SUSTech CS302 OS Lab6 Report

Title: Producer-Consumer Problem

Student: Name: __唐千栋______, ID: _11612730_______

Time: 2019 Year 4 Month 8 Day

Experimental Environment: linux ubuntu 16.04

Objective: Master the synchronization & mutual exclusion algorithm, understand the producer-consumer models, and understand the reader-writer problem. Understand the multi-threaded concurrent execution mechanism. Understand the synchronization and mutexes between threads.

Deadline: **11:59 AM, 2019-04-10**

Summit by: Blackboard

Task：

Task 1.Understand the source codes

Task 2.Edit and modify the source codes

Experiments:

1. fundamental：

   Function(功能) of APIs:

   - **pthread_create: start a new thread in the calling process.**
   - **pthread_join: wait for the thread specified by thread to terminate**
   - **pthread_mutex_lock: locks a mutex object, which identifies a mutex. If the mutex is already locked by another thread, the thread waits for the mutex to become available.**

¤ **pthread_cond_wait: atomically release mutex and cause the calling thread to block on the thread variable cond**

¤ **pthread_cond_signal: use to unblock threads blocked on a condition variable cond.**

¤ **pthread_mutex_unlock: unlocks the mutex mutex. The mutex should be owned by the calling thread.**

2. Producer-Consumer Problem

Are the data that consumers read from the buffer are produced by the same producer? No, there could be multiple producers producing to the buffer. The consumer read could be produced by multiple producers.

¤ What is the order of the consumer's read operations and the producer's write operations, and their relationship producer’s write operation would happens before consumer’s read operations, since consumer’s read operations would not happens if the buffer is empty. Consumer’s read operations would happen only when the buffer is not empty. Producer’s write operations would happen only when the buffer is not full.

¤ Briefly describe the result of the program: The program would crate two threads, a producer thread and a consumer thread. It uses a mutex lock and two conditional variables to ensure the thread synchronization. When the producer gets the mutex lock, if buffer is full, it would wait and unlock mutual exclusion. If buffer is empty, it would wake up consumer. It would write a random character to buffer. If consumer gets the mutex lock, if the buffer is empty, it would wait and unblock mutex lock. If buffer is full, it would wake up producer. It would read a character from buffer.

¤ What queue is used in this program, and its characteristics? It uses circular queue. It saves space, and it does not require moving element. It only require moving pointer.

¤ Briefly describe the mutual exclusion mechanism of this program: It uses a mutex lock and two conditional variables. It uses the mutex lock to ensure that when producer is writing, consumer would not read. And it uses a conditional variable to ensure producer would not write when buffer is full. And the other conditional variable ensures consumer would not read when buffer is empty.

3. Readers-Writers Problem

¤ What interfaces of semaphore are used in this program? sem_init, sem_wait, sem_post

¤ What are these interfaces used for? sem_init initialize the unnamed semaphore at the address pointed by sem. sem_wait decrements the semaphore pointed by sem, if the semaphore > 0, then the decrement proceeds, and the function return. If the semaphore currently has the value zero, then the call blocks until either the semaphore > 0. sem_post increment the semaphore pointed by sem. If the semaphore > 0, then another process or thread in sem_wait wake up.

¤ Can readers read at the same time? Why? Yes, because reader reads together would not cause side effect.

¤ Can writers write at the same time? Why? No, because writer write together would cause the file content inconsistent.

¤ What is the performance of the reader's synchronous reading? They would read the same content.

¤ After one writer writes, can the next writer write before one reader read? Why? Yes, because writer write before reader read do not cause inconsistent problem.

Conclusion:

Submission:

-lab6_studentID	(directory)
---lab6_report_studentID.pdf	(pdf version report)
---read.c	(c file)
---write.c	(c file)

Zip the directory with the same name and submit it