CSCI 247 Computer Systems I
Assignment Exercise 3

Objectives

Gain familiarity with POSIX Shared Memory APIs.

Submitting Your Work

Submit your C program files along with your analysis of task 5 as the Assignment Exercise 3 Submission item on the course web site. You will submit 3 files – “shm.h”, “server.c” and “client.c”. Skeleton version of these files are available to you. You must submit your program by the deadline in 2 weeks.

Background

In Assignments 1 and 2, you gained familiarity with communicating between threads within the same process. Inter-thread communication within the same process is relatively straight forward, because the Data segment for the process is shared between all threads within that process. The only challenge is to ensure you provide a means for mutual exclusion when the same data is accessed from multiple threads.

Inter-process communication (IPC) refers to communication between different process spaces. In this situation, you don’t have the luxury of sharing a common data segment. Of the many inter-process communication methods employed in software, the easiest and most efficient method when both processes belong to a common host machine is referred to as “Shared memory”.

Your task in this assignment is to explore the POSIX shared memory APIs available in the Linux OS.

Assignment Tasks

Task 1: Populate shm.h

In the file “shm.h”, define an “enum” called “StatusEnum” which has the following enumerations: INVALID, VALID and CONSUMED.

Also in the file “shm.h” typedef a structure called “ShmData” that includes the enum above and an additional “int” variable type called “data”.

Task 2: Populate Server.c

This file contains the server code to create and populate the shared memory. Research the APIs “shm_open”, “ftruntcate”, “mmap”, “munmap”, “close”, and “shm_unlink” to fully populate this file.

Task 3: Populate Client.c

This file contains the client code to map and read the contents of shared memory. It uses some of the same APIs used in the server code above. Populate this file with all the APIs listed in the comments in this file.

Task 4: Test your inter-process communication

Compile and run the Server and pass an argument in the command line. Compile and run the Client and confirm you are receiving the argument passed to the server in your client.

Task 5: Postulate how you could use this IPC

Based on your knowledge of timers from the previous assignment, postulate (by providing a small analytical paragraph) how you would use this method of IPC where a server is a producer of data and a client is a consumer of data, to periodically get the data from the producer at regular intervals.

Grading guidelines

- Working code (4 points)
- Conforming to good coding practice (4 points)
- Analysis (2 points)