determining the characteristic feature of real time computing, it doesn't necessarily mean fast. This is a misconception that arises from the fact that many hard real-time systems indeed deal with deadline in the tens of milliseconds, such as a aircraft navigation system. However there are also soft and firm real time system like food industry application which may have deadlines of 5 seconds or airline reservation system with deadline of 15 seconds. These latter examples may not be particularly fast, but satisfying that determines the success or failure of the system. Thus, real time computing is not all about fast computing.

Shreeja
161640

11. Differentiate between soft, hard and firm real-time system.

	Hard RTS	Soft RTS	Firm RTS.
i)	A hard RTOS is the one in which failure to meet even a single deadline may lead to catastrophic system failure.	A soft RTOS is the one in which performance is degraded but not destroyed by failure to meet time constraints.	A firm real-time system is one in which a few missed deadlines will not lead to total failure, but missing more than a few may lead to complete failure.
ii)	It is often critical.	It is non critical.	It is critical if more than few deadlines are missed.
iii)	Eg: avionics weapon delivery system in which pressing a button launches an air-to-air missiles.	Eg: console hockey game.	Eg: navigation controller for autonomous weed killer robot.

Shreeja
161640

figures.

i)

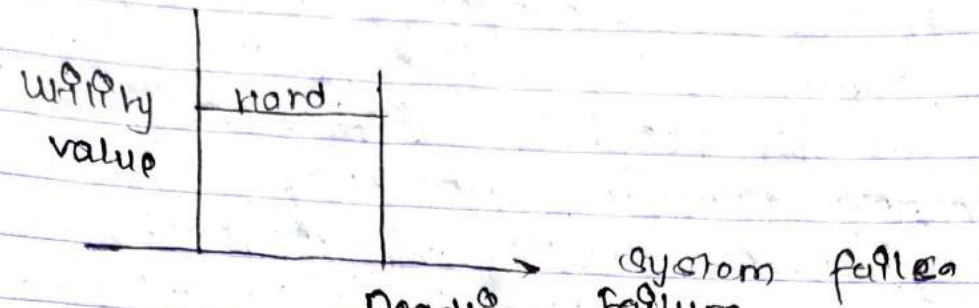


fig: Hard RTOS.

ii)

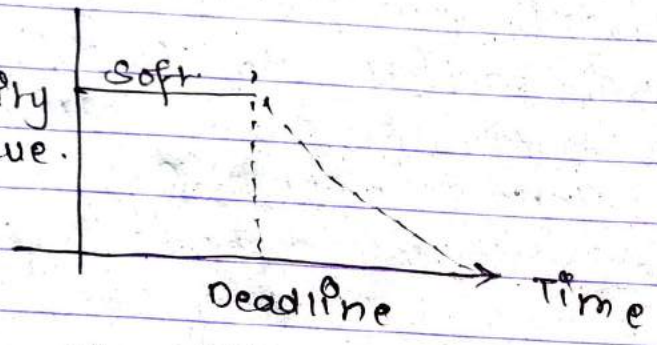


fig: Soft RTOS.

iii)

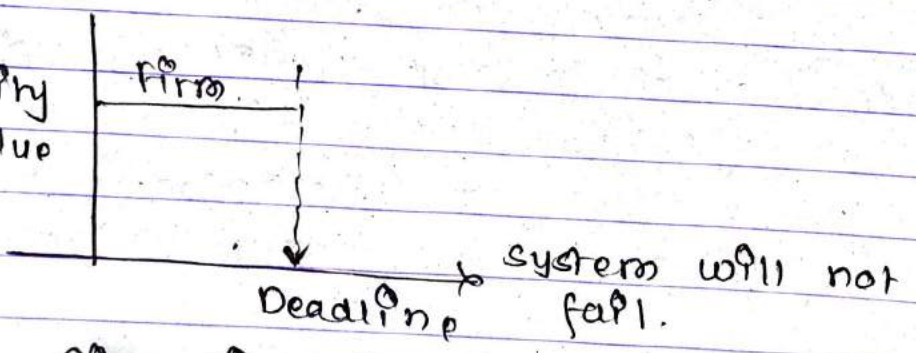


fig: Firm RTOS.

Shreeja
161640

12. A Burglar Alarm system has the following with execution period of p_i and worst case execution times of e_i, $i \in \{1,2,3,4\}$.

Task 1: capture frames from camera (at the rate of 25 fps): $p_1 = 250$ ms, $e_1 = 9$ ms.

Task 2: compare frames and determine intrusion $p_2 = 14$ ms, $e_2 = 2$ ms.

Task 3: Send signals to the Alarm to either Beep or not to Beep. $p_3 = 30$ ms $e_3 = 2$ ms.

What is overall CPU utilization factor.

Soln:

Here,

For camera: $p_1 = 250$ ms, $e_1 = 9$ ms.

For comparing frames: $p_2 = 14$ ms, $e_2 = 2$ ms.

for actuators: $p_3 = 30$ ms, $e_3 = 2$ ms.

CPU utilization (U) = ?

We know;

$$U = \sum_{i=1}^{3} \frac{e_i}{p_i}$$

$$= \frac{9}{250} + \frac{2}{14} + \frac{2}{30}$$

$$= 22.55\ \%$$

∴ Total CPU utilization is 24.55%.

Shreeja
1616/10

13. write why these are considered real time programming languages?

1) Ada.

→ Ada was originally planned to be mandatory language for all US DOD. The first version standardized by 1983 had serious problems. Though it was intended to be used for RTS but was bulky and inefficient and there was problem with multitasking. The useful constructs of Ada 95 to resolve shortcomings of Ada 83 are:

- A pragma controls how tasks are dispatched
- Pragma controls the interaction between tas scheduling.
- Pragma controls queuing policy of tasks an resource entry queues.

