山东大学 计算机科学与技术 学院

操作系统 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201900161140 | 姓名：张文浩 | | 班级： 19人工智能 |
| 实验题目：进程同步实验 | | | |
| 实验学时：2 | | 实验日期： 4.26 | |
| 实验目的：  加深对并发协作进程同步与互斥概念的理解，观察和体验并发进程同步与互斥操作的效果，分析与研究经典进程同步与互斥问题的实际解决方案。了解 Linux 系统中IPC 进程同步工具的用法，练习并发协作进程的同步与互斥操作的编程与调试技术。 | | | |
| 硬件环境： **自带的笔记本，win10** | | | |
| 软件环境：vmware的ubuntu | | | |
| 实验步骤与内容：   1. 根据指导书上完后才能实例实验的文件的创建，并执行，结果如下：     一开始使用make编译的时候一直显示ipc.h中的变量多次定义，导致编译不了，去网上找了好多方法都不行。然后猜测是编译器的问题，于是我就把gcc版本10卸载掉，重装了gcc版本9，然后就能编译通过了，可能是10版本的gcc对这个错误比较敏感，这玩意折磨了我好久。   1. 独立实验   抽烟者问题。假设一个系统中有三个抽烟者进程，每个抽烟者不断地卷烟并抽 烟。抽烟者卷起并抽掉一颗烟需要有三种材料：烟草、纸和胶水。一个抽烟者有烟 草，一个有纸，另一个有胶水。系统中还有两个供应者进程，它们无限地供应所有 三种材料，但每次仅轮流提供三种材料中的两种。得到缺失的两种材料的抽烟者在 卷起并抽掉一颗烟后会发信号通知供应者，让它继续提供另外的两种材料。这一过 程重复进行。 请用以上介绍的 IPC 同步机制编程，实现该问题要求的功能。   1. 创建ipc.h文件   #include <stdio.h>  #include <stdlib.h>  #include <sys/types.h>  #include <sys/ipc.h>  #include <sys/shm.h>  #include <sys/sem.h>  #include <sys/msg.h>  #include<sys/time.h>  #include<sys/resource.h>  #include<unistd.h>  #define BUFSZ 256  //建立或获取 ipc 的一组函数的原型说明  int get\_ipc\_id(char \*proc\_file,key\_t key);  char \*set\_shm(key\_t shm\_key,int shm\_num,int shm\_flag);  int set\_msq(key\_t msq\_key,int msq\_flag);  int set\_sem(key\_t sem\_key,int sem\_val,int sem\_flag);  int down(int sem\_id);  int up(int sem\_id);  /\*信号灯控制用的共同体\*/  typedef union semuns  {  int val;  } Sem\_uns;  /\* 消息结构体\*/  typedef struct msgbuf  {  long mtype;  char mtext[1];  } Msg\_buf;  //生产消费者共享缓冲区即其有关的变量  key\_t buff\_key;  int buff\_num;  char \*buff\_ptr;  //生产者放产品位置的共享指针  key\_t pput\_key;  int pput\_num;  int \*pput\_ptr;  //消费者取产品位置的共享指针  key\_t cget\_key;  int cget\_num;  int \*cget\_ptr;  //生产者和消费者有关的信号量  key\_t ab\_key;  key\_t ac\_key;  key\_t bc\_key;  key\_t cmtx\_key;  key\_t prod1\_key;  key\_t prod2\_key;  key\_t prod3\_key;  key\_t pmtx\_key;  int ab\_sem;  int ac\_sem;  int bc\_sem;  int prod1\_sem;  int prod2\_sem;  int prod3\_sem;  int pmtx\_sem;  int cmtx\_sem;  int sem\_val;  int sem\_flg;  int shm\_flg;   1. 