Analysis Workbook. C++-based Analysis

This section will guide you trough the steps necessary to perform simple analysis in a C++ analysis framework (psana). For more detailed description of psana consult Psana User Manual - Old.

User modules in psana are the instances of the C++ class which must inherit from a special base class `psana::Module`. There are several methods in the base class which can be overriden in the subclass (event() must always be implemented):

- `void beginJob(Event& evt, Env& env)`
- `void beginRun(Event& evt, Env& env)`
- `void beginCalibCycle(Event& evt, Env& env)`
- `void event(Event& evt, Env& env)`
- `void endCalibCycle(Event& evt, Env& env)`
- `void endRun(Event& evt, Env& env)`
- `void endJob(Event& evt, Env& env)`

To start with doing analysis one should have user release setup first as explained in Packages and Releases. If you have not done it yet create a package for your analysis with some unique name:

```
newrel ana-current my_ana_rel # to make user release
cd my_ana_rel
sit_setup # never forget this!
newpkg my_ana_pkg # to make package for analysis code
```

Good place to start writing analysis module is to create skeleton module from existing template:

```
mkdir my_ana_pkg/src my_ana_pkg/include
codegen -l psana-module my_ana_pkg my_ana_mod
```

This will create files `my_ana_pkg/include/my_ana_mod.h` and `my_ana_pkg/src/my_ana_mod.cpp` which you need to edit and fill with some useful code. Start your favorite editor:

```
vi my_ana_pkg/src/my_ana_mod.cpp my_ana_pkg/include/my_ana_mod.h
```

For example purposes we are going to use beam line data and print few interesting values on every event. For that the code will be quite simple (excluding comments):

my_ana_pkg/include/my_ana_mod.h

```
#ifndef MY_ANA_PKG_MY_ANA_MOD_H
#define MY_ANA_PKG_MY_ANA_MOD_H

#include "psana/Module.h"

namespace my_ana_pkg {

class my_ana_mod : public Module {
public:

  my_ana_mod (const std::string& name) ;

  virtual ~my_ana_mod () ;

  virtual void event(Event& evt, Env& env);

};

}
#endif // MY_ANA_PKG_MY_ANA_MOD_H
```

my_ana_pkg/src/my_ana_mod.cpp

```
#include "my_ana_pkg/my_ana_mod.h"

#include "MsgLogger/MsgLogger.h"
#include "PSEvt/EventId.h"
#include "psddl_psana/bld.ddl.h"

// This declares this class as psana module
using namespace my_ana_pkg;
PSANA_MODULE_FACTORY(my_ana_mod)

namespace my_ana_pkg {

my_ana_mod::my_ana_mod (const std::string& name)
  : Module(name)
{
}

my_ana_mod::~my_ana_mod ()
{
}

void
my_ana_mod::event(Event& evt, Env& env)
{
  // get event time
  PSTime::Time evtTime;
  boost::shared_ptr<PSEvt::EventId> eventId = evt.get();
  if (eventId.get()) {
    evtTime = eventId->time();
  }

  // get beam data
  boost::shared_ptr<Psana::Bld::BldDataEBeam> ebeam = evt.get(Source());
  if (ebeam.get()) {
    MsgLog(name(), info, "time: " << evtTime << ", charge: " << ebeam->ebeamCharge() <<
           ", energy: " << ebeam->ebeamL3Energy());
  }
}

}
```

After you finished editing your module build everything in the release:

```
scons
```

To run this analysis start pyana giving it module name and input files:

```
psana -m my_ana_pkg.my_ana_mod /reg/d/psdm/AMO/amo14110/xtc/e43-r0100-s0*
```

