undo

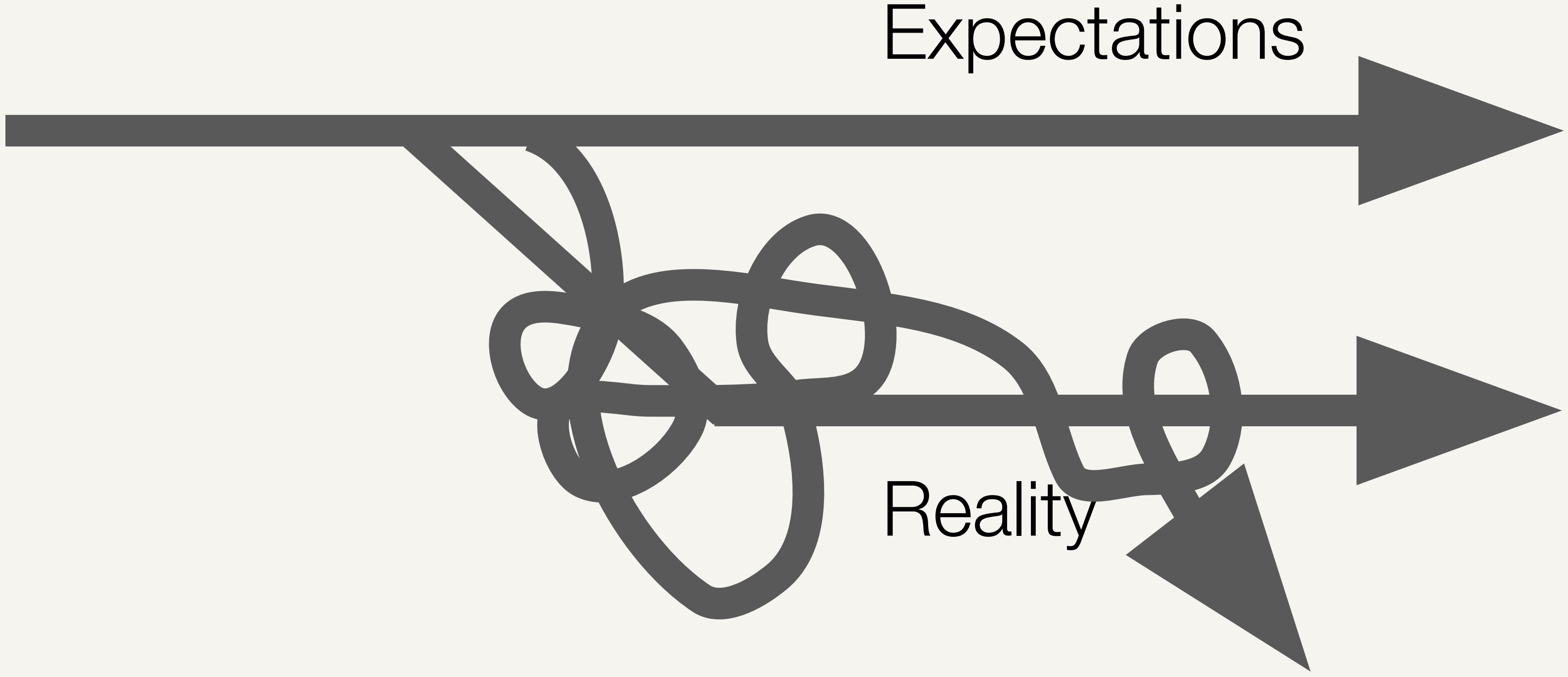
HOW TO DEBUG LINUX MULTI-THREADED CODE

GREG LAW

undo

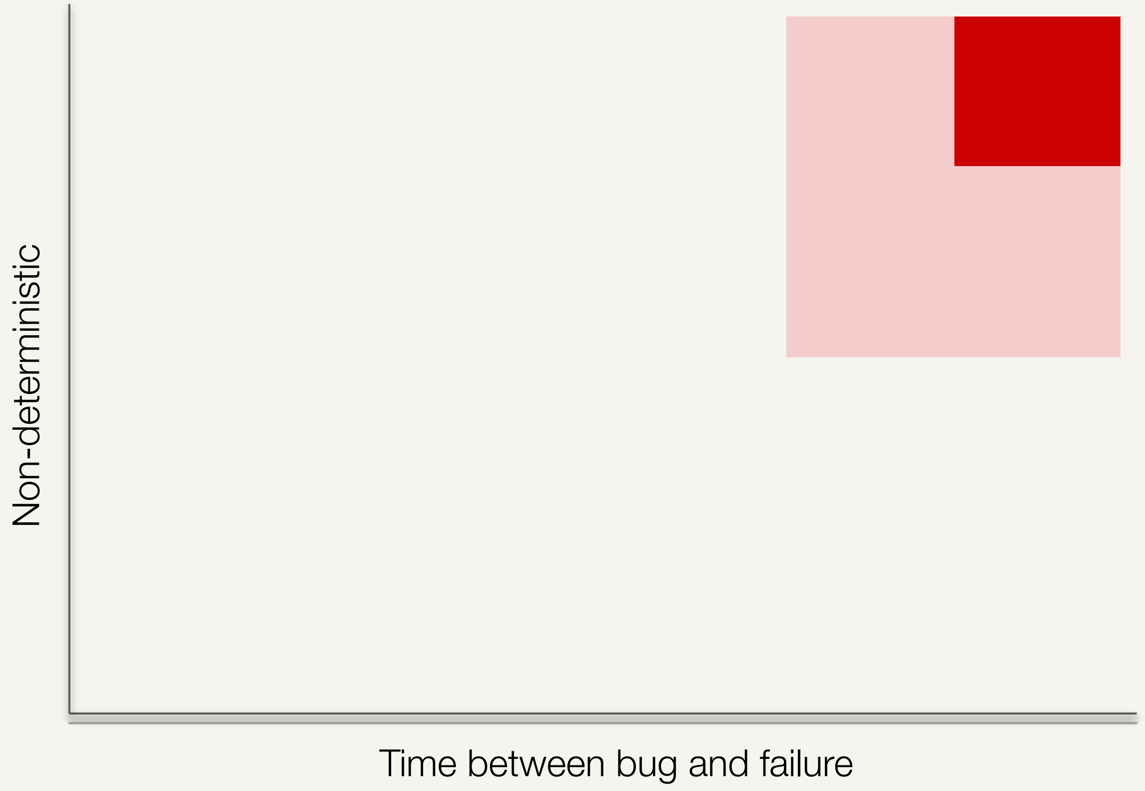
Free UDB ($7,900)

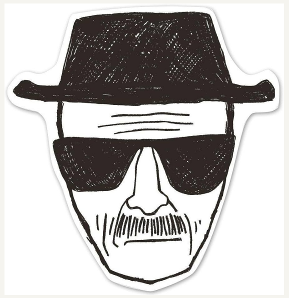
bitly


Expectations
Reality

undo

What makes bugs hard to find?





‘Heisenbugs’

WHAT IS A RACE CONDITION?

Concurrent threads of execution acting on a shared resource such that the ordering of operations affects the outputs of the program.

WHAT IS A RACE CONDITION?

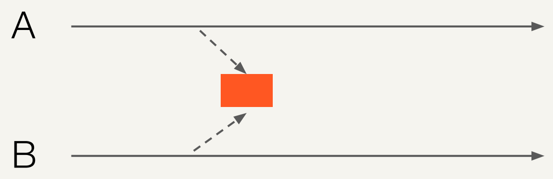
Concurrent threads of execution

WHAT IS A RACE CONDITION?

Concurrent threads of execution <u>acting on a shared resource</u>

WHAT IS A RACE CONDITION?

Concurrent threads of execution acting on a shared resource <u>such that the ordering of operations affects the outputs of the program</u>.