More additions were made to make Ada fully object oriented. They are:

- Tagged types.
- Packages
- Protected units.

The latest version of Ada has following change

- RTS annex contains:
  - additional dispatching policies.
  - support for timing events
  - support of CPU-time utilizations.

Shreeja
161640

ii) C#

→ C# is a C++ like language that, along with its operating environment, has similarities to Java and the Java Virtual machine. C# is associated with Microsoft's .NET framework for scaled down OS like windows CE which was originally intended to as RTOS for .NET platform. C# supports "unsafe code" allowing pointers to refer to specific memory locations. The object referenced by pointers must be explicitly 'pinned' disallowing the garbage collector from altering their location in memory. The garbage collector collects pinned objects & doesn't move them, thus increasing schedulability. C# support array of thread synchronization constructs: - Lock & monitor
- Mutex : interlock

Timers that are similar in functionality to the widely used win32 timer exist in C#. When constructed, timers are configured how long to wait in milliseconds before their first invocation, and are supplied an interval in milliseconds specifying period between subsequent invocation. The accuracy of these timers is machine dependent, and thus not guaranteed. C# is not appropriate for hard real time system but are suitable for soft and firm RT application.

Shreeja
161640

(pg) ~~Java~~

→ Java, in the same way as C#, is interpreted language, that is code compiles into machine independent intermediate code. Real-time Java is modification of standard Java language.

RTSJ defines real-time classes to create thread which the resident scheduler executes. Real time threads can access objects on heap and can incur delays because of garbage collections. So, RTSJ extends memory model to support memory management in a way that doesn't infer RT code's ~~availability~~ ability with the use of preallocated objects pools.

RTSJ uses priority such that highest priority thread indicates most eligible thread. It doesn't presume strict priority based dispatch policy. The resources in priority order. The priority inheritance protocol is implemented by defaut.

In RTSJ, asynchronous event facility comprises two classes: AsyncEvent and AsyncEventHandler.

RTSJ also defines two classes for programmer who want to access physical memory for Java code. They are Raw Memory Access and Physical Memory. These features make Java suitable for RT programming.

Shreeja
161640

iv) C

→ The C programming language, invented around 1972 at Bell laboratories, is a good language for low level programming. C supports machine related things like addresses, bits, bytes and characters which are handled in high-level language. These can be used effectively to control CPU's work registers, peripheral interfaces and other memory mapped ~~on~~ hardware.

C provides special variable types as register, volatile, static and constants, which allows for effective control of code generation ~~and~~ at procedural level. Automatic coercion occurs in C. This feature is very convinient but it makes it impossible for compiler to throughly type check the arguments, which can create problems.

C also provides exception handling through the use of signals, and other two mechanism, set jump and longjump. setjmp procedure call is ~~known~~ is macro that saves environment information that can be used by subsequent longjmp library call. Longjmp restore the program to the state at the time of the last setjmp call.

Overall, C language is particularly good for embedded programming as it provides for structure & flexibility without complex language restriction.

Shreeja
161640

14. Distinguish between periodic, aperiodic and sporadic tasks with examples.

⇒

i) Periodic tasks.

→ Periodic tasks are time driven i.e the characteristics are known in advance and they execute at periodical instants of time. Every J task J_i is characterized by (p_i, e_i) where p_i is task period and e_i is the worst case execution time, P_i is the minimum time period between job releases.

→ Eg: measuring pressure and other attributes in a chemical plant.

ii) Aperiodic tasks.

→ Real time systems are required to respond to external events which occur at random instants of time, the system executes a set of job in response to these events. The release time of these jobs are not known until the event triggering them occurs.

→ Eg: garbage collection in OS, activity logs, system mode changes.

Shreeja
161040

iii) Sporadic tasks.

→ Sporadic tasks are the jobs with hard deadline. These jobs execute to accomplish the mode change. Every task is characterized by arrival time and deadline.

→ eg: Pilot disengaging the autopilot system arbitrarily making system changes from cruise mode to standby mode in a transient fault tolerant system.

---

15. Is it a good idea to allow preemption on hard RTS? Discuss your view.

→ Preemption allows higher priority task to preempt the lower priority one if it interrupts the lower priority are assigned to the task based on their urgency. This is a really good feature for hard real time system since it gives good processor to task with highest priority.

Though the highest priority task always gets cpu, the preemptive priority suffers from resource hogging by higher priority tasks. This may lead to a lack of available resource for lower priority tasks. This may lead to a ~~lack of availability~~ ~~available~~ to lower-priority task facing serious problem like starvation.

Thus, it is better to allow non-pre

Shreeja
161640

emphre scheduling in hard real time systems where missing deadlines leads to catastrophic situation and where resources must not be wasted. Hence, it is not good idea to allow preemption on hard real time system.

16. What are the different ways of performing system integration?

→ Integration is the process of combining partial functionality to form the compete system functionality. RTS are usually embedded, the integration process involves both hardware and software units.

The different ways to perform system integration are:

1) A simple integration strategy

→ In any embedded OS, it is important to ensure that the tasks are running at a prescribed rate, and that context is saved and restored. This is done without performing any function within those tasks; functions are added later.

