南京航空航天大學

NANJING UNIVERSITY OF AERONAUTICS AND ASTRONAUTICS

计算机科学与技术学院

/人工智能学院

操作系统课程作业

162140222 黄钰轩

2024 年 5 月 10 日

题目 1. 桌上有一空盘，只允许存放一个水果. 爸爸专向盘中放橙子，妈妈专向盘中放苹果，女儿专等吃橙子，儿子专等吃苹果. 规定当盘空时一次只能放一个水果供吃者自用，请用 P、V 操作实现爸爸、妈妈、女儿、儿子四个并发进程的同步.

解答. 具体代码如下:

Listing 1: **fruit.c**

```c
#include <thread.h>
#include <thread-sync.h>

#define CAN_PRODUCE (fruit_in_dish == 0)
#define CAN_CONSUME (fruit_in_dish == 1)

typedef enum FruitType // 水果类型
{
    NONE, // 无
    ORANGE, // 橙子
    APPLE // 苹果
}FruitType;

int fruit_in_dish; // 盘中水果数量
FruitType fruit_type; // 盘中水果类型
mutex_t mutex, orange, apple; // 线程锁、橙子锁、苹果锁
```

```
17 sem_t fill, empty; // 信号量
18
19 void init()
20 {
21     fruit_in_dish = 0;
22     fruit_type = NONE;
23
24     mutex_init(&mutex);
25     mutex_init(&orange);
26     mutex_init(&apple);
27
28     SEM_INIT(&fill, 0);
29     SEM_INIT(&empty, 1);
30 }
31
32 void father()
33 {
34     while (1)
35     {
36         P(&empty);
37         mutex_lock(&mutex);
38         mutex_lock(&orange);
39
40         printf("put orange\n");
41         fflush(stdout);
42
43         fruit_in_dish ++ ;
44         fruit_type = ORANGE;
45
46         mutex_unlock(&orange);
47         mutex_unlock(&mutex);
48         V(&fill);
49
50         usleep(1000000);
51     }
52 }
53
54 void mother()
55 {
56     while (1)
57     {
58         P(&empty);
59         mutex_lock(&mutex);
60         mutex_lock(&apple);
61
62         printf("put apple\n");
63         fflush(stdout);
64
65         fruit_in_dish ++ ;
```

```
66          fruit_type = APPLE;
67
68          mutex_unlock(&apple);
69          mutex_unlock(&mutex);
70          V(&fill);
71
72          usleep(1000000);
73      }
74  }
75
76  void daughter()
77  {
78      while (1)
79      {
80          P(&fill);
81          mutex_lock(&mutex);
82          mutex_lock(&orange);
83
84          while (fruit_type != ORANGE)
85          {
86              mutex_unlock(&orange);
87              mutex_unlock(&mutex);
88              V(&fill);
89              P(&fill);
90              mutex_lock(&mutex);
91              mutex_lock(&orange);
92          }
93
94          printf("eat orange\n");
95          fflush(stdout);
96
97          fruit_in_dish -- ;
98          fruit_type = NONE;
99
100         mutex_unlock(&orange);
101         mutex_unlock(&mutex);
102         V(&empty);
103
104         usleep(1000000);
105     }
106 }
107
108 void son()
109 {
110     while (1)
111     {
112         P(&fill);
113         mutex_lock(&mutex);
114         mutex_lock(&apple);
```

```
115
116         while (fruit_type != APPLE)
117         {
118             mutex_unlock(&apple);
119             mutex_unlock(&mutex);
120             V(&fill);
121             P(&fill);
122             mutex_lock(&mutex);
123             mutex_lock(&apple);
124         }
125
126         printf("eat apple\n");
127         fflush(stdout);
128
129         fruit_in_dish -- ;
130         fruit_type = NONE;
131
132         mutex_unlock(&apple);
133         mutex_unlock(&mutex);
134         V(&empty);
135
136         usleep(1000000);
137     }
138 }
139
140 int main()
141 {
142     init();
143
144     for (int i = 0; i < 1; i ++ )
145     {
146         create(father);
147         create(daughter);
148         create(mother);
149         create(son);
150     }
151 }
```

执行结果下:

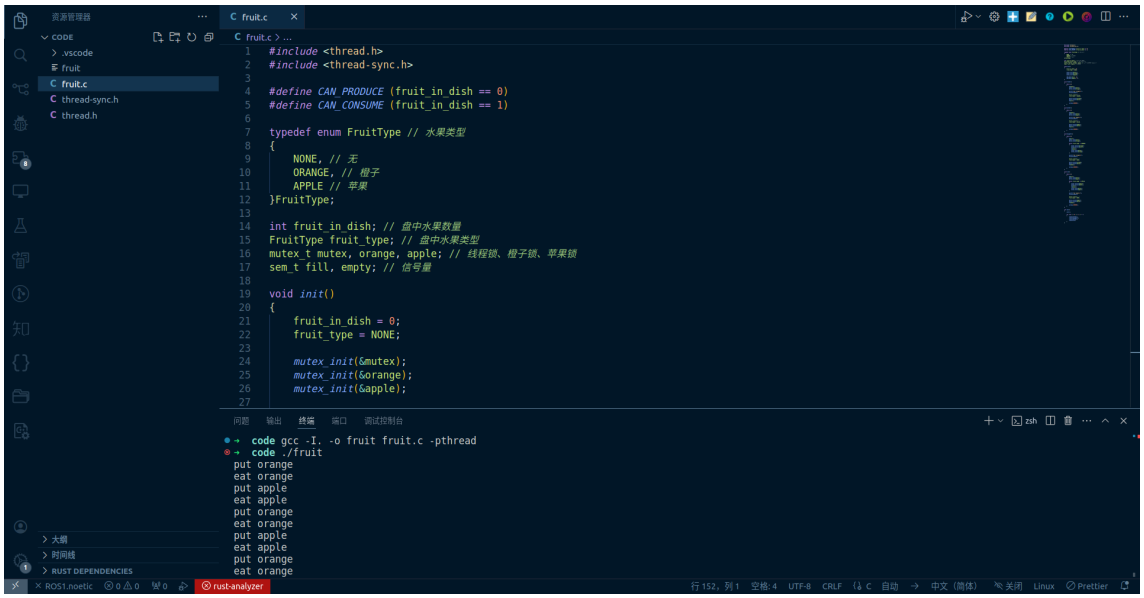

图 1: 使用信号量实现同步

题目 1 的注记. 代码中所包含的头文件是一个将 POSIX 封装过后的“最简”线程库. 本题的实现也参考了2024 南京大学操作系统课程中的示例代码.

题目 2. 假设有个南北向的桥，仅能容同方向的人顺序走过，相对方向的两个人则无法通过. 现在桥南北端都有过桥人，把每个过桥人当成一个进程，用 P、V 操作实现管理.

解答. 假定桥上最多允许 3 人同时通行，这里给出过桥过程的代码实现:

