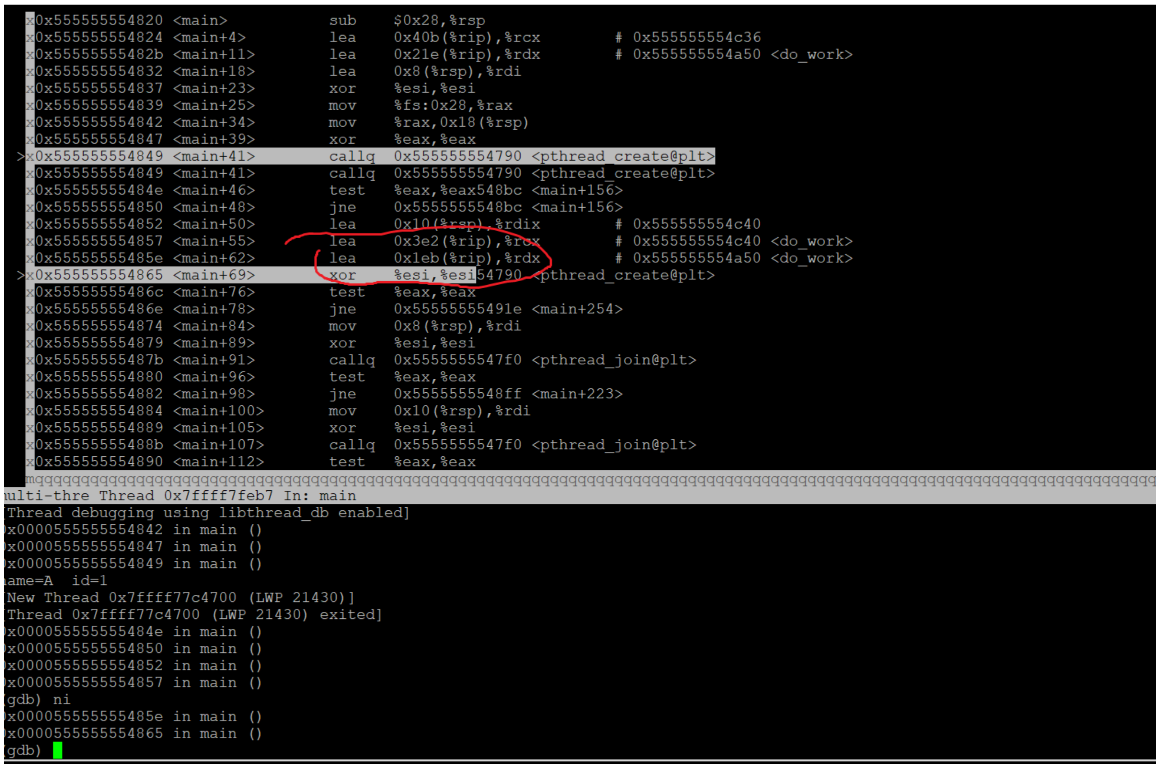
Comp3230 – Tutorial 3 Exercise 2

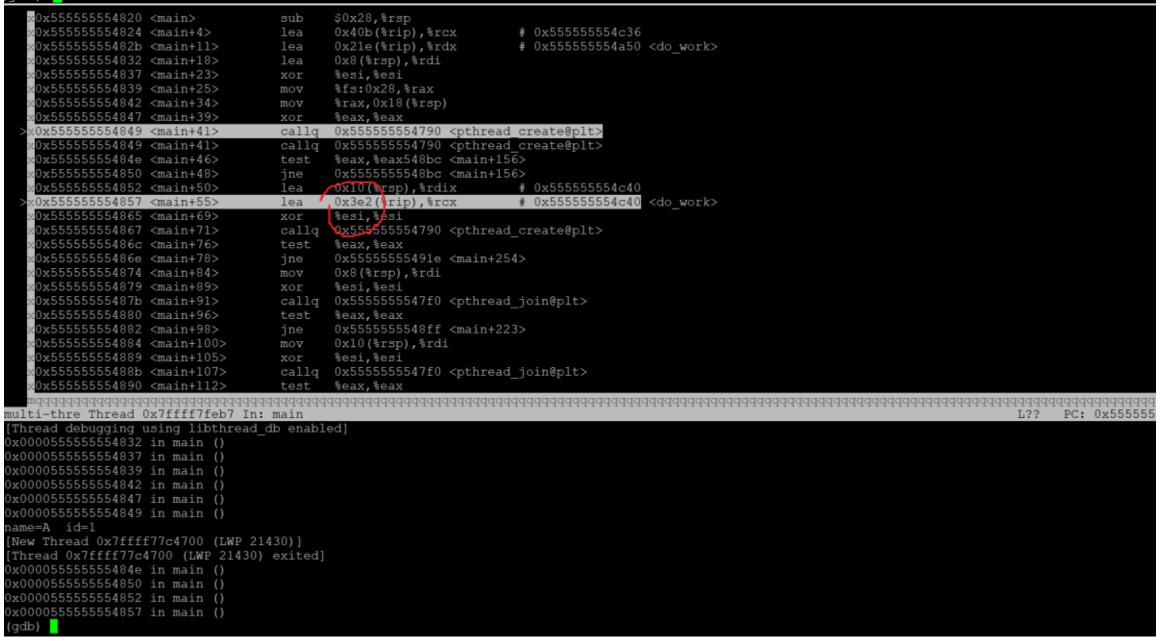
Name: Lee Aaron **UID:30335574103**

1. Sometime the second thread also passed the get_instance function, even if the first thread has already entered. Therefore, the output will some time print out both A and B. A race condition occurs.

