

Persistent Memory Leak Detector

Operating Systems (Course Code: CS303)

Group Members:

(202351129) Ram Sharma

(202351137) Kush Sonawane

(202351164) Charvi Sri Sravanam

Indian Institute of Information Technology, Vadodara
Gandhinagar Campus

October 2025

Problem Statement, Motivation & Objectives

Problem Statement:

Memory leaks in long-running programs cause gradual performance degradation. Traditional tools detect leaks only during a single run and lose data afterward.

Goal: Design a detector that logs and persists leak data for long-term analysis.

Motivation:

Memory leaks are difficult to trace in complex, long-running software such as servers or background daemons. Conventional tools lose information after each execution, making pattern detection and debugging harder.

Real-World Applications:

- Used in large-scale backend systems (e.g., web servers, database engines) for monitoring memory health.

Objectives:

- Detect dynamically unfreed memory during runtime.
- Store leak information in a persistent SQLite database.
- Compare results across multiple executions.
- Improve reliability and reduce debugging effort.

Methodology

Phase 1 – Monitoring

- Hook malloc/free using LD_PRELOAD.
- Record address, size, PID, timestamp.

Phase 2 – Persistence & Analysis

- Log data into `memory_leak.db`.
- Compare leak reports across runs via Python script.

System Setup

- OS: Ubuntu 24.04 LTS, GCC 13, SQLite 3.45
- Language: C + Python
- Run: `LD_PRELOAD=./memory_hook.so ./test_app`
- Analyze: `python3 analyze_leaks.py summary`

System Architecture

Overview:

The system architecture shows how all components of the **Persistent Memory Leak Detector** interact with each other and the Operating System to detect, log, and analyze memory leaks persistently.

Component Roles:

- **memory_hook.c** — Intercepts `malloc()` and `free()` calls using `LD_PRELOAD` and logs them to the database.
- **memory_server.c** — Acts as the user program whose memory allocations and deallocations are being monitored.
- **analyze_leaks.py** — Compares and reports memory leaks across multiple program runs.
- **live_graph.py** — Displays real-time visualization of memory usage and leak growth.
- **memory_leak.db** — SQLite database storing all logged allocation.

Control Flow:

`memory_server.c` → `memory_hook.c` → `memory_leak.db` →
`analyze_leaks.py` / `live_graph.py`

Implementation

C Layer

- Hooks into `malloc/free` and logs to SQLite.
- Thread-safe with `pthread_mutex`.

Python Layer

- Generates summaries, comparisons, and JSON reports.

Runtime Output

Execution showing memory hook initialization and detected allocations.

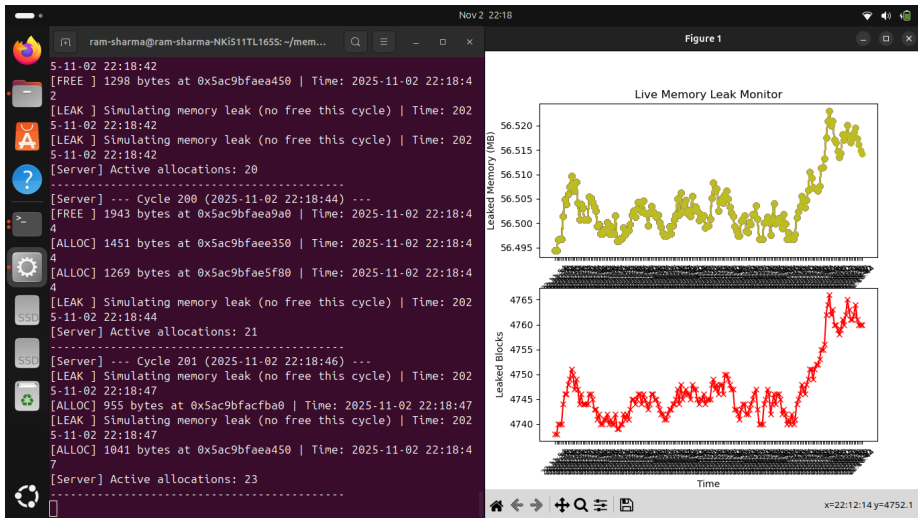
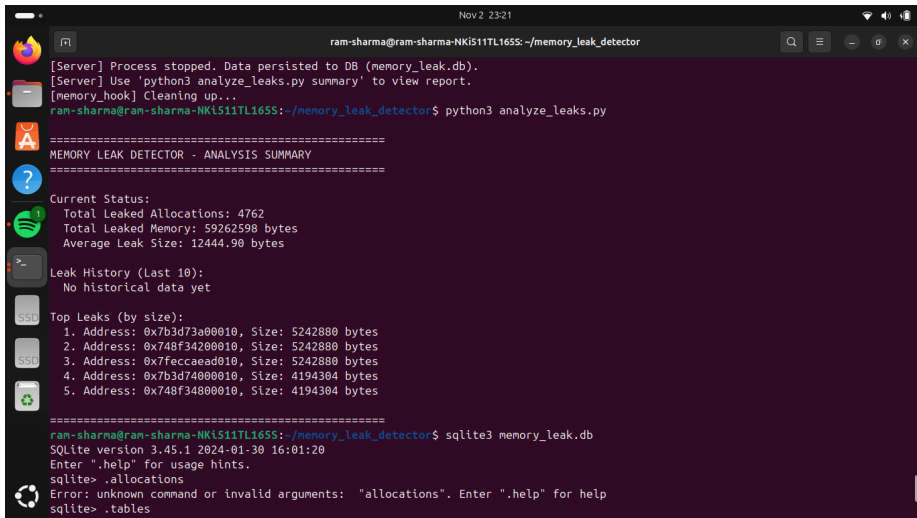


