```
void hello()
{
    std::cout << "Hello World! thread id = "
      << std::this_thread::get_id() << std::endl;
}

int main(int argc, char** argv)
{
  std::thread t(hello);
  return 0;
}
```

```
nik@Nicolas-MacBook-Air:~/GitHub/cpp_sandbox/multithreading/thread_spawn$
clang++ -std=c++14 thread1.cpp
nik@Nicolas-MacBook-Air:~/GitHub/cpp_sandbox/multithreading/thread_spawn$ ./
a.out
libc++abi.dylib: terminating
Abort trap: 6
```

```cpp
void hello()
{
   std::cout << "Hello World! thread id = "
     << std::this_thread::get_id() << std::endl;
}

int main(int argc, char** argv)
{
  std::thread t(hello);
  return 0;
}
```

```
nik@Nicolas-MacBook-Air:~/GitHub/cpp_sandbox/multithreading/thread_spawn$
clang++ -std=c++14 thread1.cpp
nik@Nicolas-MacBook-Air:~/GitHub/cpp_sandbox/multithreading/thread_spawn$ ./
a.out
libc++abi.dylib: terminating
Abort trap: 6
```

std::thread

- We ran ~thread(); when we exited from the scope of main and it destroyed the thread object.

- **If *this has an associated thread std::terminate() is called**

- We hold a thread as long as:
  (joinable() == true)

- We have to wait that the thread terminates its execution either calling join or detach