CS302
Operating System
Lab 5

Concurrency: Mutual Exclusion and Synchronization 2

April 4th , 2018

Dongping Zhang
cadongllas@gmail.com

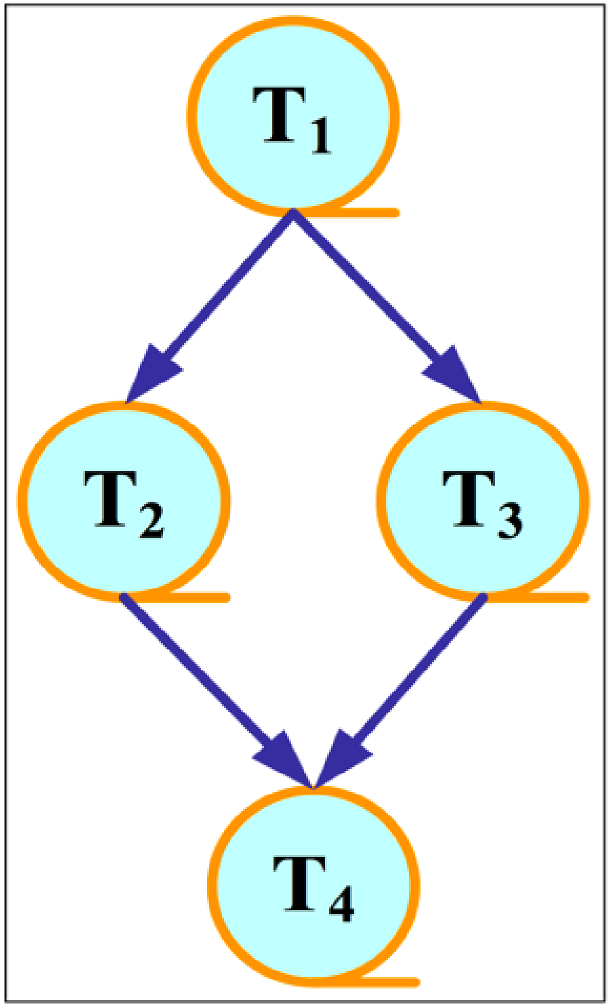
Semaphores

- Semaphore is a variable that has an integer value
  - Initialize: a nonnegative integer value
  - semWait (P): decreases the semaphore value. the value becomes negative, then the process executing the semWait is blocked.
  - semSignal (V): increases semaphore value. If the resulting value is less than or equal to zero, then a process is blocked by a semWait operation, if any, is unblocked.

Semaphores

```
struct semaphore {
       int count;
       queueType queue;
};
void semWait(semaphore s)
{
       s.count--;
       if (s.count < 0) {
         /* place this process in s.queue */;
         /* block this process */;
       }
}
void semSignal(semaphore s)
{
       s.count++;
       if (s.count<= 0) {
         /* remove a process P from s.queue */;
         /* place process P on ready list */;
       }
}
```

Semaphores



Mutual Exclusion using Semaphores

```
/* program mutualexclusion */
const int n = /* number of processes */;
semaphore s = 1;
void P(int i)
{
     while (true) {
          semWait(s);
          /* critical section */;
          semSignal(s);
          /* remainder */;
     }
}
void main()
{
     parbegin (P(1), P(2),…, P(n));
}
```

Semaphore in C

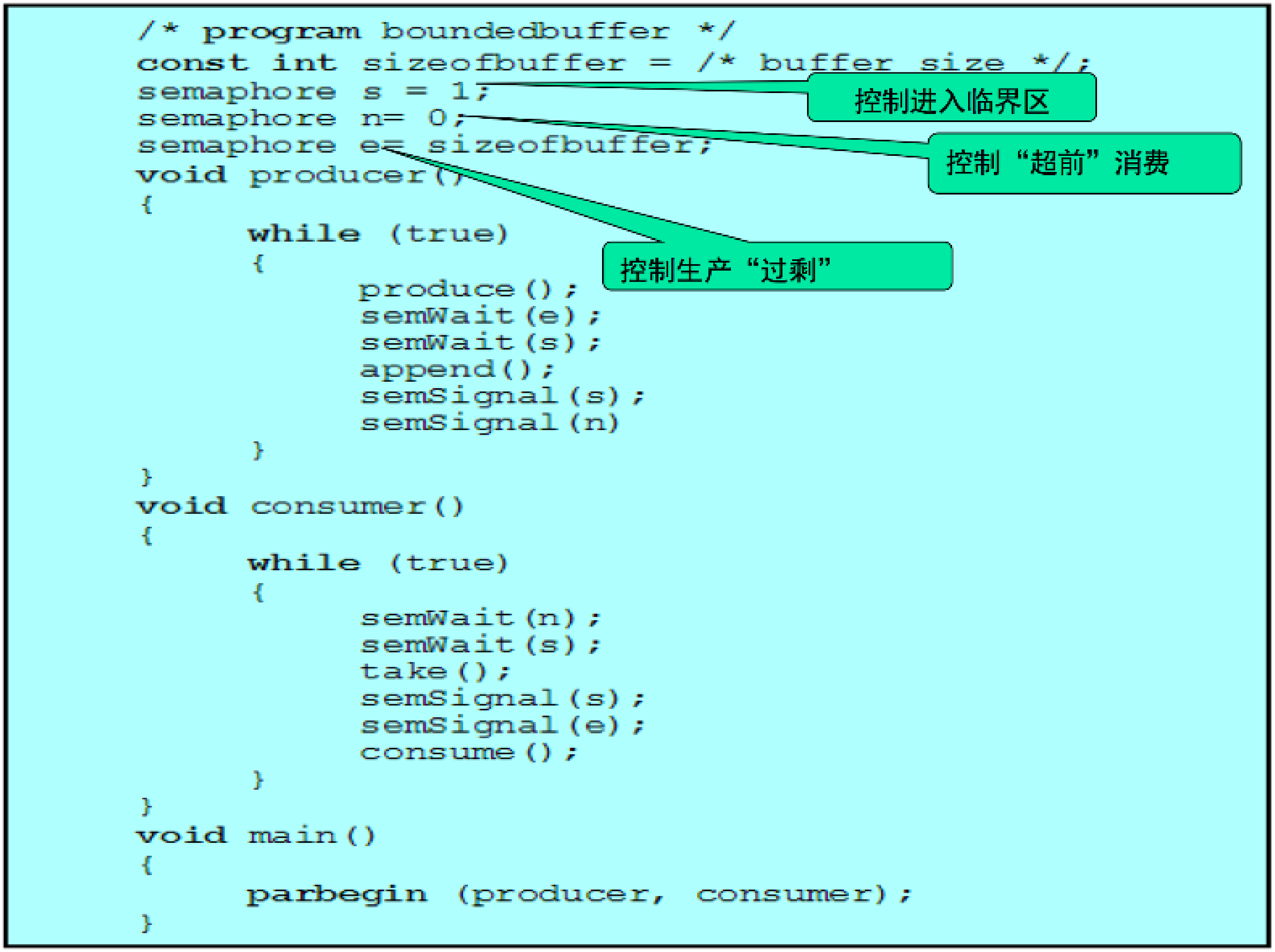
- **semaphore.c** shows how to use these functions to create, operate and remove named semaphore.
- compile semaphore.c like this: gcc semaphore.c -pthreaad -o semaphor

