**西安电子科技大学 网络与信息安全学院**

**操 作 系 统 实验报告**

**班 级： 1618011**

**学 号： 16180110065**

**姓 名： 王宇卿**

**Github： https://github.com/pitifulnoble**

**电子邮箱： pitifulnoble@outlook.com**

**2018年4月22日**

|  |
| --- |
| **实验题目：Producer-Consumer Problem** |
| **一、实验题目**  编写一个C程序，包含buffer.h、两个源程序，实现信号量控制程序解决生产者消费者的经典问题。 |
| **二、相关原理与知识**  **了解C程序库pthread.h**  **对Linux基本知识的了解**  **熟悉线程相关知识**  **对生产者-消费者模型有一定了解**  **程序中使用到了Makefile编译程序** |
| **三、实验过程**  **程序目录如下：**  **捕获**  **编译实现方法：**  **捕获**  **Main.c程序的运行结果**  **捕获** |
| ****producer-consumer.c程序运行：****  ****这个程序接受3个参数，第一个是睡眠时间sleep time.****  ****第二个是生产者的线程个数，第三个是消费者的线程个数****  ****捕获****  **问题解决。** |
| **四、实验结果与分析**  基本完成实验要求，实现了生产者和消费者线程搭配运作的模式。 |
| **五、问题总结**  **在编译源代码时遇到问题。出现如下错误：**  **捕获**  **通过查阅相关资料，发现带有ptread.h头文件的C程序源代码需要在**  **编译时加上-pthread参数。** |
| **六、源代码**  **Main.c:**  **#include <stdlib.h>**  **#include <stdio.h>**  **#include <pthread.h>**  **#define M 3**  **#define K 2**  **#define N 3**  **int A [M][K] = { {1,4}, {2,5}, {3,6} };**  **int B [K][N] = { {8,7,6}, {5,4,3} };**  **int C [M][N];**  **struct v {**  **int i; // Row**  **int j; // Column**  **};**  **void \*multiply(void \*arg){**  **struct v\* param = (struct v \*)arg;**  **int i = param->i;**  **int j = param->j;**  **C[i][j] = 0;**  **int k;**  **for (k = 0; k < K; k++)**  **C[i][j] += A[i][k] \* B[k][j];**    **free(arg); // Free data that was malloc'd earlier?**  **pthread\_exit(NULL);**  **return 0;**  **}** |
| **int main(){**  **// Perform multiplication**  **int i, j, rc;**  **pthread\_t threads[M\*N];**  **for (i = 0; i < M; i++){**  **for (j = 0; j < N; j++){**  **struct v \*data = (struct v \*) malloc(sizeof(struct v));**  **data->i = i;**  **data->j = j;**  **rc = pthread\_create(&threads[i\*N + j], NULL, multiply, (void \*)data);**  **if (rc){**  **printf("\* ERROR: pthread\_create() abnormal return value.\n");**  **exit(1);**  **}**  **}**  **}**  **for (i = 0; i < M\*N; i++)**  **pthread\_join(threads[i], NULL); // Wait for the previous threads to finish**  **// Print the matrices**  **printf("Matrix A:\n");**  **for(i = 0; i < M; i++){**  **for (j = 0; j < K; j++){**  **if (j != K-1)**  **printf("%d ", A[i][j]);**  **else**  **printf("%d", A[i][j]);**  **}**  **printf("\n");**  **}** |
| **printf("\nMatrix B:\n");**  **for(i = 0; i < K; i++){**  **for (j = 0; j < N; j++){**  **if (j != N-1)**  **printf("%d ", B[i][j]);**  **else**  **printf("%d", B[i][j]);**  **}**  **printf("\n");**  **}**  **printf("\nMatrix C:\n");**  **for(i = 0; i < M; i++){**  **for (j = 0; j < N; j++){**  **if (j != N-1)**  **printf("%d ", C[i][j]);**  **else**  **printf("%d", C[i][j]);**  **}**  **printf("\n");**  **}**  **pthread\_exit(NULL);**  **return 0;**  **}** |
| ****producer-consumer.c：****  ****#include <stdlib.h>****  ****#include <stdio.h>****  ****#include <string.h>****  ****#include <unistd.h>****  ****#include <pthread.h>****  ****#include <semaphore.h>****  ****#include <time.h>****  ****#include "buffer.h"****  ****// Global variables****  ****buffer\_item buffer[BUFFER\_SIZE];****  ****pthread\_mutex\_t mutex;****  ****sem\_t full, empty;****  ****int count, in, out;****  ****// Function prototypes****  ****int insert\_item(buffer\_item item);****  ****int remove\_item(buffer\_item \*item);****  ****void \*consumer(void \*param);****  ****void \*producer(void \*param);****  ****int main(int argc, char \*\*argv){****  ****if (argc != 4){****  ****printf("ERROR: Provide exactly three arguments.\n");****  ****exit(1);****  ****}**** |
