Snowmass '21, Sep 30th 2020

Overview

This is a tutorial on the simulation and reconstruction software for the Muon Collider, which is based on CLIC's ILCSoft. It has been setup to run on the Snowmass machine login.snowmass21.io.

Tutorial Agenda: https://indico.fnal.gov/event/45187/

Monte Carlo samples

The Monte Carlo samples used in this tutorial are stored in the directory:

```
/collab/project/snowmass21/data/muonc/
```

Signal samples

The available $^+$ hard-scattering processes are:

1. $^+$ H bb at s = 1.5 TeV, generated with PYTHIA8;
2. $^+$ HH bbbb at s = 3 TeV, generated with WIZARD;
3. $^+$ bbbb at s = 3 TeV, generated with WIZARD.

Beam-induced background sample

The Beam-Induced Background (BIB) at 1.5 TeV for one bunch-crossing has been generated by N. Mokhov with MARS15.

This has been simulated using this software with the last available detector geometry developed for the Muon Collider (CLIC_o3_v14_mod4)

All the BIB has been split in 16 files which contain a total of 2993 pseudo-events; this is an artificial subdivision used to speed up the simulation.

Simulation and reconstruction code

The Muon Collider simulation and reconstruction code is available in a singularity container on CERN's cvmfs:

```
/cvmfs/unpacked.cern.ch/registry.hub.docker.com/infnpd/mucoll-ilc-framework:1.0-centos8
```

N.B.: A docker container is also available at this link.

Steering files and ROOT macros

All the configuration files and ROOT macros used in this tutorial are available in the github repository:

```
https://github.com/MuonColliderSoft/MuC-Tutorial.git
```

Tutorial setup

1 - Log into the login.snowmass.io machine:

```
ssh -XY <account name>@login.snowmass21.io
```

2 - Access the singularity container:

```
singularity run --bind `echo $HOME` --bind /collab/project/snowmass21/data/muonc:/data /cvmfs/unpacked.cern.ch
/registry.hub.docker.com/infnpd/mucoll-ilc-framework:1.0-centos8
```

The "bind" option is used to to mount the $HOME directory and the /data directory with the Monte Carlo samples.

N.B.: The source code is saved in the `/opt/ilcsoft/v02-01-pre/` directory.

