LAB 8

23K0018

Fasih Hasan Khan

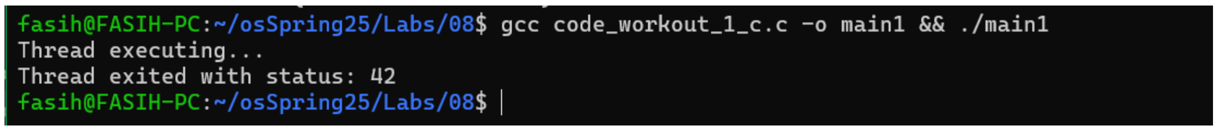
Code Workout #1

a) The variable num's pointer is passed to the runner function (typecasted to void*).

b) For now, we're using default **pthread** attributes.

c)

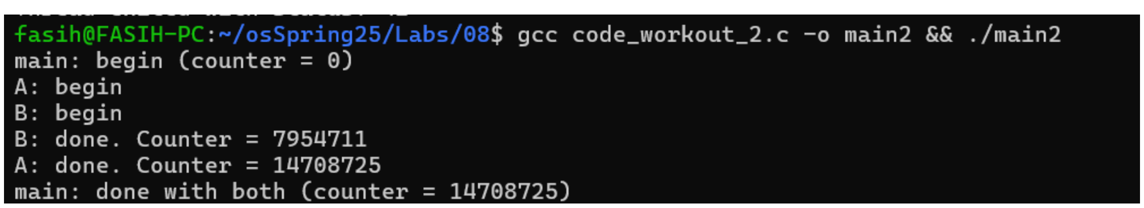
```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

void *thread_function(void *arg) {
  printf("Thread executing...\n");
  pthread_exit((void *)42);
}

int main() {
  pthread_t thread;
  pthread_create(&thread, NULL, thread_function, NULL);

  void *exit_status;
  pthread_join(thread, &exit_status);
  printf("Thread exited with status: %ld\n", (long)exit_status);

  return 0;
}
```



```
fasih@FASIH-PC:~/osSpring25/Labs/08$ gcc code_workout_1_c.c -o main1 && ./main1
Thread executing...
Thread exited with status: 42
fasih@FASIH-PC:~/osSpring25/Labs/08$ |
```

Code Workout #2

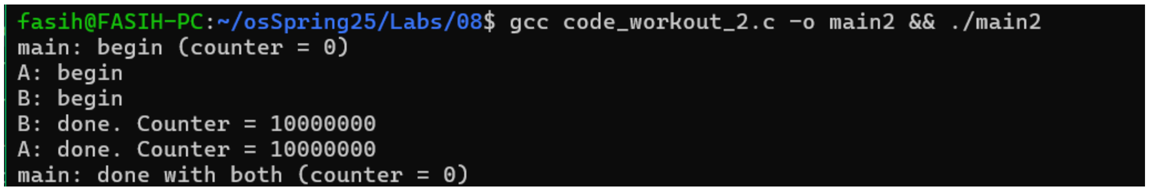
a)



```
fasih@FASIH-PC:~/osSpring25/Labs/08$ gcc code_workout_2.c -o main2 && ./main2
main: begin (counter = 0)
A: begin
B: begin
B: done. Counter = 7954711
A: done. Counter = 14708725
main: done with both (counter = 14708725)
```

b) Both threads share the same variable of **counter**. When thread A updates counter, thread B continues from where A had left instead of starting its own counter.

c) Instead of using the global counter, both threads now use the local instance of counter. This prevents them from making changes to the same variable. Which is why the global counter variable at the end is still 0.



```
fasih@FASIH-PC:~/osSpring25/Labs/08$ gcc code_workout_2.c -o main2 && ./main2
main: begin (counter = 0)
A: begin
B: begin
B: done. Counter = 10000000
A: done. Counter = 10000000
main: done with both (counter = 0)
```