

Real Time Systems. Final exam questions.

1. Describe the main differences between hard real-time and soft real-time systems. Provide an example of each and explain why it fits that category.
2. Identify two typical applications of real-time systems in industry. Explain why real-time processing is crucial for each application.
3. Tell about the role of hardware (such as processors and sensors) in a real-time system and its impact on system performance.
4. What is the importance of standards in real-time systems? Name one standard relevant to real-time systems and describe its main purpose.
5. Describe the general architecture of an RTOS. What are the primary responsibilities of the kernel in a real-time OS?
6. Explain synchronous communication in real-time operating systems and provide an example where synchronous communication is preferred.
7. How do sensors and actuators interact within a real-time system? Provide an example of such interaction in an industrial setting.
8. Explain the difference between preemptive and non-preemptive scheduling in real-time systems. When might each type be preferred?
9. Compare any two memory management techniques used in real-time systems between each other. Why is efficient memory management critical in real-time environments?
10. Describe the steps involved in designing a real-time system. Highlight the aspects unique to real-time system design compared to general software system design.
11. Describe a typical process flow for implementing a real-time system, including testing and verification phases. Why is real-time system testing particularly challenging?
12. List two tools commonly used in real-time system development and briefly describe their functions.

13. Explain the concept of deterministic testing in real-time systems. Why is determinism important, and how might it be verified?
14. Describe the key architectural features of QNX that make it suitable for real-time applications. How does it differ from other real-time operating systems?
15. Outline the steps involved in creating a real-time system from scratch. Which programming languages and frameworks might be preferred, and why?
16. Describe two methods of representing numbers in memory in a real-time system. Discuss their relevance and potential impact on performance.
17. What are the primary differences between processes and threads in real-time systems? Provide an example use case for each.
18. Explain the concept of synchronous programming in real-time systems. Provide an example of a situation that would benefit from synchronous execution.
19. Describe the role of signal processing in QNX. How can signals be used for communication between processes in real-time applications?
20. How does QNX handle resource allocation? Describe one mechanism for preventing resource contention in a real-time environment.
21. Explain link building technologies in the context of real-time systems. Why is efficient linking critical, and what challenges might arise?
22. Describe a scenario in which resource allocation and management are critical in real-time systems. How might a failure in resource management impact the system?
23. Explain how semaphores can be used for synchronization in real-time systems. Provide an example of using semaphores for inter-process communication.
24. Describe the use of timers to ensure periodicity in a real-time system. How might you configure a timer to execute a task every 10 milliseconds?
25. Explain how shared memory can be utilized for data transmission in real-time systems. Provide an example of when shared memory might be preferred over other communication methods.

26. List and explain three essential characteristics of real-time systems that differentiate them from general-purpose systems.
27. Define periodic and aperiodic tasks in real-time systems. Provide an example of each and describe the implications for system scheduling.
28. Explain the concept of latency in real-time systems. How can high latency affect the performance and reliability of a real-time application?
29. Describe the process of interrupt handling in a real-time operating system. Why is it essential for handling interrupts to be fast and predictable?
30. Tell about the importance of timing constraints in real-time systems. What are the potential risks if these constraints are not met?
31. Compare and contrast two scheduling algorithms commonly used in real-time systems. Include their strengths and limitations.
32. How does multitasking work in real-time operating systems? What mechanisms are used to switch between tasks?
33. Describe the message-passing mechanism in QNX. How does it ensure reliable communication between processes?
34. Explain the role of embedded systems in real-time applications. Give an example of an embedded real-time system in a household appliance.
35. Define priority inversion. How can priority inversion affect a real-time system, and what are some techniques to prevent it?
36. Tell about the role of reliability in real-time systems. What measures are typically taken to ensure a real-time system is reliable?
37. Describe how energy management can be handled in a real-time system, especially in battery-operated embedded systems.
38. Explain the role of thread priority in a real-time system. How would you choose the priority of different tasks?
39. Distinguish between kernel mode and user mode in real-time systems. Why is this distinction important for security and stability?
40. Describe a strategy to prevent deadlocks in real-time systems. What role does resource allocation play in preventing deadlock?

41. Define determinism in the context of real-time systems. Why is it crucial for the predictability of a real-time application?
42. What is response time analysis in real-time systems, and why is it necessary? Describe a method to calculate response time for a specific task.
43. Explain rate monotonic scheduling (RMS) in real-time systems. What assumptions does RMS rely on, and what types of tasks is it best suited for?
44. How should a real-time system handle overload situations? Describe a technique that ensures critical tasks are prioritized in these cases.
45. Describe a communication protocol commonly used in real-time systems, such as CAN or EtherCAT. What are its key characteristics?
46. Define event-driven programming in the context of real-time systems. What are the benefits and challenges of using an event-driven model?
47. What is jitter, and how does it affect the performance of a real-time system? Provide an example scenario where jitter might be problematic.
48. How errors are typically handled in real-time systems? Why is it essential for error handling mechanisms to be predictable?
49. What is a watchdog timer, and how is it used in real-time systems? Provide an example where a watchdog timer might be essential.
50. Describe an application where QNX would be suitable as the operating system. Explain how QNX's features contribute to the system's requirements.
51. Explain the difference between hard, soft, and firm real-time systems. Provide an example where each type would be applied.
52. Describe the concept of time-driven scheduling. How does it compare to event-driven scheduling in a real-time system?
53. How do real-time operating systems manage scarce resources such as CPU time and memory to ensure deadlines are met?
54. How QNX implements scheduling for real-time tasks? What unique features does QNX offer for real-time task management?
55. Explain the role of periodic tasks in real-time systems. How do systems ensure that these tasks are executed within their time constraints?

