Boost.Asio part2

making thread pool using thread_group

Concurrent Programming

Introduction

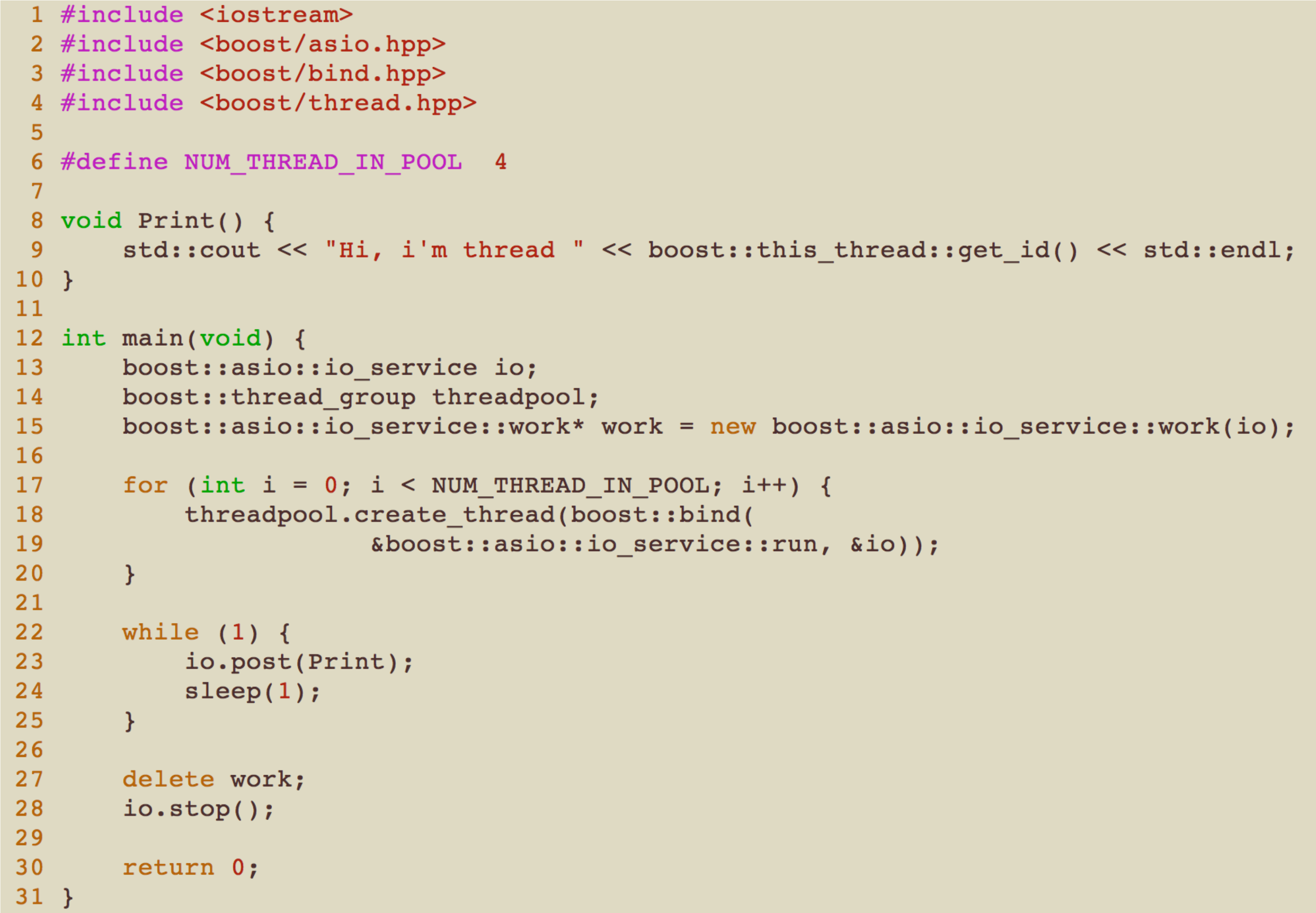
- What is the thread_group of Boost.Asio?

