作业 7
唐嘉良
2020K8009907032

1. 设有两个优先级相同的进程 T1，T2 如下。令信号量 S1，S2 的初值为 0，已知 z=2，试 问 T1，T2 并发运行结束后 x=? y=? z=?

```
线程 T1        线程 T2
y:=1;          x:=1;
y:=y+2;        x:=x+1;
V(S1);         p(S1);
z:=y+1;        x:=x+y;
P(S2);         V(S2);
y:=z+y;        z:=x+z;
```

注:请分析所有可能的情况，并给出结果与相应执行顺序。

答：T2 第三行须在 T1 第三行之后执行，且 T2 第五行须在 T1 第五行之前执行。根据这一原则，有不同的执行顺序。观察语句，发现可能存在相关的语句只有 z:=y+1、z:=x+z、y:=z+y 这三条。首先 x:=x+y 肯定在 T1 第三行和第五行之间，且它与同样在 T1 第三行和第五行之间的 z:=y+1 并无相关。而 z:=y+1 和 y:=z+y 的相对位置已经确定，所以只需要讨论 z:=x+z 的位置。

1) 如果该语句位于 z:=y+1 之前，则三条相关语句的顺序（由先到后）为 z:=x+z、z:=y+1、y:=z+y，运行结束后(x,y,z)=(5,7,4)；

2) 如果该语句位于 z:=y+1 之后且位于 y:=z+y 之前，则三条相关语句的顺序（由先到后）为 z:=y+1、z:=x+z、y:=z+y，运行结束后(x,y,z)=(5,12,9)；

3) 如果该语句位于 y:=z+y 之后，则三条相关语句的顺序（由先到后）为 z:=y+1、y:=z+y、z:=x+z，运行结束后(x,y,z)=(5,7,9)；

2. 银行有 n 个柜员,每个顾客进入银行后先取一个号,并且等着叫号,当一个柜员空闲后,就叫下一个号.
请使用 PV 操作分别实现:
//顾客取号操作 Customer_Service
//柜员服务操作 Teller_Service

答：C 语言风格伪代码设计如下：

```
int counter = n; //信号量：空闲柜员数
int customer = 0; //信号量：要服务的顾客数
int Customer_Service(){
    P(counter);//取号，如果柜员有空就直接叫号，否则进入等待队列
    call();//叫号，相当于顾客到柜台（进 buffer）
    V(customer);
}

int Teller_Service(){
    P(customer);
    serve();//服务,相当于顾客离开柜台（出 buffer）
    V(counter);
}
```

3. 多个线程的规约(Reduce)操作是把每个线程的结果按照某种运算(符合交换律和结合律) 两两合并直到得到最终结果的过程。
试设计管程 monitor 实现一个 8 线程规约的过程，随机初始化 16 个整数，每个线程通过 调用 monitor.getTask 获得 2 个数，相加后，返回一个数 monitor.putResult ，然后再 getTask( ) 直到全部完成退出，最后打印归约过程和结果。
要求:为了模拟不均衡性，每个加法操作要加上随机的时间扰动，变动区间 1~10ms。
提示: 使用 pthread_系列的 cond_wait, cond_signal, mutex 实现管程
使用 rand( )函数产生随机数，和随机执行时间。
答：设计代码如下：

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <pthread.h>
#include <syscall.h>
#include <stdbool.h>
#include <sys/time.h>

typedef struct {
    int a, b;
} Pair;

typedef struct Node {
    int num;
    struct Node * next;
} Node;

typedef struct {
    int cnt;
    Node * head;
    Node * nail;
} Queue;

void Queue_init(Queue * que) {
    que->cnt = 0;
    que->head = NULL;
    que->nail = NULL;
}

void push(Queue * que, int num) {
    Node * new_node = malloc(sizeof(Node));
    new_node->num = num;
```

```c
    if (que->head == NULL) {
        que->head = new_node;
        que->nail = new_node;
        new_node->next = NULL;
    } else {
        que->nail->next = new_node;
        que->nail = new_node;
    }
    que->cnt += 1;
}

int pop(Queue * que) {
    Node * old_node = que->head;
    int res = old_node->num;
    if (old_node->next == NULL) {
        que->head = NULL;
        que->nail = NULL;
    } else {
        que->head = old_node->next;
    }
    que->cnt -= 1;
    free(old_node);
    return res;
}

int Queue_print(Queue * que) {
    for (Node * p = que->head; p != NULL; p = p->next) {
        printf("%d ", p->num);
    }
    printf("\n");
}

typedef struct {
    Queue que;
    pthread_mutex_t lock;
    pthread_cond_t can_reduce;
    int remind_task;
} Monitor;