Expected code



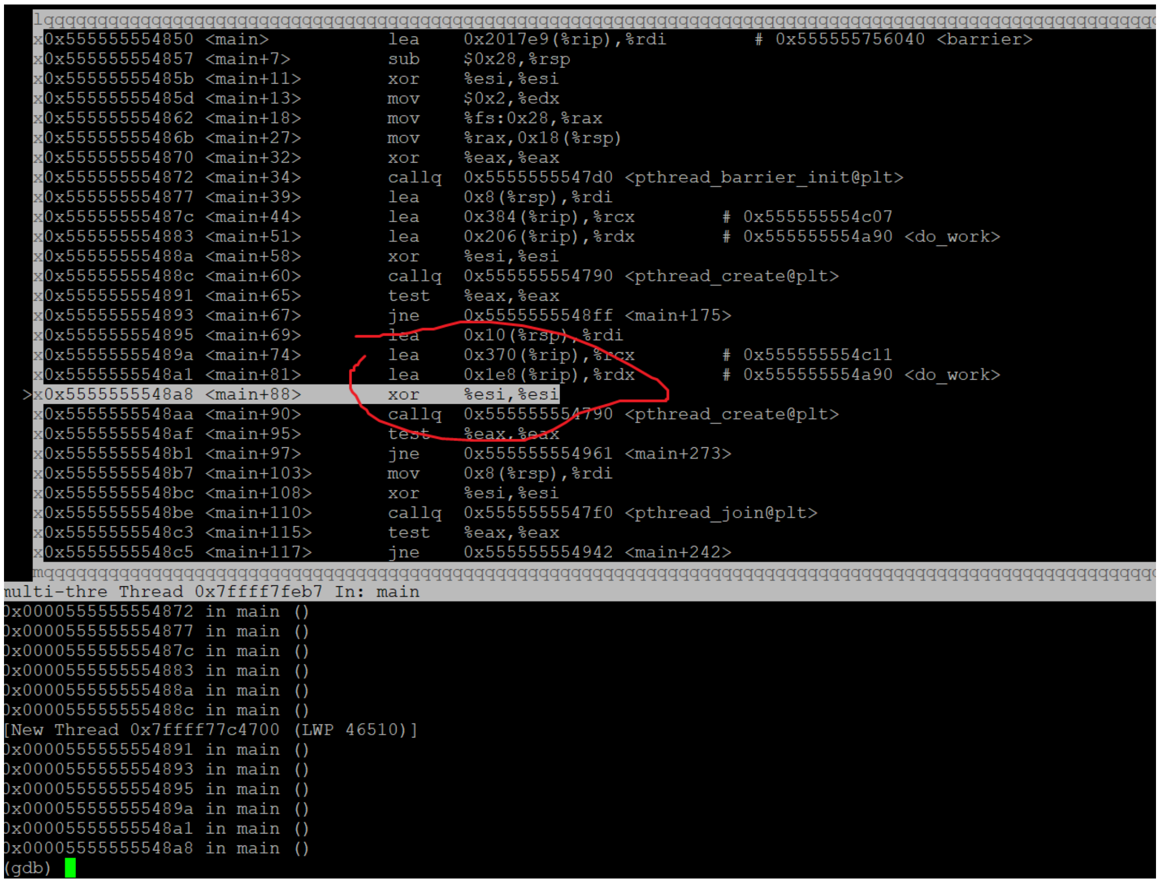
```
  x0x555555554820 <main>           sub    $0x28,%rsp
  x0x555555554824 <main+4>         lea    0x40b(%rip),%rcx        # 0x555555554c36
  x0x55555555482b <main+11>        lea    0x21e(%rip),%rdx        # 0x555555554a50 <do_work>
  x0x555555554832 <main+18>        lea    0x8(%rsp),%rdi
  x0x555555554837 <main+23>        xor    %esi,%esi
  x0x555555554839 <main+25>        mov    %fs:0x28,%rax
  x0x555555554842 <main+34>        mov    %rax,0x18(%rsp)
  x0x555555554847 <main+39>        xor    %eax,%eax
 >x0x555555554849 <main+41>        callq  0x555555554790 <pthread_create@plt>
  x0x555555554849 <main+41>        callq  0x555555554790 <pthread_create@plt>
  x0x55555555484e <main+46>        test   %eax,%eax548bc <main+156>
  x0x555555554850 <main+48>        jne    0x5555555548bc <main+156>
  x0x555555554852 <main+50>        lea    0x10(%rsp),%rdix        # 0x555555554c40
  x0x555555554857 <main+55>        lea    0x3e2(%rip),%rcx        # 0x555555554c40 <do_work>
  x0x55555555485e <main+62>        lea    0x1eb(%rip),%rdx        # 0x555555554a50 <do_work>
 >x0x555555554865 <main+69>        xor    %esi,%esi54790 <pthread_create@plt>
  x0x55555555486c <main+76>        test   %eax,%eax
  x0x55555555486e <main+78>        jne    0x55555555491e <main+254>
  x0x555555554874 <main+84>        mov    0x8(%rsp),%rdi
  x0x555555554879 <main+89>        xor    %esi,%esi
  x0x55555555487b <main+91>        callq  0x5555555547f0 <pthread_join@plt>
  x0x555555554880 <main+96>        test   %eax,%eax
  x0x555555554882 <main+98>        jne    0x5555555548ff <main+223>
  x0x555555554884 <main+100>       mov    0x10(%rsp),%rdi
  x0x555555554889 <main+105>       xor    %esi,%esi
  x0x55555555488b <main+107>       callq  0x5555555547f0 <pthread_join@plt>
  x0x555555554890 <main+112>       test   %eax,%eax
  mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
ulti-thre Thread 0x7ffff7feb7 In: main
[Thread debugging using libthread_db enabled]
)x0000555555554842 in main ()
)x0000555555554847 in main ()
)x0000555555554849 in main ()
ame=A  id=1
[New Thread 0x7ffff77c4700 (LWP 21430)]
[Thread 0x7ffff77c4700 (LWP 21430) exited]
)x000055555555484e in main ()
)x0000555555554850 in main ()
)x0000555555554852 in main ()
)x0000555555554857 in main ()
(gdb) ni
)x000055555555485e in main ()
)x0000555555554865 in main ()
(gdb) 
