Recitation – Week 5

PRANUT JAIN

Plan for today

Quiz discussion

IPC example – Sleeping Barber Problem

Quiz 1

Average: 11.5

Max: 23

The Sleeping Barber Problem

The sleeping barber problem (cont'd)

- Barber shop - *one* service provider; *many* customers
- A finite waiting queue
- One customer is served at a time
- Service provider, *barber*, sleeps when no customers are waiting
- Customer *leaves* if shop is full
- Customer *sleeps* while waiting in queue

The sleeping barber: implementation

```
#define    CHAIRS    5
semaphore customers = 0; // number of waiting customers
Semaphore barbers = 0; // number of available barbers: either 0 or 1
int  waiting = 0; // copy of customers for reading
Semaphore mutex = 1;   // mutex for accessing 'waiting'
void  barber(void)   {
      while(TRUE)      {
        down(customers); // block if no customers
        down(mutex); // access to 'waiting'
        waiting = waiting - 1;
        up(barbers); // barber is in..
        up(mutex);  // release 'waiting'
        cut_hair();      }
}
```

The sleeping barber: implementation (cont'd)

```
void  customer(void) {
    down(mutex);    // access to `waiting'
    if(waiting < CHAIRS)        {
      waiting = waiting + 1; // increment waiting
      up(customers); // wake up barber
      up(mutex); // release 'waiting'
      down(barbers); // go to sleep if barbers=0
      get_haircut();
      }
    else   {
      up(mutex);                          /* shop full  .. leave */
    }}
```

Any problem with this code? Two customers on chair at once

The sleeping barber: correct synchronization

```
#define    CHAIRS    5
semaphore customers = 0; // number of waiting customers
Semaphore barbers = 0; // number of available barbers: either 0 or 1
Semaphore  mutex = 1;   // mutex for accessing 'waiting'
Semaphore  synch = 0;   // synchronizing the service operation
int  waiting = 0; // copy of customers for reading

void  barber(void)   {
     while(TRUE)      {
       down(customers); // block if no customers
       down(mutex); // access to 'waiting'
       waiting = waiting - 1;
       up(barbers); // barber is in..
       up(mutex);  // release 'waiting'
       cut_hair();
       down(synch)   //wait for customer to leave }
}
```

The sleeping barber: correct synchronization (cont'd)

```
void  customer(void) {
     down(mutex);    // access to `waiting'
     if(waiting < CHAIRS)         {
       waiting = waiting + 1; // increment waiting
       up(customers); // wake up barber
       up(mutex); // release 'waiting'
       down(barbers); // go to sleep if barbers=0
       get_haircut();
       up(sync); //synchronize service
       }
     else    {
       up(mutex);                              /* shop full  .. leave */
     }}
```