EI_ 48398 _Megha Lohar

PAGE:
DATE: / /

Sunbeam
Institute of Information Technology.

DESD - Real Time Operating System

Question 1

What is RTOS and describe types of RTOS?

RTOS

- RTOS stands for Real Time Operating System.
- RTOS is OS in which accuracy of result is not only dependent on "correctness of the calculation", but also depends on "time duration" in which results are produced.
- Based on timing requirements there are three types of RTOS
  - ⓐ Hard realtime os
  - ⓑ Soft realtime OS
  - ⓒ Firm realtime OS

A) Hard Realtime OS

- These operating systems guarantee that critical tasks be completed within a range of time.
- For example.

A robot is hired to weld a car body. If the robot welds too early or too late, the car cannot be sold, so it is a hard real-time system that requires complete car welding by robot hardly on the time.

- If interrupt / tasks deadline miss → catastrophic effect.
- Usually doesn't have secondary storage.
- Timing : 10 to 100 "use".
- Eg. µC-OS, FreeRTOS, Xenomai RTAI,---

B) Soft Realtime OS

- This operating system provides some relaxation in the time limit.
- For example,
- * Multimedia systems, digital audio systems etc.
- * Explicit, programmer-defined and controlled processes are encountered in real-time systems.
- * A separate process is changed with handling a single external event.
- * The process is activated upon occurrence of the related event signalled by an interrupt.
- Multitasking operation is accomplished by scheduling processes for execution independently of each other.
- Each process is assigned a certain level of priority that corresponds to the relative importance of the event that it services.
- The processor is allocated to the highest priority processes.
- This type of schedule, called, priority-based preemptive scheduling is used by real time systems.
- Less time critical - If deadline miss, after product quality (not catastrophic).
- May have secondary storage.
- Timing : 4ms to 40ms
- Eg. Linux (PREEMP_RT) ...

c) Firm Realtime OS

- Like hard realtime.
- Rare miss of deadline is acceptable (not catastropic)

Question 2

Difference between GPOS and RTOS ?

	GPOS	RTOS
1.	Customization Support GPOS is little or no customizable (Linux is exception).	Customization Support RTOS is fully customizable. So that it can be used with minimal memory.
2	Interrupt Latencies GPOS have higher interrupt latencies. i.e in msec.	Interrupt Latencies RTOS handle interrupts in deterministic time and with lower latencies i.e. in usec.
3.	IPC Latencies (i) IPC buffers are allocated at runtime. (ii) Task awakening is not real-time. (iii) Signal handling is not real-time.	IPC Latencies (i) IPC buffers are preallocated (system heap). (ii) Task awakening is done in deterministic time. (iii) Signals are processed in real time.
4.	Memory Management GPOS uses more memory for IPC mechanisms, tasks etc.	4Memory Management. RTOS should have minimal memory footpoints as compared to GPOS. so that it can smobthly work on low end embedded devices.

	GPOS	RTOS
9.	Interrupt Management In GPOS, interrupts are processed in two steps. i.e. top half (non-blocking code) and bottom half (blocking code) to ensure better interrupt latency.	Interrupt Management In RTOS, ISRs are minimal & non-blocking. Also interrupt handlers are executed as highest priority tasks, so that they will be executed before any other task.
10	IO Subsystem & Device Driver. Drivers have higher latencies because they deal with rest of the OS through multiple layers.	IO subsystem & Device Drivers. Drivers and tasks are present in same address space. (i.e. kernel space), so that minimal latencies are ensured.
11.	Task Management In GPOS, processes & threads are heavy-weight i.e. have higher memory requirements.	Task Management The RTOS takes light-weight. i.e. With Minimum memory requirements.

Question 3

Explain following terms,

1) Interrupt Latency.
2) Dispatcher Latency.
3) Kernel Response Time.

PAGE:

DATE: / /

El_49398_ Megha Lohar

① Interrupt Latency.

Interrupt Latency = Maximum amount of time interrupts are disabled

+

Time to start executing the first instruction of the ISR.

Interrupt response time:

(i) Time between the reception of an interrupt and the start of user code that handles the interrupt.

(ii) For foreground / background systems AND non-preemptive kernels.

- = interrupt latency + time needed to save CPU context.

(iii) For preemptive kernels.

- = Interrupt latency + time needed to save CPU context + execution time of the kernel ISR entry function.

② Dispatcher Latency

* The term dispatch latency describes the amount of time it takes for a system to respond to a request for a process to begin operation.

