操作系统实验二 进程管理

班级：2017211314 学号：2017213508 学生：蒋雪枫

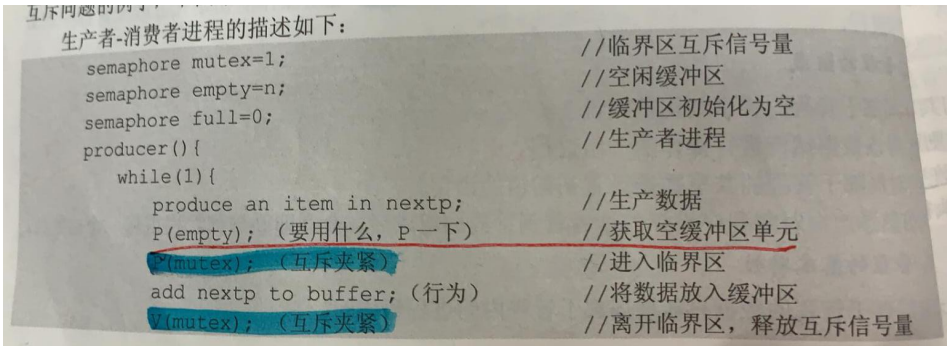
一、Background and Summary

In this chapter of process synchronization , we learned a typical process synchronization scenario named Bounded-buffer issue,which is also named by producer&consumer issue.

A Bounded-buffer,which means a section of finite size,is shared by a producer and a consumer.These two are processes of two types.Just like an array,producers cannot push to a full buffer and consumers cannot pull from an empty buffer.

In our simulation of this issue,we just let producers push integers into the buffer that increment sequentially like an queue.Meanwhile,consumers will pull integers from the buffer section in the FIFO sequence.

Pseudo code of Bounded-Buffer issue can be introduced like this:



生产者-消费者进程的描述如下：

```
semaphore mutex=1;                    //临界区互斥信号量
semaphore empty=n;                    //空闲缓冲区
semaphore full=0;                     //缓冲区初始化为空
producer(){                           //生产者进程
   while(1){
      produce an item in nextp;       //生产数据
      P(empty);（要用什么，P一下）     //获取空缓冲区单元
      P(mutex);（互斥夹紧）            //进入临界区
      add nextp to buffer;（行为）     //将数据放入缓冲区
      V(mutex);（互斥夹紧）            //离开临界区，释放互斥信号量
      V(full);（提供什么，V一下）      //满缓冲区数加1
   }
}
consumer(){                           //消费者进程
   while(1){
      P(full);                        //获取满缓冲区单元
      P(mutex);                       //进入临界区
      remove an item from buffer;     //从缓冲区中取出数据
      V(mutex);                       //离开临界区，释放互斥信号量
      V(empty);                       //空缓冲区数加1
      consume the item;               //消费数据
   }
}
```

互斥夹紧.

二、实验任务

1） 理解线程同步的思想和方法，学会用线程同步解决临界区问题，本次实验解决生产者消费者问题

2 了解 windows 系统或 linux 系统下中信号量的使用方法。

三、实验环境

Pure Linux C: GCC, Linux-cored Ubuntu OS

Usage:

##Build: gcc -pthread -o buffer BoundedBuffer.c

##Run: ./ buffer <sleep time> <number of producer threads> <first number to produce>

(eg. ./buffer 10 2 2 10)

使用 pthread 库的多线程 C 程序编译需要加上参数 -pthread，这是因为 pthread 并非 Linux 的默认库，需要静态链接。这里最后一个参数可能略显晦涩，但它仅仅是对“物品”的一个编号，用于区分不同的物品，就像流水线一样。

四、实验过程与分析

正式完成本次实验前，阅读《操作系统概念》对应的实验要求可知，我们需要先了解一下 Pthread 头文件提供的 API 使用规则和 Linux 里面的线程、进程的组织规则。另外，我们按要求设置缓冲区大小为 BUFFER_SIZE=5，互斥信号量设置为 pthread_mutex_t mutex;缓冲区放的是一些 int，我们在这里把它抽象成一些 buffer_item。

全局过程和变量说明:

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
#define TRUE 1
typedef int buffer_item;
#define BUFFER_SIZE 5
buffer_item START_NUMBER;
buffer_item buffer[BUFFER_SIZE];
pthread_mutex_t mutex;
sem_t empty;
sem_t full;
int insertPointer = 0, removePointer = 0;
int insert_item(buffer_item item);
int remove_item(buffer_item *item);
void *producer(void *param);
void *consumer(void *param);
```

一、如何创建一个线程?

查文档可知:

pthread_create 是 UNIX 环境创建线程函数，来自于<pthread.h>。

int pthread_create(pthread_t *restrict tidp, const pthread_attr_t *restrict_attr, void*(*start_rtn)(void*), void *restrict arg);第一个参数为指向线程标识符的指针。第二个参数用来设置线程属性(没有必要的话可以不设置)。第三个参数是线程运行函数的地址。最后一个参数是运行函数的参数。对于该函数，返回值：若成功则返回 0，否则返回出错编号。

创建线程和创建子进程类似，首先 pthread_t pid，然后 pthread_create(&pid...)即可。

二、信号量和互斥信号量怎么建立?

对于本项目，我们使用 pthread 提供的无名称信号量和互斥锁。其中，互斥锁采用 pthread_mutex_t 数据类型，使用 pthread_mutex_int(&mutex_lock,NULL)进行初始化,NULL 表示互斥锁的默认值，我们可以通过 pthread_mutex_lock()和 pthread_mutex_unlock()来获取和释放互斥锁。

提出以上问题之后，我们就可以设计好 main 函数的架构了。