Listing 2: **bridge.c**

```c
#include <thread.h>
#include <thread-sync.h>

#define NORTH 0
#define SOUTH 1
#define MAX_ON_BRIDGE 3

mutex_t mutex;
cond_t north_cond, south_cond;
sem_t bridge_sem;

int current_direction = NORTH; // 桥的方向
int num_on_bridge = 0;         // 桥上的行人数量
int waiting_north = 0;         // 等待向北过桥的人数
int waiting_south = 0;         // 等待向南过桥的人数
```

```
16
17  void cross_bridge() // 模拟过桥时间
18  {
19      usleep(1000000);
20  }
21
22  void init_bridge()
23  {
24      mutex_init(&mutex);
25      pthread_cond_init(&north_cond, NULL);
26      pthread_cond_init(&south_cond, NULL);
27      SEM_INIT(&bridge_sem, MAX_ON_BRIDGE);
28  }
29
30  void north_to_south(int id)
31  {
32      mutex_lock(&mutex);
33
34      waiting_north ++ ;
35      while (current_direction == SOUTH
36             || num_on_bridge >= MAX_ON_BRIDGE)
37      {
38          printf("Person %d from north wants to cross to south but has to wait\n", id);
39          cond_wait(&north_cond, &mutex);
40      }
41      waiting_north -- ;
42
43      P(&bridge_sem);
44      num_on_bridge ++ ;
45      current_direction = NORTH;
46
47      printf("Person %d is crossing from north to south\n", id);
48      mutex_unlock(&mutex);
49
50      cross_bridge();
51
52      mutex_lock(&mutex);
53      num_on_bridge -- ;
54      if (num_on_bridge == 0)
55          if (waiting_south > 0)
56          {
57              current_direction = SOUTH;
58              cond_broadcast(&south_cond);
59          }
60          else
61              cond_broadcast(&north_cond);
62      V(&bridge_sem);
63      mutex_unlock(&mutex);
64  }
```

```
65
66  void south_to_north(int id)
67  {
68      mutex_lock(&mutex);
69
70      waiting_south ++ ;
71      while (current_direction == NORTH
72             || num_on_bridge >= MAX_ON_BRIDGE)
73      {
74          printf("Person %d from south wants to cross to north but has to wait\n", id);
75          cond_wait(&south_cond, &mutex);
76      }
77      waiting_south -- ;
78
79      P(&bridge_sem);
80      num_on_bridge ++ ;
81      current_direction = SOUTH;
82
83      printf("Person %d is crossing from south to north\n", id);
84      mutex_unlock(&mutex);
85
86      cross_bridge();
87
88      mutex_lock(&mutex);
89      num_on_bridge -- ;
90      if (num_on_bridge == 0)
91          if (waiting_north > 0)
92          {
93              current_direction = NORTH;
94              cond_broadcast(&north_cond);
95          } else
96              cond_broadcast(&south_cond);
97      V(&bridge_sem);
98      mutex_unlock(&mutex);
99  }
100
101 int main()
102 {
103     init_bridge();
104
105     for (int i = 0; i < 5; i ++ )
106     {
107         create(north_to_south);
108         create(south_to_north);
109     }
110
111     return 0;
112 }
```

执行结果下:

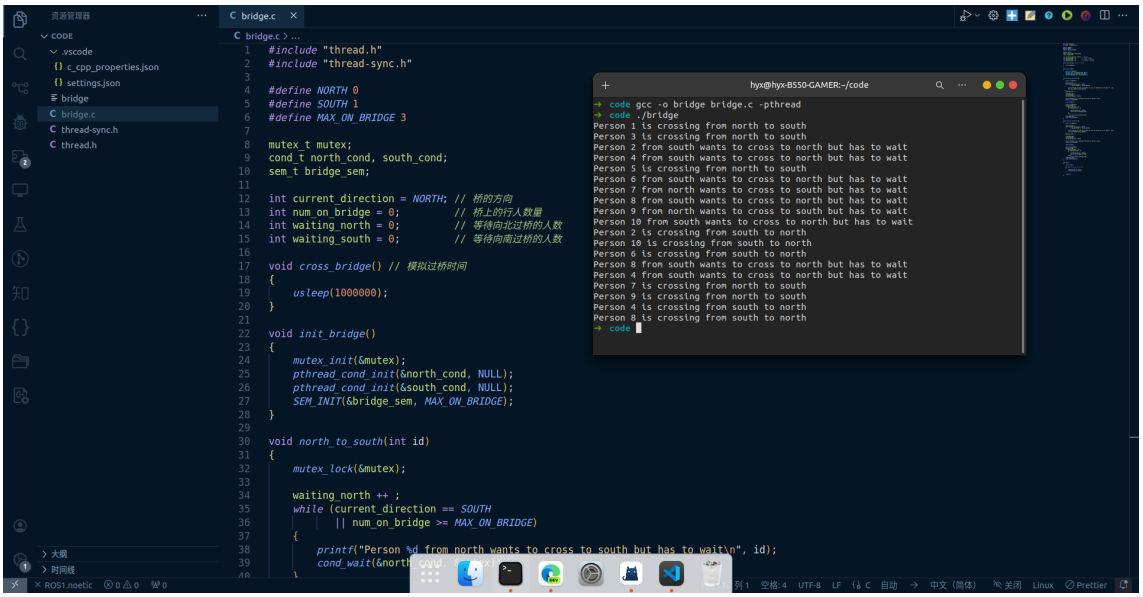

图 2: 过桥问题