|  |
| --- |
| **实验概述：** |
| **【实验目的及要求】**  **掌握Linux下生产者/消费者问题算法的实现**  **【实验原理】**  1.clone系统调用：  功能：创建一个轻进程或线程  用法：  int clone (int (\*fn)(void \*arg),void \*stack,int flag,void \*arg);  其中，  ★fn是轻进程所执行的函数；stack是轻进程所使用的栈  ★flag是CLONE\_VM,CLONE\_FS,CLONE\_FILES,CLONE\_SIGNAND,CLONE\_PID的组合  ★arg是调用过程的对应参数  clone的关键是flag的设定。  ☆CLONE\_VM表示子进程共享父进程的内存  ☆CLONE\_FS表示子进程共享父进程的文件系统  ☆CLONE\_SIGNAND表示子进程共享父进程的消息处理机制  ☆CLONE\_PID指子进程继承父进程的id号  2.sem\_wait和sem\_post系统调用：  sem\_wait(&s)和sem\_post(&s)分别相当于信号灯的P操作和V操作。其中，s是类型为sem\_t的信号灯。初始化函数sem\_init(s,0,8)。  3.pthread\_mutex\_lock和pthread\_mutex\_unlock系统调用：  pthread\_mutex\_lock(&mutex)和pthread\_rmutex\_unlock(&mutex)分别用于加锁和解锁。参数为mutex定义的互斥锁。初始pthread\_mutex\_init(&mutex,NULL)。  **【实验环境】（使用的软件）**  **gcc编译器，Linux平台，centos8系统** |
| **实验内容：** |
| **【实验方案设计】**  根据实验目的和要求，以及给出的实验原理和环境，可以给出以下方案：  1. 创建生产者和消费者进程：使用`clone()`系统调用创建两个独立的进程，一个用作生产者，一个用作消费者。在`clone()`中，设置`flag`参数为适当的组合，以确保子进程共享父进程的内存、文件系统和消息处理机制。  2. 使用信号量进行同步：使用`sem\_init()`函数初始化一个信号量，并设置初始值为资源的数量。在生产者和消费者进程中，使用`sem\_wait()`和`sem\_post()`系统调用来实现生产者和消费者之间的同步。在生产者进程中，每次生产一个项目之前，调用`sem\_wait()`减少信号量值，表示可用的资源减少；在消费者进程中，每次消耗一个项目之后，调用`sem\_post()`增加信号量值，表示可用的资源增加。  3. 使用互斥锁进行互斥访问：使用`pthread\_mutex\_init()`函数初始化一个互斥锁，在需要互斥访问的临界区代码段中，使用`pthread\_mutex\_lock()`和`pthread\_mutex\_unlock()`系统调用来加锁和解锁互斥锁。这样可以确保在同一时间只有一个进程能够访问临界区，避免竞争条件。  4. 编写生产者和消费者代码：在生产者进程中，编写代码以生成项目，并将其放入共享的缓冲区中。在消费者进程中，编写代码以从缓冲区中获取项目并进行消费。  5. 编译和运行代码：使用gcc编译器编译生产者和消费者代码，并在Linux平台的CentOS 8系统上运行生成的可执行文件。  **【实验过程】（实验步骤、记录、数据、分析）**  实验过程如下：  1. 创建生产者和消费者进程：使用`clone()`系统调用创建两个子进程，一个作为生产者，另一个作为消费者。这些进程将在共享的内存区域中进行通信。  2. 定义共享内存区域：使用`shmget()`系统调用创建一个共享内存区域，用于生产者和消费者之间的数据交换。确保设置合适的权限和大小。  3. 初始化信号量和互斥锁：使用`sem\_init()`函数初始化生产者和消费者之间的信号量，以实现同步操作。使用`pthread\_mutex\_init()`函数初始化互斥锁，以实现对共享资源的互斥访问。  4. 编写生产者代码：在生产者进程中，使用互斥锁保护共享资源，使用信号量控制生产者和消费者之间的同步。生产者从输入中读取数据，并将数据写入共享内存区域。  5. 编写消费者代码：在消费者进程中，使用互斥锁保护共享资源，使用信号量控制生产者和消费者之间的同步。消费者从共享内存区域中读取数据，并进行相应的处理。  6. 编译和运行代码：使用gcc编译器编译生产者和消费者代码，并链接所需的库。确保在编译时添加适当的选项和标志。然后运行生成的可执行文件。  **实验代码**  #define \_GNU\_SOURCE  #include "sched.h"  #include "pthread.h"  #include "stdio.h"  #include "stdlib.h"  #include "semaphore.h"  #define BUFFER\_LEN 8 //定义缓冲区长度  void producer(void \*args);  void consumer(void \*args);  pthread\_mutex\_t mutex;  sem\_t product;  sem\_t warehouse;  char buffer[BUFFER\_LEN][4];  int in,out; //产品进、出缓冲区指针  int tot=0; //缓冲区中的产品总数  main(int argc,char \*\*argv){  pthread\_mutex\_init(&mutex,NULL); //对互斥信号量进行初始化  sem\_init(&product,0,0); //对标识产品个数的信号量进行初始化  sem\_init(&warehouse,0,BUFFER\_LEN); //对标识缓冲区中空位置个数的信号量进行初始化  in=0;  out=0;  int i0=0,i1=1;  int clone\_flag, retval;  char \*stack;  clone\_flag=CLONE\_VM|CLONE\_SIGHAND|CLONE\_FS|CLONE\_FILES;  //创建两个生产者和两个消费者线程  stack=(char \*)malloc(4096);  retval=clone((void \*)producer,&(stack[4095]),clone\_flag,(void \*)&i0);  stack=(char \*)malloc(4096);  retval=clone((void \*)consumer,&(stack[4095]),clone\_flag,(void \*)&i0);  stack=(char \*)malloc(4096);  retval=clone((void \*)producer,&(stack[4095]),clone\_flag,(void \*)&i1);  stack=(char \*)malloc(4096);  retval=clone((void \*)consumer,&(stack[4095]),clone\_flag,(void \*)&i1);    exit(1);    }  void producer(void \*args){  int id=\*((int \*)args);  int i;  for(i=0;i<10;i++){  sleep(i+1);  sem\_wait(&warehouse);  pthread\_mutex\_lock(&mutex);  if(id==0)  strcpy(buffer[in],"aaa\0");  else  strcpy(buffer[in],"bbb\0");  printf("producer %d produces %s in %d\n",id,buffer[in],in);    in=(in+1)%BUFFER\_LEN;  tot++; //当前产品总数加1  printf("\*\*\*\*\*the number of products:%d\*\*\*\*\*\n",tot);  pthread\_mutex\_unlock(&mutex);  sem\_post(&product);  }  printf("producer %d is over!\n",id);  }  void consumer(void \*args){  int id=\*((int \*)args);  int i;  for(i=0;i<10;i++){  sleep(10-i);  sem\_wait(&product);  pthread\_mutex\_lock(&mutex);  printf("consumer %d get %s in %d\n",id,buffer[out],out);  out=(out+1)%BUFFER\_LEN;  tot--; //当前产品总数减1  printf("\*\*\*\*\*the number of products:%d\*\*\*\*\*\n",tot);  pthread\_mutex\_unlock(&mutex);  sem\_post(&warehouse);  }  printf("consumer %d is over!\n",id);  }  **【结论】（结果）** |
| **【小结】**  **错误1：缺少#include <unistd.h>头文件**    **错误2：缺少#include "string.h"头文件**    **运行成功：**    **从运行结果来看，貌似并没有问题** |
| **指导教师评语及成绩：** |
| **评语：**  **成绩： 指导教师签名：**  **批阅日期：** |

