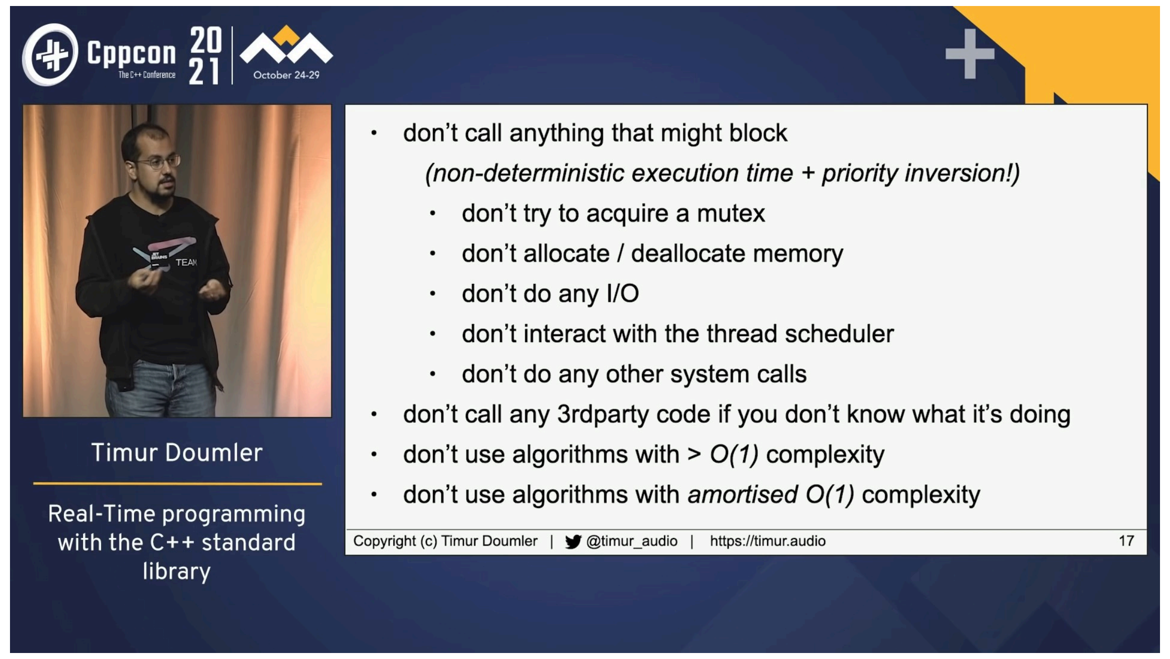
103

	GUI (System Trace)	cli (dtrace)	code	interpose	RTSan
Easy to use?	✅/⚠️	⚠️	✅	⚠️	✅
Clear?	❌	✅	✅	✅	✅
🧵Filterable?	⚠️	⚠️	✅	✅	✅
CI?	❌	⚠️	✅	✅	✅
Portable?	❌	❌	✅	⚠️	⚠️
System calls?	✅	✅	❌	⚠️	⚠️
Malloc/free?	⚠️	✅	❌	✅	✅
Lock/unlock?	✅	✅	⚠️	✅	✅
3rd party code?	✅	✅	❌	✅	✅
Notes	Different tools for different tasks	Requires disabling SIP		No raw/inline context switches/syscalls	No raw/inline context switches/syscalls

	GUI (System Trace)	**cli (dtrace)**	**code**	**interpose**	**RTSan**
Easy to use?	✅/⚠️	⚠️	✅	⚠️	✅
Clear?	❌	✅	✅	✅	✅
🧵Filterable?	⚠️	⚠️	✅	✅	✅
CI?	❌	⚠️	✅	✅	✅
Portable?	❌	❌	✅	⚠️	⚠️
System calls?	✅	✅	❌	⚠️	⚠️
Malloc/free?	⚠️	✅	❌	✅	✅
Lock/unlock?	✅	✅	⚠️	✅	✅
3rd party code?	✅	✅	❌	✅	✅
Notes	Different tools for different tasks	Requires disabling SIP		No raw/inline context switches/syscalls	No raw/inline context switches/syscalls

