Fermilab | U.S. DEPARTMENT OF ENERGY | Office of Science

Changes implemented in LArSoft to support *art* 3.03

Kyle J. Knoepfel

19 November 2019

LArSoft coordination meeting

Significance of *art* 3.03

- Nothing too significant
  - Mostly preparing for bigger changes in *art* 3.04 and 3.05
- Improving abstractions
  - Changes to prevent passing around 'this' pointers
  - (Most of the) breaking changes presented on next pages
- `art::Event::getProductTokens<MyType>` interface
  - Returns list of product tokens (or, alternatively, input tags) corresponding to the products that would be retrieved in a `getByManyType` call.
- EmptyEvent now has a time-limit option
  - Tells the framework to create new events until the time limit is exceeded

`produces` and `consumes` may be called only within modules

- Sometimes produces/consumes calls are delegated to helper functions/classes.

```
MyModule(ParameterSet const& pset)
  : EDProducer{pset}
{
  call_produces(*this);
}

void
call_produces(art::EDProducer& module)
{
  module.produces<MyType>();
}
```

`produces` and `consumes` may be called only within modules

- Sometimes produces/consumes calls are delegated to helper functions/classes.
- Only modules and “collectors” may now call produces and consumes

```
MyModule(ParameterSet const& pset)
  : EDProducer{pset}
{
  call_produces(*this);
}

void
call_produces(art::EDProducer& module)
{
  module.produces<MyType>();
}
```

```
MyModule(ParameterSet const& pset)
  : EDProducer{pset}
{
  call_produces(producesCollector());
}

void
call_produces(art::ProducesCollector& collector)
{
  collector.produces<MyType>();
}
```

- This actually simplifies code…sometimes considerably.

API changes in lardata

```
- MVAWriter writer(this, "some_name");
+ MVAWriter writer(producesCollector(), "some_name");

- recob::HitRefinerAssociator assoc(prod, evt, fHitProducerLabel, fDoWireAssns, fDoRawDigitAssns);
+ recob::HitRefinerAssociator assoc(evt, fHitProducerLabel, fDoWireAssns, fDoRawDigitAssns);

- recob::ChargedSpacePointCollectionCreator::produces(*this, "noreg");
+ recob::ChargedSpacePointCollectionCreator::produces(producesCollector(), "noreg");

# Important - notice change in how creator is constructed
- recob::ChargedSpacePointCollectionCreator spcol_pre(event, *this, "pre");
+ auto spcol_pre = recob::ChargedSpacePointCollectionCreator::forPtrs(event, "pre");

- recob::HitCollectionCreator::declare_products(*this, "uhits");
+ recob::HitCollectionCreator::declare_products(producesCollector(), "uhits");

- recob::HitCollectionCreator hitCollection_U(*this, evt, "uhits");
+ recob::HitCollectionCreator hitCollection_U(evt, "uhits");

- recob::HitCollectionAssociator::declare_products(*this);
+ recob::HitCollectionAssociator::declare_products(producesCollector());

- recob::HitCollectionAssociator hcol(*this,e,fWireModuleLabel,true);
+ recob::HitCollectionAssociator hcol(e,fWireModuleLabel,true);
```

API changes in larreco

```
- virtual void IHit3DBuilder::produces(art::EDProducer*);
+ virtual void IHit3DBuilder::produces(art::ProducesCollector&);

- virtual void IHit3DBuilder::Hit3DBuilder(art::EDProducer&, art::Event&, reco::HitPairList&, RecobHitToPtrMap&);
+ virtual void IHit3DBuilder::Hit3DBuilder(art::Event&, reco::HitPairList&, RecobHitToPtrMap&);
```

API changes in larreco

```
- virtual void IHit3DBuilder::produces(art::EDProducer*);
+ virtual void IHit3DBuilder::produces(art::ProducesCollector&);

- virtual void IHit3DBuilder::Hit3DBuilder(art::EDProducer&, art::Event&, reco::HitPairList&, RecobHitToPtrMap&);
+ virtual void IHit3DBuilder::Hit3DBuilder(art::Event&, reco::HitPairList&, RecobHitToPtrMap&);
```

- There are likely a few others
  - All changes have been incorporated in feature branches
  - Experiments should use the `feature/team_for_art_v3_03` branches to pick them up

- The summary, in broad strokes:
  - There should be considerably fewer places where the ‘this’ pointer is passed

Interface change for `util::CreateAssn(…)`

- The createAssn utilities have long required a pointer/reference to the module where the Assns is created.
  - This is due to an older *art* interface for creating art::Ptrs.
- This is no longer necessary for creating art::Ptrs, or for calling createAssn.

```
util::CreateAssn(*this, evt, seedVec, seedHitVec, seedHitAssns); // Supported, not yet
                                                                 // formally deprecated
util::CreateAssn(evt, seedVec, seedHitVec, seedHitAssns); // Preferred
```

- However, even the `CreateAssn` utilities will soon become unnecessary.
  - See page 24 of the LCM presentation from September 10:
  - https://indico.fnal.gov/event/21770/contribution/2/material/slides/0.pdf

Changes required for *art* 3.04

- Released last week
- Minor feature enhancements and some important bug fixes
- Supports only e19 (GCC 8) and c7 (Clang 7) compilers
- Python 3 enabled by default; to use Python 2, use the py2 qualifier
  - SLF6 systems must use the py2 qualifier: SciSoft SLF6 builds do not support Python 3
- Support dropped for macOS 10.13 (High Sierra)
  - macOS 10.14 (Mojave) support kept for now

Tentative plan for *art* 3.05

- Will support latest available versions of GCC and Clang
- External product version updates
- Change in interacting with services
  - Modules/services/sources must declare which services they are going to use
  - Creating `ServiceHandles` anywhere in code will be deprecated for *art* 3.05, and impossible in later versions
  - Enables ability to remove configurations of unused services from the job