创建producer.c文件   #include "ipc.h"  int main(int argc,char \*argv[])  {  int rate;  //可在在命令行第一参数指定一个进程睡眠秒数，以调解进程执行速度  if(argv[1] != NULL)  rate = atoi(argv[1]);  else  rate = 3; //不指定为 3 秒  //共享内存使用的变量  buff\_key = 101;//缓冲区任给的键值  buff\_num = 3;//缓冲区任给的长度  pput\_key = 102;//生产者放产品指针的键值  pput\_num = 1; //指针数  shm\_flg = IPC\_CREAT | 0644;//共享内存读写权限  //获取缓冲区使用的共享内存，buff\_ptr 指向缓冲区首地址  buff\_ptr = (char \*)set\_shm(buff\_key,buff\_num,shm\_flg);  //获取生产者放产品位置指针 pput\_ptr  pput\_ptr = (int \*)set\_shm(pput\_key,pput\_num,shm\_flg);  //信号量使用的变量  prod1\_key = 201;//生产者同步信号灯键值  prod2\_key = 202;//生产者同步信号灯键值  prod3\_key = 203;//生产者同步信号灯键值  pmtx\_key = 204;//生产者互斥信号灯键值  ab\_key = 301;//有C的消费者控制键值  ac\_key = 302;//有B的消费者控制键值  bc\_key = 303;//有A的消费者控制键值  sem\_flg = IPC\_CREAT | 0644;  //生产者同步信号灯初值设为缓冲区最大可用量  sem\_val = 1;  //获取生产者同步信号灯，引用标识存 prod\_sem  prod1\_sem = set\_sem(prod1\_key,sem\_val,sem\_flg);  prod2\_sem = set\_sem(prod2\_key,sem\_val,sem\_flg);  prod3\_sem = set\_sem(prod3\_key,sem\_val,sem\_flg);  //消费者初始无产品可取，同步信号灯初值设为 0  sem\_val = 0;  ab\_sem = set\_sem(ab\_key,sem\_val,sem\_flg);  ac\_sem = set\_sem(ac\_key,sem\_val,sem\_flg);  bc\_sem = set\_sem(bc\_key,sem\_val,sem\_flg);  //生产者互斥信号灯初值为 1  sem\_val = 1;  //获取生产者互斥信号灯，引用标识存 pmtx\_sem  pmtx\_sem = set\_sem(pmtx\_key,sem\_val,sem\_flg);  int i = 0;    while(1){  int d = (i++)%3;  down(pmtx\_sem);  //buff\_ptr[\*pput\_ptr] = 'A'+ d;  if(d==0){  down(prod1\_sem);  sleep(rate);  \*pput\_ptr = 0;  printf("%d把烟草和纸放入[%d]缓存区\n",getpid(),\*pput\_ptr);  up(bc\_sem);  }  else if(d==1){  down(prod2\_sem);  sleep(rate);  \*pput\_ptr = 1;  printf("%d把胶水和纸放入[%d]缓存区\n",getpid(),\*pput\_ptr);  up(ac\_sem);  }  else{  down(prod3\_sem);  sleep(rate);  \*pput\_ptr = 2;  printf("%d把烟草和胶水放入[%d]缓存区\n",getpid(),\*pput\_ptr);  up(ab\_sem);  }  //\*pput\_ptr = (\*pput\_ptr+1) % buff\_num;  up(pmtx\_sem);  }  return EXIT\_SUCCESS;  }   1. 