Monitor task_monitor;

void Monitor_init(Monitor * p) {
    p->remind_task = 15;//////////////
    Queue_init(&p->que);
```

```
    for (int i = 0; i < 16; i++) {////////
        push(&p->que, rand()%100 );
    }
    pthread_mutex_init(&p->lock, NULL);
}

bool Monitor_getTask(Monitor * p, Pair * result) {
    pthread_mutex_lock(&p->lock);

    if (p->remind_task <= 0) {
        pthread_mutex_unlock(&p->lock);
        return false;
    }//no task to do

    p->remind_task -= 1;

    while (p->que.cnt < 2) {
        pthread_cond_wait(&p->can_reduce, &p->lock);
    }//still have tasks to do, but nums in queue are not enough
    //wait until other threads put res into queue

    result->a = pop(&p->que);
    result->b = pop(&p->que);
    printf("get task: %d, %d\n", result->a, result->b);

    pthread_mutex_unlock(&p->lock);
    return true;
}

void Monitor_putResult(Monitor * p, int res) {
    pthread_mutex_lock(&p->lock);

    push(&p->que, res);
    printf("put result: %d\n", res);
    printf("The queue now is ");
    Queue_print(&task_monitor.que);

    pthread_cond_signal(&p->can_reduce);//tell other threads

    pthread_mutex_unlock(&p->lock);
}

void * func_thread_reduce(void * arg) {
    Pair task;
```

```
    while (1) {
        bool get_task = Monitor_getTask(&task_monitor, &task);
        if (!get_task) {
            printf("A thread finished!\n");
            return NULL;
        }
        sleep((rand()%10)/1000); //random time delay: 1~10ms
        int res = task.a + task.b;
        Monitor_putResult(&task_monitor, res);
    }
    return NULL;
}

int main() {
    Monitor_init(&task_monitor);
    printf("The initial 16 integers are:\n");
    Queue_print(&task_monitor.que);
    printf("Processing...\n");

    pthread_t thread[8];
    for (int i = 0; i < 8; i++) {
        pthread_create(thread + i, NULL, func_thread_reduce, NULL);
    }
    for (int i = 0; i < 8; i++) {
        pthread_join(thread[i], NULL);
    }

    printf("\nPass\nThe final result is: ");
    Queue_print(&task_monitor.que);

    return 0;
}
```

运行结果如下：（初始化的随机数范围为 0~100，每次 push 两数之和入队之后都会打印一遍当前队列）

```
ubuntu@ubuntu:~/Desktop$ ./hw
The initial 16 integers are:
83 86 77 15 93 35 86 92 49 21 62 27 90 59 63 26
Processing...
get task: 83, 86
get task: 77, 15
get task: 93, 35
get task: 86, 92
get task: 49, 21
put result: 128
The queue now is 62 27 90 59 63 26 128
get task: 62, 27
put result: 92
The queue now is 90 59 63 26 128 92
get task: 90, 59
put result: 169
The queue now is 63 26 128 92 169
get task: 63, 26
put result: 70
The queue now is 128 92 169 70
get task: 128, 92
put result: 178
The queue now is 169 70 178
get task: 169, 70
put result: 89
The queue now is 178 89
get task: 178, 89
put result: 149
The queue now is 149
put result: 89
The queue now is 149 89
A thread finished!
get task: 149, 89
put result: 267
The queue now is 267
A thread finished!
put result: 239
The queue now is 267 239
A thread finished!
put result: 220
The queue now is 267 239 220
A thread finished!
get task: 267, 239
put result: 238
The queue now is 220 238
A thread finished!
get task: 220, 238
put result: 506
The queue now is 506
A thread finished!
put result: 458
The queue now is 506 458
A thread finished!
get task: 506, 458
put result: 964
The queue now is 964
A thread finished!

Pass
The final result is: 964
ubuntu@ubuntu:~/Desktop$
```