```



```
  x0x555555554820 <main>           sub    $0x28,%rsp
  x0x555555554824 <main+4>         lea    0x40b(%rip),%rcx        # 0x555555554c36
  x0x55555555482b <main+11>        lea    0x21e(%rip),%rdx        # 0x555555554a50 <do_work>
  x0x555555554832 <main+18>        lea    0x8(%rsp),%rdi
  x0x555555554837 <main+23>        xor    %esi,%esi
  x0x555555554839 <main+25>        mov    %fs:0x28,%rax
  x0x555555554842 <main+34>        mov    %rax,0x18(%rsp)
  x0x555555554847 <main+39>        xor    %eax,%eax
 >x0x555555554849 <main+41>        callq  0x555555554790 <pthread_create@plt>
  x0x555555554849 <main+41>        callq  0x555555554790 <pthread_create@plt>
  x0x55555555484e <main+46>        test   %eax,%eax548bc <main+156>
  x0x555555554850 <main+48>        jne    0x5555555548bc <main+156>
  x0x555555554852 <main+50>        lea    0x10(%rsp),%rdix        # 0x555555554c40
 >x0x555555554857 <main+55>        lea    0x3e2(%rip),%rcx        # 0x555555554c40 <do_work>
  x0x555555554865 <main+69>        xor    %esi,%esi
  x0x555555554867 <main+71>        callq  0x555555554790 <pthread_create@plt>
  x0x55555555486c <main+76>        test   %eax,%eax
  x0x55555555486e <main+78>        jne    0x55555555491e <main+254>
  x0x555555554874 <main+84>        mov    0x8(%rsp),%rdi
  x0x555555554879 <main+89>        xor    %esi,%esi
  x0x55555555487b <main+91>        callq  0x5555555547f0 <pthread_join@plt>
  x0x555555554880 <main+96>        test   %eax,%eax
  x0x555555554882 <main+98>        jne    0x5555555548ff <main+223>
  x0x555555554884 <main+100>       mov    0x10(%rsp),%rdi
  x0x555555554889 <main+105>       xor    %esi,%esi
  x0x55555555488b <main+107>       callq  0x5555555547f0 <pthread_join@plt>
  x0x555555554890 <main+112>       test   %eax,%eax
  mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
multi-thre Thread 0x7ffff7feb7 In: main                                                    L??   PC: 0x555555
[Thread debugging using libthread_db enabled]
0x0000555555554832 in main ()
0x0000555555554837 in main ()
0x0000555555554839 in main ()
0x0000555555554842 in main ()
0x0000555555554847 in main ()
0x0000555555554849 in main ()
name=A  id=1
[New Thread 0x7ffff77c4700 (LWP 21430)]
[Thread 0x7ffff77c4700 (LWP 21430) exited]
0x000055555555484e in main ()
0x0000555555554850 in main ()
0x0000555555554852 in main ()
0x0000555555554857 in main ()
(gdb) 
```