* With a scheduler written scheduler written specifically to honor application priorities real-time applications can be developed with a bounded dispatch latency.

③ Kernel Response Time

* Time to handle the interrupt, schedule the next process & begin its execution.

Interrupt arrived → Interrupt handler → ISR call → ISR execution → Scheduler called → Decide the next process to execute → Dispatch / restore context of new process.

Kernel Response Time = Interrupt Latency + ISR Duration + Scheduler Latency + Scheduler Duration.

Interrupt Latency = Intterrupt arrived → Interrupt handler → ISR call.
ISR Duration = ISR execution
Scheduler Latency = Scheduler called.
Scheduler Duration = Decide the next process to execute.

Question 4

Describe RTOS Scheduling mechanism, earliest deadline first, rate monotonic scheduling proportional shared scheduling ?

RTOS Scheduling

* Admission control system

It verifies whether newly created task can be completed within deadline. If possible, then only task is executed further, otherwise task is rejected.

* All admitted task are guaranteed to be completed within deadline.

* RTOS use specialized scheduling algorithms,
RMA : Rate Monotonic Scheduling Algorithm.
EDF : Earliest Deadline First.
Proportional Share Algorithm.

Rate Monotonic Algorithm :

(i) The rate-monotonic scheduling algorithm schedules periodic tasks using a static priority policy with preemption.

(ii) If a lower-priority process is running and a higher-priority process becomes available to run, it will preempt the lower-priority process.

(iii) Upon entering the system, each periodic task is assigned a priority inversely based on its period : The shorter the period, the higher the priority, the longer the period, the lower the priority.

(iv) The rationale behind this policy is to assign a higher priority to tasks that require the CPU more often.

(v) Furthermore, rate-monotonic scheduling assumes that the processing time of a periodic process is the same for each CPU burst. That is, every time a process acquires the CPU, the duration of its CPU burst is the same.

RMA = lower is the period, higher is the priority.

e.g. P1 : t=20, d = 50, p = 50 (lower period => higher priority)
P2 : t=35, d=100, p=100 (higher period => lower priority).

CPU Utilization = $\frac{20}{50} + \frac{35}{100}$ = 0.75.

In RMA, max CPU utilization = n * (2^1/n-1)
Maximum 69.3%. for many tasks.

Earliest Deadline First

(i) Earliest - deadline first (EDF) scheduling dynamically assigns priorities according to deadline.

(ii) The earlier the deadline, the higher the priority; the later the deadline, the lower the priority.

(iii) Under the EDF policy, when a process becomes runnable, it must announce its deadline requirements to the system.

(iv) Priorities may have to be adjusted to reflect the deadline of the newly runnable process.

(v) This algorithm can give 100% CPU utilization.

(vi) Early the deadline, higher will be the priority.

e.g. P1 : t=25, d=50, p=50
P2 : t = 35, d=80, p=80.

Question 5

What is priority inversion and what is solution to overcome it ?

(i) Priority inversion is a situation in which a low-priority task executes while a higher priority task waits on it due to resource contentions.

(ii) A high task priority implies a more stringent deadline.

(iii) In a priority based, preemptive scheduling system, the kernel schedules higher priority tasks first & postpones lower priority tasks until either all of the higher priority tasks are completed or the higher priority tasks voluntarily relinquish the CPU.

(iv) In real time embedded system, the kernel strives to make the schedulability of the highest priority task deterministic.

(v) To do this, the kernel must preempt the currently running task and switch the context to run the higher priority task that has just become eligible, all within a known time interval.

(vi) This system scheduling behavior is the norm when these tasks are independent of each other.

(vii) Task interdependancy is inevitable when task share resources and synchronizing activities.

(viii) Priority inversion occurs when task interdependancy exits among tasks with different priorities.

EI_49398 _ Megha Lohar

PAGE:
DATE: / /

Priority Inversion Problem.

Priority becomes inverted and deadlock develops in certain situations when using semaphore.

Solution:

1. Disabling all interrupts to protect critical sections.
2. A priority ceiling.
3. Priority Inheritance
4. Avoid blocking.
5. Random boosting.

Question 6

What is jiffy?

(i) Jiffy: In computing, a jiffy was originally the time bet^n two ticks of the system timer interrupt.

(ii) It is not an absolute time interval unit, since its duration depends on the clock interrupt frequency of the particular hardware platform.

