Threads & Signals

- Dealing with signals can be complicated
  even with a process-based paradigm.
  Introducing threads into the picture makes things even more complicated.
- Each thread has its own signal mask (*see* pthread_sigmask()),
  but the signal disposition is shared by all threads in the process.
  - This means that individual threads can block signals,
    but when a thread modifies the action associated with a given signal,
    all threads share the action.
  - Thus, if one thread chooses to ignore a given signal,
    another thread can undo that choice by restoring the default disposition or
    installing a signal handler for the signal.
- Signals are delivered to a single thread in the process.
  - If the signal is related to a hardware fault or expiring timer,
    the signal is sent to the thread whose action caused the event.
  - **Other signals**, on the other hand, are **delivered** to an **arbitrary thread**.
- To send a signal to a thread, we call pthread_kill(tid,signo).

C++11 Threads

- **C++ includes built-in support for**
  - **threads**
  - **mutual exclusion**
  - **condition variables and**
  - **futures**

TO COMPILE:
g++ prog.cpp -pthread -std=c++11 -Wall -o prog

```
#include <iostream>
#include <thread>
using namespace std;

void thrFunc()
{
  cout << "In aux thread" << endl;
}

int main()
{
  thread t(thrFunc);
  cout << "In main thread ..." << endl;
  t.join();
  cout << "...back to main thread" << endl;
  return 0;
}
```

```
#include <iostream>
#include <thread>

using namespace std;

void func(int x)
{
  cout << "Inside thread: received
           parameter = " << x << endl;
}

int main()
{
  int i = 10;
  cout << "Launching thread ... parameter
=" << i << endl;
  thread t(func, i);
  t.join();
  cout << "Thread ended" << endl;
  return 0;
}
```

C++11 Threads - parameters

```cpp
#include <iostream>
#include <thread>
#include <string>
using namespace std;

// The thread function can have multiple parameters
// ... but all them are passed by value

void func(int i, double d, const std::string &s)
{
  cout << i << ", " << d << ", " << s << endl;
}

int main()
{
  thread t(func, 10, 1.75, "hello");
  t.join();

  return 0;
}
```

```cpp
#include <iostream>
#include <thread>
#include <string>
using namespace std;

// To pass a parameter by reference
// it must be wrapped in a std::ref object
// (see the call below)

void func(int &i, double &d,string &s)
{
  cout << i << ", " << d << ", " << s << endl;
  i++;
  d--;
  s = s + " world";
}

int main()
{
  int a = 10; double b = 1.75; string c = "hello";
  thread t(func, ref(a), ref(b), ref(c));
  t.join();
  cout << a << ", " << b << ", " << c << endl;
  return 0;
}
```