Figure: Terminal showing memory hook initialization + allocations

Leak Summary Analysis

SQLite output summarizing total leaked allocations and memory bytes.



```
Nov 2 23:21
ram-sharma@ram-sharma-NKi511TL165S: ~/memory_leak_detector

[Server] Process stopped. Data persisted to DB (memory_leak.db).
[Server] Use 'python3 analyze_leaks.py summary' to view report.
[memory_hook] Cleaning up...
ram-sharma@ram-sharma-NKi511TL165S:~/memory_leak_detector$ python3 analyze_leaks.py

=====
MEMORY LEAK DETECTOR - ANALYSIS SUMMARY
=====

Current Status:
  Total Leaked Allocations: 4762
  Total Leaked Memory: 59262598 bytes
  Average Leak Size: 12444.90 bytes

Leak History (Last 10):
  No historical data yet

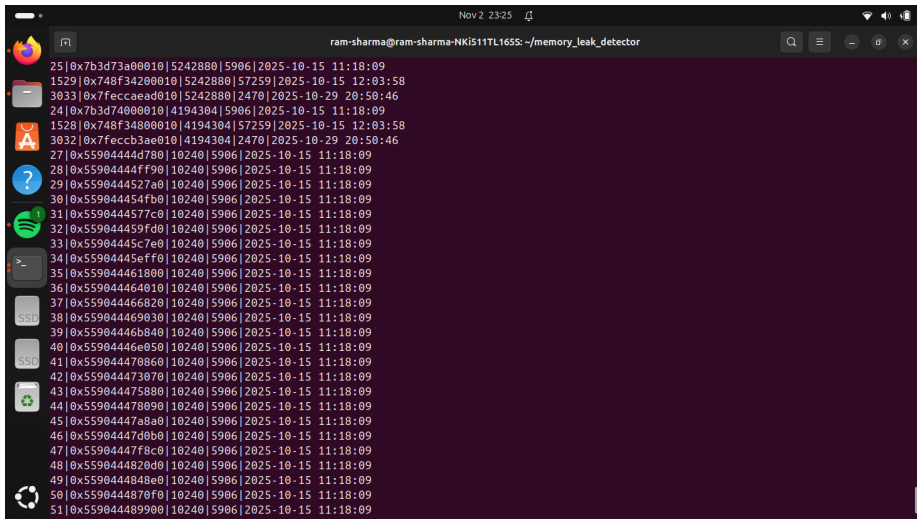
Top Leaks (by size):
  1. Address: 0x7b3d73a00010, Size: 5242880 bytes
  2. Address: 0x748f34200010, Size: 5242880 bytes
  3. Address: 0x7feccaead010, Size: 5242880 bytes
  4. Address: 0x7b3d74000010, Size: 4194304 bytes
  5. Address: 0x748f34800010, Size: 4194304 bytes

=====
ram-sharma@ram-sharma-NKi511TL165S:~/memory_leak_detector$ sqlite3 memory_leak.db
SQLite version 3.45.1 2024-01-30 16:01:20
Enter ".help" for usage hints.
sqlite> .allocations
Error: unknown command or invalid arguments: "allocations". Enter ".help" for help
sqlite> .tables
```

Figure: SQLite query result and summary table from analysis

Persistent Leak Database

Historical leak data stored in `memory_leak.db` across multiple runs.