Function	Description
sem_open	Opens/creates a named semaphore for use by a process
sem_wait	lock a semaphore
sem_post	unlock a semaphore
sem_close	Deallocates the specified named semaphore
sem_unlink	Removes a specified named semaphore

Producer/Consumer Problem (生产者消费者问题)

- One or more producers are generating data and placing these in a buffer
- A single consumer is taking items out of the buffer one at time
- Only one producer or consumer may access the buffer at any one time

Producer/Consumer Problem



```
/* program boundedbuffer */
const int sizeofbuffer = /* buffer size */;
semaphore s = 1;
semaphore n= 0;
semaphore e= sizeofbuffer;
void producer()
{
      while (true)
      {
            produce();
            semWait(e);
            semWait(s);
            append();
            semSignal(s);
            semSignal(n)
      }
}
void consumer()
{
      while (true)
      {
            semWait(n);
            semWait(s);
            take();
            semSignal(s);
            semSignal(e);
            consume();
      }
}
void main()
{
      parbegin (producer, consumer);
}
```

控制进入临界区
控制“超前”消费
控制生产“过剩”

Readers/Writers Problem

- There is a data area shared among a number of processes. The data area could be a file, a block of main memory, or even a bank of processor registers. There area number of processes that only read the data area(readers) and a number that only write to the data area(writers). The conditions that must be satisfied are as follows:
  - ➢Any number of readers may simultaneously read the file
  - ➢Only one writer at a time may write to the file
  - ➢If a writer is writing to the file, no reader may read it

读者-写者问题

- 有两组并发进程
  - 读者和写者,共享一组数据区
- 要求
  - 允许多个读者同时执行读操作
  - 不允许读者、写者同时操作
  - 不允许多个写者同时操作

读者-写者问题

- 互斥关系
  - 读者和写者不能同时进入共享数据区
  - 多个写者不能同时进入共享数据区
  - 多个读者可以同时进入共享数据区
- 同步关系
  - 读者进入缓冲区，写者必须等待
  - 写者进入缓冲区，读者必须等待

信号量描述

- 读者来
  - 无读者、写者，新读者可以读
  - 有写者等，但有其它读者正在读，则新读者也可以读
  - 有写者写，新读者等
- 写者来
  - 无读者，新写者可以写
  - 有读者，新写者等待
  - 有其它写者，新写者等待

读者-写者问题

- 两个进程
  - ➢Reader、Writer
  - ➢读者与写者间的互斥信号量：Wmutex＝1
  - ➢多个读者间的互斥信号量：Rmutex＝1
- Readcount：正在读取的进程数目
  - ➢Readcount＝0时允许写

读者部分

```
wait(rmutex);
If readcount=0 then
            wait(wmutex);
    Readcount:=readcount+1;
signal(rmutex);
……执行读取操作
wait(rmutex);
Readcount:=readcount-1
if readcount=0 then
            signal(wmutex);
signal(rmutex);
```

写者部分

```
wait(wmutex);
......执行写操作
signal(wmutex);
```

Report for Lab 5

- Please complete the report
- Please complete ”read.c” and “write.c”, and your output should be same to “output_sample.txt”
- Name your document as **OS_Lab5_XXX_YYYYYYYY.zip**, contains :
  - OS_Lab5_XXX_YYYYYYYY.doc
  - read.c
  - write.c
- Replace XXX with your name and replace YYYYYYYY with your student id
- Suche as **OS_Lab5_张三_**12345678.zip
- Check the **blackboard** for deadline