3 - Check that the Muon Collider software is properly set up. If everything is OK, you should see the following outputs with no error messages:

```
usage: Running DD4hep Simulations: [-h] [--steeringFile STEERINGFILE]
                                   [--compactFile COMPACTFILE]
                                   [--runType {batch,vis,run,shell}]
                                   [--inputFiles INPUTFILES [INPUTFILES ...]]
                                   [--outputFile OUTPUTFILE]
                                   [-v {1,2,3,4,5,6,7,VERBOSE,DEBUG,INFO,WARNING,ERROR,FATAL,ALWAYS}]
                                   [--numberOfEvents NUMBEROFEVENTS]
                                   [--skipNEvents SKIPNEVENTS]
                                   [--physicsList PHYSICSLIST]
                                   [--crossingAngleBoost CROSSINGANGLEBOOST]
                                   [--vertexSigma X Y Z T]
                                   [--vertexOffset X Y Z T]
                                   [--macroFile MACROFILE] [--enableGun]
                                   [--enableG4GPS] [--enableG4Gun]
                                   [--dumpParameter]
                                   [--enableDetailedShowerMode]
                                   [--dumpSteeringFile]
                                   [--output.inputStage {1,2,3,4,5,6,7,VERBOSE,DEBUG,INFO,WARNING,ERROR,FATAL,ALWAYS}]
                                   [--output.kernel {1,2,3,4,5,6,7,VERBOSE,DEBUG,INFO,WARNING,ERROR,FATAL,ALWAYS}]
                                   [--output.part {1,2,3,4,5,6,7,VERBOSE,DEBUG,INFO,WARNING,ERROR,FATAL,ALWAYS}]
                                   [--output.random {1,2,3,4,5,6,7,VERBOSE,DEBUG,INFO,WARNING,ERROR,FATAL,ALWAYS}]
                                   [--random.seed RANDOM.SEED]
                                   [--random.type RANDOM.TYPE]
                                   [--random.luxury RANDOM.LUXURY]
                                   [--random.replace_gRandom RANDOM.REPLACE_GRANDOM]
                                   [--random.file RANDOM.FILE]
                                   [--random.enableEventSeed]
                                   [--gun.energy GUN.ENERGY]
                                   [--gun.particle GUN.PARTICLE]
                                   [--gun.multiplicity GUN.MULTIPLICITY]
                                   [--gun.phiMin GUN.PHIMIN]
                                   [--gun.phiMax GUN.PHIMAX]
                                   [--gun.thetaMin GUN.THETAMIN]
                                   [--gun.thetaMax GUN.THETAMAX]
                                   [--gun.direction GUN.DIRECTION]
                                   [--gun.distribution {uniform,costheta,eta,pseudorapidity,ffbar}]
                                   [--gun.isotrop GUN.ISOTROP]
                                   [--gun.position GUN.POSITION]
                                   [--part.enableDetailedHitsAndParticleInfo]
                                   [--part.keepAllParticles PART.KEEPALLPARTICLES]
                                   [--part.minDistToParentVertex PART.MINDISTTOPARENTVERTEX]
                                   [--part.minimalKineticEnergy PART.MINIMALKINETICENERGY]
                                   [--part.printEndTracking PART.PRINTENDTRACKING]
                                   [--part.printStartTracking PART.PRINTSTARTTRACKING]
                                   [--part.saveProcesses PART.SAVEPROCESSES]
                                   [--part.userParticleHandler PART.USERPARTICLEHANDLER]
                                   [--field.stepper FIELD.STEPPER]
                                   [--field.equation FIELD.EQUATION]
                                   [--field.eps_min FIELD.EPS_MIN]
                                   [--field.eps_max FIELD.EPS_MAX]
                                   [--field.min_chord_step FIELD.MIN_CHORD_STEP]
                                   [--field.delta_chord FIELD.DELTA_CHORD]
                                   [--field.delta_intersection FIELD.DELTA_INTERSECTION]
                                   [--field.delta_one_step FIELD.DELTA_ONE_STEP]
                                   [--field.largest_step FIELD.LARGEST_STEP]
                                   [--action.calo ACTION.CALO]
                                   [--action.mapActions ACTION.MAPACTIONS]
                                   [--action.tracker ACTION.TRACKER]
                                   [--guineapig._parameters GUINEAPIG._PARAMETERS]
                                   [--guineapig.particlesPerEvent GUINEAPIG.PARTICLESPEREVENT]
                                   [--lcio._parameters LCIO._PARAMETERS]
