References

Ben-Ari, *Principles of Concurrent and Distributed Programming*, second edition, Addison-Wesley, 2006.
Buhr, et. al., "Monitor Classification", *Computing Surveys*, March 1995.

General monitor

All procedures are mutually exclusive. Each monitor has

- One entry queue
- One queue for each condition variable
- One waiting queue
- One signaler queue

General actions

waitC(*cond*)
Blocked on condition queue for *cond*

signalC(*cond*)
Signaler moved to signaler queue
Signaled moved from condition queue to wait queue

Monitor is unlocked
Monitor chooses from one of the queues which process gets to enter

Types of monitors

The type of monitor is determined by how the monitor chooses which process gets to enter. Each queue has a specific precedence:

- E – entry precedence
- W – waiting precedence
- S – signaler precedence

The type of monitor is determined by the relative precedence the monitor gives to the queues.

Buhr, Signal and Continue, Mesa Semantics

$E < W < S$

The monitor design for C++. However, the condition queue is not guaranteed to be FIFO.

Because S has the highest precedence, the process that executes the signal statement is always picked. The signaler queue is not necessary, because the signaler is not queued. Because $E < W$, separate queues are needed for waiting and entry.

There is no guarantee to the waiting process that the boolean expression it waited on is still true, because another process may have changed the value of the expression between the signal execution and the resumption of the waiting process. This phenomenon is called a *spurious wakeup*. Requires a loop on the boolean expression instead of an `if` on the expression.

Known as Priority Non-Blocking (PNB) in Buhr, because the signaler is not blocked.

Recommended by Buhr because performance is better than Signal and Urgent Wait, and the signaler continuing is more intuitive for programmers.

Ben-Ari, Signal and Urgent Wait, Hoare Semantics
$E < S < W$

Immediate resumption requirement (IRR).

The monitor design for Ben-Ari pseudocode and C--.

Because W has higher precedence than S, the signaler process is blocked if any processes are waiting on *cond*. The waiting queue is not necessary, because the waiter is not queued. Because $E < S$, separate queues are needed for signaler and entry.

The signaler can guarantee to the waiting process that the boolean expression it waited on is still true, because the waiting process resumes immediately after being signaled with no interleaving after execution of signalC(*cond*). You can program the waitC(*cond*) with an `if` on the boolean expression.

When possible, for ease of analyzing concurrency, the execution of signalC(*cond*) should be the last statement of a monitor procedure. That placement minimizes interleaving between the waiting process and the signaling process.

Known as Priority Blocking (PB) in Buhr, because the signaler is blocked.

Java, Wait and Notify
$E = W < S$

Java has no condition variables. Hence, there is a single queue for processes that have executed waitC().

Because S has the highest priority, the process that executes the signalC() statement (called `notify()` in Java) is always picked and the signaler queue is not necessary. However, `notify()` moves a process from the waiting queue to the entry queue where it must compete with processes that have not begun executing their monitor operations ($E = W$). Method `notifyAll()` moves all processes from the waiting queue to the entry queue.

Common coding style is for the signaler to execute `notifyAll()` to guarantee that the correct waiting process is unblocked, and for the waiting processes to loop on their boolean expressions.

Known as No Priority Non-Blocking (NPNB) in Buhr, because the processes in the waiting queue do not have priority over the processes in the entry queue, and the signaler is not blocked.

Criticized by Buhr, "In all cases, the no-priority property complicates the proof rules, makes performance worse, and makes programming more difficult. …Therefore, we have rejected all no-priority monitors from further consideration."