四川大学计算机学院、软件学院

实验报告

学号：2022141460155 姓名： 林诺晗 专业：计算金融 班级：计金班 第 11 周

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 课程名称 | 操作系统实验 | | 实验课时 | 2 |
| 实验项目 | 进程同步实验 | | 实验时间 | 2024.05.09 |
| 实验目的 | 1）加强对进程同步和互斥的理解，学会使用信号量解决资源共享问题。  2）熟悉Linux 进程同步原语。  3）掌握信号量wait/signal 原语的使用方法，理解信号量的定义、赋初值及wait/signal操作。 | | | |
| 实验环境 | VMware Workstation 17 Pro  Ubuntu 22.04.3 LTS | | | |
| 实验内容（算法、程序、步骤和方法） | 编写程序，使用Linux操作系统中的信号量机制模拟实现生产者-消费者问题。设有一个生产者和一个消费者，缓冲区可以存放产品，生产者不断生成产品放入缓冲区，消费者不断从缓冲区中取出产品，消费产品。  **实验要求**  -使用两个线程来模拟生产者和消费者  -使用pthread库提供的线程操作，需要包含头文件pthread.h  -使用POSIX的无名信号量机制，需要包含头文件semaphore.h  **实验相关原理：**  **Linux进程/线程同步简介**  互斥锁：保证资源独占  自旋锁：与互斥量类似，但是等待自旋锁时，进程不会释放CPU，而是一直占用CPU。  条件变量：等待和通知，一般与互斥锁合用  读写锁：与互斥锁类似，不过读写锁允许更高的并行性  记录锁（文件锁）：在读写锁的基础上进一步细分被锁对象的粒度  信号量：条件变量的升级版  **Linux信号量机制**  POSIX信号量  --有名信号量：  基于内存的信号量  常用于多线程间的同步，也可用于相关进程间的同步  用于进行进程同步时，需要放在进程间的共享内存区中  --无名信号量：  通过IPC名字进行进程间的同步  特点是把信号量值保存在文件中  既可用于线程，也可用于相关进程，甚至是不相关的进程  System V信号量  使用相对复杂，在内核中维护  **POSIX信号量操作函数**  #include<semaphore.h>    **System V信号量操作**  #include <sys/sem.h>  int semget(key\_t key, int num\_sems, int sem\_flags);  int semctl(int sem\_id, int sem\_num, int cmd, union semun arg);  int semop(int sem\_id, struct sembuf \*sops, size\_t nsops);  使用步骤：  使用semget()函数创建或获取信号量。不同进程通过使用同一个信号量键值来获得同一个信号量。  使用semctl()函数的SETVAL操作初始化信号量。  使用semop()函数进行信号量的PV操作，这是实现进程同步或互斥的核心工作。  如果不需要信号量，则从系统中删除它，此时使用shmctl()函数的IPC\_RMID操作。  **源代码：**  #include <stdio.h>  #include <pthread.h>  #include <semaphore.h>  #include <unistd.h>  #define BUFFER\_SIZE 10  int buffer = 0; // 缓冲区  sem\_t empty; // 缓冲区为空的信号量  sem\_t full; // 缓冲区为满的信号量  pthread\_mutex\_t mutex; // 互斥锁，保护缓冲区  void\* producer(void\* arg) {  printf("input something to buffer:\n");  scanf("%d", &buffer); // 从标准输入读取一个整数作为产品  sem\_wait(&empty); // 等待缓冲区变空  pthread\_mutex\_lock(&mutex); // 进入临界区  printf("Produced item: %d\n", buffer);  sem\_post(&full); // 通知消费者缓冲区有产品  pthread\_mutex\_unlock(&mutex); // 离开临界区  return NULL;  }  void\* consumer(void\* arg) {  while (1) {  sem\_wait(&full); // 等待缓冲区变满  pthread\_mutex\_lock(&mutex); // 进入临界区  printf("read product from buffer:%d\n", buffer); // 读取并输出产品  sem\_post(&empty); // 通知生产者缓冲区变空  pthread\_mutex\_unlock(&mutex); // 离开临界区  break;  }  return NULL;  }  int main() {  pthread\_t p, c;  // 初始化信号量和互斥锁  sem\_init(&empty, 0, BUFFER\_SIZE);  sem\_init(&full, 0, 0);  pthread\_mutex\_init(&mutex, NULL);  // 创建生产者和消费者线程  pthread\_create(&p, NULL, producer, NULL);  pthread\_join(p, NULL);  pthread\_create(&c, NULL, consumer, NULL);  pthread\_join(c, NULL);  // 销毁信号量和互斥锁  sem\_destroy(&empty);  sem\_destroy(&full);  pthread\_mutex\_destroy(&mutex);  printf("The End..\n");  return 0;  }  IMG_256    ****编译程序****：使用编译器（如gcc）将.c文件编译成可执行文件。  pthread库并非Linux操作系统的默认库，编译时加上-lpthread选项，以调用该链接库.  gcc cons.c -o cons -lpthread  IMG_256  ****运行程序****：编译成功后，会生成一个可执行文件，程序运行如下： | | | |
| 结论  （结果） | 实验成功，按照相关要求完成程序编写及实现，并熟练掌握了相关命令的使用。  使用Linux操作系统中的信号量机制模拟实现了生产者-消费者问题。 | | | |
| 小结 | 1）加强了对进程同步和互斥的理解，学会使用信号量解决资源共享问题。  2）熟悉了Linux 进程同步原语。  3）掌握了信号量wait/signal 原语的使用方法，理解信号量的定义、赋初值及wait/signal操作。 | | | |
| 指导老师  评议 |  | | | |
| 成绩评定： | 指导教师签名： | | |