This approach involves establishing a baseline of running kernel components. It ensures that the interrupts are handled properly and all the cycles are running at

their prescribed rates, without worry about interferen ce from application code. Once the baseline is established, small sections of application code are added and the cycle rates verified. If an error is detected, it is patched. If patch succeeds in restoring cycle rates properly, then more code is added. This ensures that the system is grown incrementally, with an appropriate baseline at each stage of integration

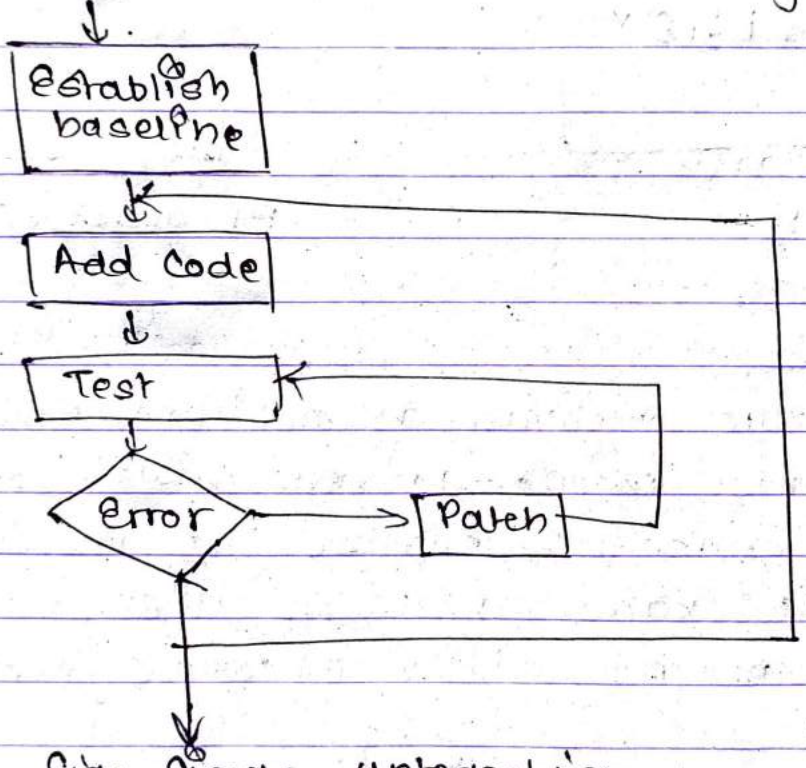


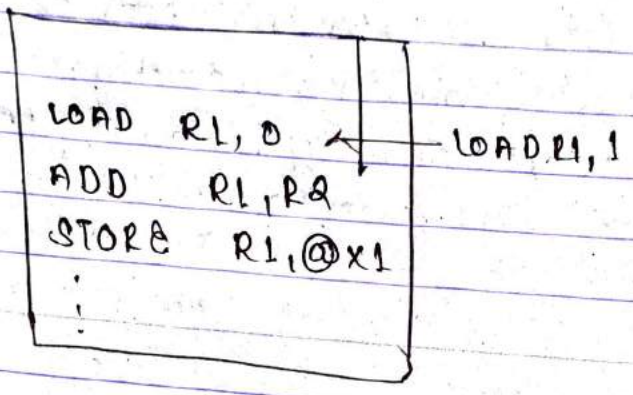
fig: Simple integration

ii) Patching

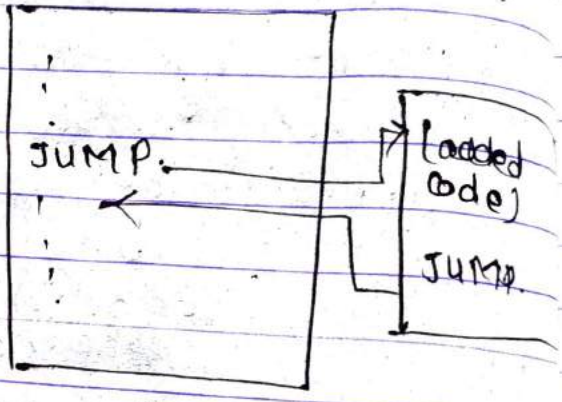
→ It is the process of correcting errors directly on the target machine. It allows minor errors detected during the integration process to be corrected on targeted machine directly without

Shreeja
181640

undergoing the tedious process of correcting on a new module. Patching of software written in object-oriented languages is very difficult because of lack of a straightforward mapping from source code on the object code.



In patch



An oversized patch

(iii) Probe effect.

→ ~~while~~ software systems do not explicitly deal with electrons (except as ensemble behavior), uncertainty arises because the more closely a system is examined more likely the examination process will affect system.

The ~~software~~ system integration can be done with the use of various integration tools like multimeter, oscilloscopes, logic analyzers etc.

Shreeja
161640

17. What are different POSIX API available for locking resources by tasks?

→ The different POSIX API available for locking resources by task are:

i) int pthread_mutex_lock (pthread_mutex_t *mutex)

→ This routine locks the mutex object referenced by mutex. If the object is already locked by another thread, this call will block the calling thread until it is unlocked.

ii) int pthread_mutex_trylock (pthread_mutex_t * mutex)

→ This routine is called to lock the mutex object referenced by mutex. However, if the mutex is already locked, it returns immediately with a "busy" error code.

iii) In Posix, semaphores are not included in pthread library. The necessary declaration related to semaphores are contained in semaphore.h library. They are:

→ int sem_wait (sem_t *sem)

→ int sem_post (sem_t * sem)

wait on semaphore locks the semaphore. If value is greater than 0, do

Shreeja
161640

In order to lock the memories there are procedures like: mlockall (int) & munlockall(void)

i) mlockall
→ It tries to lock all the memory down, this includes the program text, data, heap and stack. Locking a process includes shared libraries that can be mapped in and other shared memory areas that process may be using.

ii) munlockall (void).
→ munlockall unlocks all the locked memory.

Memory can be locked down in parts as well:

i) mlock.
→ It locks down the adress range being specified. If mlock is called for range of memory, then calling munlockall unlocks all memory locked with mlock. It can cause segmentation fault if an address is passed to where there is not any memory for executable code.

Shreeja
16.16.40

18. How can scaled number and BAM be used for optimizing the performance of real time system?

→ Scaled number and BAM are used in optimizing the performance of real time operating system. In virtually all computers, integer operations are faster than floating point numbers. Thus to eliminate this inefficiency caused by floating point calculation scaled number and BAM is used.