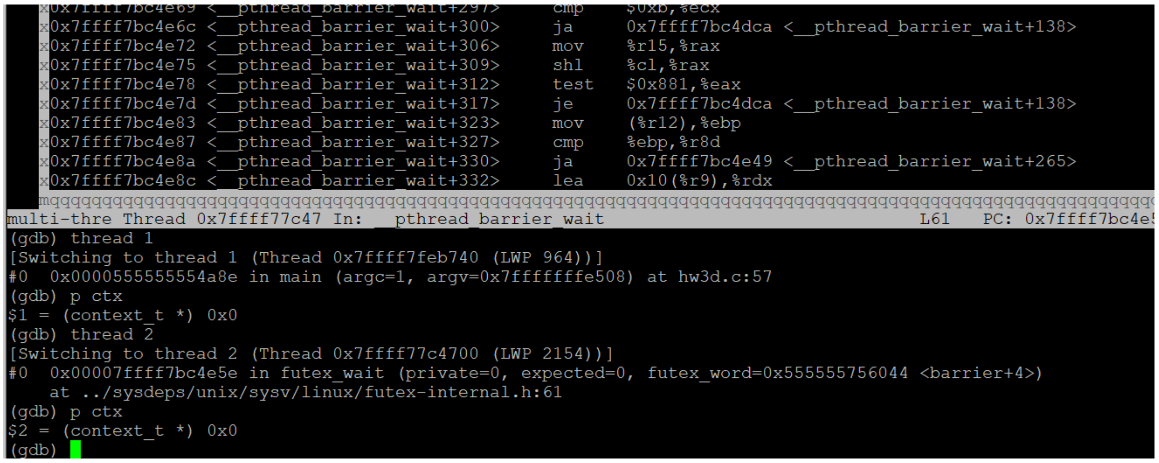
When there is the second threading in coming the %esi will not long be xor %esi %esi. It has changed the value, which meant the ctx is no longer a null pointer.

