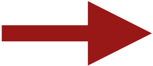
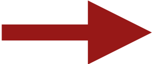
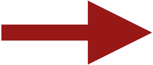
```
std::vector<std::future<void>> futures(10);

for(unsigned i = 0; i<10; ++i)
  futures[i] = std::async(std::launch::any, f, i);

for(auto&f: futures )
  f.wait();
```

```cpp
std::mutex m;
using lock = std::lock_guard<std::mutex>;
std::map<std::thread::id,bool> ids;

void f(unsigned i)
{
    lock lk{m};
    auto id = std::this_thread::get_id();

    std::cout << "thread #"<<i<< " id = " << id << std::endl;
    ids.insert(std::make_pair(id, false));
}
```







std::async + std::future

```
std::mutex m;
using lock = std::lock_guard<std::mutex>;
std::map<std::thread::id,bool> ids;

void f(unsigned i)
{
    lock lk{m};
    auto id = std::this_thread::get_id();

    std::cout << "thread #"<<i<< " id = " << id << std::endl;
    ids.insert(std::make_pair(id, false));
}


std::vector<std::future<void>> futures(10);

for(unsigned i = 0; i<10; ++i)
  futures[i] = std::async(std::launch::any, f, i);

for(auto&f: futures )
  f.wait();
```

Possible output

```
nik@Nicolas-MacBook-Air:~/GitHub/cpp_sandbox/multithreading/thread_spawn$ ./a.out
thread #0 id = 0x1041f0000
thread #1 id = 0x104273000
thread #2 id = 0x1042f6000
thread #3 id = 0x1041f0000
thread #4 id = 0x104273000
thread #5 id = 0x1042f6000
thread #6 id = 0x1041f0000
thread #7 id = 0x104273000
thread #8 id = 0x1042f6000
thread #9 id = 0x1041f0000
Actual thread spawned = 3
0x1041f0000
0x104273000
0x1042f6000
```