In scaled number, the least significant bit (LSB) of an integer variable can is assigned a real-number scale factor. Scaled numbers can be added and subtracted together as well as multiplied and divided by constants - but not by other scaled number. The result are then converted to floating point only at last computing step, thus saving processing time.

Eg:

Suppose for an analog-to-digital converter least significant bit is of the two's complement 16-bit integer has value $a = 0.0001$ m/s². then any acceleration can be represented up to the maximum value of $(2^{15}-1)\ 0.0001\ m/s^2 = 3.2767\ m/s^2$. The 16 bit number 0000 0000 0000 1101, for instance represents acceleration of

Shreeja
161640

$19 \times 0.0001 \ m/s^2 = 0.0019 \ m/s^2$.

Another type of scaled number is based on the property that adding 180° to any angle is analogous to taking its complement. This technique is called Binary Angular Measurement (BAM).

Consider the LSB of n-bit word to be 2^{-n}. $2^{(n-1)} \cdot 180°$ with most significant bit (MSB) of 180°. The range of any angle θ represented this way is $0° \leq \theta \leq 360° - 2^{-(n-1)} \cdot 180°$. A 16-bit BAM word is shown as follows.

180	90	45	22.5	...	0.0054931

For better accuracy, BAM can be extended to multiple words. Each n-bit word has maximum value of:

$$(2 - 2^{-(n-1)}) \cdot 180° = 360° - LSB$$

with granularity:

$$2^{-(n-1)} \cdot 180° = LSB.$$

Let us consider 16-bit BAM word.

0000 0000 1010 0110

It corresponds to the angle:

$$166 \cdot 2^{-15} \cdot 180° = 0.911°.$$

Shreeja
161640

BAM words can be added and subtracted as well as multiplied and divided by constants.

Thus, in this way, the scaled number and BAM make floating point operations efficient and optimizes the performance of RTS.

---

19. Differentiate between PIP and PCP for avoiding resource contentions.

→ Priority Inversion Protocol (PIP) is a critical resource sharing protocol which is used for sharing critical resources among different tasks. This allows the sharing of critical resource among different processes without the occurance of unbounded priority inversions. When a task goes through priority inversions, then priority of the lower priority task, which has the critical resource is increased by the priority inheritance mechanism. It allows this task to use the critical resource as early as possible without going through the preemption. It avoids the unbounded priority inversion.

Priority ceiling Protocol (PCP) is an extension of priority inversion Protocol (PIP) & Highest Locker Protocol (HLP). It solves the problem of unbounded priority inversion of priority inheri-

Shreeja
161640

tance protocol, deadlock and chain blocking of highest locker protocol. and also minimizes the inheritance-related version which was also a limitation of ~~HLP~~ HLP. It is not a greedy approach like PIP. In PCP, it is possible that a task may be denied for access ~~at~~ although the resource is free.

Priority Inheritance Protocol	Priority Ceiling Protocol
→ It is a critical resource sharing protocol used for sharing critical resources among different tasks.	It is a critical resource sharing protocol which is extension of PIP and HLP.
→ It overcomes the limitations of traditional resource sharing technique.	It overcomes the limitations of PIP and HLP.
→ It requires minimum support from OS.	It requires maximum support from OS.
→ It can not prevent the deadlock.	while it prevents tasks from going into deadlock
→ Tasks using PIP may suffer chain blocking.	while the task using PCP can not suffer from chain blocking.

Shreeja
161640

→ It is the simplest protocol among all resource sharing protocols.	~~write~~ It is the complex and most efficient one among all.
→ It can not minimize the inheritance related inversions.	~~write~~ It is able to ~~main~~ minimize the inheritance related inversion.
→ It solves the problems of unbounded priority inversion.	It solves the problem of unbounded priority inversion, deadlock and chain blocking.
→ In PIP, highest priority task can be denied access, if resource is free..	~~write~~ In PCP, highest priority task can be denied access although resource is free - if priority value is less than current system ceiling.
→ It is usually used in small application.	→ It is used in large applications.

Shreeja
1C16CS40

Q. How can fault tolerance be achieved in case of Real Time Systems?

→ Fault tolerance is the tendency to function in the presence of hardware or software failures. In real-time systems, fault tolerance includes design choices that transform hard RT deadlines into soft. The various ways in which fault tolerance can be achieved are:

i ~~Spatial fault Tolerance~~

→ Spatial fault-tolerance can be achieved by using redundant hardware. Two or more pairs of ~~software~~ redundant hardware devices provide inputs to the system.

Under spatial fault-tolerance, the approaches are

→ checkpoints.

→ It is a way to increase fault tolerance in which the intermediate results are written in a memory at a fixed location in code for diagnostic purpose. These locations are called checkpoints and can be used during system operation and during system verification.

→ Recovery block approach.

→ In this technique fault-tolerance can be increased by using

checkpoints in conjunction with predetermined reset points in software. These reset points mark recovery blocks in the software. At the end of recovery block, the checkpoints are tested for "reasonableness". If the results are not reasonable, then processing resumes with a prior recovery block.

ii) Software black boxes.

→ It is related to checkpoint and is used in certain mission critical systems to recover data to prevent future disasters. Its objective is to recreate the sequence of events that leads to the software failure for the purpose of identifying the faulty code.

iii) N-Version Programming.

→ In any system, a state can be entered where the system is rendered ineffective or locks up due to some unitested flow-of-control in the software for which there is no escape. In order to reduce this sort of catastrophic error, a redundant processors are added to the system. The redundant processors can use a voting scheme to decide on outputs, or mor dificult. There are two processors, master and slave.

Shreeja
161640

iv) Built in test softwares.

→ Built in test softwares can increase fault tolerance by providing ongoing diagnostic of the underlying software. BITS is especially important in embedded systems.

eg: if an I/O channel is functioning incorrectly as determined by its onboard circuitry, the software may be able to shut off the channel & redirect the I/O.

