//

// main.c

// ExamTest

//

// Created by 李毓琪 on 2021/10/24.

//

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <pthread.h>

#define N 5

**typedef** **int** data;

pthread\_t id1,id2,id3,id4,id5;

pthread\_mutex\_t Cust,Prod;

**int** fd[2];

**int** i = 1;

**int** num = 0;

**void** full(){

**while**(num >= 5);

}

**void** empty(){

**while**(num <= 0);

}

**void** Customer(**int** n)

{

**char** buffer[50];

**while**(1){

sleep(1);

empty();

pthread\_mutex\_lock(&Cust);

read(fd[0],buffer,50);

printf("Customer %d Receive: %s\n",n,buffer);

num--;

pthread\_mutex\_unlock(&Cust);

}

}

**void** Producter(**int** n)

{

**char** buffer[50];

**while**(1){

sleep(1);

full();

pthread\_mutex\_lock(&Prod);

sprintf(buffer,"%d",i);

write(fd[1],buffer,50);

printf("Producter %d Send %s\n",n,buffer);

i++;

num++;

pthread\_mutex\_unlock(&Prod);

}

}

**int** main(**int** argc, **const** **char** \* argv[]) {

pipe(fd);

pthread\_mutex\_init(&Cust,**NULL**);

pthread\_mutex\_init(&Prod,**NULL**);

pthread\_create(&id1,**NULL**,(**void** \*)Customer,1);

pthread\_create(&id3,**NULL**,(**void** \*)Customer,3);

pthread\_create(&id5,**NULL**,(**void** \*)Producter,5);

pthread\_create(&id2,**NULL**,(**void** \*)Producter,2);

pthread\_create(&id4,**NULL**,(**void** \*)Producter,4);

pthread\_join(id1,**NULL**);

pthread\_join(id2,**NULL**);

pthread\_join(id3,**NULL**);

pthread\_join(id4,**NULL**);

pthread\_join(id5,**NULL**);

close(fd);

}

表格

低可信度描述已自动生成通过使用pthread\_mutex线程锁代替互斥信号量，线程锁与互斥信号量类似，相比于互斥信号量的sem\_wait以及sem\_post，pthread\_mutex\_lock,pthread\_mutex\_unlock更为直观。使用管道替代原先的队列，队列创建繁琐相比而言管道使用简单，此处也可使用消息队列进行替代。通过while循环替代full以及empty的PV操作，但是如果程序运行过快即将程序中的sleep注释掉或者生产者数量过多时则可能会出现产品数量超过上限的现象。

二次改进的程序：

//

// main.c

// ExamTest

//

// Created by 李毓琪 on 2021/10/24.

//

#include <stdio.h>

#include <signal.h>

#include <stdlib.h>

#include <unistd.h>

#include <pthread.h>

#define N 5

**typedef** **int** data;

pthread\_t id1,id2,id3,id4,id5,controller;

pthread\_mutex\_t Cust,Prod;

**int** fd[2];

**int** i = 1;

**int** num = 0;

**int** tag = 0;

**void** Customer(**int** n)

{

**char** buffer[50];

**while**(1){

sleep(1);

pthread\_mutex\_lock(&Cust);

read(fd[0],buffer,50);

printf("Customer %d Receive: %s\n",n,buffer);

num--;

tag = 0;

}

}

**void** Producter(**int** n)

{

**char** buffer[50];

**while**(1){

sleep(1);

pthread\_mutex\_lock(&Prod);

num++;

sprintf(buffer,"%d",i++);

write(fd[1],buffer,50);

printf("Producter %d Send %s\n",n,buffer);

tag = 0;

}

}

**void** Controller(){

**while**(1){

usleep(100);

**if** (num > 0 && tag == 0) {

tag = 1;

pthread\_mutex\_unlock(&Cust);

}**else** **if**(num < 5 && tag == 0){

tag = 1;

pthread\_mutex\_unlock(&Prod);

}

}

}

**int** main(**int** argc, **const** **char** \* argv[]) {

pipe(fd);

pthread\_mutex\_init(&Cust,**NULL**);

pthread\_mutex\_init(&Prod,**NULL**);

pthread\_create(&id1,**NULL**,(**void** \*)Customer,1);

pthread\_create(&id3,**NULL**,(**void** \*)Customer,3);

pthread\_create(&id5,**NULL**,(**void** \*)Producter,5);

pthread\_create(&id2,**NULL**,(**void** \*)Producter,2);

pthread\_create(&id4,**NULL**,(**void** \*)Producter,4);

pthread\_create(&controller,**NULL**,(**void** \*)Controller,**NULL**);

pthread\_join(id1,**NULL**);

pthread\_join(id2,**NULL**);

pthread\_join(id3,**NULL**);

pthread\_join(id4,**NULL**);

pthread\_join(id5,**NULL**);

close(fd);

}

图片包含 文本

描述已自动生成通过创建一个Controller线程控制所有生产者以及消费者的进行，Controller中通过全局变量tag实现了生产的以及消费者的PV(mutex)操作，以及通过判断产品的数量选择让生产者或者消费者解除阻塞状态。任务分配交由Controller主要因为如果使用phread\_lock进行full,empty的PV操作时可能会出现多个线程同时解除阻塞状态。