- Practice

What is the thread_group?

- thread_group provides for a collection of threads that are related in some fashion
- New threads can be added to the group with *add_thread()* and *create_thread()* member function
- You can use the *post()* function of the io_service to pass a job to the thread in the group

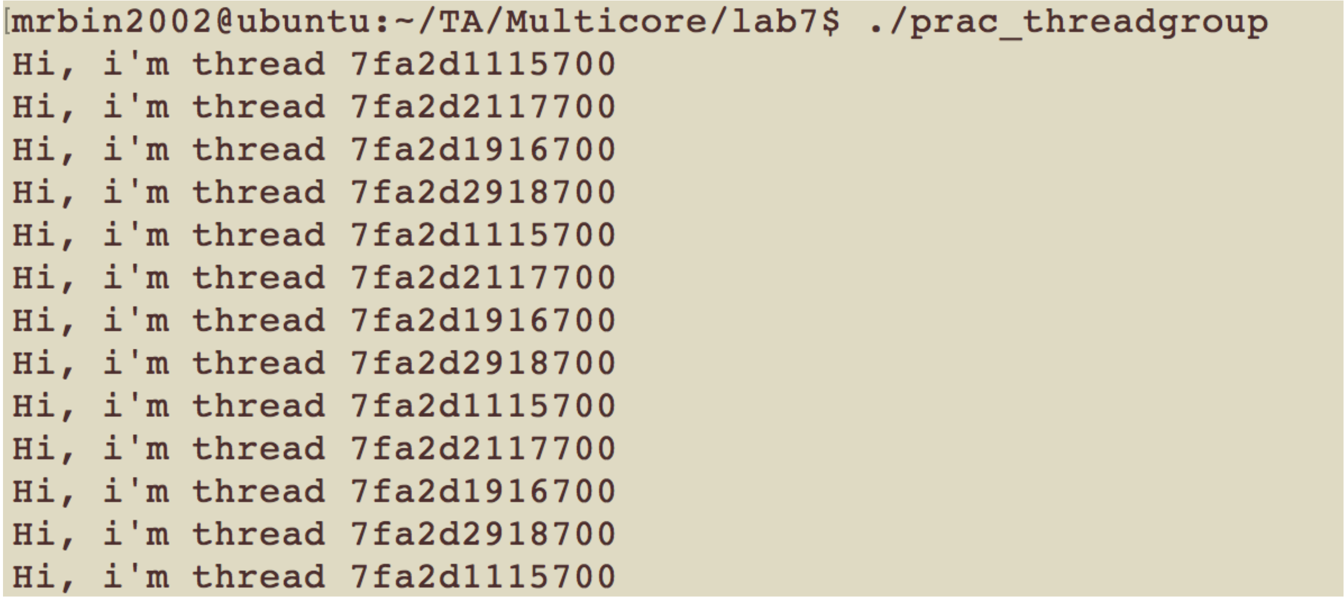
Simple thread pool with thread_group



```cpp
#include <iostream>
#include <boost/asio.hpp>
#include <boost/bind.hpp>
#include <boost/thread.hpp>

#define NUM_THREAD_IN_POOL  4

void Print() {
    std::cout << "Hi, i'm thread " << boost::this_thread::get_id() << std::endl;
}

int main(void) {
    boost::asio::io_service io;
    boost::thread_group threadpool;
    boost::asio::io_service::work* work = new boost::asio::io_service::work(io);

    for (int i = 0; i < NUM_THREAD_IN_POOL; i++) {
        threadpool.create_thread(boost::bind(
                    &boost::asio::io_service::run, &io));
    }

    while (1) {
        io.post(Print);
        sleep(1);
    }

    delete work;
    io.stop();

    return 0;
}
```