Although BITS ~~add~~ is important part of embedded system, it add significantly to worst case time-loading analysis. Thus, this must be considered when selecting BITS.

Q1. What is the difference between priority inversion and unbounded priority inversion? Explain with an example using timeline and three set of tasks. How does priority inheritance protocol solve the case of priority inversion.

→ Priority inversion occurs when a high priority task is forced to wait for the release of shared resource owned by a lower priority task. There are two types of priority inversions:

→ Bounded → Unbounded.

Shreeja
161610

They occur when two types of tasks attempt to access a single shared resource. A shared resource can be anything that must be used by two or more tasks in a mutually exclusive fashion. The period of time that a task has a lock on a shared resource is called the task's critical section or critical region.

3) Bounded priority inversion.

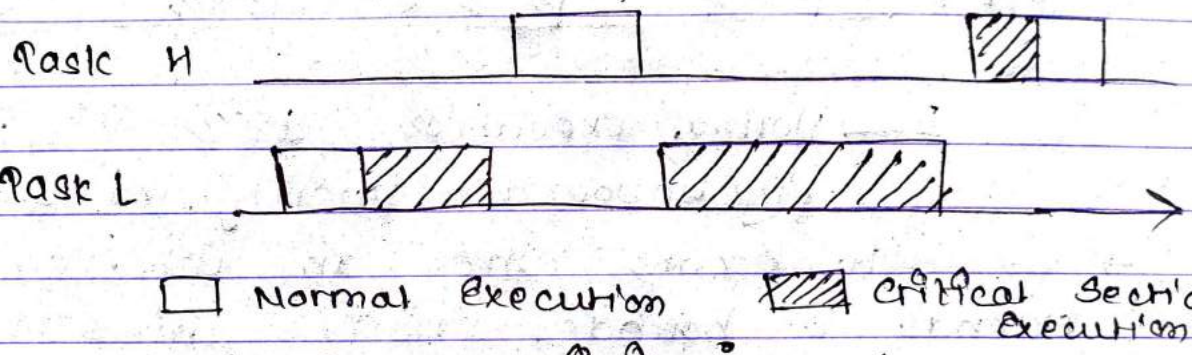


fig: Bounded priority inversion

Bounded priority inversion occurs when low-priority Task L acquires a lock on a shared resource, but before releasing the resource is preempted by high-priority Task H. Task H attempts to acquire the resource but is forced to wait for Task L to finish its critical section. Task L continues running until it releases the resource, at which point Task H acquires the resource and resumes executing. The worst-case wait time for Task H is equal to the length of critical section of Task L. Bounded priority inversions won't generally hurt an application provided

Shreeja
16/640

the critical section of Task L executes in timely manner.

(ii) Unbounded priority inversion

→

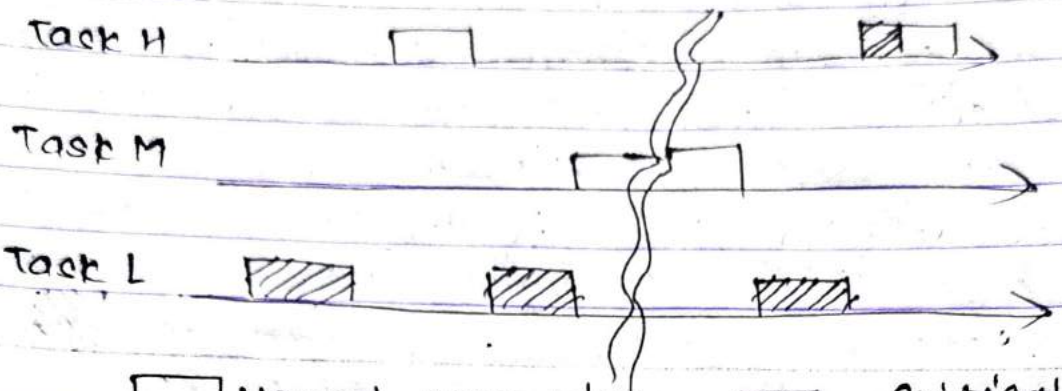
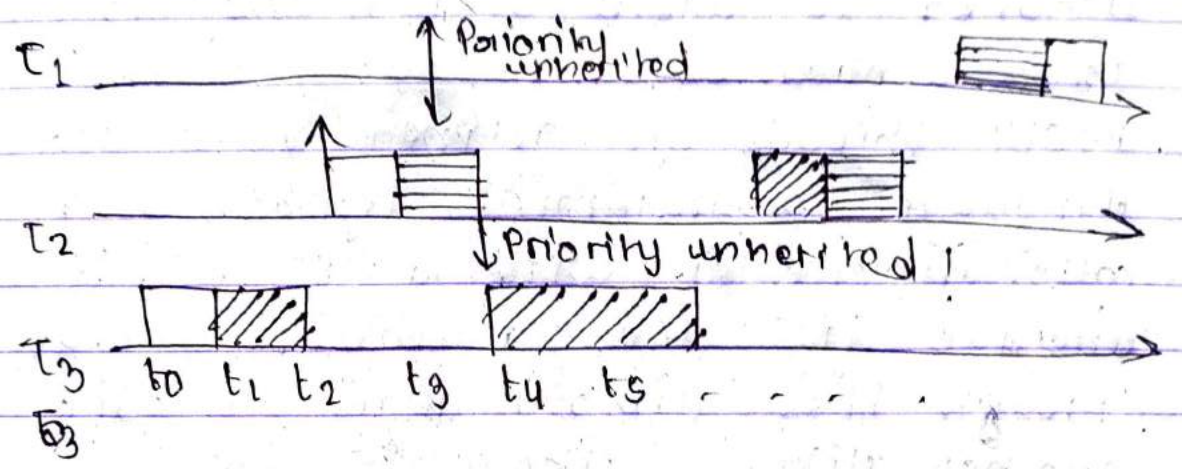


fig: unbounded priority inversion

→ It occurs when an intervening task extends a bounded priority inversion. Let us suppose the medium priority task M preempts L during the execution of Task L's critical section. Task M runs until it relinquishes control of the processor. Only when Task M turns over ~~current~~ control can Task L finish executing its critical section and release shared resource. This extension of critical section leads to unbounded priority inversion. Task H can acquire resource only when task L release the resource.