The screenshot shows a terminal window with a dark purple background. The title bar indicates the user is `ram-sharma` at `ram-sharma-NKi511TL1655` in the directory `~/memory_leak_detector`. The terminal displays a list of 32 rows of data, each representing a memory leak entry. The data is formatted as a pipe-separated string: `id|hex_address|size|timestamp|timestamp`. The first row is `25|0x7b3d73a00010|5242880|5906|2025-10-15 11:18:09`. The second row is `1529|0x748f34200010|5242880|57259|2025-10-15 12:03:58`. The third row is `3033|0x7feccae010|5242880|2470|2025-10-29 20:50:46`. The fourth row is `24|0x7b3d74000010|4194304|5906|2025-10-15 11:18:09`. The fifth row is `1528|0x748f34800010|4194304|57259|2025-10-15 12:03:58`. The sixth row is `3032|0x7fecb3ae010|4194304|2470|2025-10-29 20:50:46`. The seventh row is `27|0x55904444d780|10240|5906|2025-10-15 11:18:09`. The eighth row is `28|0x55904444ff90|10240|5906|2025-10-15 11:18:09`. The ninth row is `29|0x5590444527a0|10240|5906|2025-10-15 11:18:09`. The tenth row is `30|0x559044454fb0|10240|5906|2025-10-15 11:18:09`. The eleventh row is `31|0x55904445577c0|10240|5906|2025-10-15 11:18:09`. The twelfth row is `32|0x559044459fd0|10240|5906|2025-10-15 11:18:09`. The thirteenth row is `33|0x55904445c7e0|10240|5906|2025-10-15 11:18:09`. The fourteenth row is `34|0x55904445eff0|10240|5906|2025-10-15 11:18:09`. The fifteenth row is `35|0x559044461800|10240|5906|2025-10-15 11:18:09`. The sixteenth row is `36|0x559044464010|10240|5906|2025-10-15 11:18:09`. The seventeenth row is `37|0x559044466820|10240|5906|2025-10-15 11:18:09`. The eighteenth row is `38|0x559044469030|10240|5906|2025-10-15 11:18:09`. The nineteenth row is `39|0x55904446b840|10240|5906|2025-10-15 11:18:09`. The twentieth row is `40|0x55904446e050|10240|5906|2025-10-15 11:18:09`. The twenty-first row is `41|0x559044470860|10240|5906|2025-10-15 11:18:09`. The twenty-second row is `42|0x559044473070|10240|5906|2025-10-15 11:18:09`. The twenty-third row is `43|0x559044475880|10240|5906|2025-10-15 11:18:09`. The twenty-fourth row is `44|0x559044478090|10240|5906|2025-10-15 11:18:09`. The twenty-fifth row is `45|0x55904447a8a0|10240|5906|2025-10-15 11:18:09`. The twenty-sixth row is `46|0x55904447d0b0|10240|5906|2025-10-15 11:18:09`. The twenty-seventh row is `47|0x55904447f8c0|10240|5906|2025-10-15 11:18:09`. The twenty-eighth row is `48|0x5590444820d0|10240|5906|2025-10-15 11:18:09`. The twenty-ninth row is `49|0x5590444848e0|10240|5906|2025-10-15 11:18:09`. The thirtieth row is `50|0x5590444870f0|10240|5906|2025-10-15 11:18:09`. The thirty-first row is `51|0x559044489900|10240|5906|2025-10-15 11:18:09`. The terminal window has a sidebar on the left with various application icons and a top bar with system status icons.

```
25|0x7b3d73a00010|5242880|5906|2025-10-15 11:18:09
1529|0x748f34200010|5242880|57259|2025-10-15 12:03:58
3033|0x7feccae010|5242880|2470|2025-10-29 20:50:46
24|0x7b3d74000010|4194304|5906|2025-10-15 11:18:09
1528|0x748f34800010|4194304|57259|2025-10-15 12:03:58
3032|0x7fecb3ae010|4194304|2470|2025-10-29 20:50:46
27|0x55904444d780|10240|5906|2025-10-15 11:18:09
28|0x55904444ff90|10240|5906|2025-10-15 11:18:09
29|0x5590444527a0|10240|5906|2025-10-15 11:18:09
30|0x559044454fb0|10240|5906|2025-10-15 11:18:09
31|0x55904445577c0|10240|5906|2025-10-15 11:18:09
32|0x559044459fd0|10240|5906|2025-10-15 11:18:09
33|0x55904445c7e0|10240|5906|2025-10-15 11:18:09
34|0x55904445eff0|10240|5906|2025-10-15 11:18:09
35|0x559044461800|10240|5906|2025-10-15 11:18:09
36|0x559044464010|10240|5906|2025-10-15 11:18:09
37|0x559044466820|10240|5906|2025-10-15 11:18:09
38|0x559044469030|10240|5906|2025-10-15 11:18:09
39|0x55904446b840|10240|5906|2025-10-15 11:18:09
40|0x55904446e050|10240|5906|2025-10-15 11:18:09
41|0x559044470860|10240|5906|2025-10-15 11:18:09
42|0x559044473070|10240|5906|2025-10-15 11:18:09
43|0x559044475880|10240|5906|2025-10-15 11:18:09
44|0x559044478090|10240|5906|2025-10-15 11:18:09
45|0x55904447a8a0|10240|5906|2025-10-15 11:18:09
46|0x55904447d0b0|10240|5906|2025-10-15 11:18:09
47|0x55904447f8c0|10240|5906|2025-10-15 11:18:09
48|0x5590444820d0|10240|5906|2025-10-15 11:18:09
49|0x5590444848e0|10240|5906|2025-10-15 11:18:09
50|0x5590444870f0|10240|5906|2025-10-15 11:18:09
51|0x559044489900|10240|5906|2025-10-15 11:18:09
```

Figure: Database table view showing persistent stored data

References

- Linux Man Pages — LD_PRELOAD
- SQLite Documentation — <https://www.sqlite.org>
- Valgrind Developers, Memcheck: A Memory Error Detector, 2024.
Available: <https://valgrind.org/docs/manual/mc-manual.html>
→ Conceptual comparison for professional memory leak detection methodology.