Concurrency

1. Explain the distinctions among concurrent, parallel, and distributed.

2. Explain the motivation for concurrency. Why do people write concurrent programs? What accounts for the increased interest in concurrency in recent years?

3. Describe the implementation levels at which parallelism appears in modern systems, and the levels of abstraction at which it may be considered by the programmer.

4. What is a race condition? What is synchronization?

5. What is a context switch? Preemption?

6. Explain the concept of a dispatch loop. What are its advantages and disadvantages with resepct to multithreaded code?

7. Explain the distinction between a multiprocessor and a cluster; between a processor and a core.

8. What does it mean for memory in a multiprocessor to be uniform? What is the alternative?

9. Explain the coherence problem for multicore and multiprocessor caches.

10. What is a vector machine? Where does vector technology appear in modern systems?

11. Explain the differences among coroutines, threads, lightweight processes, and heavyweight processes.

12. What is a quasiparallelism?

13. Describe the bag of tricks programming model.

14. What is busy-waiting? What is its principal alternative?

15. Name four explicitly concurrent programming languages.

16. Why don't message-passing programs require explicit synchronization mechanisms.

17. What are the tradeoffs between language-based and library-based implementations of concurrency?

18. Explain the difference between data parallelism and task parallelism.

19. Describe six different mechanisms commonly used to create new threads of control in a concurrent program.

20. In what sense is `fork/join` more powerful than co-begin?

21. What is a thread pool in Java? What purpose does it serve?

22. What is meant by a two-levelthread implementation?

23. What is a ready list?

24. Describe the progressive implementation of scheduling, preemption, and (true) parallelism on top of coroutines.

25. What is mutual exclusion? What is a critical section?

26. What does it mean for an operation to be atomic? Explain the difference between atomicity and condition synchronization.

27. Describe the behavior of a `test_and_set` instruction. Show how to use it to build a spin lock.

28. Describe the behavior of the `compare_and_swap` instructions. What advantages does it offer in comparison to `test_and_set`?

29. Explain how a reader-writer lock differs from an "ordinary" lock.

30. What is a barrier? In what types of programs are barriers common?

31. What does it mean for an algorithm to be nonblocking? What advantages do nonblocking algorithms have over algorithms based on locks?

32. What is sequential consistency? Why is it difficult to implement?

33. What information is provided by a memory consistency model? What is the relationship between hardware-level and language-level memory models.

34. Explain how to extend a preemptive uniprocessor scheduler to work correctly on a multiprocessor.

35. What is a spin-then-yield lock?

36. What is a bounded buffer?

37. What is a semaphore? What operations does it support? How do binary and general semaphores differ?

38. What is a monitor? How do monitor condition variables differ fro semaphores?

39. Explain the difference between treating treating monitor signals as hints and treating them as absolutes.

40. What is a monitor invariant? Under what circumstances must it be guaranteed to hold?

41. Describe the nested monitor problem and some potential solutions.

42. What is deadlock?

43. What is a conditional region? How does it differ from a monitor?

44. Summarize the synchronization mechanisms of Ada 95, Java, and C#. Contrast them with one another, and with monitors and conditional critical regions. Be sure to explain the features added to Java 5.

45. What is transactional memory? What advantages does it offer over algorithms based on locks? What challenges will need to be overcome before it enters widespread use?

46. Describe the semantics of the HPF/Fortran 95 `forall` loop. How does it differ from `do concurrent`?

47. Why might pure functional languages be said to provide a particularly attractive setting for concurrent programming?

48. What are futures? In what languages do they appear? What precautions must the programmer take when using them?

49. Explain the difference between `AND` parrallelism and `OR` parallelism in Prolog.

50. Describe three ways in which processes or threads commonly name their communication partners.

51. What is a datagram?

52. Why, in general, might a `send` operation need to block?

53. What are the three principal synchronization options for the sender of a message? What are the tradeoffs among them?

54. What are `gather` and `scatter` operations in a message-passing program? What are marshalling and unmarshalling?

55. Describe the tradeoffs between explicit and implicit message receipt.

56. What is a remote procedure call (RPC)? What is a stub compiler?

57. What are the obstacles to transparency in an RPC system?

58. What is a rendezous? How does it differ from a remote procedure call?

59. Explain the purpose of a `select` statement in Ada or Go.

60. What semantic and pragmatic challenges are introduced by the ability to "peek" inside messages before they are received?