Further pthread Practice

2021 — 03 — 23

Today's lab is much the same as yesterday: there are a bunch of snippets in this .zip file to work through. Answer each question below for the relevant .c file.

1. Write the `fizzbuzz` function used as an entrypoint to the threads. You are given a pointer to an integer as an argument. Convert to:
   - `"Fizz"` if it divides by 3
   - `"Buzz"` if it divides by 5
   - `"no div"` if it divides by neither, so we don't have to do int->string conversion!
2. 1.c prints our numbers in order, because it gets the return value from each thread and prints those. They print in order, because we print when we join each thread, and we join each thread in the order we started them.
   Move the printing logic so that each thread prints the string it calculated instead. Are they still in order? Why, or why not?
3. Fully comment 3.c and explain how the code works, and what it does. You can ignore the sleeping logic — I've commented that for us, and it's not something we covered in lectures!
4. Modify 3.c so that it accepts only a pointer to a `bool` as input, rather than the struct. Calculate the total in the thread and return an address using `pthread_exit()`[1]. Print the total in the main thread after catching the return value.
5. Implement a program which starts *two* pthreads. One reads input from the user, and adds numbers to a total. The other prints the current total value every 3 seconds. After 30 seconds, the main thread sets a bool to false that signals to both threads that they should join, just like in 3.c. You can use `sleep(int seconds)` from `unistd.h` to make your threads sleep.

[1] NOTE: You probably don't want to return an address to the thread's *stack*, because the stack will collapse when the function joins. Instead, the address should be heap-allocated, and free'd at the end of your `main` thread.