Interview Questions

Karl Solomon

December 20, 2024

1 AMZN

1.1 Behavioral Coaching

most often seen for this role: deliver results, then ownership, then be curious

- Behavioral: list of toughest challenges faced on the job
- Good Framework to answer the behavioral
  - Solving problems which positively impact your team, company or external customers
  - A time when you weren't satisfied with the status quo and made improvements
  - Incorporating feedback from a customer to improve a product or service
  - Taking initiative and ownership of problems which fall outside the scope your day to day duties
  - Acquiring new skills in order to solve a problem
  - Figuring out solutions to ambiguous situations (data driven, dive deep, learn and be curious)
  - Learning from past mistakes or outcome which weren't ideal
  - Delivering an important project under a tight deadline
  - What was the scope or impact of across the organization or customer base (the broader the impact the better)
- 12 Leadership Principles:
  1. **Customer Obsession**
     Refusal to cut scope at SYK. Just put in more hours to get the job done on time and with sales' desired features.
  2. **Ownership**
     I was quite literally the software owner at SYK. I did everything from software architecture, component selection, driver development, wrote scheduler/dispatcher, in-field upgrade (USB). Cared highly about finding bugs fast at Nvidia. Integrated coverity, gcov, and unit tests into the RISCV Core RM. This identified numerous bugs and code bloat prior to Ada/Hopper tapeout.
  3. **Invent and Simplify**
  4. **Be right often**
  5. **Learn and be curious**
     Currently exploring some of the more recent c++ features (ranges, modules, in order to improve my ability to write cleaner and more efficient code.)
  6. **Hire and develop the best**
  7. **Insist on highest standards**
  8. **Think big**
  9. **Bias for action**

10. **Frugality**
11. **Earn Trust**
    X-Functional team's trust to be fully responsible for the software. They had confidence that any bug we identified I could resolve. Also Amimon's trust. SYK had a very strong-arm relationship with their clients, however I understood that we were making a lot of unrealistic demands of them. They understood that I could help translate what requests were reasonable and which were not.
12. **Dive deep**
    Autofocus

1.2 Technical Coaching

See HERE for how they hire

- Generic Feedback/Setup
  - Make sure to ask questions
  - Prioritize working solution. But very much interested in optimal approach. Try to identify best data structures/algorithms.
  - Make sure to discuss WHY I'm making specific decisions in my implementation.
  - Many recent ocus on concurrency/multithreading in low-level
  - Virtual white-board/shared text editor. Will not actually be running code
  - See here for details:
- Specifics
  - Concurrency and Multithreading
    * Basic concepts of threads and processes
    * Thread synchronization mechanisms (mutexes, semaphores, locks)
    * Race conditions and deadlocks
    * Atomic operations
  - Thread-Safe Data Structures:
    * Concurrent collections
    * Concurrent collections (e.g., ConcurrentQueue, ConcurrentBag)
    * Lock-free data structures
    * Understanding the differences between thread-safe and non-thread-safe collections
  - Design Patterns for Concurrency:
    * Producer-Consumer pattern
    * Readers-Writer pattern
    * Thread pool pattern
  - Language-Specific Concurrency Features for C++:
    * std::thread
    * <atomic>
  - Callback Mechanisms:
    * Function pointers
    * Delegates (in languages that support them)
    * Lambda expressions
  - Performance Considerations:
    * Understanding the overhead of different synchronization mechanisms
    * Balancing thread safety with performance
  - Testing Multithreaded Code:

    * Techniques for writing unit tests for concurrent code
    * Tools for detecting race conditions and deadlocks

- Distributed Systems Concepts:
  While not directly related to this problem, understanding concepts like eventual consistency and distributed locking can be beneficial
- Algorithms for Concurrent Operations:
  * Compare-And-Swap (CAS) operations
  * Lock-free algorithms
- Memory Models:
  * Understanding memory barriers and volatile variables
  * Cache coherence issues in multi-core systems
- Practice Problems:
  * Implement a thread-safe singleton
  * Create a simple producer-consumer queue
  * Implement a basic thread pool
  * Solve classic concurrency problems like the dining philosophers problem
  * Write a few test cases in addition to the solution
  * Remember, for interviews, it's not just about knowing the solutions, but also being able to explain your reasoning, discuss trade-offs, and analyze the performance and correctness of your solutions.
  * Lastly, be prepared to write code on a whiteboard or in a simple text editor. Practice implementing these concepts without relying on an IDE's features.

1.3 Behavioral implementation

1.4 Technical implementation