User Manual

Introduction

1.1 Overview

The Multi-Agent Robot System is designed to simulate the collaboration of 50 robots within a confined space, estimating the width of a randomly placed exit. The robots communicate their estimates, calculate an average exit width, and display the results.

1.2 Purpose

The purpose of this system is to demonstrate inter-process communication, thread management, and shared memory usage in a multi-agent environment.

1.3 System Requirements

- C++ Compiler
- POSIX Threads Library
- System with IPC and Semaphore Support

1.4 Installation

- Clone the repository or download the source code.
- Compile the code using a C++ compiler.
- g++ -o 1 Project.cpp -pthread
- Run the executable.
- ./1

2. Getting Started

2.1 Running the Program

Execute the compiled program in the terminal. The program will simulate robot placements, estimate exit widths, and display the results.

2.2 Input Parameters

There are no specific input parameters. The system generates random robot positions and exit width for each run.

3. System Operation

3.1 Robot Placement

- Robots are randomly placed within a 100 by 100 room.
- The exit is randomly positioned with a width ranging between 16 to 26 units.

3.2 Exit Width Estimation

- Each robot estimates the exit width based on its distance from the exit.
- Estimation accuracy fluctuates within a defined range.

3.3 Inter-Process Communication

- Shared memory and message queues facilitate communication among robots.
- Robots send and receive estimated exit widths via IPC.

3.4 Thread Management

- Threads are used to calculate average exit width with neighboring robots.
- Thread synchronization ensures accurate width calculations.

4. Results

4.1 Displaying Robot Estimates

- Estimated exit width for each robot is displayed during program execution.

4.2 Average Exit Width Calculation

- The program calculates and displays the average exit width based on robot estimates.
- The true exit width is displayed for comparison.

5. Troubleshooting

5.1 Common Issues

- Compilation Errors: Ensure a C++ compiler is installed.
- Runtime Errors: Check system support for IPC and semaphores.

5.2 Debugging Tips

- Use print statements for debugging.
- Review the console output for error messages.

6. Appendix

6.1 Glossary

- IPC: Inter-Process Communication
- POSIX: Portable Operating System Interface
- Semaphore: Synchronization mechanism
- Thread: Lightweight process

6.2 References

- POSIX Threads Documentation
- C++ Standard Library Documentation

This user manual provides an overview of the Multi-Agent Robot System, including installation instructions, system operation, and troubleshooting tips. For more details on specific functionalities, refer to the source code comments and related documentation.