```

```
                                        [--lcio.mcParticleCollectionName LCIO.MCPARTICLECOLLECTIONNAME]
                                        [--meta.eventParameters META.EVENTPARAMETERS [META.EVENTPARAMETERS ...]]
                                        [--meta.runNumberOffset META.RUNNUMBEROFFSET]
                                        [--meta.eventNumberOffset META.EVENTNUMBEROFFSET]
                                        [--filter.calo FILTER.CALO]
                                        [--filter.filters FILTER.FILTERS]
                                        [--filter.mapDetFilter FILTER.MAPDETFILTER]
                                        [--filter.tracker FILTER.TRACKER]
                                        [--physics.decays PHYSICS.DECAYS]
                                        [--physics.list PHYSICS.LIST]
                                        [--physics.pdgfile PHYSICS.PDGFILE]
                                        [--physics.rangecut PHYSICS.RANGECUT]
                                        [--physics.rejectPDGs PHYSICS.REJECTPDGS]
                                        [--physics.zeroTimePDGs PHYSICS.ZEROTIMEPDGS]

optional arguments:
    -h, --help show this help message and exit
    --steeringFile STEERINGFILE, -S STEERINGFILE
            Steering file to change default behaviour
    --compactFile COMPACTFILE
            The compact XML file
    --runType {batch,vis,run,shell}
            The type of action to do in this invocation
            batch: just simulate some events, needs numberOfEvents, and input file or gun
            vis: enable visualisation, run the macroFile if it is set
            run: run the macroFile and exit
            shell: enable interactive session
    --inputFiles INPUTFILES [INPUTFILES ...], -I INPUTFILES [INPUTFILES ...]
            InputFiles for simulation .stdhep, .slcio, .HEPEvt, .hepevt, .hepmc, .pairs files are supported
    --outputFile OUTPUTFILE, -O OUTPUTFILE
            Outputfile from the simulation,only lcio output is supported
    -v {1,2,3,4,5,6,7,VERBOSE,DEBUG,INFO,WARNING,ERROR,FATAL,ALWAYS}, --printLevel {1,2,3,4,5,6,7,VERBOSE,DEBUG,INFO,WARNING,
ERROR,FATAL,ALWAYS}
            Verbosity use integers from 1(most) to 7(least) verbose
            or strings: VERBOSE, DEBUG, INFO, WARNING, ERROR, FATAL, ALWAYS
    --numberOfEvents NUMBEROFEVENTS, -N NUMBEROFEVENTS
            number of events to simulate, used in batch mode
    --skipNEvents SKIPNEVENTS
            Skip first N events when reading a file
    --physicsList PHYSICSLIST
            Physics list to use in simulation
    --crossingAngleBoost CROSSINGANGLEBOOST
            Lorentz boost for the crossing angle, in radian!
    --vertexSigma X Y Z T
            FourVector of the Sigma for the Smearing of the Vertex position: x y z t
    --vertexOffset X Y Z T
            FourVector of translation for the Smearing of the Vertex position: x y z t
    --macroFile MACROFILE, -M MACROFILE
            Macro file to execute for runType 'run' or 'vis'
    --enableGun, -G enable the DDG4 particle gun
    --enableG4GPS enable the Geant4 GeneralParticleSource. Needs a macroFile (runType run)or use it with the shell (runType shell)
    --enableG4Gun enable the Geant4 particle gun. Needs a macroFile (runType run) or use it with the shell (runType shell)
    --dumpParameter, --dump
            Print all configuration Parameters and exit
    --enableDetailedShowerMode
            use detailed shower mode
    --dumpSteeringFile print an example steering file to stdout
    --output.inputStage {1,2,3,4,5,6,7,VERBOSE,DEBUG,INFO,WARNING,ERROR,FATAL,ALWAYS}
            Output level for input sources
    --output.kernel {1,2,3,4,5,6,7,VERBOSE,DEBUG,INFO,WARNING,ERROR,FATAL,ALWAYS}
            Output level for Geant4 kernel
    --output.part {1,2,3,4,5,6,7,VERBOSE,DEBUG,INFO,WARNING,ERROR,FATAL,ALWAYS}
            Output level for ParticleHandler
    --output.random {1,2,3,4,5,6,7,VERBOSE,DEBUG,INFO,WARNING,ERROR,FATAL,ALWAYS}
            Output level for Random Number Generator setup
    --random.seed RANDOM.SEED
    --random.type RANDOM.TYPE
    --random.luxury RANDOM.LUXURY
    --random.replace_gRandom RANDOM.REPLACE_GRANDOM
    --random.file RANDOM.FILE
    --random.enableEventSeed
            If True, calculate random seed for each event basedon eventID and runID
            Allows reproducibility even whenSkippingEvents
    --gun.energy GUN.ENERGY
    --gun.particle GUN.PARTICLE
    --gun.multiplicity GUN.MULTIPLICITY
    --gun.phiMin GUN.PHIMIN
            Minimal azimuthal angle for random distribution
    --gun.phiMax GUN.PHIMAX
```