创建smoker1.c文件   #include "ipc.h"  int main(int argc,char \*argv[])  {  int rate;  //可在在命令行第一参数指定一个进程睡眠秒数，以调解进程执行速度  if(argv[1] != NULL)  rate = atoi(argv[1]);  else  rate = 3; //不指定为 3 秒  //共享内存 使用的变量  buff\_key = 101; //缓冲区任给的键值  buff\_num = 3; //缓冲区任给的长度  cget\_key = 103; //消费者取产品指针的键值  cget\_num = 1; //指针数  shm\_flg = IPC\_CREAT | 0644; //共享内存读写权限  //获取缓冲区使用的共享内存，buff\_ptr 指向缓冲区首地址  buff\_ptr = (char \*)set\_shm(buff\_key,buff\_num,shm\_flg);  //获取消费者取产品指针，cget\_ptr 指向索引地址  cget\_ptr = (int \*)set\_shm(cget\_key,cget\_num,shm\_flg);  //信号量使用的变量  prod1\_key = 201;//生产者同步信号灯键值  prod2\_key = 202;//生产者同步信号灯键值  prod3\_key = 203;//生产者同步信号灯键值  pmtx\_key = 204;//生产者互斥信号灯键值  cmtx\_key = 205;//消费者互斥信号灯键值  ab\_key = 301;//有C的消费者控制键值  ac\_key = 302;//有B的消费者控制键值  bc\_key = 303;//有A的消费者控制键值  sem\_flg = IPC\_CREAT | 0644;  //生产者同步信号灯初值设为缓冲区最大可用量  sem\_val = 1;  //获取生产者同步信号灯，引用标识存 prod\_sem  prod1\_sem = set\_sem(prod1\_key,sem\_val,sem\_flg);  prod2\_sem = set\_sem(prod2\_key,sem\_val,sem\_flg);  prod3\_sem = set\_sem(prod3\_key,sem\_val,sem\_flg);  //消费者初始无产品可取，同步信号灯初值设为 0  sem\_val = 0;  ab\_sem = set\_sem(ab\_key,sem\_val,sem\_flg);  ac\_sem = set\_sem(ac\_key,sem\_val,sem\_flg);  bc\_sem = set\_sem(bc\_key,sem\_val,sem\_flg);  //消费者互斥信号灯初值为1  sem\_val = 1;  //消费者互斥信号灯，引用标识存 cmtx\_sem  cmtx\_sem = set\_sem(cmtx\_key,sem\_val,sem\_flg);  //生产者互斥信号灯初值为 1  sem\_val = 1;  //获取生产者互斥信号灯，引用标识存 pmtx\_sem  pmtx\_sem = set\_sem(pmtx\_key,sem\_val,sem\_flg);  while(1)  {  \*cget\_ptr = 0;  down(bc\_sem);  down(cmtx\_sem);  sleep(rate);  printf("%d 吸烟者1号在第[%d]缓冲区得到了烟草和纸\n",getpid(),\*cget\_ptr);  //\*cget\_ptr = (\*cget\_ptr+1) % buff\_num;  up(cmtx\_sem);  up(prod1\_sem);  }  return EXIT\_SUCCESS;  }   1. 创建smoker2.c文件   #include "ipc.h"  int main(int argc,char \*argv[])  {  int rate;  //可在在命令行第一参数指定一个进程睡眠秒数，以调解进程执行速度  if(argv[1] != NULL)  rate = atoi(argv[1]);  else  rate = 3; //不指定为 3 秒  //共享内存 使用的变量  buff\_key = 101; //缓冲区任给的键值  buff\_num = 3; //缓冲区任给的长度  cget\_key = 103; //消费者取产品指针的键值  cget\_num = 1; //指针数  shm\_flg = IPC\_CREAT | 0644; //共享内存读写权限  //获取缓冲区使用的共享内存，buff\_ptr 指向缓冲区首地址  buff\_ptr = (char \*)set\_shm(buff\_key,buff\_num,shm\_flg);  //获取消费者取产品指针，cget\_ptr 指向索引地址  cget\_ptr = (int \*)set\_shm(cget\_key,cget\_num,shm\_flg);  //信号量使用的变量  prod1\_key = 201;//生产者同步信号灯键值  prod2\_key = 