which should produce output similar to this:

```
[info:my_ana_pkg.my_ana_mod] time: 2010-07-31 20:05:30.043360843-07, charge: 0.233403, energy: 4393.24
[info:my_ana_pkg.my_ana_mod] time: 2010-07-31 20:05:30.093381914-07, charge: 0.248759, energy: 4380.69
[info:my_ana_pkg.my_ana_mod] time: 2010-07-31 20:05:30.143398446-07, charge: 0.243547, energy: 4384.99
[info:my_ana_pkg.my_ana_mod] time: 2010-07-31 20:05:30.193412744-07, charge: 0.249897, energy: 4393.3
[info:my_ana_pkg.my_ana_mod] time: 2010-07-31 20:05:30.243427879-07, charge: 0.248667, energy: 4389.08
[info:my_ana_pkg.my_ana_mod] time: 2010-07-31 20:05:30.293446716-07, charge: 0.245666, energy: 4392.14
[info:my_ana_pkg.my_ana_mod] time: 2010-07-31 20:05:30.343468975-07, charge: 0.251093, energy: 4389.05
[info:my_ana_pkg.my_ana_mod] time: 2010-07-31 20:05:30.393488859-07, charge: 0.253302, energy: 4389.31
...
```

This is probably the simplest module that you can create and it does not use any interesting features such as module parameters. Here is an example of slightly more advanced module, it's job is to count fraction of events with beam energy above certain threshold. Replace code in module files with this:

my_ana_pkg/include/my_ana_mod.h

```
#ifndef MY_ANA_PKG_MY_ANA_MOD_H
#define MY_ANA_PKG_MY_ANA_MOD_H

#include "psana/Module.h"

namespace my_ana_pkg {

class my_ana_mod : public Module {
public:

  my_ana_mod (const std::string& name) ;

  virtual ~my_ana_mod () ;

  virtual void event(Event& evt, Env& env);

  virtual void endJob(Event& evt, Env& env);

private:
  double m_threshold;
  unsigned long m_count;
  unsigned long m_total;
};

}
#endif // MY_ANA_PKG_MY_ANA_MOD_H
```

my_ana_pkg/src/my_ana_mod.cpp

```
#include "my_ana_pkg/my_ana_mod.h"

#include "MsgLogger/MsgLogger.h"
#include "psddl_psana/bld.ddl.h"

// This declares this class as psana module
using namespace my_ana_pkg;
PSANA_MODULE_FACTORY(my_ana_mod)

namespace my_ana_pkg {

my_ana_mod::my_ana_mod (const std::string& name)
  : Module(name)
  , m_threshold(0.0)
  , m_count(0)
  , m_total(0)
{
  m_threshold = config("threshold");
}

my_ana_mod::~my_ana_mod ()
{
}

void
my_ana_mod::event(Event& evt, Env& env)
{
  // get beam data
  boost::shared_ptr<Psana::Bld::BldDataEBeam> ebeam = evt.get(Source());
  if (ebeam.get()) {
    if (ebeam->ebeamL3Energy() > m_threshold) {
      ++ m_count;
    }
    ++ m_total;
  }
}

void
my_ana_mod::endJob(Event& evt, Env& env)
{
    MsgLog(name(), info, "Fraction of events with energy>" << m_threshold
            << " is " << m_count*100.0/m_total << "%");
}

}
```

This module requires input parameter (threshold) from configuration file. Create file psana.cfg in current directory with this contents:

```
[psana]
modules = my_ana_pkg.my_ana_mod

[my_ana_pkg.my_ana_mod]
threshold = 4400
```

And run the job, module name is specified in config file so it is not needed on the command line:

```
% scons
% psana /reg/d/psdm/AMO/amo14110/xtc/e43-r0100-s0*
[info:my_ana_pkg.my_ana_mod] Fraction of events with energy>4400 is 15.9316%
```

More interesting example is to run two instances of the same module in a job with different set of parameters. For this just replace the content of psana.cfg with this:

```
[psana]
modules = my_ana_pkg.my_ana_mod:thresh4390 my_ana_pkg.my_ana_mod:thresh4400

[my_ana_pkg.my_ana_mod:thresh4390]
threshold = 4390

[my_ana_pkg.my_ana_mod:thresh4400]
threshold = 4400
```

and run again:

```
% psana /reg/d/psdm/AMO/amo14110/xtc/e43-r0100-s0*
[info:my_ana_pkg.my_ana_mod:thresh4390] Fraction of events with energy>4390 is 55.613%
[info:my_ana_pkg.my_ana_mod:thresh4400] Fraction of events with energy>4400 is 15.9316%
```