--gun.thetaMin GUN.THETAMIN
--gun.thetaMax GUN.THETAMAX
--gun.direction GUN.DIRECTION
direction of the particle gun, 3 vector
--gun.distribution {uniform,cos(theta),eta,pseudorapidity,ffbar}
choose the distribution of the random direction for theta
Options for random distributions:
'uniform' is the default distribution, flat in theta
'cos(theta)' is flat in cos(theta)
'eta', or 'pseudorapidity' is flat in pseudorapity
'ffbar' is distributed according to 1+cos^2(theta)
Setting a distribution will set isotrop = True

--gun.isotrop GUN.ISOTROP
isotropic distribution for the particle gun
use the options phiMin, phiMax, thetaMin, and thetaMax to limit the range of randomly distributed directions
if one of these options is not None the random distribution will be set to True and cannot be turned off!

--gun.position GUN.POSITION
position of the particle gun, 3 vector
--part.enableDetailedHitsAndParticleInfo
Enable lots of printout on simulated hits and MC-truth information
--part.keepAllParticles PART.KEEPALLPARTICLES
Keep all created particles
--part.minDistToParentVertex PART.MINDISTTOPARENTVERTEX
Minimal distance between particle vertex and endpoint of parent after
which the vertexIsNotEndpointOfParent flag is set

--part.minimalKineticEnergy PART.MINIMALKINETICENERGY
MinimalKineticEnergy to store particles created in the tracking region
--part.printEndTracking PART.PRINTENDTRACKING
Printout at End of Tracking
--part.printStartTracking PART.PRINTSTARTTRACKING
Printout at Start of Tracking
--part.saveProcesses PART.SAVEPROCESSES
List of processes to save, on command line give as whitespace separated string in quotation marks
--part.userParticleHandler PART.USERPARTICLEHANDLER
Optionally enable an extended Particle Handler
--field.stepper FIELD.STEPPER
--field.equation FIELD.EQUATION
--field.eps_min FIELD.EPS_MIN
--field.eps_max FIELD.EPS_MAX
--field.min_chord_step FIELD.MIN_CHORD_STEP
--field.delta_chord FIELD.DELTA_CHORD
--field.delta_intersection FIELD.DELTA_INTERSECTION
--field.delta_one_step FIELD.DELTA_ONE_STEP
--field.largest_step FIELD.LARGEST_STEP
--action.calo ACTION.CALO
set the default calorimeter action
--action.mapActions ACTION.MAPACTIONS
create a map of patterns and actions to be applied to sensitive detectors
example: SIM.action.mapActions['tpc'] = "TPCSDAction"
--action.tracker ACTION.TRACKER
set the default tracker action
--guineapig._parameters GUINEAPIG._PARAMETERS
--guineapig.particlesPerEvent GUINEAPIG.PARTICLESPEREVENT
Set the number of pair particles to simulate per event.
Only used if inputFile ends with ".pairs"
If "-1" all particles will be simulated in a single event

