Glossary

Actor

An actor is the execution of a sequential program, i.e., a linear sequence of statements totally ordered in time. Other terms commonly used to identify an actor are: process, task, and thread.

Atomic

Two actions A1 and A2 are atomic to each other if they cannot interleave, i.e., if one of the following two conditions holds:

1. End (A1) $<$ Start (A2)
2. End (A2) $<$ Start (A1)

Busy Waiting

Busy waiting is a situation in which an actor continuously checks a condition to enter its critical section, i.e., it steadily consumes processing resources while it awaits the condition to become true.

Critical Section

A critical section of an actor is a sequence of statements that contains access to shared data.

Deadlock

A deadlock is a situation where a number of actors permanently block each other, i.e., they wish to enter their critical section, but no actor can succeed.

Livelock

A situation in which a set of actors remains active (they are not blocked, as holds for a deadlock) but there are execution sequences in which no actor ever enters its critical section.

Lock Variable

A lock variable L is a synonym of a binary semaphore S. L may be in two states: acquired and released. These states correspond to the values 0 and 1 of S. There are two operations defined on L: Acquire (L) and Release (L). These operations correspond to P (S) and V (S), respectively.

Monitor

A monitor is an abstract data object that encapsulates data structures and their operations into a single module. The interface of the monitor consists of a set of procedures that operate on the data hidden within the monitor. The procedures of a monitor are executed under mutual exclusion.

If an actor executes a procedure of a monitor, the actor is said to be inside the monitor. Typically, when an actor P calls a monitor procedure, it is first checked if any other actor is currently inside the monitor. If so, then P will be suspended until the other actor has left the monitor. If no other actor is inside the monitor, P may enter.

A common way to implement the mutual exclusion on monitors is using a lock variable. In that case, entering the monitor corresponds to acquiring the lock, whereas leaving the monitor corresponds to releasing the lock.

It is often necessary to execute a monitor's procedure only if some condition holds that is evaluated inside the monitor (since it may depend on the data hidden within the monitor). In such a case, condition variables are used. The value of a condition variable is a queue of actors delayed on the corresponding condition. There are the following two operations defined on a condition variable cv:

wait (cv) : This causes the actor executing wait (cv) to delay and to be placed at the end of cv's queue. After executing wait (cv) the actor immediately leaves the monitor, so that other actors can enter.

signal (cv) : This causes the actor at the head of cv's queue to be awakened. If the queue of cv is empty, signal (cv) has no effect.

The difference between a condition variable and a semaphore is that a condition variable is just a signalling device and has no counter associated with it; i.e., it does not accumulate signals for later use.

Mutual Exclusion

Mutual exclusion is a synchronization condition that requires that at most one actor of a given set of actors may be in its critical section at an instant of time.

Process

See actor.

Race Condition

A race condition is a situation where two or more actors read and write some shared data and the outcome depends on the relative timing of the actors.

Real Time System

A real time system is a system in which the time that is necessary to produce the results of an operation is significant. We distinguish between hard real time systems and soft real time systems. A hard real time system is a real time system that only works correctly if the results of an operation are produced within a specified deadline. A soft real time system is a real time system that will still function correctly if a deadline for producing the results of an operation is occasionally missed.

Semaphore

A semaphore is a nonnegative integer-valued variable. There are the following two operations defined on a semaphore S:

P (S) : delay until $S > 0$;
$S := S - 1$;

V (S) : $S := S + 1$;

When an actor is delayed on a semaphore S, it will only be awakened by another actor executing a V (S) operation on S. If more than one actor is delayed on S, only one (the choice is implementation defined) can be reactivated by a V (S) operation at an instant of time. P (S) and V (S) are atomic operations.

A semaphore that can take any nonnegative value is called a general semaphore. A semaphore that takes only the values 0 and 1 is called a binary semaphore.

Sometimes Signal (S) and Wait (S) are used in place of P (S) and V (S), respectively.

Shared Variable

A shared variable is variable that is concurrently accessed by two or more actors.

Starvation

Starvation is a situation in which an actor wants to enter its critical section but never succeeds.

Task

See actor.

Thread

See actor.