Valgrind Memory Debugging Lab Manual

Introduction

Valgrind is a powerful tool that helps developers detect memory management and threading bugs. In this lab, you will practice using Valgrind to identify and fix common memory-related errors in C and C++ programs.

Objectives

- Detect memory leaks, invalid memory access, use-after-free, and double-free errors.
- Understand Valgrind's output messages.
- Practice fixing memory issues in both C and C++.

Prerequisites

- Basic knowledge of C and C++ programming.
- GCC and G++ installed.
- Valgrind installed (Ubuntu: sudo apt install valgrind).

Folder Structure

valgrind_lab/

|-- Makefile

|-- README.md

|-- bad_access.c

|-- memory_leak.c

|-- use_after_free.c

|-- double_free.c

|-- bad_access.cpp

|-- memory_leak.cpp

|-- use_after_delete.cpp

|-- double_delete.cpp

Instructions

1. Setup

- Download and unzip the provided lab archive.
- Navigate into the lab folder.

Valgrind Memory Debugging Lab Manual

2. Compile Programs

make

3. Run Programs with Valgrind

For C programs: make PROGRAM=bad_access valgrind-c

For C++ programs: make PROGRAM=bad_access valgrind-cpp

4. Analyze Valgrind Output

- Look for errors such as "Invalid write," "Invalid read," "Memory leak detected," "Use of uninitialized value," etc.
- Identify the exact lines causing errors.

5. Fix Errors

- Modify the programs to eliminate memory issues.
- Recompile and re-run under Valgrind.

6. Clean Up

make clean

Learning Outcomes

- Use Valgrind to identify memory leaks and invalid memory access.
- Interpret Valgrind reports effectively.
- Write safer C and C++ code with proper memory management.

Solution Key

Corrected C and C++ programs included fixing bad access, memory leaks, use-after-free, and double-free errors by properly managing memory (bounds checking, freeing, setting pointers to NULL).