--lcio._parameters LCIO._PARAMETERS
--lcio.mcParticleCollectionName LCIO.MCPARTICLECOLLECTIONNAME
Set the name of the collection containing the MCParticle input.
Default is "MCParticle".

--meta.eventParameters META.EVENTPARAMETERS [META.EVENTPARAMETERS ...]
Event parameters to write in every event. Use C/F/I ids to specify parameter type. E.g parameterName/F=0.42 to set a float parameter
--meta.runNumberOffset META.RUNNUMBEROFFSET
The run number offset to write in slcio output file. E.g setting it to 42 will start counting runs from 42 instead of 0
--meta.eventNumberOffset META.EVENTNUMBEROFFSET
The event number offset to write in slcio output file. E.g setting it to 42 will start counting events from 42 instead of 0
--filter.calo FILTER.CALO
default filter for calorimeter sensitive detectors;
this is applied if no other filter is used for a calorimeter

--filter.filters FILTER.FILTERS
list of filter objects: map between name and parameter dictionary
--filter.mapDetFilter FILTER.MAPDETFILTER
a map between patterns and filter objects, using patterns to attach filters to sensitive detector

```
    --filter.tracker FILTER.TRACKER
            default filter for tracking sensitive detectors; this is applied if no other filter is used for a tracker
    --physics.decays PHYSICS.DECAYS
            If true, add decay processes for all particles.
            Only enable when creating a physics list not based on an existing Geant4 list!

    --physics.list PHYSICS.LIST
            The name of the Geant4 Physics list.
    --physics.pdgfile PHYSICS.PDGFILE
            location of particle.tbl file containing extra particles and their lifetime information
            For example in $DD4HEP/examples/DDG4/examples/particle.tbl

    --physics.rangecut PHYSICS.RANGECUT
            The global geant4 rangecut for secondary production
              Default is 0.7 mm as is the case in geant4 10
              To disable this plugin and be absolutely sure to use the Geant4 default range cut use "None"
             Set printlevel to DEBUG to see a printout of all range cuts,
              but this only works if range cut is not "None"

    --physics.rejectPDGs PHYSICS.REJECTPDGS
            Set of PDG IDs that will not be passed from the input record to Geant4.
              Quarks, gluons and W's Z's etc should not be treated by Geant4

    --physics.zeroTimePDGs PHYSICS.ZEROTIMEPDGS
            Set of PDG IDs for particles that should not be passed to Geant4 if their properTime is 0.
              The properTime of 0 indicates a documentation to add FSR to a lepton for example.
<!-- Loading shared library : /opt/ilcsoft/v02-01-pre/MarlinDD4hep/v00-06/lib/libMarlinDD4hep.so.0.6.0 (libMarlinDD4hep.so)-->
<!-- Loading shared library : /opt/ilcsoft/v02-01-pre/DDMarlinPandora/v00-11/lib/libDDMarlinPandora.so.0.11.0 (libDDMarlinPandora.so)-->
<!-- Loading shared library : /opt/ilcsoft/v02-01-pre/MarlinReco/v01-26/lib/libMarlinReco.so.1.26.0 (libMarlinReco.so)-->
<!-- Loading shared library : /opt/ilcsoft/v02-01-pre/PandoraAnalysis/v02-00-01/lib/libPandoraAnalysis.so.2.0.1 (libPandoraAnalysis.so)-->
<!-- Loading shared library : /opt/ilcsoft/v02-01-pre/LCFIVertex/v00-08/lib/libLCFIVertexProcessors.so.0.8.0 (libLCFIVertexProcessors.so)-->
<!-- Loading shared library : /opt/ilcsoft/v02-01-pre/CEDViewer/v01-17/lib/libCEDViewer.so.1.17.0 (libCEDViewer.so)-->
<!-- Loading shared library : /opt/ilcsoft/v02-01-pre/Overlay/mc/lib/libOverlay.so.0.22.0 (libOverlay.so)-->
<!-- Loading shared library : /opt/ilcsoft/v02-01-pre/MarlinFastJet/v00-05-02/lib/libMarlinFastJet.so.0.5.2 (libMarlinFastJet.so)-->
<!-- Loading shared library : /opt/ilcsoft/v02-01-pre/LCTuple/v01-12mc/lib/libLCTuple.so.1.12.0 (libLCTuple.so)-->
<!-- Loading shared library : /opt/ilcsoft/v02-01-pre/MarlinKinfit/v00-06/lib/libMarlinKinfit.so.0.6.0 (libMarlinKinfit.so)-->
<!-- Loading shared library : /opt/ilcsoft/v02-01-pre/MarlinTrkProcessors/mc/lib/libMarlinTrkProcessors.so.2.11.0 (libMarlinTrkProcessors.so)-->
<!-- Loading shared library : /opt/ilcsoft/v02-01-pre/MarlinKinfitProcessors/v00-04-01/lib/libMarlinKinfitProcessors.so.0.4.1 (libMarlinKinfitProcessors.so)-->
<!-- Loading shared library : /opt/ilcsoft/v02-01-pre/ILDPerformance/v01-07/lib/libILDPerformance.so.1.7.0 (libILDPerformance.so)-->
<!-- Loading shared library : /opt/ilcsoft/v02-01-pre/ClicPerformance/HEAD/lib/libClicPerformance.so.2.3.0 (libClicPerformance.so)-->
<!-- Loading shared library : /opt/ilcsoft/v02-01-pre/Clupatra/v01-03/lib/libClupatra.so.1.3.0 (libClupatra.so)-->
<!-- Loading shared library : /opt/ilcsoft/v02-01-pre/Physsim/v00-04-01/lib/libPhyssim.so.0.4.1 (libPhyssim.so)-->
<!-- Loading shared library : /opt/ilcsoft/v02-01-pre/LCFIPlus/v00-08/lib/libLCFIPlus.so.0.8.0 (libLCFIPlus.so)-->
<!-- Loading shared library : /opt/ilcsoft/v02-01-pre/FCalClusterer/v01-00-01/lib/libFCalClusterer.so (libFCalClusterer.so)-->
<!-- Loading shared library : /opt/ilcsoft/v02-01-pre/ForwardTracking/v01-14-mucoll-01/lib/libForwardTracking.so.1.14.0 (libForwardTracking.so)-->
<!-- Loading shared library : /opt/ilcsoft/v02-01-pre/ConformalTracking/v01-10/lib/libConformalTracking.so.1.10.0 (libConformalTracking.so)-->
<!-- Loading shared library : /opt/ilcsoft/v02-01-pre/LICH/v00-01/lib/libLICH.so.0.1.0 (libLICH.so)-->
<!-- Loading shared library : /opt/ilcsoft/v02-01-pre/Garlic/v03-01/lib/libGarlic.so.3.1.0 (libGarlic.so)-->

Usage: Marlin [OPTION] [FILE]...
    runs a Marlin application

Running the application with a given steering file:
    Marlin steer.xml

    Marlin [-h/-?] print this help information
    Marlin -x print an example steering file to stdout
    Marlin -c steer.xml check the given steering file for consistency
    Marlin -u old.xml new.xml consistency check with update of xml file
    Marlin -d steer.xml flow.dot create a program flow diagram (see: http://www.graphviz.org)

Example:
To create a new default steering file from any Marlin application, run
    Marlin -x > mysteer.xml
and then use either an editor or the MarlinGUI to modify the created steering file
to configure your application and then run it. e.g. :
    Marlin mysteer.xml > marlin.out 2>&1 &

Dynamic command line options may be specified in order to overwrite individual steering file parameters, e.g.:
    Marlin --global.LCIOInputFiles="input1.slcio input2.slcio" --global.GearXMLFile=mydetector.xml
           --MyLCIOOutputProcessor.LCIOWriteMode=WRITE_APPEND --MyLCIOOutputProcessor.LCIOOutputFile=out.slcio steer.xml

    NOTE: Dynamic options do NOT work together with Marlin options (-x, -f) nor with the MarlinGUI
```

4 - Check out the configuration files and the ROOT macros:

```
git clone https://github.com/MuonColliderSoft/MuC-Tutorial.git

cd  MuC-Tutorial && git checkout v2.0
```

1 - Running the simulation and reconstruction

Simulation step

The simulation program is part of the DD4Hep package, which uses as main components the ROOT geometry package, which is used for construction and visualization of geometry, and the Geant4 simulation toolkit, which can be interfaced via DD4hep to perform detector simulation in complex detector designs.

