Concurrency Terminology

Synchronization: Ensuring temporal coordination (and hence correct cooperation) between processes. Ultimately relies on the atomicity (indivisibility) of some operations.

Example: P1= i:=i+1; P2= i:= i-1

Note: process $\sim$ thread for this discussion...

Cooperation

Are values computed within one process visible in another?

Are values computed within one process consulted in another?

Do different temporal orderings produce different results?

Does it matter? (Consistency criteria)

Value Sharing Mechanisms

Mechanisms for making values visible to another process.

Sharing address space - whole or parts.

Messaging.

Concurrency Terminology

If it does matter that different temporal orderings produce different results, force a particular ordering (serialization)...

Mutual Exclusion (naive): ensures that only one process does a particular activity at a time. One process doing an activity *excludes* all the others from doing it at that time.

Activity?

Concurrency Terminology

Critical Section (naive): is the piece of code that only one process is supposed to be executing at a time.

$\Longrightarrow$ All synchronization involves waiting.

Textbook examples vs real life

Mutual Exclusion

Ensure:

Exclusion : If one process is executing in its critical section, then no other process can be executing in its critical section

Progress : If no process is executing in its critical section, then the selection of a process that will be allowed to enter its critical section cannot be postponed indefinitely

Bounded waiting : There must exist an upper bound on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted

Concurrency Mechanisms

Low-level mutual exclusion mechanisms (software, hardware)

Encapsulated mutual exclusion mechanisms (locks, semaphores, conditional critical regions, monitors)

Languages and mathematical formalisms for concurrent programming (Brinch Hansen, Hoare, Milner)

Common concurrency situations (“patterns”?) and recipes using different mechanisms (dining philosophers, readers-writers)

Software Mutual Exclusion

Provide mutually exclusive access of N processes to a single shared variable...

Busy-waiting loops at the core of algorithms (Dekker, Peterson, Lamport)

Mutual Exclusion by Disabling Interrupts

Long critical sections?

Security?

Multiprocessors

Mutual Exclusion with Special Instructions

Why are ordinary instructions not sufficient?

Test-and-set instruction

```
repeat forever
   while test-and-set (lock) do no-op;
   ...critical section...
   lock = false;
   ...remainder...
```

Summary

- Communication among processes/threads is typically done through shared variables
- Critical sections identify pieces of code that cannot be executed in parallel by multiple processes, typically code that accesses and/or modifies the values of shared variables
- Synchronization primitives are required to ensure that only one process executes in a critical section at a time