Bugged code



```
lqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
x0x555555554850 <main>            lea    0x2017e9(%rip),%rdi        # 0x555555756040 <barrier>
x0x555555554857 <main+7>          sub    $0x28,%rsp
x0x55555555485b <main+11>         xor    %esi,%esi
x0x55555555485d <main+13>         mov    $0x2,%edx
x0x555555554862 <main+18>         mov    %fs:0x28,%rax
x0x55555555486b <main+27>         mov    %rax,0x18(%rsp)
x0x555555554870 <main+32>         xor    %eax,%eax
x0x555555554872 <main+34>         callq  0x5555555547d0 <pthread_barrier_init@plt>
x0x555555554877 <main+39>         lea    0x8(%rsp),%rdi
x0x55555555487c <main+44>         lea    0x384(%rip),%rcx        # 0x555555554c07
x0x555555554883 <main+51>         lea    0x206(%rip),%rdx        # 0x555555554a90 <do_work>
x0x55555555488a <main+58>         xor    %esi,%esi
x0x55555555488c <main+60>         callq  0x555555554790 <pthread_create@plt>
x0x555555554891 <main+65>         test   %eax,%eax
x0x555555554893 <main+67>         jne    0x5555555548ff <main+175>
x0x555555554895 <main+69>         lea    0x10(%rsp),%rdi
x0x55555555489a <main+74>         lea    0x370(%rip),%rcx        # 0x555555554c11
x0x5555555548a1 <main+81>         lea    0x1e8(%rip),%rdx        # 0x555555554a90 <do_work>
>x0x5555555548a8 <main+88>         xor    %esi,%esi
x0x5555555548aa <main+90>         callq  0x555555554790 <pthread_create@plt>
x0x5555555548af <main+95>         test   %eax,%eax
x0x5555555548b1 <main+97>         jne    0x555555554961 <main+273>
x0x5555555548b7 <main+103>        mov    0x8(%rsp),%rdi
x0x5555555548bc <main+108>        xor    %esi,%esi
x0x5555555548be <main+110>        callq  0x5555555547f0 <pthread_join@plt>
x0x5555555548c3 <main+115>        test   %eax,%eax
x0x5555555548c5 <main+117>        jne    0x555555554942 <main+242>
mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
multi-thre Thread 0x7ffff7feb7 In: main
0x0000555555554872 in main ()
0x0000555555554877 in main ()
0x000055555555487c in main ()
0x0000555555554883 in main ()
0x000055555555488a in main ()
0x000055555555488c in main ()
[New Thread 0x7ffff77c4700 (LWP 46510)]
0x0000555555554891 in main ()
0x0000555555554893 in main ()
0x0000555555554895 in main ()
0x000055555555489a in main ()
0x00005555555548a1 in main ()
0x00005555555548a8 in main ()
(gdb)
```

It is still xor %esi, %esi and don't have any changes. Which meant the ctx is still the null pointer in the second thread, the written data in first thread has not followed by the second thread. Therefore, the second thread will still enter the get instance funciton.