In the MuC-Tutorial/tutorial/1-mumuHHbbbb/ directory you can find two examples of the configuration files used by the simulation command: `ddsim`

To go to the directory use the command line:

```
cd ~/MuC-Tutorial/tutorial/1-mumuHHbbbb/
```

All the options can be passed as arguments to the command line, but probably the easiest way is to set everything in the steering file and then invoke for example:

```
ddsim --steeringFile sim_steer_Hbb.py > sim.out 2>&1
```

Looking in the configuration file you can see for example:

```
SIM.compactFile = "/opt/ilcsoft/v02-01-pre/detector-simulation/geometries/CLIC_o3_v14_mod4/CLIC_o3_v14.xml"        geometry
definition
SIM.inputFiles = ["/data/samples/HH/mumu2H2bb750.stdhep"]                                                       Input
file
SIM.outputFile = "mumu_H_bb.slcio"
Output file
SIM.numberOfEvents =
10                                                                                                  Number of events
```

Use the output of the command `ddsim -h` to see all the other parameters that you can set.

Digitization/reconstruction step

This second step is done using the command: `Marlin`: Modular Analysis and Reconstruction for the LINear Collider. As described in the main github page of the program: the idea is that every computing task is implemented as a processor (module) that analyses data in an LCEvent and creates additional output collections that are added to the event. The framework allows to define the processors (and their order) that are executed at runtime in a simple steering file. Via the steering file you can also define named parameters (string, float, int - single and arrays) for every processor as well as for the global scope. By using the framework users don't have to write any code that deals with the IO they simply write processors with defined callbacks, i.e. init(), processRunHeader(), processEvent(), end().

You can have a list of all the available processors with all their parameters using the follow command. Note that the file `processor.txt` can be see as an example steering file for the Marlin program:

```
Marlin -x > processor.txt
```

We can now use the output of the previous step to reconstruct the simulated events.

In the same directory MuC-Tutorial/tutorial/1-mumuHHbbbb/ run the reconstruction:

```
Marlin reco_steer_Hbb.xml > reco.out 2>&1
```

This command produces two output files: `Output_REC.slcio` with all the collections produced (you can specify which collections need to be saved in the file) and `histograms.root` which contains some diagnostics plots and trees.

Giving a quick look at the steering file you can see that it is an xml file enclosed in the `<marlin> ... </marlin>` tag. Then there are three main sections:

1) execute section (ordered list of processors to be executed):

```
<execute>
  <processor name="MyAIDAProcessor"/>
  <processor name="MyTestProcessor"/>
  <processor name="MyLCIOOutputProcessor"/>
</execute>
```

2) global section (global settings):

```
<global>
  <parameter name="LCIOInputFiles"> input.slcio </parameter>
  <parameter name="MaxRecordNumber" value="1000" />
</global>
```

3) processor section (processor configuration, there is one section for each processors of the "execute" list):

```
<processor name="MyLCIOOutputProcessor" type="LCIOOutputProcessor">
  <parameter name="LCIOOutputFile" type="string"> Output_DST.slcio </parameter>
  <parameter name="DropCollectionTypes" type="StringVec">
    SimCalorimeterHit
    SimTrackerHit
  </parameter>
  <parameter name="LCIOWriteMode" type="string" value="WRITE_NEW"/>
  <parameter name="SplitFileSizekB" type="int">1048576 </parameter>
  <parameter name="Verbosity" type="string">WARNING </parameter>
</processor>
```

To produce final ntuples for the analysis:

```
Marlin lctuple_steer.xml > ntuples.out 2>&1
```

The final root file with ntuples is: `JetHistograms.root`

For exercise, 1000 events of double Higgs production are already simulated and reconstructed, and you can run a simple macro `invariant_mass.C` for the events analysis:

In order to run the macro:

```
root -l
.L invariant_mass.C
invariant_mass()
```

The argument is used to chose the plot you want to see:

- 0: number of reconstructed jets
- 1: pseudorapidity of jets in the event
- 2: transverse momentum of jets in the event
- 3: phi of jets in the event
- 4: energy of jets in the event
- 5: invariant mass of jet pair associated to Higgs with highest pT
- 6: invariant mass of jet pair associated to Higgs with lowest pT

As example:

```
root -l
.L invariant_mass.C
invariant_mass(5)
```

the plot of the invariant mass of jet pair associated to Higgs with highest pT is shown

Some useful tools

The default output format of the reconstruction package is an LCIO (**L**inear **C**ollider **I/O**) file. You can inspect these files using these commands:

1) To print the list of saved collection and their sizes use:

```
anajob Output_REC.slcio
```

2) To see the details of all the collection of the first event:

```
dumpevent Output_REC.slcio 1 > event1.txt
```