```
int main(int argc, char *argv[])
{
    //TODO:用户指定 sleepTime,生产者消费者线程个数,运行时输入即可
    int sleepTime, producerThreads, consumerThreads;
    int i, j;
    if(argc != 5)
    {
        //错误处理
        fprintf(stderr, "Useage: <sleep time> <producer threads> <consumer threads> <start number>\n");
        return -1;
    }
    //字符串自动转整数
    sleepTime = atoi(argv[1]);
    producerThreads = atoi(argv[2]);
    consumerThreads = atoi(argv[3]);
    START_NUMBER = atoi(argv[4]);
    //创建互斥信号量,调用 pthread API
    pthread_mutex_init(&mutex, NULL);
    //创建 full 和 empty 信号量,并且赋予初值
    //这里第一个参数是指信号量的指针,第二个参数 0 表示仅有该进程下线程可以访问该信号量,第三个表示初始化数值
    sem_init(&full, 0, BUFFER_SIZE);
    sem_init(&empty, 0, 0);
    pthread_t pid, cid;
    for(i = 0; i < producerThreads; i++){
        //创建 producer 线程,个数由用户指定,去运行 producer 过程
        pthread_create(&pid,NULL,&producer,NULL);
    }
    for(j = 0; j < consumerThreads; j++){
        //同理
        pthread_create(&cid,NULL,&consumer,NULL);
    }
    //sleep for a while
    sleep(sleepTime);
    return 0;
}
```

然后是根据模型，来编写生产者和消费者的基本代码：

```
void *producer(void *param)
{
    buffer_item item;
    while(TRUE) {
        sleep(2);
        sem_wait(&full);
        pthread_mutex_lock(&mutex);
        item = START_NUMBER++;
        insert_item(item);
        printf("Producer %u produced %d \n", (unsigned int)pthread_self(), item);
        pthread_mutex_unlock(&mutex);
        sem_post(&empty);
    }
}

void *consumer(void *param)
{
    buffer_item item;
    while(TRUE){
        sleep(2);
        sem_wait(&empty);
        pthread_mutex_lock(&mutex);
        remove_item(&item);
        printf("Consumer %u consumed %d \n", (unsigned int)pthread_self(), item);
        pthread_mutex_unlock(&mutex);
        sem_post(&full);
    }
}
```

对于具体的 push 和 pull 操作：

```
int insert_item(buffer_item item)
{
    buffer[insertPointer] = item;
    insertPointer = (insertPointer + 1) % BUFFER_SIZE;
    return 0;
}

int remove_item(buffer_item *item)
{
    *item = buffer[removePointer];
    //buffer[removePointer]=0  可要可不要~
    removePointer = (removePointer + 1) % BUFFER_SIZE;
    return 0;
}
```

这里，*item 表示我们传递的是指针，在过程中会给 item 赋值，并且该值会返回给调用它的过程，记录到 consumer 的 item 变量中，用于打印调试信息。而 insert 的时候只用吧这个数字(序号)，我们是在 producer 过程中得到了 item 的值(序号)，直接在 insert 的时候往 buffer 里面填如就好了。

运行结果：

```
jxf@2BB3NCKF8U1RJ2A:/mnt/d/zjj$ gcc -pthread -o buffer BoundedBuffer.c
BoundedBuffer.c: In function ‘producer’:
BoundedBuffer.c:43:3: warning: implicit declaration of function ‘sleep’ [-Wimplicit-fun
ction-declaration]
   sleep(2);
   ^~~~~

jxf@2BB3NCKF8U1RJ2A:/mnt/d/zjj$ ./buffer 10 2 2 10
Producer 1545406208 produced 10
Producer 1536952064 produced 11
Consumer 1520043776 consumed 10
Consumer 1528497920 consumed 11
Producer 1545406208 produced 12
Consumer 1520043776 consumed 12
Producer 1536952064 produced 13
Consumer 1528497920 consumed 13
Producer 1545406208 produced 14
Consumer 1520043776 consumed 14
Producer 1536952064 produced 15
Consumer 1528497920 consumed 15
Producer 1545406208 produced 16
Consumer 1520043776 consumed 16
Producer 1536952064 produced 17
Consumer 1528497920 consumed 17
jxf@2BB3NCKF8U1RJ2A:/mnt/d/zjj$
```

五、实验总结

纸上得来终觉浅，绝知此事要躬行。本次实验让我结合 BoundedBuffer 问题展开分析，动手完成了本次代码后，收益匪浅，用到了以前不太熟悉的 C 语言语法，也进一步认识到了在 Linux 操作系统下对于进程同步的机制，接触到了 Linux style 的 semaphore。

学习是无止境的，进程同步本身还是比较复杂的一回事，期末也可能是一大考点，之后学生也会多加练习。