Valgrind Memory and profiling

valgrind --tool=X synthetic CPU → runs → **Your program** 10-50x slower → reports → **Errors / out file** stderr, *.out.<pid> → read with → **Viewer** ms_print, *_annotate

CONCEPTS

Compile with -g and -O0 or -O1 so traces show file:line; skip -O2 inlining and sanitizers (ASan clashes with valgrind).

Memcheck is the default tool; --tool= picks another. Each tool is a separate run, they do not combine.

Reports show only what the run executed: a bug on an untested code path stays invisible.

RUNNING

valgrind [opts] ./<prog> [args]
Run program under valgrind

valgrind --tool=<name> ./<prog>
Pick memcheck, callgrind, massif

valgrind -q ./<prog>
Quiet: only print errors

valgrind -s ./<prog>
List errors and suppressions used

valgrind --trace-children=yes ./<prog>
Follow exec'd child processes

RUNNING (cont.)

valgrind --num-callers=30 ./<prog>
Deeper stack traces

valgrind --log-file=<file> ./<prog>
Write messages to a file

valgrind --error-exitcode=<n> ./<prog>
Exit code n when errors found

MEMCHECK LEAKS

--leak-check=full
List every leak with stack

--show-leak-kinds=all
Include reachable and indirect

--errors-for-leak-kinds=<set>
Which leaks count as errors

Leak kinds

Kind	Meaning
definitely lost	No pointer left: a real leak
indirectly lost	Only held by lost blocks
possibly lost	Only interior pointers remain
still reachable	Pointer exists at exit
suppressed	Matched a suppression

MEMCHECK LEAKS (cont.)

Fix definitely lost first; indirect ones vanish when their owner is freed. Still reachable is usually harmless.

MEMCHECK ERRORS

--track-origins=yes
Trace uninitialised values to source

Reading an error

Message	Usually means
Invalid read/write	Out of bounds or freed memory
Jump on uninitialised	Branch on unset variable
Invalid free()	Double free or bad pointer
Mismatched free	malloc/ new/new[] mix
Overlap in memcpy	Source and dest overlap
Syscall uninitialised	Unset bytes sent to kernel

MEMCHECK ERRORS (cont.)

The first error is often the cause of later ones; fix top-down and rerun. Read the 'Address ... is N bytes after a block' line.

SUPPRESSIONS

--gen-suppressions=all
Print a suppression per error

--suppressions=<file>
Load extra suppression file

Suppress by narrow frames (fun:/obj:, with ... wildcards) so you hide a library's noise, not your own bugs.

CALLGRIND AND CACHEGRIND

valgrind --tool=callgrind ./<prog>
Record call graph and costs

callgrind_annotate callgrind.out.<pid>
Text report by function

callgrind_annotate --inclusive=yes <file>
Sort by inclusive cost

CALLGRIND AND CACHEGRIND (cont.)

callgrind_annotate --tree=caller <file>
Show callers per function

kcachegrind callgrind.out.<pid>
Browse graph in a GUI

--toggle-collect=<function>
Collect only inside a function

valgrind --tool=cachegrind ./<prog>
Cache and branch simulation

cg_annotate cachegrind.out.<pid>
Report cachegrind results

MASSIF AND DHAT

valgrind --tool=massif ./<prog>
Heap usage over time

ms_print massif.out.<pid>
Graph and snapshot tree

THREAD CHECKERS

valgrind --tool=helgrind ./<prog>
Find races, lock misuse, deadlocks