Shreeja
1616410

Priority Inversion offers a simple solution to the problem of unbounded priority inversion. Here, the tasks are dynamically changed so that the priority of any task in critical ~~syster~~ region gets the priority of highest pending task.



If a task, T_1 is blocked by τ_2 and $T_1 > \tau_2$, task τ_2 inherits the priority of τ_1 as long as it blocks τ_1. when τ_1 exist critical section it blocks τ_1. when τ_2 exits the critical section that caused the blocks, it reverts to the priority it had before it entered the section. The highest priority task τ_1 relinquishes the processor whenever it seeks to lock the semaphore guarding a critical section that is already locked by other job. If task τ_1, is blocked by τ_2 & $T_1 > T_2$, task τ_2 inherits the priority of τ_1 as long as it blocks τ_1. when τ_2 exits the critical section that caused the block, it reverts to the priority it had before it entered the section.

Shreeja
161640

Q2. How can POSIX compliant threads be created? Explain the APIs used for creating and locking semaphores.

→ POSIX stands for Portable Operating System Interface. It provides compliance criteria for operating system services and is designed to allow applications programs to write application that easily. The POSIX threads are defined as a set of C language programming types and procedure calls. Implemented with a pthread.h header/ include file and a thread library, though this library may be part of another library like. libc.

POSIX provides counting semaphores and binary semaphores to enable processes running in different address spaces, to synchronize and communicate ~~with shared~~ using shared memory

The different types of APIs used for creating and locking semaphores are:

1) ~~sem~~ initializes

→

Shreeja
161640

i) int sem_init (sem_t *sem, int pshared, unsigned int value).
→ Initializes the semaphore object pointed by 'sem'.

ii) int sem_destroy (sem_t *sem)
→ It destroys a semaphore object and frees up the resources it might hold.

iii) sem_t *sem_open (const char *name, int oflag, ...).
int sem_close (sem_t *sem)
int sem_unlink (const char *name)
~~int sem_wait (sem_t *sem)~~
→ They are used in conjunction with other process.

iv) int sem_wait (sem_t *~~sem~~ sem)
→ It suspends the ~~th~~ calling thread until the semaphore pointed to by 'sem' has non-zero count. It decreases semaphore count.

v) int sem_trywait (sem_t *sem)
→ It is non-blocking variant of sem_wait

vi) int sem_post (sem_t *sem)
→ It increases count of semaphore pointed to by 'sem'.

Shreeja
161640

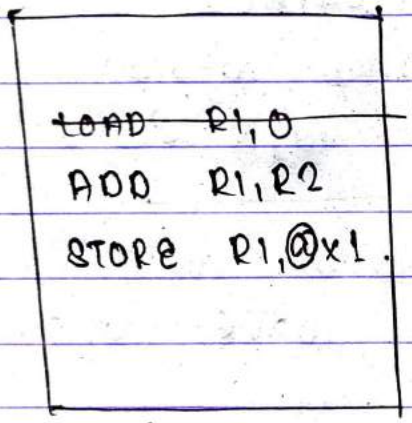
iii) int sem_getvalue (sem_t * sem, int * sval)
→ It stores the current count of the semaphore 'sem' in 'sval'.

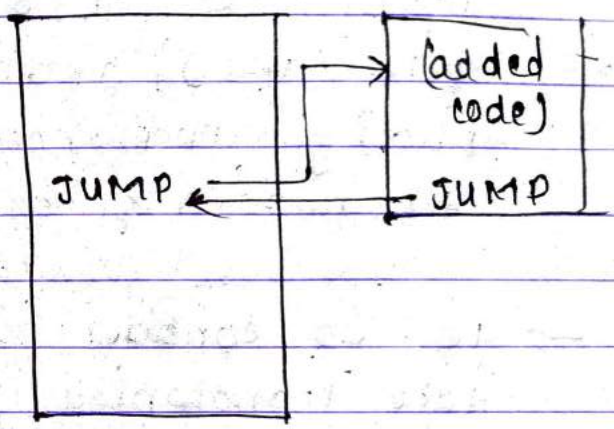
Q3. Differentiate between probing and patching why and how are they used?
→

Patching	Probing.
→ It is the process of correcting errors directly on target machine.	It is based on Heisenberg uncertainty principle and states that uncertainty in system arises because the more closely a system is examined, the more likely the examination process will affect system.
→ It allows minor errors to be solved on machine itself.	It is not about fixing error but close examination can bring change in the behavior of the system.

Streeja
161640

Patching helps to correct the source codes remotely and ~~creating~~ creating new load modules. It requires expert command of the opcodes for the target machine. unless a macro-assembly level patch facility is available. Eg: A 1 was supposed to be added to register 1 (R1) instead of a 0. This error can be changed easily by altering the memory location containing the LOAD R1, 0 instruction to LOAD R1, 1.





in-line patch.

Eg for Probe:

an engineer debugging the pasta sauce bottling system discovers the deadlines are not being meet. ~~After~~ Some debugging code is added to print preliminary result. After adding debugging code problem goes away. But as soon as debugging code is removed it reappears.

Shreeja
161640

Qu. The following set of periodic tasks are to be scheduled by RMA:

Tasks	Period - Dead line(pi)	Execution - time (ei)
A.	3	1
B	2	1
C	6	4

i) if tasks A, B and C both use a shared resource R, protected by S (task A uses resource R for 2ms & C uses same for 4 ms ii) and Real time kernel allows priority inheritance then, are above task all scheduable ? explain.

