|  |  |  |
| --- | --- | --- |
| **Realtime System?**  Correctness and Execution time of the result are guaranteed. When a certain deadline is  met. -correct result -specified time frame (meets a deadline)  **Requirements**? -Predictability- Reliability- minimal Delay/latency  - Correctness and Execution time of the result are guaranteed  **2 most important deadlines?** -correct result and have to meet deadlines  **Three types of a Realtime systems?**(Andere Fragestellung: which are the different kinds of realtime systems, give one exampleeach)  **- Hard:**missing a deadline is a total system failure (airbag)  **- Soft:**the usefulness of a result degrades after its deadline thereby degrading the  system's quality of service(warning systems)  **- Firm:** in frequent deadline is misses are tolerable but may the usefulness of a result  is zero after its deadline (car - ignition point optimizer for motor)  **Classification/Properties?**-centralised or distributed RTS -interactive or autonomic systems  - hierarchical or independent system -time-driven or event driven RTS  - consequences of missing deadline -reliability and fault tolerance  - cyclic or asynchronous scheduling  **Operating System?**Operating system fulfil requirements  - small ticks -Examples **Definition for:** - Relative Time  **-Watch Dogs:**function of other components monitored. If a possible malfunction is detected  -Alarm clock  **-Timer:**- enable set the time start the time → gives a signal at the end of day  **Main requirements for realtime operating systems?**  - timer with small ticks (ns). Synchronization to a time standard  **Tasks? if the task is non-interruptilble, it can have only 3 states:** - ready -running -completed  **there is only one way to change states:** -ready (may be not necessary) nunning completed  **when you use explicit plan there are only the states:** - running - completed  **if the task is interruptible, there are 5 states:** -ready -running -blocked –suspended –completed **Running:**When a task is actually executing it is said to be in the Running state. It is  currently utilizing the processor. If the processor on which the OS is running only  has a single core then there can only be one task in the Running state at any  given time. **Ready:**Ready tasks are those that are able to execute but are not currently executing because a different task of equal or higher priority is already in the Running state.  **Blocked:**A task is said to be in the Blocked state if it is currently waiting for either a  temporal or external event. For example, if a task calls “sleep()” it will block until the delay period has expired a temporal event. Tasks can also block to wait for queue, semaphore, event group, notification or semaphore event. Tasks in the Blocked state normally have a 'timeout' period, after which the task will be timeout, and be unblocked, even if the event the task was waiting for has not occurred. Tasks in the Blocked state do not use any processing time and cannot be selected to enter the Running state.  **Suspended:**Like tasks that are in the Blocked state, tasks in the Suspended state cannot be  selected to enter the Running state, but tasks in the Suspended state do not have a  time out. Instead, tasks only enter or exit the Suspended state when explicitly  commanded to do so through the “Suspend()” and “Resume()” API calls respectively.  **Embedded Systems?**-an embedded system is a computer that does not look a computer  **Requirements of Embedded Systems?**-at least one CPU -has a physical context  -provides dedicated set of services to the end user -does not provide general computing services to the end user -often part of a larger environment -have often real time constraints  **Minor page fault/major page fault?**-page faults cause an interrupt -if this happens, you need more memory -lack memory and forbid RTS from having page fault  **Absolute vs. relative timestamps? Relativ:**-displays the number of minutes, hours, days, week or years ago a post was published  **Absolute:**-display the exact date and time a post was published  **Scheduling?**-The Process of creating a schedule deciding how to order this tasks and how to commit resources between the variety of possible tasks  **Feasible Schedule?**-when each job can be completed with its individual deadline  **EDF (earliest deadline first)?**-has dynamic priorities, shorter deadline → higher priority  -executable task with high p will always interrupt a task with lower priority  **RMS( Rate Monotonic Scheduling)?**-has a static priorities: shortest period → high priority  -preemptive -if a system not schedulable with RMS can not be schedulable with any other static priority  **EDD (earliest Due Date)?**-non preemptive -for periodic tasks with equal arrival time | **Different between necessary test and sufficient Test?**  **Necessary test (test fails = no feasible schedule):**-if one of the appropriate necessity test fails then there is no feasible schedule  -if one or more or all necessity test are fulfilled then there may be or may not be a feasible schedule  **Sufficient:Test (suff T. exists → feasible schedule):**-if you find at least (delta) suf. necessity test, than the task package is feasible schedulable  -if you can not find a suffice necessity text, than a feasible suf. may exist or not  **Priority inversion?**  -a high priority is blocked because a task of lower priority is using the resource  **Difference between process and thread?**  **Process:**-Haeavy unit of resource allocation and of protection-process creation is very costly in terms of resources -program execution a process is relatively slow  -process cannot access to memory belonging to another process -process switching is time consuming -process can contain several T’s  **Thread:**-light weight unit of CPU utilization-creation is very economical  -program executing using threads are comparatively faster  -can access to memory area belonging to other threads within the same process  **TCB(Thread Control block)?**  -thread state -program counter -CPU Registers -CPU Scheduling info  -memory might info -pointer to the process control block  **PCB (Process CB):**-process state -cpu registers -cpu scheduling info -accounting info  -I/O status info -memory might info  **Race Condition? how avoid?**-when 2 or more tasks access common data  -avoided by allowing access to critical selection for 1 task only at a time → mutual exclusion -mathematical solution → semaphore  **Mutex?**a mutex is a special semaphore with n==1 (also called binary semaphore). it tells you only free of not and the queue respectively  **Semaphore?**semaphoren ist eine speicher sperre. mutex up → freigeben, mutex down→ sperrenmutex <0 → fehler  **what issue do we have with concurrencies?-Deadlock:** one waits for another  -**livelock:** one triggers the other (distributed endless loop) -race condition  **Job?**-a job is a single cpu-time requirement to perform a computational sequence  **Execution Time?**-the duration of a specific job between job request and job completion -min net exc (ideal case) -max net exc (worst case)  **Sensor?**-converts a physical or chemical messure into electrical signals  **Assembling?**-Physical signal → sensor → low pass filter → sampling → quantization → coding  **My quist Theorem?**-sampling frequency : needs to be at least twice the maximum of the recorded signal -we need for that a low pass filter  **If sample slowly?**-higher freq. signals wp2 be transferred  **Aliasing?**-causes a false lower frequency component to appear in the simpled data of a signal  **Anti Aliasing?**-Remove all higher frequency signals  **Processor Architecture?-ISP:** Processors that come with a dedicated with a pre -defined Instruction set- similar CPU's like in PC's  **-ASIP:** are optimized in Hrd and software for the specific use of the system  **-ASP:** optimal performance clue to complexity should only be done for processors that are needed for a single purpose  **Microcontroller?**  **Development Process:**-develoop it on pc  -cross compile it on pc -deploy the executable onto the NC  **Components:-**Timer -Memory -Interface -ADC (Analog Digital Converter)  -DAC (Digital Analog Converter)  **Distributed Realtime systems?**-avoid collision on the Network line by connecting smaller parts with each other with only one connection  **Petri Nets? simple process model with:**-3 elements: places, transitions and arcs  -graphical and mathematical description -formal semantics and allows for analysis  **Enabled transmission:**-if each of its input contains at least one token  -an enabled token can fire -when it fires it consumes a token from each input and places it in each output **liveness:**-a transition is live if it can never deadlock  **safeness:**-the number of tokens in a place never executes one |  |