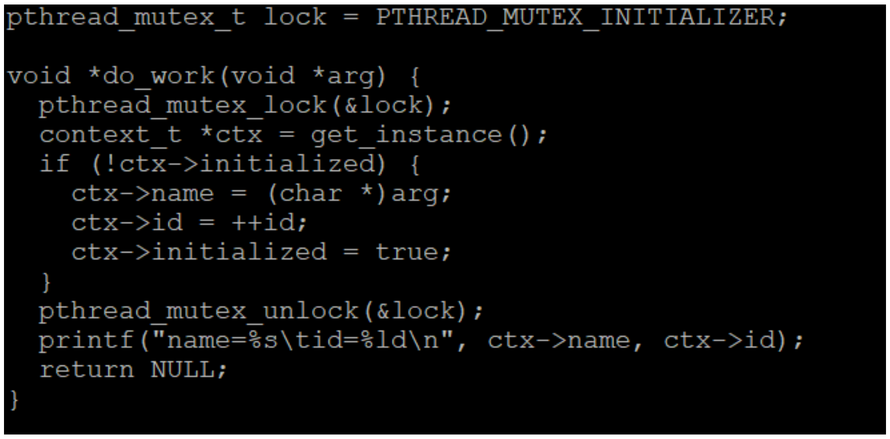
```
x0x7ffff7bc4e69 <__pthread_barrier_wait+297>    cmp    $0xb,%ecx
x0x7ffff7bc4e6c <__pthread_barrier_wait+300>    ja     0x7ffff7bc4dca <__pthread_barrier_wait+138>
x0x7ffff7bc4e72 <__pthread_barrier_wait+306>    mov    %r15,%rax
x0x7ffff7bc4e75 <__pthread_barrier_wait+309>    shl    %cl,%rax
x0x7ffff7bc4e78 <__pthread_barrier_wait+312>    test   $0x881,%eax
x0x7ffff7bc4e7d <__pthread_barrier_wait+317>    je     0x7ffff7bc4dca <__pthread_barrier_wait+138>
x0x7ffff7bc4e83 <__pthread_barrier_wait+323>    mov    (%r12),%ebp
x0x7ffff7bc4e87 <__pthread_barrier_wait+327>    cmp    %ebp,%r8d
x0x7ffff7bc4e8a <__pthread_barrier_wait+330>    ja     0x7ffff7bc4e49 <__pthread_barrier_wait+265>
x0x7ffff7bc4e8c <__pthread_barrier_wait+332>    lea    0x10(%r9),%rdx
mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
multi-thre Thread 0x7ffff77c47 In: __pthread_barrier_wait                              L61   PC: 0x7ffff7bc4e5
(gdb) thread 1
[Switching to thread 1 (Thread 0x7ffff7feb740 (LWP 964))]
#0  0x0000555555554a8e in main (argc=1, argv=0x7fffffffe508) at hw3d.c:57
(gdb) p ctx
$1 = (context_t *) 0x0
(gdb) thread 2
[Switching to thread 2 (Thread 0x7ffff77c4700 (LWP 2154))]
#0  0x00007ffff7bc4e5e in futex_wait (private=0, expected=0, futex_word=0x555555756044 <barrier+4>)
    at ../sysdeps/unix/sysv/linux/futex-internal.h:61
(gdb) p ctx
$2 = (context_t *) 0x0
(gdb)
```

Both of the ctx are null pointer

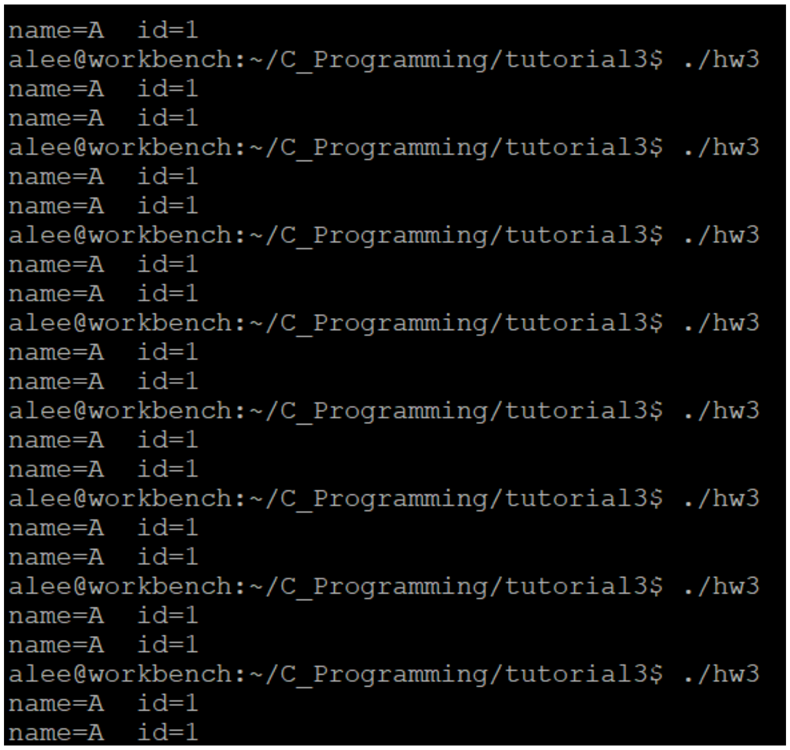
2.

Original code:

Add pthread_mutex_lock and pthread_mutex_unlock to inscribe the do_work function. When the first thread enters the do_work function it will block the second thread to enter the do_work function until it has finished the job. Therefore, the thread will become asynchronous.



