Tutorial

LCIO

LCIO is the event data model of Marlin and thus of MarlinTPC. It defines the data classes which are stored, for instance a `lcio::TrackerHit`, which is a reconstructed 3D hit with spatial coodrinates and the deposited charge. All these objects are derived from `lcio::LCObject`.

LCIO objects are stored in so called `lcio::LCCollections`. Each collection contains LCObject of a certain kind, for instance `lcio::TrackerHits`. Each collection has a user given name which should describe which sort of data is stored.

Examples:

- A collection with `TrackerHits` from the TPC is called *TPCHits*
- Another collecton, also containing `TrackerHits` but from a silicon hodoscope, whould be called *HodoscopeHits*
- A collection with `lcio::Tracks` from the TPC standalone reconstruction could be alled *TPCTracks*, while the final tracks from both TPC and silicon hodoscope might be named *Tracks*.

A lcio file consists of a sequence of `lcio::LCEvents` which contain all the collections which have been reconstructed.

Example:

Event Nr	Collection Name	Collection Type
1		
	TPCHits	`lcio::TrackerHit`
	TPCTracks	`lcio::Track`
2		
	TPCHits	`lcio::TrackerHit`
3		
	TPCHits	`lcio::TrackerHit`
	TPCTracks	`lcio::Track`

The events don't necessarily have the same collections. In the second event for instance no tracks have been found, which means there is no *TPCTracks* collection.

Marlin

Marlin is a framework to reconstruct and analyse lcio data. It works on an event by event basis and opens the lcio files for you, provides GEAR geometry data and condiditons data from the Linear Collider Conditions Data toolkit (LCCD).

The individual computing tasks are implemented as so called *Processors*, which allows good modularity of the code. The processor gets an lcio event and its *processEvent* method is called. This can modify the event and add collections.

To use the MarlinTPC processors you have to set the MARLIN_DLL variable (see Compiling MarlinTPC, section *Compiling with cmake*, item D).

Steering Files

Marlin is controlled by steering files. In these files it it defined which lcio files to process and which processors to use, as well as the steering parameters for the individual processors.

Example: Searching for straight tracks using a hough transformation (TrackFinderHoughTrafoProcessor):

```
<marlin>
 <execute>
  <processor name="MyTrackFinderHoughTrafoProcessor"/>
  <processor name="MyLCIOOutputProcessor"/>
 </execute>

 <global>
  <parameter name="LCIOInputFiles"> TPCHits.slcio </parameter>
  <parameter name="Verbosity" options="DEBUG0-4,MESSAGE0-4,WARNING0-4,ERROR0-4,SILENT"> WARNING0
 </global>

 <processor name="MyLCIOOutputProcessor" type="LCIOOutputProcessor">
 <!--Writes the current event to the specified LCIO outputfile. Needs to be the last ActiveProces
  <!-- name of output file -->
  <parameter name="LCIOOutputFile" type="string">TPCHits_and_TrackCandidates.slcio </parameter>
  <!--write mode for output file:  WRITE_APPEND or WRITE_NEW-->
  <parameter name="LCIOWriteMode" type="string">WRITE_NEW </parameter>
</processor>

 <processor name="MyTrackFinderHoughTrafoProcessor" type="TrackFinderHoughTrafoProcessor">
 <!--TrackFinderHoughTrafoProcessor uses the hough transformation to find hits on track, track pa
  <!--Name of the input collection-->
  <parameter name="InputCollectionName" type="string" lcioInType="TrackerHit">TimePixHits </param
  <!--Maximum distance of a hit to the track in mm-->
  <parameter name="MaximumDistanceToTrack" type="double">0.5 </parameter>
  <!--minimum number of hits on one track-->
  <parameter name="NHitsOnTrack" type="int">5 </parameter>
  <!--Name of the output collection-->
  <parameter name="OutputCollectionName" type="string" lcioOutType="Track">TimePixTrackCandidates
  <!--if not 0 the output collection is set transient-->
  <parameter name="SetOutputTransient" type="int">0 </parameter>
</processor>

</marlin>
```

Although this is a very simple example it is already quite long. Let's go through the XML file step by step.

- After the starting `<marlin>` tag there is the `<execute>` section. Here every processor which is going to be executed is listed. In this case it's only two processors: The track finder and the LCIOOutputProcessor.
- The second section is the global Marlin configuration (`<global>`). In this example only the two most important parameters are set:
  - The name of the lcio input file
  - The verbosity level. Marlin uses a streamlog system which defines vebosity levels from DEBUG0 (most verbose) to ERROR4 (only the most important errors). SILENT suppresses all output. The example level WARNING0 shows all warnings and errors.
- The following sections configures the individual processors. For each processor there is one section which defines it's parameter (to be correct one section for each instance o the processor. One can run the same processor several times with different parameters. This is why in the `<execute>` section the user specified name *myLCIOOutputProcessor* is given instead of the processor type (`LCIOOutputProcessor`).
  - The *myLCIOOutputProcessor* instance of the `LCIOOutputProcessor` gets the name of output file (cannot be the same as the input file) and the write mode (WRITE_NEW).

- Data processors like the `TrackFinderHoughTrafoProcessor` usually work on specific collection. These collection names (usually one input and one output collection) are specified in this section. The `TrackFinderHoughTrafoProcessor` also needs some additional parameters: The minimal number of hits to accept the track, and the maximal allowed distance of a hit from the track. They are also defined in this section.

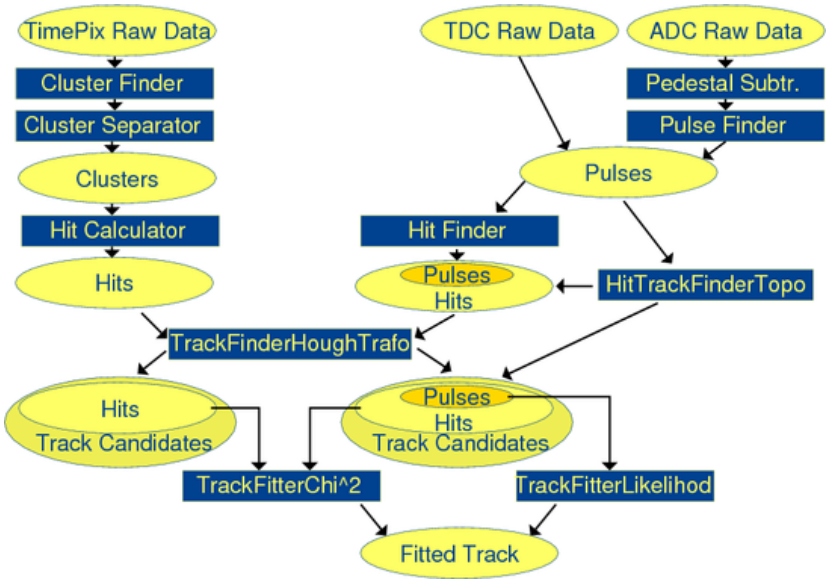
Examples

There are more complete examples which come together with MarlinTPC. See Example Steering Files section for more details.

Reconstruction Work Flow

Up to now MarlinTPC consists of more than 50 Processors from which you plug together the task you want to perform. The following coarse view of the possible reconstruction flows is not complete, for instance a track seeder is missing before the track fitter. For information about the individual Processors please refer to the doxygen documentation.

- Coarse view of the reconstruction flow:



FIXME: Improve information about reconstruction flow and available processors.

-- MartinKillenberg - 25 Jul 2008

This topic: ILCTPC > UserWorkbookTutorial
Topic revision: r4 - 2008-07-25 - MartinKillenberg