KNOW HOW TO USE THE TOOLS

Thread Sanitizer (tsan)

Valgrind / Helgrind / DRD

GDB

Lightweight logging

Time-travel: thread fuzzing / chaos mode

AN EXAMPLE RACE

```
#include <thread>

int
main(void)
{
    int ret = 42;

    {
        std::jthread t0([&ret]() {ret = 1;});
        std::jthread t1([&ret]() {ret = 0;});
    }

    return ret;
}
```

ThreadSanitizer

Use ThreadSanitizer to catch the race in simple_race.cpp

Compile and run:

```
g++ -g -fsanitize=thread simple_race.cpp -lpthread
./a.out
```

```
sysctl vm.mmap_rnd_bits=30
```

HELGRIND

```
valgrind --tool=helgrind ./a.out
```

DRD

Use DRD to catch the race in simple_race.cpp

```
valgrind --tool=drd ./a.out
```

MUTEXES ARE NOT ALWAYS THE ANSWER

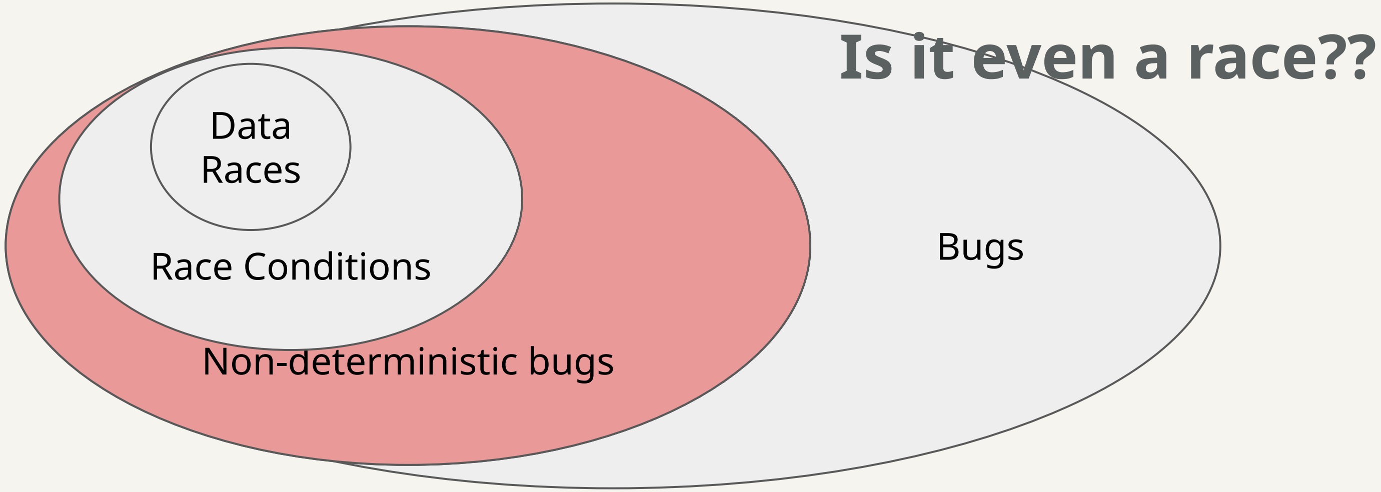
```diff
 #include <thread>
+#include <mutex>

 int main(void)
 {
     int ret = 42;
+    std::mutex m;

     {
-        std::jthread t0([&ret]() {ret = 1;});
-        std::jthread t1([&ret]() {ret = 0;});
+        std::jthread t0([&ret, &m]() {m.lock(); ret = 1; m.unlock();});
+        std::jthread t1([&ret, &m]() {m.lock(); ret = 0; m.unlock();});
     }
```

"DATA RACES" ARE JUST ONE KIND OF RACE

- Race with another process or the OS more common.
- Race with the filesystem, signals, process exit, etc.
- Time-of-check to time-of-use (TOCTOU, TOCTTOU or TOC/TOU)



GDB: Much maligned but actually good!

OH: GDB isn’t good at debugging threads.

OH: When I compile with debuginfo the races go away.

SLEEPING BY SYNCHRONIZATION

```
 #include <thread>
+#include <unistd.h>

 int main(void)
 {
     int ret = 42;
     {
              std::jthread t0([&ret]() {ret = 1;});
-             std::jthread t1([&ret]() {ret = 0;});
+             std::jthread t1([&ret]() {usleep(10000); ret = 0;};
     }
```

L3

Sometimes, logging is all we got

We can do better than printf though

THREAD FUZZING

undo

VISIT UNDO.IO TO LEARN MORE

undo

Free UDB ($7,900)

bitly