→ Let us consider that the task above are rate monotonically schedulable.
Then,
Highest priority task A will have response time equal to its execution time, so,

$$R_A = 1.$$

The next highest priority task is B, it will have its response time calculated as following:

Shreeja
16/6/10

Initially;

$R_B = 1$.

The recursive values following R_B^0 are derived as:

$$R_B^1 = e_B + \left\lceil \frac{R_B}{P_A} \right\rceil e_A$$

$$= 1 + \left\lceil \frac{1}{3} \right\rceil 1.$$

$$= 1 + 1$$

$$= 2.$$

Similarly;

$$R_B^2 = e_B + \left\lceil \frac{R_B^1}{P_A} \right\rceil e_A$$

$$= 1 + \left\lceil \frac{2}{3} \right\rceil 1$$

$$= 1 + 1 = 2$$

Similarly, for the task with lowest priority C, the response time is;

$$R_c = e_c = 1.$$

The recursive values following R_c^0 are derived as;

$$R_c^1 = e_c + \left\lceil \frac{R_c^0}{R_A} \right\rceil e_A + \left\lceil \frac{R_c}{P_B} \right\rceil e_B$$

$$= 1 + \left\lceil \frac{1}{3} \right\rceil 1 + \left\lceil \frac{1}{2} \right\rceil 1$$

$$= 1 + 1 + 1 = 3.$$

Shreeja
161640

Again;

$$R_c^2 = e_c + \left\lceil \frac{R_c^1}{P_A} \right\rceil \cdot e_A + \left\lceil \frac{R_c^1}{P_B} \right\rceil \cdot e_B$$

$$= 1 + \left\lceil \frac{9}{9} \right\rceil \cdot 1 + \left\lceil \frac{9}{2} \right\rceil \cdot 1$$

$$= 1 + 1 + 2$$

$$= 4.$$

Here, $R_c^1 \neq R_c^2$ thus the tasks are not schedulable.

---

25. What is difference between mutex and semaphore? How can they be used to avoid resource contention.

→ The difference between mutex and semaphore are as following.

Semaphore	Mutex
→ It is a signaling mechanism.	It is a locking mechanism.
→ It exists as integer variable.	It exists as an object.
→ It allows multiple program threads to access a fine instance of resources.	It allows multiple program thread to access a single resource but not simultaneously.

Shreeja
161640

Semaphore	Mutex
→ Semaphore value can be change by any process acquiring or releasing the resource	→ Mutex object lock is released only by the process that has acquired the lock on it.
→ Semaphore can be categorized into counting semaphore and binary semaphore.	Mutex is not categorized further.
→ Semaphore value is modified using wait() and signal() operation.	The mutex object is locked and unlocked by other processes requesting or releasing the resource.
→ If all resources are being used, the process requesting for resource performs wait() operation on block itself if semaphore count becomes greater than one.	If a mutex object is already locked, the process requesting for resource waits & is queued by the system till lock is released.

Shreeja
161640

Resource contention refers to the conflict over access to shared resource such as random access memory, disk storage, cache memory, internal bus and external network devices. Mechanisms like mutex, semaphores and queues are used to solve it

A mutex provides mutual exclusion and synchronizes the resources. It is created with unique name at the start of the program. It ensures that only one thread can acquire the mutex at a time and enter the critical section

→ wait (mutex);

...

Critical section

...

* signal (mutex).

A semaphore is signalling mechanism and a thread that is waiting on a semaphore can be signalled by another thread. It uses wait & signal. wait decreases value of s, if it is positive. If s is negative or zero, then no operation is performed.

wait(s)

{ while (s<=0);

s--;

}

Shreeja
161640

Signal (S)
{
S++;
}

26. How can you find out the maximum no. of task in a single hyperperiod? A real time system consists of tasks $T_1(4,1)$, $T_2(6,2)$, $T_3(10,2)$. Propose the appropriate frame size to ~~above~~ schedule the above mentioned task using cyclic executives or clock driven scheduling.

→ Hyperperiod is major cycle in cyclic executive which is equal to the least common multiple (LCM) of the period, that is, lcm($p_1, \ldots p_n$). It is the minimum time required to execute required to executed allocated to the processor, ensuring that the deadlocks and periods of all the process are ~~mad~~ met.

The maximum no. of task in a single hyperperiod is equal to lcm of all the processes.

→

Shreeja

Given,

Three tasks are T1(4,1), T2(6,2), T3(10,2).

Tp	ep	pp	dp
T1		4	1
T2		6	0
T3		10	0

Here the maximum execution time among these tasks is 2. The frame must be equal to the maximum execution time of these tasks.
It means the ~~fi~~ frame is greater than or equal to 2.

Now, Hyperperiodicity =

pp	4	6	10
fp	2, 4	2, 3, 6	2, 4, 5, 10

Now,

P1° \ f	2	4	5	6	10
4	4	8	10	12	20
6	2<4 (T)	4<4 (X)	5<4 (F)		
10	2<6 (T)	4<6 (T)			
	2<10 (T)	4<10 (T)			

Shreeja
1616410

	f	2	4	5	10
P_i	2f	4	8	10	20
4		2<4 (T)	4<4 (F)	5<4 (F)	10<4 (F)
8		2<6 (T)	4<6 (T)	5<6 (T)	10<6 (F)
10		2<10 (T)	4<10 (T)	5<10 (T)	10<10 (F)

∴ The number of frames required is 2.