56. Define interrupt latency in a real-time system. What factors contribute to interrupt latency, and how can it be minimized?
57. Describe the concept of load balancing in a real-time system. How can load balancing affect the performance and reliability of the system?
58. Explain the Inter-Process Communication (IPC) mechanisms available in QNX. How does QNX ensure efficient and safe communication between processes?
59. Compare dynamic and static scheduling algorithms in real-time systems. What are the advantages and disadvantages of each approach?
60. Define preemption in the context of real-time systems. How does preemption affect the behavior of real-time applications?
61. What strategies can be used to ensure both the reliability and availability of a real-time system in a critical application?
62. Explain the concept of fault tolerance in real-time systems. Provide an example of how fault tolerance is implemented in a safety-critical system.
63. Describe the difference between semaphores and mutexes in real-time systems. How would you choose between them for synchronization?
64. What are hard real-time constraints, and what techniques are used to guarantee that they are met?
65. How do real-time systems synchronize threads or processes to ensure mutual exclusion and avoid race conditions?
66. How is memory dynamically allocated in real-time operating systems? What are the challenges associated with memory allocation in real-time environments?
67. Explain how task deadlines and priority assignment work in real-time systems. How do scheduling algorithms ensure deadlines are met?
68. Define the role of the kernel in a real-time operating system. What specific features does it need to support real-time task execution?
69. Compare event-driven and time-driven models of real-time systems. In what situations would each model be more appropriate?

70. What are the key performance metrics used to evaluate real-time operating systems? Discuss how they are important for ensuring real-time system reliability.
71. How does QNX handle resource allocation to ensure that real-time tasks are prioritized? How does it avoid conflicts between processes?
72. Explain the concept of time-partitioning in real-time systems. How is this approach useful in systems with mixed criticality levels?
73. What are the challenges of managing databases in real-time systems? How do real-time database systems handle time constraints?
74. List and describe at least three key features that a real-time operating system should provide for supporting real-time task execution.
75. How do real-time systems utilize multi-core processors for scheduling tasks? What challenges arise when scheduling tasks across multiple cores in real-time systems?
76. Outline the main phases in the development of a real-time system. What specific considerations must be taken into account during each phase?
77. How can time-dependent task execution be achieved in real-time systems? Provide an example of a system that requires this type of task execution.
78. Describe the lifecycle of a process in QNX. How does QNX manage process states and transitions?
79. What is a critical section in real-time systems, and what techniques can be used to prevent concurrency issues when accessing shared resources?
80. List and explain three task synchronization mechanisms that are commonly used in real-time operating systems.
81. What happens when a task misses its deadline in a real-time system? How do real-time systems deal with these situations?
82. Compare the differences between hard and soft real-time scheduling. Which type of scheduling would you apply to a streaming media application, and why?

- 83.Explain the concept of memory partitioning in real-time operating systems.
How does it help in managing critical tasks?
- 84.What are the key principles to consider when designing a time-critical system?
Provide examples of design decisions that affect time constraints.
- 85.How is interrupt handling managed in multi-core real-time systems? What challenges arise when interrupts need to be distributed across cores?
- 86.How does a real-time operating system (RTOS) like QNX benefit Internet of Things (IoT) applications? Provide an example of an IoT device using real-time capabilities.
- 87.Explain how multi-threading is used in real-time systems. What are the challenges associated with multi-threading in critical applications?
- 88.What are the advantages and disadvantages of preemptive versus non-preemptive scheduling in real-time systems? In which scenarios would each be preferred?
- 89.How does QNX manage real-time events, and how are events handled in real-time systems to ensure timely responses?
- 90.What is Quality of Service (QoS) in real-time systems? How can QoS be managed to ensure that critical tasks receive the required resources?
- 91.Describe different methods for communication between tasks in real-time systems. How do these methods differ from general-purpose system communication techniques?
- 92.What is the priority inheritance protocol, and how does it solve the priority inversion problem in real-time systems?
- 93.Compare Rate Monotonic Scheduling (RMS) and Earliest Deadline First (EDF) scheduling. In which types of systems is each scheduling algorithm best suited?
- 94.Describe event-driven scheduling in real-time systems. How does it differ from time-driven scheduling, and in what scenarios is it preferable?

95. Explain the difference between deterministic and non-deterministic behavior in real-time systems. Why is determinism crucial for real-time system performance?
96. Tell about the security challenges faced by real-time systems. How do real-time systems ensure security without compromising performance?
97. How is fault detection implemented in real-time systems? Provide an example of a fault detection mechanism used in critical systems.
98. Compare and contrast a real-time operating system (RTOS) with a general-purpose operating system. What features make an RTOS more suitable for time-sensitive applications?
99. How does QNX handle inter-process communication to avoid deadlocks in real-time systems? What strategies can be used to prevent deadlock in multi-process environments?
100. Explain how task scheduling is handled in distributed real-time systems. What challenges arise from managing real-time tasks in a distributed environment?