Senior C++ Interview Notes

1■■ Environment Setup

Aspect	Details
Concepts	Compiler & C++ version, OS/Kernel, CPU Architecture, Libraries, Build systems, Containerization
Commands	g++ --version clang++ --version std::cout << __cplusplus; uname -a lsb_release -a uname -m ldd --version strings $(g++ -print-file-name=libstdc++.so) | grep GLIBCXX
Interview Notes	Explain host environment, reproducibility (Docker, CI/CD), library version mismatches.

2■■ Cross-Compilation & Toolchains

Aspect	Details
Concepts	Host vs Target vs Build, Toolchain, Sysroot, Real-world cross builds.
Commands	arm-linux-gnueabihf-g++ main.cpp -o main_arm x86_64-w64-mingw32-g++ main.cpp -o main.exe cmake -DCMAKE_TOOLCHAIN_FILE=toolchain-arm.cmake ..
Interview Notes	Draw flow: Host → Cross-Compiler → Target. Discuss linking (static vs dynamic).

3■■ Optimization Techniques

Aspect	Details
Concepts	Compiler flags, LTO, PGO, modern C++ (move, constexpr), cache optimization.
Commands	g++ -O2 main.cpp -o main g++ -O3 -flto main.cpp -o main g++ -fprofile-generate main.cpp -o prog && ./prog g++ -fprofile-use main.cpp -o prog
Interview Notes	Trade-offs: speed vs size. Show profiling before/after optimization.

4■■ Profiling & Benchmarking Tools

Aspect	Details
Concepts	CPU, memory, cache, system-level profiling, micro-benchmarks.
Commands	g++ -pg main.cpp -o prog && ./prog && gprof prog gmon.out perf stat ./prog perf record ./prog && perf report valgrind --tool=cachegrind ./prog Google Benchmark: BENCHMARK(Function)
Interview Notes	gprof=function, perf=system, Valgrind=memory, Benchmark=precision.

5■■ Debugging, Sanitizers & Static Analysis

Aspect	Details
Concepts	Memory errors, UB, race conditions, static analysis.
Commands	g++ -fsanitize=address -g main.cpp -o prog g++ -fsanitize=thread -g main.cpp -o prog g++ -fsanitize=undefined -g main.cpp -o prog clang-tidy main.cpp cppcheck main.cpp
Interview Notes	Sanitizers catch runtime bugs, static analysis for CI/CD, trade-offs in speed.

6■■ Embedded & Real-World Scenarios

Aspect	Details
Concepts	Bare-metal, freestanding, linker scripts, cross-build for Windows/ARM, Docker.
Commands	arm-none-eabi-g++ -mcpu=cortex-m4 -mthumb -ffreestanding -nostdlib main.cpp -o firmware.elf file firmware.elf
Interview Notes	Constraints: low RAM, no dynamic memory, debugging via JTAG/Emulators.

7■■ Interview Tips & Workflow

Aspect	Details
Tips	Clarify host/target/build. Always explain profiling→optimization loop. Mention real-world experience.
Workflow Diagram	Host → Toolchain → Compilation → Target Binary → Profiling → Debugging → Optimized App