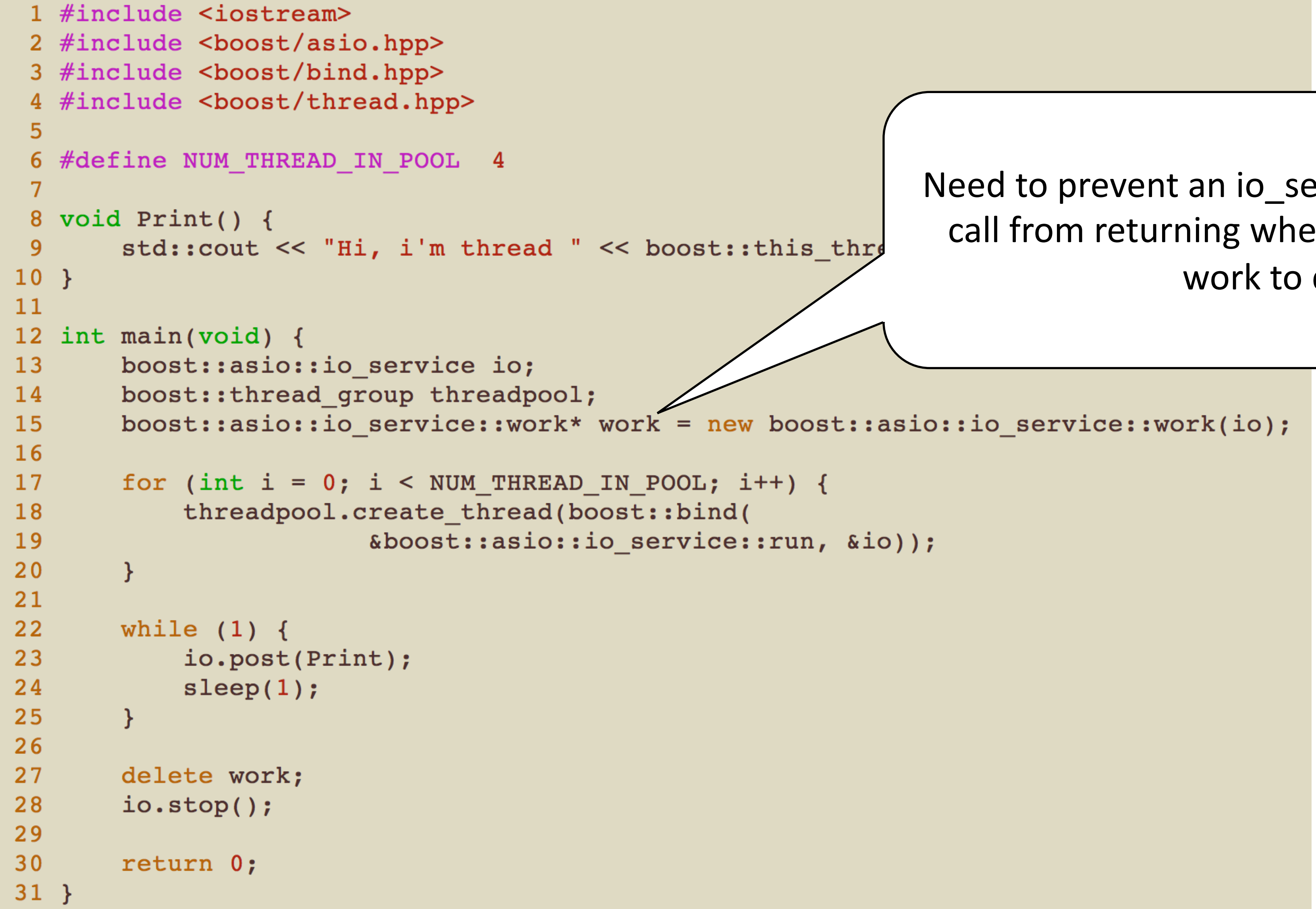
Simple thread pool with thread_group

```
$ g++ -o prac_threadgroup prac_threadgroup.cpp –lboost_system –lboost_thread
```



```
mrbin2002@ubuntu:~/TA/Multicore/lab7$ ./prac_threadgroup
Hi, i'm thread 7fa2d1115700
Hi, i'm thread 7fa2d2117700
Hi, i'm thread 7fa2d1916700
Hi, i'm thread 7fa2d2918700
Hi, i'm thread 7fa2d1115700
Hi, i'm thread 7fa2d2117700
Hi, i'm thread 7fa2d1916700
Hi, i'm thread 7fa2d2918700
Hi, i'm thread 7fa2d1115700
Hi, i'm thread 7fa2d2117700
Hi, i'm thread 7fa2d1916700
Hi, i'm thread 7fa2d2918700
Hi, i'm thread 7fa2d1115700
```