202;//生产者同步信号灯键值  prod3\_key = 203;//生产者同步信号灯键值  pmtx\_key = 204;//生产者互斥信号灯键值  cmtx\_key = 205;//消费者互斥信号灯键值  ab\_key = 301;//有C的消费者控制键值  ac\_key = 302;//有B的消费者控制键值  bc\_key = 303;//有A的消费者控制键值  sem\_flg = IPC\_CREAT | 0644;  //生产者同步信号灯初值设为缓冲区最大可用量  sem\_val = 1;  //获取生产者同步信号灯，引用标识存 prod\_sem  prod1\_sem = set\_sem(prod1\_key,sem\_val,sem\_flg);  prod2\_sem = set\_sem(prod2\_key,sem\_val,sem\_flg);  prod3\_sem = set\_sem(prod3\_key,sem\_val,sem\_flg);  //消费者初始无产品可取，同步信号灯初值设为 0  sem\_val = 0;  ab\_sem = set\_sem(ab\_key,sem\_val,sem\_flg);  ac\_sem = set\_sem(ac\_key,sem\_val,sem\_flg);  bc\_sem = set\_sem(bc\_key,sem\_val,sem\_flg);  //消费者互斥信号灯初值为1  sem\_val = 1;  //消费者互斥信号灯，引用标识存 cmtx\_sem  cmtx\_sem = set\_sem(cmtx\_key,sem\_val,sem\_flg);  //生产者互斥信号灯初值为 1  sem\_val = 1;  //获取生产者互斥信号灯，引用标识存 pmtx\_sem  pmtx\_sem = set\_sem(pmtx\_key,sem\_val,sem\_flg);  while(1)  {  \*cget\_ptr = 1;  down(ac\_sem);  down(cmtx\_sem);  sleep(rate);  printf("%d 吸烟者2号在第[%d]缓冲区得到了胶水和纸\n",getpid(),\*cget\_ptr);  //\*cget\_ptr = (\*cget\_ptr+1) % buff\_num;  up(cmtx\_sem);  up(prod2\_sem);  }  return EXIT\_SUCCESS;  }   1. 创建smoker3.c文件   #include "ipc.h"  int main(int argc,char \*argv[])  {  int rate;  //可在在命令行第一参数指定一个进程睡眠秒数，以调解进程执行速度  if(argv[1] != NULL)  rate = atoi(argv[1]);  else  rate = 3; //不指定为 1 秒  //共享内存 使用的变量  buff\_key = 101; //缓冲区任给的键值  buff\_num = 3; //缓冲区任给的长度  cget\_key = 103; //消费者取产品指针的键值  cget\_num = 1; //指针数  shm\_flg = IPC\_CREAT | 0644; //共享内存读写权限  //获取缓冲区使用的共享内存，buff\_ptr 指向缓冲区首地址  buff\_ptr = (char \*)set\_shm(buff\_key,buff\_num,shm\_flg);  //获取消费者取产品指针，cget\_ptr 指向索引地址  cget\_ptr = (int \*)set\_shm(cget\_key,cget\_num,shm\_flg);  //信号量使用的变量  prod1\_key = 201;//生产者同步信号灯键值  prod2\_key = 202;//生产者同步信号灯键值  prod3\_key = 203;//生产者同步信号灯键值  pmtx\_key = 204;//生产者互斥信号灯键值  cmtx\_key = 205;//消费者互斥信号灯键值  ab\_key = 301;//有C的消费者控制键值  ac\_key = 302;//有B的消费者控制键值  bc\_key = 303;//有A的消费者控制键值  sem\_flg = IPC\_CREAT | 0644;  //生产者同步信号灯初值设为缓冲区最大可用量  sem\_val = 1;  //获取生产者同步信号灯，引用标识存 prod\_sem  prod1\_sem = set\_sem(prod1\_key,sem\_val,sem\_flg);  prod2\_sem = set\_sem(prod2\_key,sem\_val,sem\_flg);  prod3\_sem = set\_sem(prod3\_key,sem\_val,sem\_flg);  //消费者初始无产品可取，同步信号灯初值设为 0  sem\_val = 0;  ab\_sem = set\_sem(ab\_key,sem\_val,sem\_flg);  ac\_sem = set\_sem(ac\_key,sem\_val,sem\_flg);  bc\_sem = set\_sem(bc\_key,sem\_val,sem\_flg);  //消费者互斥信号灯初值为1  sem\_val = 1;  //消费者互斥信号灯，引用标识存 cmtx\_sem  cmtx\_sem = set\_sem(cmtx\_key,sem\_val,sem\_flg);  //生产者互斥信号灯初值为 1  sem\_val = 1;  //获取生产者互斥信号灯，引用标识存 pmtx\_sem  pmtx\_sem = set\_sem(pmtx\_key,sem\_val,sem\_flg);  while(1)  {  \*cget\_ptr = 2;  down(ab\_sem);  down(cmtx\_sem);  sleep(rate);  printf("%d 吸烟者3号在第[%d]缓冲区得到了烟草和胶水\n",getpid(),\*cget\_ptr);  //\*cget\_ptr = (\*cget\_ptr+1) % buff\_num;  up(cmtx\_sem);  up(prod3\_sem);  }  return EXIT\_SUCCESS;  }   1. 创建makefile文件   b\_obj = smoker2.o ipc.o  c\_src = smoker3.c ipc.c  c\_obj = smoker3.c ipc.o  all: smoker1 smoker2 smoker3 producer  smoker1: $(a\_obj)  gcc $(a\_obj) -o smoker1  smoker1.o: $(a\_src) $(hdrs)  gcc $(opts) $(a\_src) -lrt  smoker2: $(b\_obj)  gcc $(b\_obj) -o smoker2  smoker2.o: $(b\_src) $(hdrs)  gcc $(opts) $(b\_src) -lrt  smoker3: $(c\_obj)  gcc $(c\_obj) -o smoker3  smoker3.o: $(c\_src) $(hdrs)  gcc $(opts) $(c\_src) -lrt  producer: $(p\_obj)  gcc $(p\_obj) -o producer  producer.o: $(p\_src) $(hdrs)  gcc $(opts) $(p\_src) -lrt  clean:  rm smoker1 smoker2 smoker3 producer \*.o   1. make编译   运行producer，速度设置为1  运行三个smoker，速度设置为1  运行结果如下： | | | |
| 结论分析与体会：  通过本次实验，我了解到信号量是某个实例可用资源的计数，初始为该实例可用资源的数量，而每当线程需要使用，则调用wait()方法减少信号量，释放资源时调用signal()方法增加信号量，故信号量为0表示所有资源都在被使用，线程使用资源的请求不被允许。  信号量主要分为计数信号量和二进制信号量，前者主要针对一个实例有多个资源的情况，值域不受限制，而后者信号量仅为0或1，也就是说线程之间访问该资源是互斥的，也可称作互斥锁。  在操作系统中，信号量sem是一整数。在sem大于等于零时代表可供并发进程使用的资源实体数，但sem小于零时则表示正在等待使用临界区的进程数。显然，用于互斥的信号量sem初值应该大于零。信号量数值仅能由P,V原语操作改变。执行一次P操作意味着请求分配一个单位资源，因此S的值减1；当S<0时，表示已经没有可用资源，请求者必须等待别的进程释放该类资源，它才能运行下去。而执行一个V操作意味着释放一个单位资源，因此S的值加1；若S=0，表示有某些进程正在等待该资源，因此要唤醒一个等待状态的进程，使之运行下去。P,V原语成对出现，在互斥操作时，他们处于同一个进程，但在同步操作时，他们不处于一个进程内。私有信号量是指只与制约进程有关而不是与整组并发进程有关的信号量。利用P,V原语实现进程同步的方法分为三步：首先为各并发进程设置私用信号量，然后为私用信号量赋初值，最后利用P,V原语和使用信号量规定各进程的执行顺序。  总之，这次实验收获很大，对信号量进程同步问题有了很深的了解。 | | | |