OBL3-OS March 15, 2019

## Synchronization

### 1. The principle of process isolation in an operating system means that processes must not have access to the address spaces of other processes or the kernel. However, processes also need to communicate.

#### (a) Give an example of such communication.

Shared memory is one method of communication between processes.

#### (b) How does this communication work?

If Process 1 and Process 2 are executing simultaneously and they share resources or information, if process 2 need to use some of the data or information generated by process 1 it will check the record stored in shared memory to see the information generated by process 1.

#### (c) What problems can result from inter-process communication?

There can be problems of process 1 generating too much information and having to wait for process 2 to use the information before having the storage space to generate more.

### 2. What is a critical region? Can a process be interrupted while in a critical region? Explain.

The protected region where both concurrent processes share information is the critical region, this area can only be accessed by one process at the time.

### 3. Explain the difference between busy waiting (polling) versus blocking (wait/signal) in the context of a process trying to get access to a critical section.

Polling is when the process in question sends a request to get access to the region in question, wait/signal is when the process simply gives notice that it wants to use and waits for access to be released by the other process using it

### 4. What is a race condition? Give a real-world example.

A race condition is when a device or system attempts to preform two or more operations at the same time, but the processes need to be done in a specific order.

### 5. What is a spin-lock, and why and where is it used?

A lock which causes the thread that tries to access it to loop until the lock is available.  
often used in the operating system for things that only require very short access periods.

#### 6. List the issues involved with thread synchronization in multi-core architectures. Two lock algorithms are MCS and RCU (read-copy-update). Describe the problems they attempt to address. What hardware mechanism lies at the heart of each?

MCS locks can eliminate much of the cache-line bouncing experienced by simpler locks.  
RCU achieves scalability improvements by allowing reads to occur concurrently with updates.

## 2 Deadlocks

### What is the difference between resource starvation and a deadlock?

Starvation occurs when one or more threads in your program are blocked from gaining access to a resource.   
Deadlock occurs when two or more threads are waiting on a condition that cannot be satisfied.

### What are the four necessary conditions for a deadlock? Which of these are inherent properties of an operating system?

* Limited access to resources
* No preemption
* Multiple independent requests
* Circular chain of requests

### How does an operating system detect a deadlock state? What information does it have available to make this assessment?

It can be done simply by checking if a given process takes longer then it should and then flag it for recovery, this can lead to false positives.

## 3 Scheduling

### 1. Uniprocessor scheduling

#### (a) When is first-in-first-out (FIFO) scheduling optimal in terms of average response time? Why?

If we have a fixed number of tasks, and those tasks only need the processor, this will ensure that ever task gets done in order and to completion.

#### (b) Describe how Multilevel feedback queues (MFQ) combines first-in-first-out, shortest job first, and round robin scheduling in an attempt at a fair and efficient scheduler. What (if any) are its shortcomings?

Has a set of Round Robin queues, each with separate priorities, where the high priority queues have shot time slices and the low priority queues have long time slices.  
All tasks start in the highest queue, and if it exceeds the time slices it drops one level.

MFQ combines several of the methods of other queue types, which means it is more varied in its responses, but suffer from a “Jack-of-all-trades” where it doesn’t do any of the things as good as the pure method.

### 2. Multi-core scheduling

#### (a) Similar to thread synchronization, a uniprocessor scheduler running on a multi-core system can be very inefficient. Explain why (there are three main reasons). Use MFQ as an example.

Contention for scheduler spinlock  
Cache slowdown due to ready list data structure pinging from one CPU to another  
Limited cache reuse: thread’s data from last time it ran is often still in its old cache

#### (b) Explain the concept of work-stealing.

Idle processors can steal work from other processors