If you are interest only in the number of events saved use:

```
lcio_event_counter Output_REC.slcio
```

You can also merge or split an LCIO files:

```
Singularity> lcio_merge_files -h

usage: lcio_merge_files <output-file> <input-file1> [[input-file2],...]

Singularity> lcio_split_file -h

usage: lcio_splitfile infilename outfilename sizeInBytes
```

You can also write a ROOT tuple for a more standard analysis. For this purpose you can use the LCTuple processor: it creates a ROOT TTree with a column wise ntuple from LCIO collections. You can see an example of use in the `MuC-Tutorial/config/reco/lctuple_steer.xml` file.

```
Marlin MuC-Tutorial/config/reco/lctuple_steer.xml > lctuple.out
```

This command produces the ROOT file: `lctuple_example.root`

Event display

To display events use the package CEDViewer:

```
ced2go -d /opt/ilcsoft/muonc/detector-simulation/geometries/MuColl_v1/MuColl_v1.xml Output_REC.slcio
```

2 - A look at the beam-induced background (BIB)

To get acquainted with the beam-induced background (BIB) two ROOT files have been prepared and are available in /data/ntuples/BIB (within the singularity container):

- mcParts_ntuple_BIB.root contains a TTree with the BIB Monte Carlo particles (MCpartTuple);
- allHits_ntuple_BIB.root contains TTrees with the reconstructed hits of the tracker, the ECAL and HCAL calorimeters, and the muon detercors (TrackerHitsTuple, CaloHitsTuple, MuonHitsTuple).

The macros to run over the TTrees and fill some significant histograms are in the directory:

```
cd ~/MuC-Tutorial/tutorial/2-BIB
```

Instructions are in the readme.txt file, available locally or at

https://github.com/MuonColliderSoft/MuC-Tutorial/blob/master/tutorial/2-BIB/readme.txt

3 - Overlaying the BIB to signals

To overlay background events from an additional set of LCIO files you can use the Overlay processor. You can list all files of the BIB and then decide which quota (i.e. the number of artificial pseudo events) to use. You can also decide a priori a set of time cuts to apply to each detector. Remember that signal and BIB are overlaid in the digitization step so the signal and BIB samples must be generated and simulated prior to overlaying.

You can check the example in the tutorial/3-mu_BIB directory. In this exercise you are going to learn how to check the reconstruction differences with and without the BIB.
The chosen signal is an event with 1000 prompt muons, with uniform pt between 0 and 10 GeV. The signal has been already simulated for you, now you are going to reconstruct it.

First going in the right directory:

```
cd ~/MuC-Tutorial/tutorial/3-mu_BIB/
```

To reconstruct the signal without the BIB digit:

```
Marlin reco_sig_only_steer.xml > log_sig_only.log
```

It will produce an `histograms_sig_only.root` output.

To reconstruct the signal overlayed with the BIB (10/2993 of the BIB bunch crossing is used as example):

```
Marlin reco_sig_and_BIB_steer.xml > log_sig_and_BIB.log
```

It will produce an `histograms_sig_and_BIB.root` output.

Now we are going to compare the fitted track parameters in the two cases.
A root file with a full signal+BIB reconstruction, a signal only reconstruction and a macro for the analysis has been prepared for you.

In order to run the macro:

```
root -l
.L compare_sig_and_BIB.C
compare_sig_and_BIB()
```

You can play with the cuts on the maximum chi2 (first argument) and on the number of hits (second argument) to clean your track sample. The third argument is used to chose the plot you want to see:

- 0. chi2/ndof
- 1: number of hits
- 2: track pt
- 3: D0
- 4: Z0
- 5: Omega

```
root -l
.L compare_sig_and_BIB.C
compare_sig_and_BIB(5,6,3)
```

In this way a maximum chi2/ndof of 5 and a minimum number of 6 hits is required, and the D0 comparison is shown.

You can also measure the track pT resolution as a function of PT with the track_pt_resolution.C macro. You should simply type:

```
root -l
.L track_pt_resolution.C
track_pt_resolution()
```

In this way you can compare the track pt resolution with and without the BIB.

Advanced topics

Modifying an existing processor

If you want to modify an existing processor you can follow these simple steps:

1. clone the repository of the processor in your working directory. Look into the following repositories:
    a. Official ILCSoft repository
    b. Official Muon Collider repository
2. modify it
3. compile it following these steps:
    a. create a BUILD directory separated from the sources
        i. `mkdir BUILD; cd BUILD`
    b. create the make files:
        i. `cmake -C $ILCSOFT/ILCSoft.cmake -DBOOST_INCLUDEDIR=/usr/include/boost173 -DBOOST_LIBRARYDIR=/usr/lib64/boost173 ../`
    c. compile it as usual (this create a library under the `lib` directory)
        i. `make`
4. modify the MARLIN_DLL variable: you have to remove the old library path and add the new ones (this variable can not store duplicate libraries)

Creating a new processor

A script has been prepared to easily create a new processor, for details see here.

