Exam INF 223

23 November 2010

1h30 – No Documentation Allowed

1 Concurrence

1.1 Threads Posix (2 points)

We'd like to block a thread until a certain variable is set with a certain value. Is the following solution correct? Justify your answer. Provide a correct solution.

```
int x;
pthread_mutex_t lock;
void set_x(int y) {
  pthread_mutex_lock(&lock);
  x = y;
  pthread_mutex_unlock(&lock);
}
```

```
void wait_for_x_equal(int y){
  while (true) {
    pthread_mutex_lock(&lock);
    int r = x;
    pthread_mutex_unlock(&lock);
    if (r == y) break;
    sleep (t);
    }
}
```

1.2 Threads Java (2.5 points)

Considering two semaphores - S1 and S2, initialised as follows:

```
Semaphore S1 = new Semaphore (n);
Semaphore S2 = new Semaphore (0);
```

Is the following producer/consumer scheme correct, considering that the `put` and `get` methods belong to the same object? Justify your answer. Provide a correct solution and explain.

```
synchronized put(Object o){
  S1.acquire();
  last = (last+1)%size;
  buffer[last] = o;
  S2.release();
  }
```

```
synchronized Object get(){
   S2.acquire();
   Object o = buffer[first];
   first = (first+1)%size;
   S1.release();
   return o;
}
```

2 Communications

2.1 Sockets (1.5 point)

What problem can arise when exchanging data (e.g. of type `int`) between two remote processes that communicate via sockets (irrespectively of whether UDP or TCP is used)? Provide a solution.

2.2 Sockets (2 points)

How will a server that is waiting on `accept` be unblocked by a remote call? What should the server do once the connection is established?

3 Design Patterns

3.1 Questions (6 points)

1. Identify a classical design pattern within the design of a known middleware technology (e.g. RMI or CORBA).

   Note: alternatively (as a last resort only!), choose one of the design patterns presented in the course, in order to address questions 2, 3 and 4.

2. Briefly indicate the advantages of using this design pattern in the context of the targeted middleware.

3. Depict the static structure of the selected design pattern (e.g. via an UML class diagram).

4. Describe the runtime behaviour of the selected design pattern (e.g. via an UML sequence diagram).

4 Use Case

4.1 Thread version (2 points)

In the Thread-based version of the Use Case: explain the mechanism used for establishing the communication between the Threads; and indicate the synchronisation-related properties that this mechanism introduces.

4.2 Socket version (2 points)

In the Socket-based version of the Use Case: explain the way in which the completely-connected communication graph is being created in order to minimise the number of opened sockets.

4.3 CORBA Version (2 points)

In the CORBA-based version of the Use Case: explain the way in which the nodes manage to contact each-other, while detaining no initial information about one another. Indicate an alternative way of achieving the same result, based on a solution presented in the course notes.