(iii) Stratus also defines the microJiffy, being 1/65,536 of a regular Jiffy.

(iv) Jiffies = 32 bit ⇒ incremented on each tick interrupt
Jiffies = 64 bit ⇒ memory shared with jiffies.

j
jiffies = sec * Hz

Question 7

What is the difference between Hard and Soft real-time systems?

Hard Real-time Systems	Soft Real-time Systems
(i) Hard response time is required	(i) Soft response time is required.
(ii) Data integrity is short term.	(i) Data integrity is long term.

Hard Real-time Systems	Soft-Real-time Systems.
(iii) Size of the data file is small or medium.	(iii) Size of the data file is large.
(iv) Peak load performance is predictable.	(iv) Peak load performance is degraded.
(v) Hard real-time systems have little laxity & generally provide full deadline compliance.	(v) Soft real-time systems are more flexible. They have greater laxity & can tolerate certain amounts of deadline misses.
(vi) Safety critical systems are typically a hard real-time system.	(vi) Linux and many OS provide a soft real time system.

Question 8

What is priority inheritance?

Priority inheritance.

(i) The Priority inheritance protocol is a resource access control protocol that raises the priority of a task, if that task holds a resource being requsested by a higher priority task, to the same priority level as the heigher priority task.

(ii) Basic concept of PIP:

The basic concept of PIP is that when a task goes through priority inversion, the 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.

(iii) Advantages of PIP:

It allows the different priority tasks to share the critical resources.

The most prominent advantage with priority inheritance protocol is that it avoids the unbounded priority inversion.

(iv) Disadvantages of PIP:

① Deadlock
② Chain Blocking.

Question 9

When should we re-enable the interrupts in an ISR and why?

EI_49298_Megha Lohar.

PAGE:
DATE: / /

Question 10

What are Hard & Soft Interrupts?

	Hard Interrupts	Soft Interrupts
①	Hardware interrupt is an interrupt generated from an external device or hardware.	Software interrupt is the interrupt that is generated by any internal system of the computer
②	It do not increment the program counter.	It increment the program counter.
③	Hardware interrupt can be invoked with some external device such as request to start an I/O or occurrence of a hardware failure.	Software interrupt can be invoked with the help of INT instruction.
④	It has lowest priority than software interrupts.	It has highest priority among all interrupts.
⑤	It is an asynchronous event.	It is synchronous event.
⑥	Hardware interrupts can be classified into two types. they are. (1) maskable interrupt, (2) Non-maskable interrupt.	Software interrupts can be classified into two types. (1) Normal interrupts. (2) Exception.

Question 41

What is difference between Aperiodic and Periodic Task? Explain RTOS APIs for the same.

	Aperiodic Task	Periodic Task
①	It can occur at random instants.	It repeats itself after a certain time interval.
②	These tasks are not controlled by clock interrupts	These tasks are controlled by clock interrupts.
③	The time interval between occurrence of two consecutive tasks can be zero	The time interval between occurrence of two consecutive tasks can't be zero
④	Aperodic tasks generally include soft real-time tasks.	Periodic tasks generally include soft and hard real-time tasks both.
⑤	To meet deadline of all instances of an aperiodic task is quite difficult.	Deadline of all instances of periodic task can be meet easily.
⑥	It includes interactive task with users	It includes vast majority of internal tasks.
⑦	Example: Logging task in a distributed system.	Example: Taking information from sensor at a time interval.

~~Question 1~~2.

RTOS API :

- API calls. At the heart of an RTOS is the kernel, which comprises the task scheduler and a number of services available to be called by application programs.
- Control of the scheduler and access to these services is by means of the kernel's application program interface (API).

Question 12

What is preemptive and non-preemptive scheduling ?

Preemptive Scheduling

(i) Preemptive scheduling is a CPU scheduling technique that works by dividing time slots of CPU to a given process.

(ii) The time slot given might be able to complete the whole process or might not be able to it.

(iii) When the burst time of the process is greater than CPU cycle, it is placed back into the ready queue & will execute in the next chance.

(iv) This scheduling is used when the process switch to ready state.

(v) Algorithms that are backed by preemptive scheduling are round-robin (RR), priority, SRTF (Shortest remaining time first).

(vi) The current process may give up CPU volunteerily or paused forcibly (for high priority process or upon completion of its time quantum).

Non-preemptive Scheduling.

(i) Non-preemptive Scheduling technique the process takes the resource (CPU time) and holds it till the process gets terminated or is pushed to the waiting state.

