Rec. 14. Concurrency (CON)

Page: CON01-C. Acquire and release synchronization primitives in the same module, at the same level of abstraction
Page: CON02-C. Do not use volatile as a synchronization primitive
Page: CON03-C. Ensure visibility when accessing shared variables
Page: CON04-C. Join or detach threads even if their exit status is unimportant
Page: CON05-C. Do not perform operations that can block while holding a lock
Page: CON06-C. Ensure that every mutex outlives the data it protects
Page: CON07-C. Ensure that compound operations on shared variables are atomic
Page: CON08-C. Do not assume that a group of calls to independently atomic methods is atomic
Page: CON09-C. Avoid the ABA problem when using lock-free algorithms

Risk Assessment Summary

<table>
<thead>
<tr>
<th>Rule</th>
<th>Severity</th>
<th>Likelihood</th>
<th>Remediation Cost</th>
<th>Priority</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>CON01-C</td>
<td>Low</td>
<td>Probable</td>
<td>Medium</td>
<td>P4</td>
<td>L3</td>
</tr>
<tr>
<td>CON02-C</td>
<td>Medium</td>
<td>Probable</td>
<td>Medium</td>
<td>P8</td>
<td>L2</td>
</tr>
<tr>
<td>CON04-C</td>
<td>Low</td>
<td>Unlikely</td>
<td>High</td>
<td>P1</td>
<td>L3</td>
</tr>
<tr>
<td>CON05-C</td>
<td>Low</td>
<td>Probable</td>
<td>High</td>
<td>P2</td>
<td>L3</td>
</tr>
<tr>
<td>CON06-C</td>
<td>Medium</td>
<td>Probable</td>
<td>Medium</td>
<td>P8</td>
<td>L2</td>
</tr>
<tr>
<td>CON07-C</td>
<td>Medium</td>
<td>Probable</td>
<td>Medium</td>
<td>P8</td>
<td>L2</td>
</tr>
<tr>
<td>CON08-C</td>
<td>Low</td>
<td>Probable</td>
<td>Medium</td>
<td>P4</td>
<td>L3</td>
</tr>
<tr>
<td>CON09-C</td>
<td>Medium</td>
<td>Unlikely</td>
<td>High</td>
<td>P2</td>
<td>L3</td>
</tr>
</tbody>
</table>

Related Rules and Recommendations

- CON04-C. Join or detach threads even if their exit status is unimportant
- CON30-C. Clean up thread-specific storage
- CON31-C. Do not destroy a mutex while it is locked
- CON32-C. Prevent data races when accessing bit-fields from multiple threads
CON33-C. Avoid race conditions when using library functions

CON34-C. Declare objects shared between threads with appropriate storage durations

CON35-C. Avoid deadlock by locking in a predefined order

CON37-C. Do not call signal() in a multithreaded program

CON43-C. Do not allow data races in multithreaded code

CON50-CPP. Do not destroy a mutex while it is locked

POS04-C. Avoid using PTHREAD_MUTE X_NORMAL type mutex locks

POS44-C. Do not use signals to terminate threads

POS47-C. Do not use threads that can be canceled asynchronously

POS48-C. Do not unlock or destroy another POSIX thread’s mutex

POS49-C. When data must be accessed by multiple threads, provide a mutex and guarantee
no adjacent data shared between POSIX threads with appropriate storage durations

POS51-C. Avoid deadlock with POSIX threads by locking in predefined order

POS53-C. Do not use more than one mutex for concurrent waiting operations on a condition variable not forcibly terminate execution