Simple thread pool with thread_group



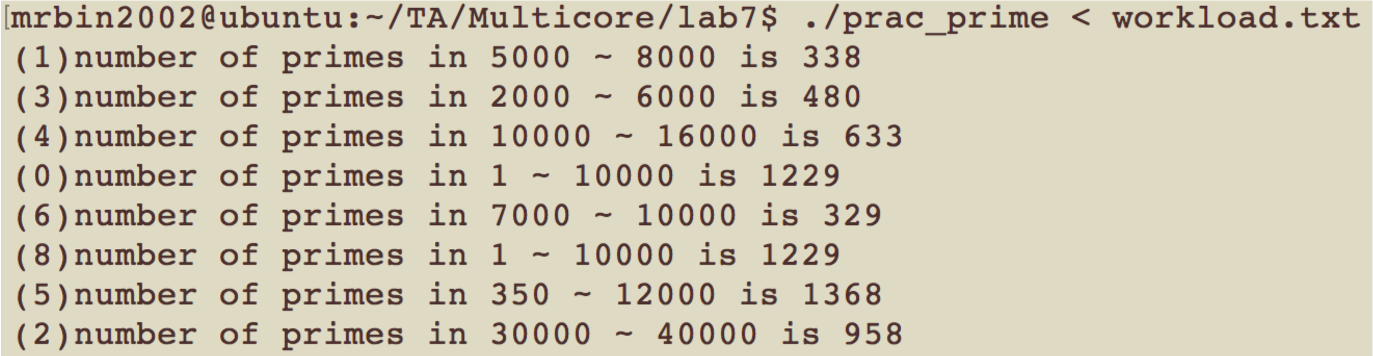
```cpp
#include <iostream>
#include <boost/asio.hpp>
#include <boost/bind.hpp>
#include <boost/thread.hpp>

#define NUM_THREAD_IN_POOL  4

void Print() {
    std::cout << "Hi, i'm thread " << boost::this_thre
}

int main(void) {
    boost::asio::io_service io;
    boost::thread_group threadpool;
    boost::asio::io_service::work* work = new boost::asio::io_service::work(io);

    for (int i = 0; i < NUM_THREAD_IN_POOL; i++) {
        threadpool.create_thread(boost::bind(
                    &boost::asio::io_service::run, &io));
    }

    while (1) {
        io.post(Print);
        sleep(1);
    }

    delete work;
    io.stop();

    return 0;
}
```

Need to prevent an io_service object's run() call from returning when ther is no more work to do

Practice

- Implement the program using thread_group
  - Two integer values *range_start* & *range_end* are given from the standard input
  - Pass this range and sequence number to a worker thread in the thread pool
  - Worker thread calculates the number of prime number in given range and send the result, sequence number to the single printer thread that is not in the thread pool
  - Single printer thread prints each result in a line like

    (*sequence_number*) number of primes in *range_start* ~ *range_end* is *result*
  - Increase sequence number and repeat it until the *range_start* is given as -1



```
mrbin2002@ubuntu:~/TA/Multicore/lab7$ ./prac_prime < workload.txt
(1)number of primes in 5000 ~ 8000 is 338
(3)number of primes in 2000 ~ 6000 is 480
(4)number of primes in 10000 ~ 16000 is 633
(0)number of primes in 1 ~ 10000 is 1229
(6)number of primes in 7000 ~ 10000 is 329
(8)number of primes in 1 ~ 10000 is 1229
(5)number of primes in 350 ~ 12000 is 1368
(2)number of primes in 30000 ~ 40000 is 958
```

The order of result's sequence number could be mixed, but single result should be printed in a single line

Thank You