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`

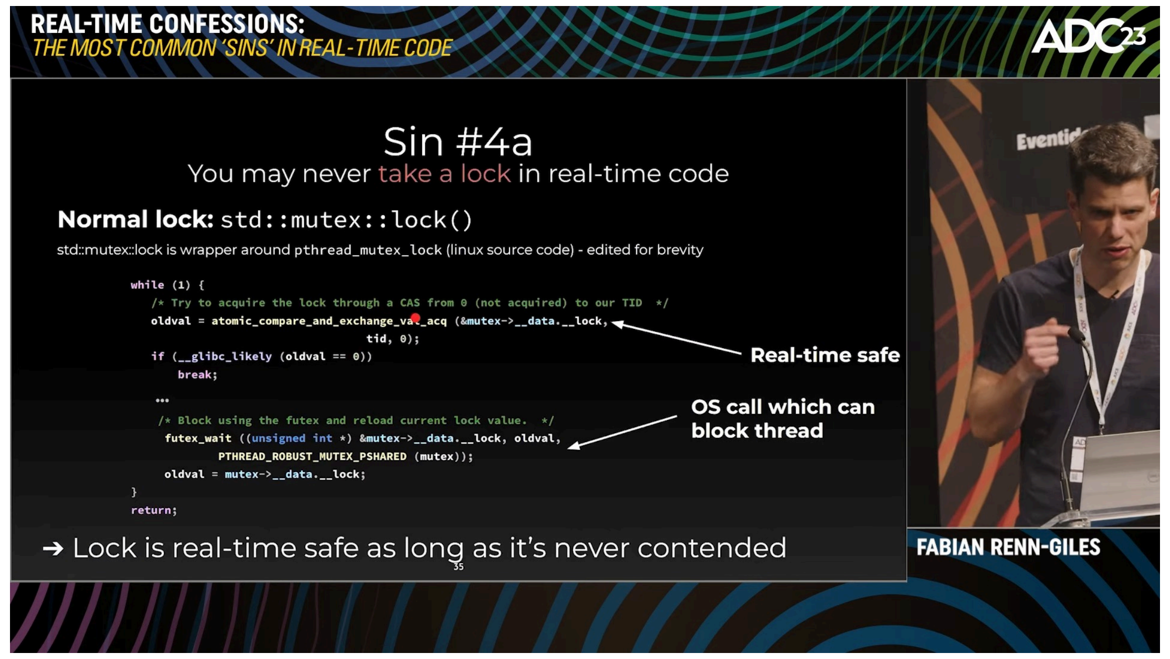


Timur Doumler

Real-Time programming with the C++ standard library

- 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

Copyright (c) Timur Doumler | @timur_audio | https://timur.audio 17



Sin #4a

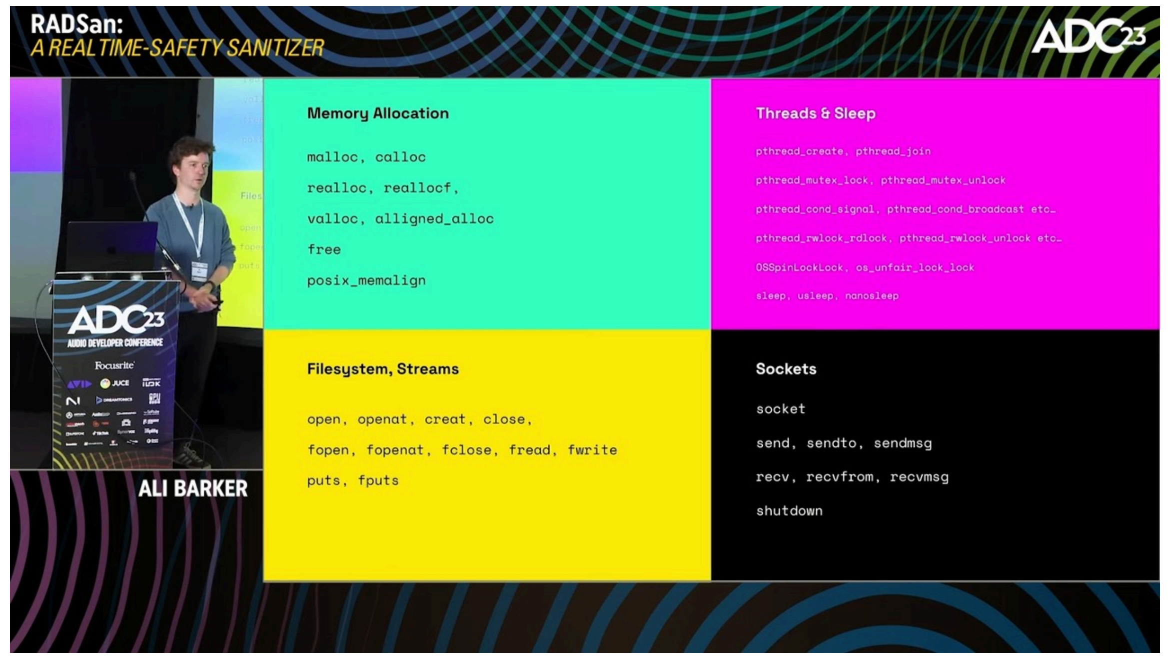
You may never take a lock in real-time code

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

std::mutex::lock is wrapper around `pthread_mutex_lock` (linux source code) - edited for brevity

```
while (1) {
  /* Try to acquire the lock through a CAS from 0 (not acquired) to our TID  */
  oldval = atomic_compare_and_exchange_val_acq (&mutex->__data.__lock,
                                                tid, 0);
  if (__glibc_likely (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



Memory Allocation

malloc, calloc
realloc, reallocf,
valloc, alligned_alloc
free
posix_memalign

Threads & Sleep

pthread_create, pthread_join
pthread_mutex_lock, pthread_mutex_unlock
pthread_cond_signal, pthread_cond_broadcast etc…
pthread_rwlock_rdlock, pthread_rwlock_unlock etc…
OSSpinLockLock, os_unfair_lock_lock
sleep, usleep, nanosleep

Filesystem, Streams

open, openat, creat, close,
fopen, fopenat, fclose, fread, fwrite
puts, fputs

Sockets

socket
send, sendto, sendmsg
recv, recvfrom, recvmsg
shutdown