LINUX MULTI-THREADED CLIENTSERVER USING SHARED MEMORY

Contents

Overview

An overview of a multi-threaded server-client program that uses shared memory and semaphores to generate random integers and distribute them to many clients is given in this documentation. While clients read the random numbers from shared memory, the server generates them and saves them there. Semaphores are used to control the synchronization between the server and clients.

Objectives

The objective of this application is to demonstrate the use of shared memory and semaphores for inter-process communication in a multi-threaded environment. The server generates random numbers continuously and stores them in shared memory. Multiple clients connect to the server and read the random numbers from the shared memory, ensuring synchronization using semaphores.

Declaration and Includes

Global Variables:

sem_t *sem :- Declares semaphore variable

int server_fd :- Server file descriptor

client_count :- Counter to count clients

Includes:

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <semaphore.h>
```

Code Functionality

Server Application (server.c)

Functionality:

The server application listens for incoming client connections.

It generates random two-digit numbers every second and stores them in shared memory.

Multiple client-handling threads are created to manage client connections. Each thread handles up to one client, with new threads created for additional clients.

Components:

generate_random_numbers: Thread function responsible for generating random numbers and storing them in shared memory.

handle_clients: Thread function responsible for handling multiple client connections. It accepts new client connections and manages them.

Usage:

Compile the server code: gcc -o server server.c

Run the server: ./server

Client Application (client.c)

Functionality:

The client application connects to the server and continuously reads random numbers from shared memory.

It prints the random numbers received from the server.

Components:

main: The main function of the client application. It connects to the server and reads random numbers from shared memory.

Usage:

Compile the client code: gcc -o client client.c

Create a shell script with n number of clients

Run the shell with n clients: ./test.sh

Debug

Use the command in terminal ps -ef | grep server then pstree -p <pid>

Use the command in terminal ipcs -s to check semaphore identifier.

Conclusion

This multi-threaded client-server program shows how to utilize semaphores for synchronization and shared memory for IPC in a Linux environment. It offers a reliable way to manage several client connections while maintaining data consistency and appropriate resource allocation.