**TESTING DOCUMENTATION FOR**

**LINUX MULTI-THREADED CLIENTSERVER USING SHARED MEMORY**

**Contents**

[Overview 3](#_Toc167976550)

Test Plan....................................................................................................................3

[Testing Covers 3](#_Toc167976555)

[Testing Approach 4](#_Toc167976556)

Test Cases..................................................................................................................4

Test Execution...........................................................................................................5

[Conclusion 5](#_Toc167976557)

# **Overview**

This documentation provides an overview of a multi-threaded server-client application designed to generate random numbers and share them with multiple clients using shared memory and semaphores. The server generates random numbers and stores them in shared memory, while clients read these numbers from the shared memory. The synchronization between the server and clients is managed using semaphores.

**Test Plan**

**Objectives:**

* Verify that the server can handle multiple client connections concurrently.
* Ensure the shared memory is correctly updated and accessed by multiple clients.
* Validate the system's performance and reliability with both typical and edge case scenarios.

**The testing covers:**

* Unit Testing: Testing individual functionalities (connection handling, shared memory access)
* System Testing: End-to-end testing of the entire client-server system to ensure all requirements are met.

**Testing Approach:**

* **Define Test Cases**: Detailed test cases for the server and client based on the requirements.
* **Prepare Test Data**: Create sample data that covers all possible scenarios including edge cases.
* **Execute Tests**: Run the tests and record the results.
* **Bug Reporting**: Document any defects found during testing.
* **Regression Testing**: Re-test after bug fixes to ensure issues are resolved without introducing new problems.

**Test Cases**

**Server-Client Connection**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case # | Description | Input | Expected Result | Status (P/F) |
| TC1 | Connect a single client to the server | Client connection request | Client connected, server acknowledges | P |
| TC2 | Connect multiple clients to the server concurrently | Multiple client connections | All clients connected, server acknowledges each | P |
| TC3 | Disconnect a client | Client disconnection request | Client disconnected, server acknowledges | P |

**Shared Memory Access**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case # | Description | Input | Expected Result | Status (P/F) |
| TC4 | Write to shared memory by one client | Client sends message | Message written to shared memory | P |
| TC5 | Read from shared memory by another client | Client reads message | Correct message read from shared memory | P |
| TC6 | Concurrent access to shared memory | Multiple clients send messages | Messages written and read correctly with synchronization | P |

**System Testing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case # | Description | Input | Expected Result | Status (P/F) |
| ST1 | End-to-end test with typical data | Client messages | Correct messages exchanged and shared memory updated | P |
| ST2 | End-to-end test with edge cases | Edge case messages | System handles edge cases correctly | P |
| ST3 | Performance test with large data set | High volume of messages | System performs efficiently without errors | P |

**Test Execution**

· **Setup Test Environment**: Configure the test environment, ensuring access to the server, clients, shared memory, and necessary tools.

· **Run Test Cases**: Execute each test case, recording the results in the provided status column.

· **Document Defects**: For any test case that fails, document the defect, including steps to reproduce, expected vs. actual results, and severity.

· **Review and Fix**: Review the defects with the development team, implement fixes, and re-test as necessary.

· **Final Validation**: After all defects are resolved, perform a final round of testing to ensure the system is ready for production.

### **Conclusion**

The testing documentation provides a comprehensive plan to ensure the client-server system functions as expected. By following this plan, we aim to deliver a robust and reliable system.