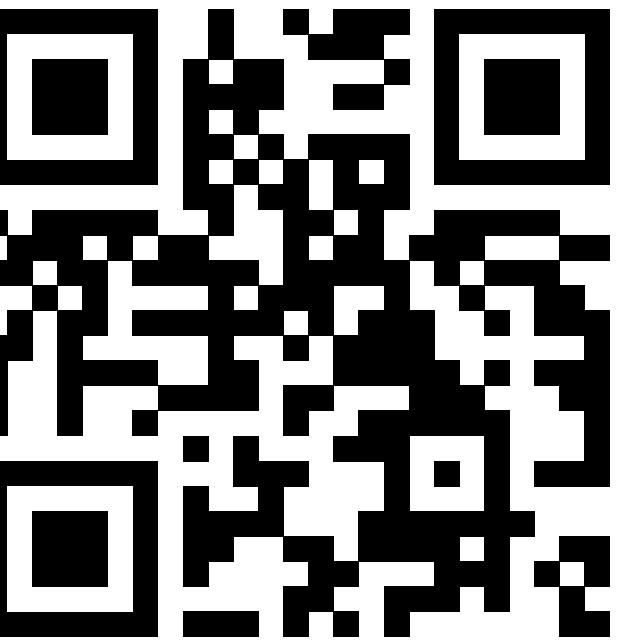


Thanks

- Timur Doumler
 - “Real-time programming with the standard library”
- Fabian Renn-Giles
 - “Real-time confessions”
- David Trevelyan, Ali Barker & Chris Apple
 - “RADSan: A real-time safety sanitizer”
- Doug Wyatt
 - dtrace script, noalloc, nolock



Cppcon 2021 | October 26-29

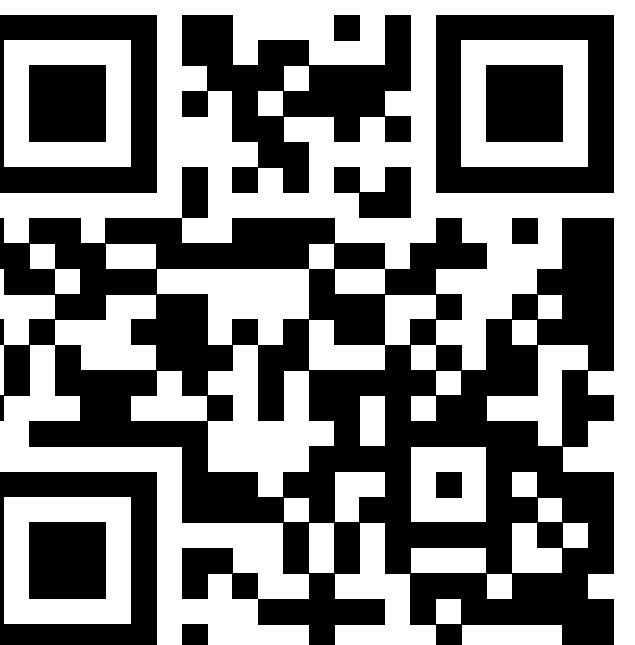


Timur Doumler

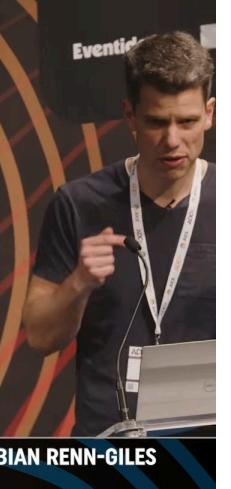
Real-Time programming with the C++ standard library

Copyright (c) Timur Doumler | [@timur_audio](#) | <https://timur.audio>

- don't call anything that might block (*non-deterministic execution time + priority inversion!*)
 - don't try to acquire a mutex
 - don't allocate / deallocate memory
 - don't do any I/O
 - don't interact with the thread scheduler
 - don't do any other system calls
 - don't call any 3rdparty code if you don't know what it's doing
 - don't use algorithms with $> O(1)$ complexity
 - don't use algorithms with *amortised* $O(1)$ complexity



REAL-TIME CONFESSIONS:
THE MOST COMMON 'SINS' IN REAL-TIME CODE ADC23



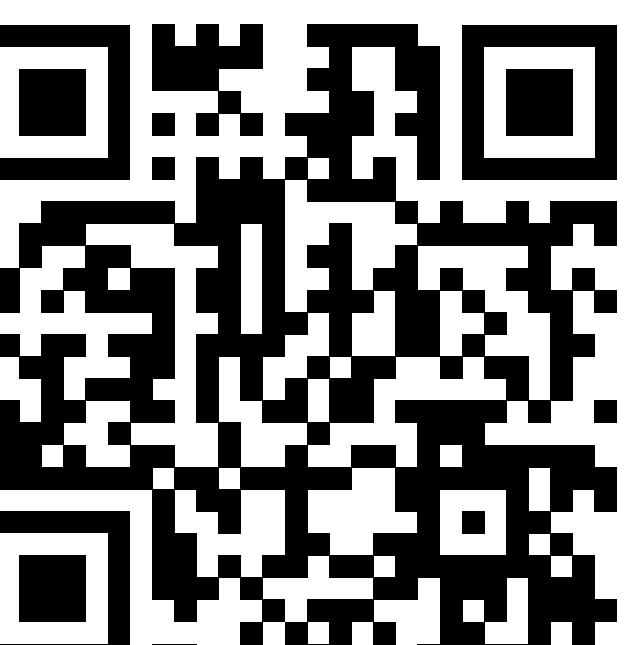
FABIAN RENN-GILES

Sin #4a
You may never take a lock in real-time code

Normal lock: `std::mutex::lock()`

```
std::mutex::lock is wrapper around pthread_mutex_lock (linux source code) - edited for brevity
white (1)
/* Try to acquire the lock through a CAS from 0 (not acquired) to our TID */
oldval = atomic_compare_and_exchange_volatile(&mutex->__data___.lock, 0, __TID);
if (!__atomic_load_nlikely(&oldval == 0))
    break;
...
/* Block using the futex and reload current lock value. */
futex_wait((unsigned int *) &mutex->__data___.lock, oldval,
            PTHREAD_ROBUST_MUTEX_PSHARED(mutex));
oldval = mutex->__data___.lock;
return;
```

→ Lock is real-time safe as long as it's never contended



RADSan:
A REALTIME-SAFETY SANITIZER ADC23



ALI BARKER

Memory Allocation	Threads & Sleep	Sockets
<code>malloc, calloc, realloc, reallocf, valloc, aligned_alloc free posix_memalign</code>	<code>pthread_create, pthread_join pthread_mutex_lock, pthread_mutex_unlock pthread_cond_signal, pthread_cond_broadcast etc. pthread_rwlock_rdlock, pthread_rwlock_unlock etc. osSemaphoreCreate, osUnfairLockCreate sleep, usleep, nanosleep</code>	<code>socket send, sendto, sendmsg recv, recvfrom, recvmsg shutdown</code>
Filesystem, Streams		
<code>open, openat, creat, close, fopen, fopenat, fclose, fread, fwrite puts, fputs</code>		

Catching Real-Time Safety Violations

David Rowland

 X @drowaudio

Questions?

Slides/video:

drowaudio.github.io/presentations

