## 2.6 RT-Thread——单生产者多消费者的设计

### 2.6.1 设计方案

设置四个信号量：

①mutex：实现对于共享资源——碟子的互斥

②empty：实现碟子资源的控制，初始有两个空位

③apple：生产苹果的信号量

④banana：生产香蕉的信号量

三个线程分别为：productor\_tid、consumer\_son\_tid和consumer\_daughter\_tid，

导出msh命令列表：MSH\_CMD\_EXPORT(productor\_consumer, productor\_consumer sample);

### 2.6.2 实现方法

/\*

\* productor\_consumer.c

\*

\* created: 2022/1/12

\* author:

\*/

#include <rtthread.h>

#include <stdio.h>

#include <stdlib.h>

#define THREAD\_PRIORITY 0

#define THREAD\_STACK\_SIZE 2048

#define THREAD\_TIMESLICE 5

struct rt\_semaphore mutex;//互斥量

struct rt\_semaphore empty;//信号量

struct rt\_semaphore apple;

struct rt\_semaphore banana;

static rt\_thread\_t productor\_tid = RT\_NULL;

static rt\_thread\_t consumer\_son\_tid = RT\_NULL;

static rt\_thread\_t consumer\_daughter\_tid = RT\_NULL;

void productor\_entry(void \*parameter)

{

int num = 10; //父亲一共给10个水果

while (num--)

{

rt\_sem\_take(&empty, RT\_WAITING\_FOREVER);//同步

rt\_sem\_take(&mutex, RT\_WAITING\_FOREVER);//互斥

int now = rand() % 2; //考虑放什么水果

if(now == 0)

{

rt\_sem\_release(&apple);

rt\_kprintf("father put a apple\n");

}

else

{

rt\_sem\_release(&banana);

rt\_kprintf("father put a banana\n");

}

rt\_sem\_release(&mutex);

rt\_thread\_mdelay(200);

}

}

void consumer\_son\_entry(void \*parameter)

{

while(1)

{

rt\_sem\_take(&apple, RT\_WAITING\_FOREVER);//同步

rt\_sem\_take(&mutex, RT\_WAITING\_FOREVER);//互斥

rt\_kprintf("son get a apple\n");

rt\_sem\_release(&empty); //取走苹果

rt\_sem\_release(&mutex);

rt\_thread\_mdelay(200);

}

}

void consumer\_daughter\_entry(void \*parameter)

{

while(1)

{

rt\_sem\_take(&banana, RT\_WAITING\_FOREVER);//同步

rt\_sem\_take(&mutex, RT\_WAITING\_FOREVER);//互斥

rt\_kprintf("daughter get a banana\n");

rt\_sem\_release(&empty); //取走香蕉

rt\_sem\_release(&mutex);

rt\_thread\_mdelay(200);

}

}

int productor\_consumer(void)

{

rt\_sem\_init(&mutex, "mutex", 1, RT\_IPC\_FLAG\_FIFO); //初始化信号量

rt\_sem\_init(&empty, "empty", 2, RT\_IPC\_FLAG\_FIFO);

rt\_sem\_init(&apple, "apple", 0, RT\_IPC\_FLAG\_FIFO);

rt\_sem\_init(&banana, "banana", 0, RT\_IPC\_FLAG\_FIFO);

//父亲

productor\_tid = rt\_thread\_create("father",

productor\_entry, RT\_NULL,

THREAD\_STACK\_SIZE,

THREAD\_PRIORITY, THREAD\_TIMESLICE);

if (productor\_tid != RT\_NULL)

rt\_thread\_startup(productor\_tid);

//儿子

consumer\_son\_tid = rt\_thread\_create("son",

consumer\_son\_entry, RT\_NULL,

THREAD\_STACK\_SIZE,

THREAD\_PRIORITY + 1, THREAD\_TIMESLICE);

if (consumer\_son\_tid != RT\_NULL)

rt\_thread\_startup(consumer\_son\_tid);

//女儿

consumer\_daughter\_tid = rt\_thread\_create("daughter",

consumer\_daughter\_entry, RT\_NULL,

THREAD\_STACK\_SIZE,

THREAD\_PRIORITY + 1, THREAD\_TIMESLICE);

if (consumer\_daughter\_tid != RT\_NULL)

rt\_thread\_startup(consumer\_daughter\_tid);

return 0;

}

/\* 导 出 到 msh 命 令 列 表 中 \*/

MSH\_CMD\_EXPORT(productor\_consumer, productor\_consumer sample);

