Signal/slots from libsigc++

- Introduction
- Signals
- Slots
- Connections
- Using adapters

http://minos.phy.bnl.gov/~bviren/talks/

Introduction to Libsigc++

- Small “modern” C++ templated library once part of `GTK--`, now separate project.
- Provides **compile-time** type safe signal/slot mechanism.
- No preprocessor, no extensions to C++ language.
- A signal(er) is just an object, many types can be slots.
- http://libsigc.sourceforge.net/

Signals

A signal is a templated functional object:

```
SigC::Signal2<void,double,double*> NewData;
```

It is parametrized by the type of return, and args. The number indicates the number of args.

Emitting the signal:

```
double data[100];
// ...
NewData(100,data);
// or
NewData.emit(100,data);
// or if it is part of a class:
obj.NewData(100,data);
// or
obj.NewData.emit(100,data);
```

Slots

Slots can be (almost) anything. From Sigc++ docs a slot can be one of:

- member function of any object derived from SigC::Object
- function object derived from SigC::Object
- static, global or friend function
- static function object
- member function to a static object

Connections

Various ways to make a connection:

- `sig.connect(slot(obj,Obj::&method));`
- `sig.connect(slot(&func));`
- `sig.connect(sigc_obj.slot());`
- `sig.connect(other_sig.slot());`

The `connect()` method returns a `Connection` object which can be used to disconnect later (or can be ignored):

```
Connection c;
c = obj.sig.connect(slot(&func));
//...
c.disconnect();
```

Using Sigc++ Adapters

Sigc++ comes with some useful adapters:

- bind:

```
void func(int);
Signal0<void> sig;
sig.connect(bind(slot(&func), 20));
```

- slot_class:

```
class YourClass : public SigC::Object {...};
class MyClass {...};
Signal0<void> sig;
sig.connect(slot(yourobj,&YourClass::method));
sig.connect(slot_class(myobj,&MyClass::method));
```