As first step copy the script in your working directory:

```
cp /opt/ilcsoft/v02-01-pre/Marlin/v01-17/examples/copy_new_Processor.sh .
```

Fix the "basic_folder" path in the script at line 9 (check the right Marlin version) and also another bug:

```
sed -i 's/v01-12/v01-17/g' copy_new_Processor.sh

sed -i 's|inform=.*|inform="export MARLIN_DLL=\${MARLIN_DLL}:${DIR}/lib/lib${PROJECTNAME}Processor.so"|g'
copy_new_Processor.sh
```

Run the script with the name of the new processor. `Usage: ./copy_new_Processor.sh [old_processor new_processor].` As you can see this script does a copy of an exist processor, so you have to use the given basic example.

First you need to export a variable with the name of the processor and then call the script as in this example. Note that the final name of the processor should be: `${PROJECTNAME}Processor`

```
export PROJECTNAME=FirstExample

./copy_new_Processor.sh /opt/ilcsoft/v02-01-pre/Marlin/v01-17/examples/mymarlin ${PROJECTNAME}Processor
```

This script creates a directory (in the example FirstExampleProcessor) with the code for a very basic processor. It also modifies your .bashrc file adding the new setting for the `MARLIN_DLL` variable, but as reported if you use another shell, you should set it by yourself!

```
source ~/.bashrc
```

At this point you have a new running processor. You can modify the sources in the FirstExampleProcessor/src directory and then recompile it just giving `make` and `make install` from the FirstExampleProcessor/build directory.

To test it first verify that the library has been recognized, so look for the name of the processor in the output of the command:

```
Marlin -x > xml/mysteer.xml
```

You should find at the begin of the file `mysteer.xml` a line starting with `<!-- Loading shared library :` end ending with the path of the new installed library (in the previous example you should find libFirstExampleProcessor.so).

In this file you find also the basic configuration of the processor:

```
<processor name="MyFirstExampleProcessor" type="FirstExampleProcessor">
<!--FirstExampleProcessor does whatever it does ...-->
<!--Name of the MCParticle collection-->
<parameter name="CollectionName" type="string" lcioInType="MCParticle">MCParticle </parameter>
<!--verbosity level of this processor ("DEBUG0-4,MESSAGE0-4,WARNING0-4,ERROR0-4,SILENT")-->
<!--parameter name="Verbosity" type="string">DEBUG </parameter-->
</processor>
```

Add you processor to the execute section at the beginning of the file:

```
<execute>
<!--processor name="MyEventSelector"/-->
<!--if condition="MyEventSelector"-->
<processor name="MyAIDAProcessor"/>
<processor name="MyTestProcessor"/>
<processor name="MyFirstExampleProcessor"/>
<processor name="MyLCIOOutputProcessor"/>
<!--/if-->
</execute>
```

And set a valid input file for the global parameter `LCIOInputFiles`

If you exec `Marlin xml/mysteer.xml` in the output you should see lines like this one:

```
[ DEBUG "MyFirstExampleProcessor"] processing event: 0 in run: 0
```

Docker container

All the software is distributed through a docker container. Images for the Muon Collider software are available at DockerHub

For further details about the Docker setup please refer to the guide

To exec it after download you can for example the following command:

```
docker run -ti --rm --env DISPLAY=${DISPLAY} --env USER=${USER} --env HOME=/home/${USER} --user=$(id -u
$USER):$(id -g $USER) -v <your working directory>:/home/${USER} -v /cvmfs:/cvmfs -w /home/${USER} -v ${HOME}/.Xaut
hority:/home/${USER}/.Xauthority --net=host --entrypoint /bin/bash infnpd/mucoll-ilc-framework:1.0-centos8
```

First we define some environment variables used to export the graphical interface: `--env DISPLAY=${DISPLAY} --env USER=${USER} --env HOME=/home/${USER}`

We define also the local user as the user logged in the container (otherwise you login as root): `--user=$(id -u $USER):$(id -g $USER)`

We mount some local directories inside the container: `-v <your working directory>:/home/${USER} -v /cvmfs:/cvmfs`

With the next option we define the working directory inside the container: `-w /home/${USER}`

The last two options tell the container to share the host's networking namespace: `-v ${HOME}/.Xauthority:/home/${USER}/.Xauthority --net=host`

Then you need to source the environment setting:

```
source /opt/ilcsoft/v02-01-pre/init_ilcsoft.sh
```