SHIVAM KACHHADIYA

+91-9712798452 | kshivamp012@gmail.com | https://linkedin.com/in/shivam012 |
https://github.com/shivamkachhadiya

SUMMARY

C++ systems and low-latency developer specializing in lock-free concurrency, high-frequency order matching engines, and POSIX-based Linux system programming. Experienced in designing high-throughput messaging frameworks and order-book-style matching systems **with emphasis on microsecond-level latency and predictable performance.**

TECHNICAL SKILLS

Languages: C++ (C++17/C++20), C, Python, Java, C#.NET
Systems & Low-Latency: Linux, POSIX APIs, Linux Internals, fork/exec, Signals, IPC, File Descriptors, Memory Management, High-Performance C++
Concurrency: Multithreading, Mutex, Condition Variables, std::atomic, Lock-Free Data Structures, Memory Ordering, Cache Optimization, False-Sharing Avoidance
Networking: TCP/IP, Sockets, High-Throughput Messaging
Tools: Git, CMake, Wireshark
Core CS: Data Structures, Algorithms, Operating Systems, Computer Architecture, STL

PROJECTS

High-Frequency Order Matching Engine | C++17, Multithreading, Concurrency, Low-Latency Systems

• Engineered an exchange-style limit order book implementing price–time priority (FIFO) matching similar to modern stock exchanges
• Designed bid/ask books using unordered_map + deque, enabling O(1) order insertion, cancellation, and execution
• Built a multithreaded producer–consumer architecture for concurrent order ingestion and matching
• Ensured thread-safe execution using mutexes, locks, and condition variables
• Eliminated heap allocations on the hot path, reducing latency spikes and improving execution consistency under load
• Implemented best bid/ask discovery, market orders, trade generation, and order cancellation
• Stress-tested engine with 100K–1M+ simulated orders, validating throughput and scalability
• Achieved microsecond-level average execution latency through data-structure and memory optimizations

Lock-Free Messaging Framework | C++20, Atomics, Systems Programming, Concurrency

• Designed and implemented a cache-aware lock-free Single-Producer Single-Consumer (SPSC) ring buffer enabling ultra-low latency inter-thread communication
• Replaced mutex/condition-variable synchronization with std::atomic and acquire–release memory ordering, eliminating context switching and lock contention

• Achieved 10–20M+ messages/sec throughput with sub-microsecond median latency in single-threaded ping-pong scenarios, ~15–25× better than mutex-based queues under sustained load.
• Implemented cache-line alignment (alignas(64)) and padding to prevent false sharing, significantly improving multi-core scalability
• Enabled zero-copy message passing using preallocated buffers, eliminating heap allocations on the hot path
• Built concurrent benchmarking suite to measure latency, throughput, and CPU utilization against traditional blocking queues
• Focused on systems-level optimization including memory layout tuning, reduced cache misses, and predictable low-latency execution

Linux Command Line Shell | C++17, POSIX, Linux System Programming, Process Management

• Engineered a modular Unix-style command line shell from scratch using low-level POSIX system calls for direct OS interaction
• Implemented process lifecycle management using fork(), execvp(), wait()/waitpid() for spawning and controlling child processes
• Built I/O redirection (> , >>) and pipelines (|) using dup2(), pipe(), file descriptor manipulation, replicating real Linux shell behavior
• Developed built-in commands (cd, pwd, ls, stat, mkdir, rm, cp, touch, cat, history) using getcwd, chdir, opendir, readdir, stat, open, read, write
• Designed command parser + dispatcher architecture for extensibility and clean separation of concerns
• Added background execution & job handling, preventing zombie processes via proper wait handling
• Optimized runtime by minimizing dynamic allocations and using efficient STL containers on hot paths
• Structured project using CMake, modular compilation, and header abstraction for maintainability

EDUCATION

M.Tech in Computer Science – Systems Programming & Low-Latency Computing Focus
Vellore Institute of Technology (VIT), Vellore | CGPA: 8.65 | 2025–Present

Master of Computer Applications (MCA)
SRM Institute of Science & Technology, Chennai | CGPA: 9.26 | 2023–2025

Bachelor of Computer Applications (BCA)
Atmiya University, Rajkot | CGPA: 8.89 | 2020–2023

ACHIEVEMENTS

• Solved 400+ Data Structures and Algorithms problems on LeetCode and GeeksforGeeks
• Secured All India Rank 370 in VITMEE 2025 (Computer Science)
• Active GitHub contributor focused on C++ concurrency and Linux system programming