实验报告说明

**1．实验项目名称：**要用最简练的语言反映实验的内容。要求与实验指导书中相一致。

**2．实验类型：**一般需说明是验证型实验还是设计型实验，是创新型实验还是综合型实验。

**3．实验目的与要求**：目的要明确，要抓住重点，符合实验指导书中的要求。

**4．实验原理：**简要说明本实验项目所涉及的理论知识。

**5．实验环境**：实验用的软硬件环境（配置）。

**6．实验方案设计（思路、步骤和方法等）**：这是实验报告极其重要的内容。概括整个实验过程。

对于**操作型实验**，要写明依据何种原理、操作方法进行实验，要写明需要经过哪几个步骤来实现其操作。

对于**设计型和综合型实验**，在上述内容基础上还应该画出流程图、设计思路和设计方法，再配以相应的文字说明。

对于**创新型实验**，还应注明其创新点、特色。

**7．实验过程（实验中涉及的记录、数据、分析）：**写明具体上述实验方案的具体实施，包括实验过程中的记录、数据和相应的分析。

**8．结论（结果）：**即根据实验过程中所见到的现象和测得的数据，做出结论。

**9．小结：**对本次实验的心得体会、思考和建议。

**10．指导教师评语及成绩：**指导教师依据学生的实际报告内容，用简练语言给出本次实验报告的评价和价值。

**注意：**

* 实验报告将记入实验成绩；
* 每次实验开始时，交上一次的实验报告，否则将扣除此次实验成绩。