---

27. How can proper selection of variable reduce its memory utilization

→ Memory utilization in one area ~~than~~ can be reduced at the expense of another. Memory utilization is less of a problem than it has been in the past, but occassionally a system needs to be designed in which the available main memory is small (specially in embedded system)

The automatic variables increase the loading in the stack area of memory, whereas global variable appear in the BSS area. By forcing variables to either local or global, ~~can~~ relief can be purchased in one area of memory at the expense of the other, thus balancing the memory load

In addition, intermediate result calculations that are computed explicitly

Shreeja
1st 40

require a variable either in the stack or RAM area, depending on whether it is local or global. The intermediate value can be forced into a register instead of omitting the intermediate calculation. Here to illustrate, let us consider this C code fragment that calculates one root of a quadratic:

discriminant = b * b - 4 * a * c;
root = (-b + sqrt(discriminant)) * 0.5 / a;

this code could be replaced by:

root = (-b + sqrt(b * B - 4 * a * c)) * 0.5/a;

which saves one floating point variable and thus by at least 4 bytes of memory. In addition, this eliminates at least one STORE macroinstruction, reducing time-loading as well.

Q8. How can response time analysis be done from a system using Round Robin scheduling?

→ To analyze response time of a round-robin system let us consider there are n processes in the ready queue and no ~~one~~ new one arrives after the system starts, and none terminate prematurely. The release time is arbitrary, although all processes are ready ~~to~~ at the same

Shreeja
161640

time, the order of execution is not predetermined, but fixed.

Assume all process have maximum end-to-end execution time, C. Suppose that each process P_i has a different maximum execution time, C_i. Then, letting $C = \max\{C_1, \ldots, C_n\}$ yields a reasonably upper bound for the system performance and allows the use of this model.

let time slice be q.

each process, would get 1/n of the CPU time in chunks of q time units, and each process would wait no longer that (n-1) q time units until its next time up.

Since each process requires at most $\lceil C/q \rceil$ time units to complete, the waiting time will be, $n-1\ q \left\lceil \frac{C}{q} \right\rceil$ so,

$$T = (n-1) \left\lceil \frac{C}{q} \right\rceil q + C.$$

Eg: Suppose, there are 5 processes with a maximum execution time of 500 ms. The time quantum is 100 ms. Hence, $n = 5$, $C = 500$, $q = 100$ which yields

$$T = (5-1) \left\lceil \frac{500}{100} \right\rceil 100 + 500$$

$$= 4 \times 5 \times 100 + 500$$

$$= 2000 + 500 = 2500.$$

Shreeja
161640

29. What are tools available for Real time system integration?

→ Integration is the process of combining partial functionality to form the overall system functionality. The various test tools used for real time system integration are:

i) Multimeter.

→ The use of multimeter in the debugging of RTS may seem odd nowadays, but it is an important tool in embedded system where the software controls or reads analogue values through hardware. It measures voltage current, or power and can be used to validate the analogue input to or output.

ii) Oscilloscope

→ Like multimeter, it is not always regarded as a software debugging tool, but it is useful in embedded software environment. Oscilloscopes range from basic single-trace variety to storage oscilloscopes with multiple traces. It can be used for validating interrupt integrity, discrete signal issuance and receipt, and for monitoring clocks.

Shreeja
1616 40

(iii) Logic Analyzer

→ The logic analyzer is connected to the system under test by connecting probes The logical analyzer can be used for timing instructions, timing code or as in-circuit emulator etc.

(iv) ~~Logic Analyzer~~ : Timing Instruction

→ Logic analyzer can be used to time an individual macroinstruction segments of code, or an entire process. To time an individual instruction, the engineer finds a memory location in the code segment of memory containing the desired instruction. It is set to trigger & logical analyzer will display the difference betn fetch of first instruction and next one.

(v) ~~Logic Analyzer~~ : Timing Code.

→ Logic analyzer also provides an accurate method for measuring time-to-complete for any periodic task. To measure the total elapsed time for any task in the system, set the logic analyzer to trigger on the starting and ending address and opcode for the first instruction of task. It should be first instruction of interrupt handler.

Shreeja
16164o

vi) ~~logic analyzer~~ : In-Circuit emulator.
→ In-circuit emulator are useful in software patching and for single-stepping through critical portion of code. In-Circuit emulators are not typically useful in timing tests, however, because subtle timing can be introduced by emulator.

vii) Software simulators.
→ When integrating and debugging embedded systems, software simulators are often needed to stand in for hardware & inputs that do not exist or that are not readily available.

viii) Hardware Prototypes
→ In the absence of actual hardware system under control, simulation hardware may be used instead of software simulation. Hardware simulators simulate real life system inputs and can be useful for integration and testing, but are not always reliable testing the underlying algorithms, for which real data from live devices are needed.

Shreeja
161640

30. What are POSIX signals? How can they be generated?

→ Signals are software representation of interrupts or execution occurances. Signals asynchronously alter the control flow of a task. It is important to point out that no routine should be called from a signal handler that might cause the handler to block - it makes it impossible to predict which resources might be unavailable.

Signals in POSIX have limitations like:

- → Lack of signal queuing
- → No signal delivery order
- → Poor information content.
- → Asynchrony.

These are improved in POSIX.4.

In POSIX, sigaction defines all the details that a process need to know when a signal arrives. As real-time signals can be queued, the queuing option for real-time signal is chosen by setting bit SA_SIGINFO in the sa_flags field of sigaction structure of the signal.

→

Shreeja
161640

```
struct sigaction {
    void (*sa_handler)();
    sigset_t sa_mask;
    int sa_flags;
    void (*sa_sigaction)(int, siginfo_t*, void
};

int sigaction(int sig, const struct
        sigaction * reaction, struct
        sigaction * oldreaction);
```

Real time signal can carry extra data, SA_SIGINFO increases information delivered, SA_SIGINFO is set, then the signal handler have additional parameter siginfo_t. sigqueue() includes an application specified value, signals can be specified as offsets from SIGRTMIN. All signals sent with sigqueue() are queued in numeric order.