|  |
| --- |
| 北 京 邮 电 大 学  实 验 报 告  课程名称： 操作系统原理  院系：计算机学院（国家示范性软件工程学院）  班级： 2021211318  姓名： 万志恒  学号： 2021212413  教师： 赵方 金昕  成绩：  2023年 春季学期 |
| 实验目的 学习 pthread 库函数的使用  掌握共享锁和信号量的使用方法  使用多线程实现生产者-消费者问题 实验环境 硬件：PC微型计算机、8核、16.0G内存、467G硬盘  软件：Vmware、Ubuntu操作系统、Vi、GCC、GDB   1. **实验任务及内容**   生产者写入缓冲区和消费者从缓冲区读数的具体流程，生产者首先要获得互斥锁，并且判断写指针+1 后是否等于读指针，如果相等则进入等待状态，等候条件变量 notfull；如果不等则向缓冲区中写一个整数，并且设置条件变量为 notempty，最后 释放互斥锁。消费者线程与生产者线程类似。流程图如下：    生产者-消费者实验源代码结构流程  源程序：  #include <stdio.h>  #include <stdlib.h>  #include <unistd.h>  #include <pthread.h>  #include <time.h>  #include <string.h>  #define BUFFER\_SIZE 10  #define OVER -1  struct Buffer{  int data[BUFFER\_SIZE]; //缓冲区数组  int readPos; //读的位置  int writePos;//写的位置  pthread\_mutex\_t lock; //互斥锁  pthread\_cond\_t notEmpty; //非空信号  pthread\_cond\_t notFull; //非满信号  }buffer;  void writeData(struct Buffer\* buffer, int data);  int readData(struct Buffer\* buffer);  void\* ProducerThread();  void\* ConsumerThread();  void initBuffer(struct Buffer\* buffer);  int main(){  pthread\_t producerThread;  pthread\_t consumerThread;  void \*retval;    initBuffer(&buffer);  pthread\_create(&producerThread, NULL, ProducerThread, NULL);  pthread\_create(&consumerThread, NULL, ConsumerThread, NULL);  pthread\_join(producerThread, &retval);  pthread\_join(consumerThread, &retval);    printf("Task Over.\n");  }  void writeData(struct Buffer\* buffer, int data){  pthread\_mutex\_lock(&buffer->lock); //获取互斥锁  while((buffer->writePos + 1) % BUFFER\_SIZE == buffer->readPos){  pthread\_cond\_wait(&buffer->notFull, &buffer->lock); /\*等待状态变量 b->notFull，不满则跳出阻塞\*/  }  /\*写入数据\*/  buffer->data[buffer->writePos] = data;  if(data!=OVER){  printf("Producer: Put data %d to buffer.\n", data);  }else printf("Producer: end of input.\n");  buffer->writePos = (buffer->writePos+1)%BUFFER\_SIZE;  pthread\_cond\_signal(&buffer->notEmpty); //设置缓冲区非空信号  pthread\_mutex\_unlock(&buffer->lock); //释放互斥锁  }  int readData(struct Buffer\* buffer){  int data;  pthread\_mutex\_lock(&buffer->lock); //获取互斥锁  while(buffer->writePos == buffer->readPos){  pthread\_cond\_wait(&buffer->notEmpty, &buffer->lock); /\*等待状态变量 b->notEmpty，不空则跳出阻塞\*/  }  /\*提取数据\*/  data = buffer->data[buffer->readPos];  if(data!=OVER){  printf("Consumer: Read data %d from buffer.\n", data);  }else printf("Consumer: end of read.\n");  buffer->readPos = (buffer->readPos+1)%BUFFER\_SIZE;  pthread\_cond\_signal(&buffer->notFull); //设置缓冲区非信号  pthread\_mutex\_unlock(&buffer->lock); //释放互斥锁  return data;  }  void\* ProducerThread(){  int i;  for(i=0; i<1000; i++){    //生产者写入随机数  writeData(&buffer, i);  }  writeData(&buffer, OVER);//结束写入  }  void\* ConsumerThread(){  int data;  while(1){  data=readData(&buffer);  if(data == OVER) break;  }  }  void initBuffer(struct Buffer\* buffer){  memset(buffer->data, 0, sizeof(buffer->data));  buffer->readPos=0;  buffer->writePos=0;  pthread\_mutex\_init(&buffer->lock, NULL);  pthread\_cond\_init(&buffer->notEmpty, NULL);  pthread\_cond\_init(&buffer->notFull, NULL);  }  运行结果：   实验心得及体会 实践了互斥锁和条件变量的使用，对并发和互斥有了更深入的了解与认识；接触了许多Linux提供的线程API，熟悉了对这些并发相关API的使用。对线程的操作有了更深的认识。 |