Week 8

Exercise 66

../66–3/main.cc

```
#include "main.ih"

int main(int argc, char const **argv)
{
  future<string> fut = async(threadFun);

  size_t count = 0;

  while (true)
  {
    this_thread::sleep_for(chrono::seconds(1));
    cerr << "inspecting: " << ++count << ’\n’;

    if (fut.wait_for(chrono::seconds(0)) == future_status::ready)
    {
      cout << "done \n";
      return 1;
    }
    // inspect whether a thread indicates
    // to end the program. If so, end it.
  }
}

// If we were to run multiple threads we could have a vector of futures, then
// rather than checking if our one future object is ready we check if any of
// the futures is ready, if one (or however many is preferred) is/are ready
// the program returns. We can keep track howmany are done with a simple
// and keep track of which futures are done with a vector of bools so we dont
// check futures that have already been counted.
```

../66–3/main.ih

```
#define ERR(msg) printf("%s : %d", (msg), __LINE__)

#include <thread>
#include <chrono>
#include <future>
#include <iostream>
#include <future>

using namespace std;

string threadFun();
```

../66–3/threadFun.cc

```
#include "main.ih"

string threadFun()
{
  cerr << "entry \n";

  this_thread::sleep_for(chrono::seconds(5));
  cerr << "first cerr \n";

  this_thread::sleep_for(chrono::seconds(5));
  cerr << "second cerr \n";

  return "end the program";
}
```