| ****// Retrieve command line arguments****  ****const long int stime = strtol(argv[1], NULL, 0);****  ****const long int num\_producer = strtol(argv[2], NULL, 0);****  ****const long int num\_consumer = strtol(argv[3], NULL, 0);****  ****// Initialize****  ****int i;****  ****srand(time(NULL));****  ****pthread\_mutex\_init(&mutex, NULL);****  ****sem\_init(&empty, 0, BUFFER\_SIZE); // All of buffer is empty****  ****sem\_init(&full, 0, 0);****  ****count = in = out = 0;****  ****// Create the producer and consumer threads****  ****pthread\_t producers[num\_producer];****  ****pthread\_t consumers[num\_consumer];****  ****for(i = 0; i < num\_producer; i++)****  ****pthread\_create(&producers[i], NULL, producer, NULL);****  ****for(i = 0; i < num\_consumer; i++)****  ****pthread\_create(&consumers[i], NULL, consumer, NULL);****  ****// Sleep before terminating****  ****sleep(stime);****  ****return 0;****  ****}****  ****// Insert item into buffer.****  ****//Returns 0 if successful, -1 indicating error**** |
| ****int insert\_item(buffer\_item item){****  ****int success;****  ****sem\_wait(&empty);****  ****pthread\_mutex\_lock(&mutex);****  ****// Add item to buffer****  ****if( count != BUFFER\_SIZE){****  ****buffer[in] = item;****  ****in = (in + 1) % BUFFER\_SIZE;****  ****count++;****  ****success = 0;****  ****}****  ****else****  ****success = -1;****  ****pthread\_mutex\_unlock(&mutex);****  ****sem\_post(&full);****    ****return success;****  ****}****  ****// Remove an object from the buffer, placing it in item.****  ****// Returns 0 if successful, -1 indicating error****  ****int remove\_item(buffer\_item \*item){****  ****int success;****    ****sem\_wait(&full);****  ****pthread\_mutex\_lock(&mutex);****    ****// Remove item from buffer to item****  ****if( count != 0){ \*item = buffer[out];****  ****out = (out + 1) % BUFFER\_SIZE;****  ****count--;****  ****success = 0;****  ****}****  ****else****  ****success = -1;**** |
| ****pthread\_mutex\_unlock(&mutex);****  ****sem\_post(&empty);****    ****return success;****  ****}****  ****void \*producer(void \*param){****  ****buffer\_item item;****  ****while(1){****  ****sleep(rand() % 5 + 1); // Sleep randomly between 1 and 5 seconds****    ****item = rand();****  ****if(insert\_item(item))****  ****printf("Error occured\n");****  ****else****  ****printf("Producer produced %d\n", item);****  ****}****  ****}void \*consumer(void \*param){****  ****buffer\_item item;****  ****while(1){****  ****sleep(rand() % 5 + 1); // Sleep randomly between 1 and 5 seconds****  ****if(remove\_item(&item))****  ****printf("Error occured\n");****  ****else****  ****printf("Consumer consumed %d\n", item);****  ****}****  ****}**** |