CSE 2431

Systems 2

SP 23

Midterm 2 Exam Key Points

**B-1 Threads**

1. What factors lead to increased concurrency in applications since about 2005?

2. What two options/approaches are covered in the class slides for building concurrent applications? What are the advantages and disadvantages of each approach?

3. Name the three common programming models for multithreaded applications.

4. What state do multiple threads which are part of the same process share (see below for more specific questions)?

5. Do threads which are part of the same process share a page table base register (PTBR) value (in other words, do their PTBR registers hold the same value)?

6. Do threads which are part of the same process share an instruction pointer (IP) (in other words, do their Ips hold the same value)?

7. Do threads which are part of the same process share a stack segment?

8. Besides sharing an address space (other than stack segments), what else do multiple threads which belong to the same process share (see slide 20)?

9. Be able to name the three common thread operations identified in the slides, and what each operation does (and the comparable operation for processes).

10. Name and understand the characteristics of the two approaches to OS support for threads in the slides (User-level thread libraries, kernel threads). What are the advantages and disadvantages of each of these approaches?

11. Be able to define non-deterministic behavior of programs in terms of the relationship between input and output for the same program run multiple times (Non-determinism means that the same program, run multiple times with the same input, produces different output).

12. What is the definition of a critical section?

13. What does it mean for a critical section to execute atomically?

14. How can races in a critical section lead to non-deterministic results if mutual exclusion is not insured?

15. What is the definition of mutual exclusion for critical sections?

16. Be able to name and explain the operations for locks (allocate/initialize, acquire, release).

**B-2 Locks**

17. Be able to name the lock implementation goals, and briefly explain each (Correctness (3 requirements), Fairness, Performance).

18. When can locks be implemented with interrupts? What are the disadvantages of doing this, even when it is possible? In what kinds of systems can this not be done?

19. What is the problem with implementing a lock with load and store instructions and a shared lock variable?

20. Briefly describe Peterson’s Algorithm for implementation of a lock (software-based solution; shared turn variable and lock array with one element for each thread). Why does Peterson’s Algorithm not work in modern systems (Answer: Cache coherency; different copes of a cached lock value might not have a consistent (equal) value)?

21. Briefly explain the atomic xchg instruction (Test-And-Set or TAS).

22. Briefly explain the atomic CompareAndSwap instruction.

23. To what value is a shared lock variable initialized in order to be used for a lock with TAS or CompareAndSwap?

24. How can TAS or CompareAndSwap be used to acquire a lock using busy waiting?

25. How can a lock be released using TAS or CompareAndSwap?

26. What is a spinlock? Why are basic spinlocks unfair?

27. When can spinlocks be fast?

28. When are spinlocks slow?

29. What is the cost/waste/loss with spinlocks both with and without yield?

**B-3 QLocks+CV**

30. Understand the code for a QLock on slide 18 (you do not need to be able to write the code, but you should be able to answer questions about how it works).

31. What is the purpose of the guard member (guard lock)?

32. Where is spin-waiting used in the implementation of the QLock? Why is using a spinlock here a reasonable approach?

33. If a thread gets the QLock and releases it while other threads are waiting in the queue for the lock, will the thread that released it be able to get the lock again during its time slice (time quantum) before the threads which are waiting can get the lock?

34. If a thread which holds the lock l->lock releases it when other threads are waiting, why does it not just set l->lock to false after removing the first thread waiting in the queue from the queue and unparking it?

35. What is the race condition in the code on slide 18, and in what situation/case can the race occur (how many threads must be waiting, when must a context switch take place for the thread(s) that is/are waiting, and when must the lock be released for the race to occur)?

36. How does the code on slide 20 fix the race condition in the code on slide 18 (How does set\_park() fix the race)?

37. When is spin-waiting or blocking better: How does it depend on whether the system has a uniprocessor or a multi-core processor?

38. How can a two-phase lock as described in the slides (slide 23) provide a good bound on the waiting time for a lock as compared with the optimal waiting time?

39. Be able to explain the two concurrency objectives discussed (mutual exclusion and ordering). Which can be solved with locks? Which can be solved with condition variables/semaphores?

40. Join for threads is an example of which concurrency objective?

41. What is a condition variable? Be able to state what the two operations on condition variables (wait and signal) do. Are they atomic?

42. Why does the correct code for thread\_join and thread\_exit using condition variables on slide 34 work correctly?

43. Be able to explain why the code for multiple producers and multiple consumers for the producer-consumer problem on slide 75 works correctly. If a broadcast signal is not used to wake all threads, why are two condition variables needed?

44. Know the rules of thumb for condition variables on slide 76.

**B-4 Semaphores**

45. Be able to describe the difference between condition variables and semaphores as described on slide 11.

46. Understand the three semaphore operations: allocate/initialize, wait, signal/post. Are they atomic?

47. How can thread\_join and thread-exit be implemented with semaphores (slide 13)?

48. What does it mean to say condition variable/locks and semaphores are equivalent concurrency mechanisms?

49. How can a lock be built from a semaphore (slides 16-17)?

50. How can a semaphore be built from a lock and a CV (slides 19-24)?

51. How does the code on slide 31 work for the producer-consumer problem with a bounded buffer of size N, assuming both semaphores are initialized correctly? How could we increase the concurrency of the solution?

52. Understand the two different types of semaphores we discussed (binary semaphores versus counting semaphores). Which type is used for locks? Which type is used for the empty and full semaphores in the producer-consumer code?

53. Understand the version of the reader-writer problem we discussed (multiple readers can read concurrently, but only one writer can write; any waiting writer must wait on any readers who are reading, and writers can starve).

54. Understand the code for reader-writer locks on slides 33-34.

**B-5 Deadlock**

55. Be able to name the three specific types of concurrency bugs identified in the class slides from the 2008 concurrency study cited (atomicity bugs, ordering bugs, deadlock bugs).

56. Be able to recognize and example of each of the three specific types of bugs identified by the answer to the previous question (see the slides for examples of each type of bug).

57. How can atomicity bugs in concurrent software be fixed/corrected (what mechanism can be used)?

58. How can ordering bugs in concurrent software be fixed/corrected (what mechanisms can be used)?

59. What is the definition of deadlock (slide 10)?

60. What 4 conditions are required in a system in order for deadlock to occur (slide 29)?

61. Be able to state/explain the meaning of each of the 4 required conditions for deadlock (see slides 30, 33, 36 and 39). Be sure you understand each definition; you do not simply want to memorize the definitions!

62. How can each of the 4 required conditions for deadlock be eliminated/removed from a system (be able to say how each can be removed; see the slides immediately following the 4 slide numbers mentioned in the prior point)?

63. How many of the 4 conditions for deadlock must be eliminated/removed in order to prevent deadlock in a system?