Project 5: 设计一个线程池 & 生产者消费者问题

Chentao Wu吴晨涛
Professor
Dept. of CSE, SJTU
wuct@cs.sjtu.edu.cn

课程目标

- 使用Pthreads设计一个线程池，支持client.c调用相关API
- 使用Pthreads解决有界缓冲区的生产者消费者问题

线程池-client.c

- client.c 在 final-src-osc10e/ch7/project-1/posix/文件夹下

- 线程池的用户使用下面的API：
  - void pool_init();

    --初始化线程池
  - int pool_submiit(void (*somefunction)(void *p), void *p);

    --向线程池传递任务，执行somefunction(void *p)函数，其中 *p 为somefunction() 函数的参数
  - void pool_shutdown(void);

    --当所有任务完成时终止线程

线程池-client.c

- client.c 执行的是add函数，传入参数是data结构体，add函数将data结构体中的两个整数相加并输出。所以是pool_submit(&add, &work);

```c
#include <stdio.h>
#include <unistd.h>
#include "threadpool.h"
struct data
{
    int a;
    int b;
};
void add(void *param)
{
    struct data *temp;
    temp = (struct data*)param;
    printf("I add two values %d and %d result = %d\n",temp->a, temp->b, temp->a + temp->b);
}
int main(void)
{
    // create some work to do
    struct data work;
    work.a = 5;
    work.b = 10;
    // initialize the thread pool
    pool_init();
    // submit the work to the queue
    pool_submit(&add,&work);
    // may be helpful
    //sleep(3);
    pool_shutdown();
    return 0;
}
```

线程池-threadpool.c

- threadpool.c 在 final-src-osc10e/ch7/project-1/posix/文件夹下

- 需要对用户调用的函数进行实现：
  - pool_init()函数：创建线程，初始化互斥锁和信号量
  - pool_submit()函数：传入任务，线程池将任务放入任务队列，涉及入队操作
  - worker()函数：在队列非空时，将队头任务出队，调用execute()函数进行执行
  - pool_shutdown()函数：终止线程，释放线程池

- 保证同一时间只有一个线程修改任务队列，需要用到互斥锁 pthread_mutex_t
- 保证任务队列非空才出队、非满才入队，需要用到信号量 sem_t

生产者消费者问题

- 要求使用标准计数信号量和互斥锁来解决生产者-消费者问题。生产者和消费者作为单独的线程运行，将数据项移入或移出缓冲区

- main()的命令行参数包括睡眠时间、生产者线程数量和消费者线程数量
- main()函数包括以下六个主要步骤：

```
#include "buffer.h"

int main(int argc, char *argv[]) {
  /* 1. Get command line arguments argv[1],argv[2],argv[3] */
  /* 2. Initialize buffer */
  /* 3. Create producer thread(s) */
  /* 4. Create consumer thread(s) */
  /* 5. Sleep */
  /* 6. Exit */
}
```

Figure 7.15 Outline of skeleton program.

生产者消费者问题

- 生产者线程和消费者线程一直循环，直到被主线程取消
- while循环体内部，随机休眠一段时间，然后执行插入和删除
- 用信号量 empty 和 full 检查缓冲区是否有空位可插入/有数据项可取出，用互斥锁 mutex 保证一次只有一个线程可以访问缓冲区

```
#include <stdlib.h> /* required for rand() */
#include "buffer.h"

void *producer(void *param) {
   buffer_item item;

   while (true) {
      /* sleep for a random period of time */
      sleep(...);
      /* generate a random number */
      item = rand();
      if (insert_item(item))
         fprintf("report error condition");
      else
         printf("producer produced %d\n",item);
}

void *consumer(void *param) {
   buffer_item item;

   while (true) {
      /* sleep for a random period of time */
      sleep(...);
      if (remove_item(&item))
         fprintf("report error condition");
      else
         printf("consumer consumed %d\n",item);
}
```

Figure 7.16 An outline of the producer and consumer threads.

作业及评分

自行阅读课本第七章的 Programming Projects 部分，并完成以下两个任务，完成后共计13分。

- （课本习题 5分）使用Pthread API实现线程池，将threadpool.c和client.c补充完整，线程池可以为client的add任务分配线程进行执行
- （课本习题 5分）使用Pthread API的标准计数信号量和互斥锁来解决生产者-消费者问题
- （报告 2分）做一个简单的报告解释你的代码，报告建议不超过2页（防内卷）
- （Bonus 1分）本题线程池的核心线程数固定为3。请查阅相关资料，简单讨论线程池的核心线程数量过大或者过小有没有影响？如何合理地设置线程池的核心线程数量？