(ii) No process is interrupted until it is completed, and after that processor switches to another process.

(iii) Algorithms that are based on non-preemptive scheduling are non-preemptive priority and shortest job first.

(iv) The current process gives up CPU volunteerily (for IO, terminate or yield).

(v) The CPU scheduler picks next process for the execution.

(vi) If each process yields CPU so that other process can get CPU for the execution, it is referred as "Co-operative scheduling".

Question 13

How to manage Timer hardware in RTOS? Explain RTOS APIs for the same.

* Hardware timer are used for timing & counting operations allowing the processor to carry on with some other process while the timer process run.

* Basic timer operation has been where a clock input a counting a counting register to measure time or count external events.

* If functionality can be extended by using additional

registers to store timer values. creating a ccp. module. CCP stands for capture / compare / PWM.

API :

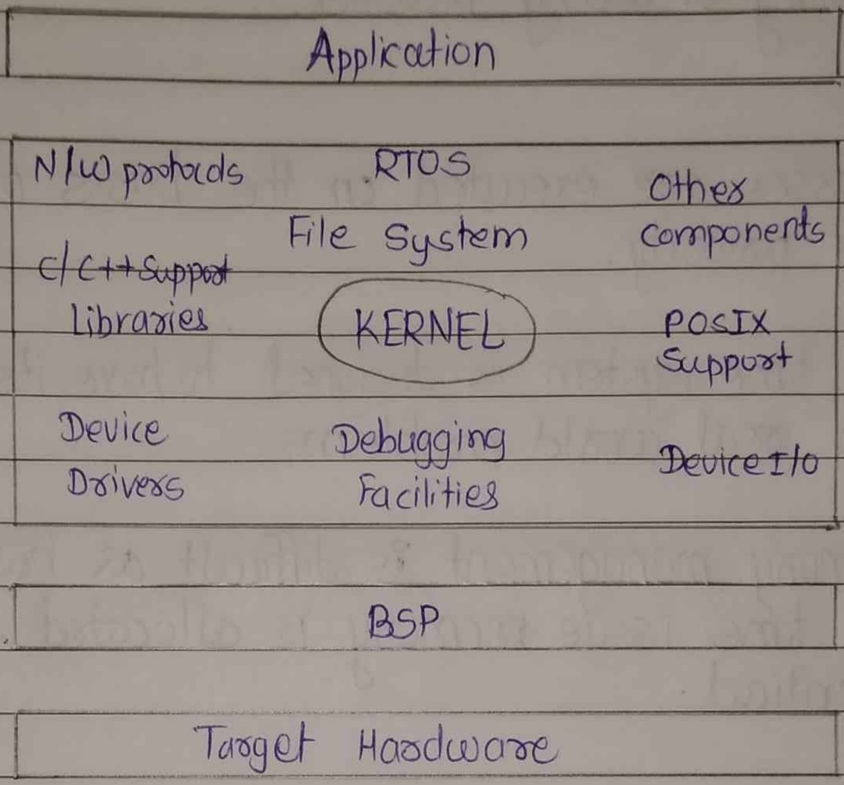
(i) At the heart of an RTOS is the kernel which comprises the task.

(ii) Scheduler & number of service available to be called by the scheduler & access to these services is by means of the kernels application program. interface (API)

(iii) API is differ from one RTOS to another, although there are some stdard like POSIX. but some characteristic are common to many RTOSes.

Question 14.

Explain architecture of any one RTOS in detail?



RTOS

① Whenevee embedded system comes into picture it is always a combination of hardware like μC or μP & software like a firmware or OS.

② An operating system forms the base of all the electronic and manages both the hardware & software within any electronic device.

③ The term operating system is not only limited to unix & windows for computers but also extent to μC.

④ One such operating system that can run on μC is called as Real time operating system.

Why RTOS?

1. Availability of drivers.
2. Scheduled files.
3. Flexibility of adding features.

(v)* Processes are executed on the basis of the order of their priority.

(vi)* Real time system is designed to have its execution for the real world problems.

(vii) Memory management is difficult as based on the real time issue memory is allocated, which itself is critical.

(viii) Applications:
Controlling aircraft or nuclear reactor, scientific research equipments.

(ix) Examples: Vx Works, QNX, Windows CE.

(x) Most popular RTOS.
Deos (DDC-I), embos (SEGGER), FreeRTOS (Amazon), Integrity (Green Hills software)...