CSCI 344/715
Lecturer: Dr. Simina Fluture

Lecture # 4

Topics: *Signal and exit* monitors
Single Resource monitors
Synchronization problems using (Signal and Exit) monitors
Producer - Consumer (Bounded Buffer)
Readings: Web lecture and Class notes
[SH] ch5 (5.2)

Signal and Exit monitors

Single Resource Monitor
There is a shared resource over which we need the mutual Exclusion condition satisfied.
The resource can be *in use* or it can be *available*.

***Shared resource**:*

Suppose that we need Mutual Exclusion over the USE of the resource.

Two ***operations over the resource***:

One ***condition variable***:

Two ***operations on the condition**:*

Initialization code:
Because we cannot directly check the condition variable, we will use a shared Boolean variable ***inUse***

Fig. Single Resource Monitor

Execution code for each thread:

*Code for the **service methods**:*

Synchronization Problems

The Bounded Buffer , Producer-Consumer problem

A producer thread and a consumer thread communicate by sharing a buffer having *n* slots. Let's consider that the buffer is represented by a circular array that can be filled up with messages. The producer sends a message to the consumer by depositing the message at the rear of the buffer. The consumer receives a message by fetching the one at the front of the buffer. Synchronization is required so that a message is not deposited if the buffer is Full and a message is not fetched if the buffer is Empty.

Shared resource:

Two *types of threads*:

Operations (methods) over the shared resource:

Two ***condition variables***:

Initialization code

```
Size =
Rear =
Front =
Count =                    // number of full slots
```

Count will be used in order to check the conditions.
Full:
Empty:

Fig. BoundedBuffer Monitor

Execution code for the threads:

```
Consumer( ) {
   While(True) {
      BoundedBuffer.Fetch( );
      Consume( );
   }
}
```

```
Producer ( ) {
   While(True) {
         Produce( );
         BoundedBuffer.Deposit( );
   }
}
```

Note: the conditions variables are declared only within the monitors and their value are not directly visible to the programmer.

*Code for the **service methods***