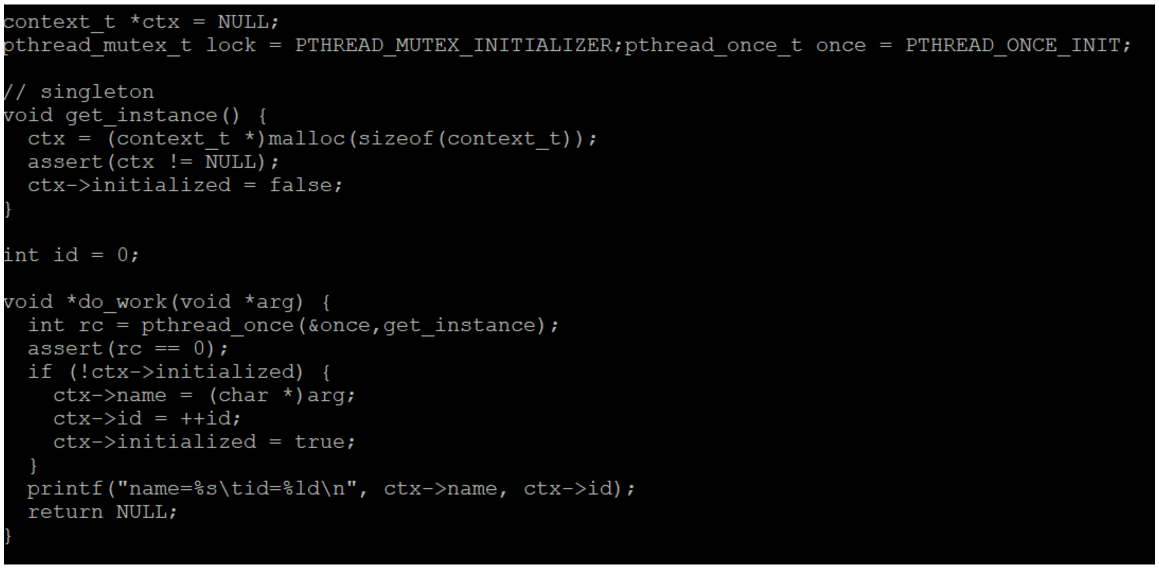
```
pthread_mutex_t lock = PTHREAD_MUTEX_INITIALIZER;

void *do_work(void *arg) {
  pthread_mutex_lock(&lock);
  context_t *ctx = get_instance();
  if (!ctx->initialized) {
    ctx->name = (char *)arg;
    ctx->id = ++id;
    ctx->initialized = true;
  }
  pthread_mutex_unlock(&lock);
  printf("name=%s\tid=%ld\n", ctx->name, ctx->id);
  return NULL;
}
```



```
name=A  id=1
alee@workbench:~/C_Programming/tutorial3$ ./hw3
name=A  id=1
name=A  id=1
alee@workbench:~/C_Programming/tutorial3$ ./hw3
name=A  id=1
name=A  id=1
alee@workbench:~/C_Programming/tutorial3$ ./hw3
name=A  id=1
name=A  id=1
alee@workbench:~/C_Programming/tutorial3$ ./hw3
name=A  id=1
name=A  id=1
alee@workbench:~/C_Programming/tutorial3$ ./hw3
name=A  id=1
name=A  id=1
alee@workbench:~/C_Programming/tutorial3$ ./hw3
name=A  id=1
name=A  id=1
alee@workbench:~/C_Programming/tutorial3$ ./hw3
name=A  id=1
name=A  id=1
alee@workbench:~/C_Programming/tutorial3$ ./hw3
name=A  id=1
name=A  id=1
```

Improvement code

Use the pthread_once replaces context_t *ctx = get_instance(). Under the pthread_once command the second thread will not enter the get_instance function again, as the first thread has already entered. It will save the time to avoid enter the get_instance function twice.



```c
context_t *ctx = NULL;
pthread_mutex_t lock = PTHREAD_MUTEX_INITIALIZER;pthread_once_t once = PTHREAD_ONCE_INIT;

// singleton
void get_instance() {
  ctx = (context_t *)malloc(sizeof(context_t));
  assert(ctx != NULL);
  ctx->initialized = false;
}

int id = 0;

void *do_work(void *arg) {
  int rc = pthread_once(&once,get_instance);
  assert(rc == 0);
  if (!ctx->initialized) {
    ctx->name = (char *)arg;
    ctx->id = ++id;
    ctx->initialized = true;
  }
  printf("name=%s\tid=%ld\n", ctx->name, ctx->id);
  return NULL;
}
```