## *GArSoft Notes*

### Machines

There are dunegpvm##.fnal.gov nodes, where ## is anything from 01 to 15. Node 01 tis SLF6; the rest are SLF7. Build on dunebuild01 or dunebuild02, which are faster as they have 16 rather than 4 cores. They are both SLF7.

You need to talk with Kirby & Herner about jobs that need O[106] CPU hour level.  
  
The canonical guidance is that each directory on $DUNEpnfs and $DUNEscr should have less than 2000 files. You’ve actually gotten by with 10,000. By the way, $DUNEscr has 4 petabytes!

About the dang grid… well you have something that sort-of works. Assume NOTHING is mounted and all you can do is copy on & off $DUNEpnfs – and the pnfs disk does weird stuff. To copy off them from a condor node, you need ifdc cp. You can debug scripts that run on the condor nodes by running them in a script with pretty much nothing defined. Ken Herner gave you the following handy options to the jobsub\_submit command for a job that needs more than 2 but less than 3Gig of memory:

jobsub\_submit -N $NJOB -q --group=dune --memory=3000

--resource-provides=usage\_model=DEDICATED,OPPORTUNISTIC,OFFSITE

--lines

='+SingularityImage=\"/cvmfs/singularity.opensciencegrid.org/fermilab/fnal-wn-sl6:latest\"'

--append\_condor\_requirements

='(TARGET.HAS\_SINGULARITY==true||TARGET.IFOS\_INSTALLED=?=\"SL6\")'

file:///dune/app/users/bellanto/GArTPC/work/RunJobs/condor\_job\_script\_20M

This lets jobs go offsite to machines with either SL6 or the Singularity SF6 container. WAIT! Some institutions, FSZ in particular, have some different container and that makes Laura’s flux files invisible.

Setting the memory requirement as low as possible is important, as the worker nodes often have much more than 4 or 3 G and will signal their availability to take another job based on how much memory they have left. The default runtime is 8 hours = 28800 sec, and you can revive jobs that go into hold at that point with jobsub\_release; but in fact those jobs are probably hosed anyway in all likelihood. May as well jobsub\_rm ‘em and restart. This is a very unreliable system, it seems.

In Jan 2020 Collaboration meeting, Ken shows slides re a “grace” revival feature. I copy from his slides:

* Not enabled by default; the following job attributes must be set: +FERMIHTC\_AutoRelease=True   
  # +FERMIHTC\_GraceMemory: This defines the amount of   
  # memory in MB that will be ADDED to the original one  
  # For example, 1024 MBs  
  +FERMIHTC\_GraceMemory=1024  
  # +FERMIHTC\_GraceLifetime: This defines the amount of   
  # walltime (in seconds) that will be ADDED to the   
  # original one # For example, 10 minutes (600 seconds) +FERMIHTC\_GraceLifetime=600
* Set these values with the -l or --lines options to jobsub  
  [submit]  
  lines\_0=+FERMIHTC\_AutoRelease=True  
  lines\_1=+FERMIHTC\_GraceMemory=1024  
  lines\_2=+FERMIHTC\_GraceLifetime=600  
  lines\_3,4,5...=other\_stuff\_you\_usually\_put

From trying to get jobsub to run in the MINERvA framework, I have the following notes: “/minerva/app/users/bellanto (which contains my cmtuser area) is visible as read-only to the gridjob; and one can spool stuff off to the /pnfs area after the job ends, using the –d <tag> <dir> option to the jobsub\_submit command. That option creates a symbol $CONDOR\_DIR\_<tag> which you can use in the scripts that are executed to specify a local area that the job uses; and then, after the job exits, another machine copies files off that local area to /pnfs. The inverse operation, to get files into the Fermigrid jobs, seems to involve the –f <file> option.”

Debugging: You’ve got allinia (which is also arm forge) and an alias setup in GArSoftsetup.sh. Try e.g. DDTart -n 10 -o dbgen.root -c prodgenie.fcl and ignore the message about the old version. Also ignore the message about not finding the source file for *art*; set your breakpoints by source file name and line number. Finally, ignore messages to the effect that the source is newer than the executable; because the install step moved the executable, the source is always newer than the (wrong) executable that allinia gets its date from. Now you might ask “Which version of the source file name?”. Tom says if the source file was moved to the build area it should be found already by allinia. Oh! Look! valgrind appears to be available on these nodes via ups!

Tom Junk found out how to run gdb on the batch worker to get a stack trace. His example is for running a readoutsim step:

Instead of just running the readoutsimjob step, do this:

setup gdb

gdb --args art -c readoutsimjob.fcl -o readoutsim\_0025.root genie\_gen\_0025.root <<EOF

r

bt

quit

EOF

To detach processes running interactively, use the screen command:

screen spawn a sorta subprocess in this terminal.

screen –list see what screens exist & their state.

screen –d pid.tty.host detach screen from the terminal it is in.

screen –r reattach the screen into this terminal.

### About art

To understand the expansion of a fcl file, try fhicl-expand, which will use $FHICL\_FILE\_PATH to locate the file. Also, there is the handy art --debug-config. Help at art --help is reasonably useful. And! config\_dumper will give you the fcl file contents of the art modules that created an art event data file such as reco.root, the output of the reconstruction art job.

In a ---.fcl file, the term @local:: means “In this file, after all the #includes have been expanded”. So for example, if prodgenie.fcl contains a line

#include “Geometry.fcl”

and Geometry.fcl includes a parameter named standard\_geo, as in

standard\_geo:

{

ForceUseFCLOnly: false

.

.

.

}

then when prodgenie.fcl says

services:

{

Geometry: @local::standard\_geo

.

.

.

}

that means that prodgenie.fcl has ForceUseFCLOnly defined for it.

Evidently, if you create a new fcl file in the srcs/garsoft area, and then build, that fcl file gets copied to the appropriate subdirectories in the local products and build areas. It is the one in the build area that gets taken by an art job by default if you don’t have a fcl file of that name in the directory from which you launch art. You have $FHiCL defined to point to that build area.

cetskelgen is a device to automatically make art modules, which are also called plugins. In art, a “module” is a dynamic library containing classes somehow. They call services and mostly interact with events; they never invoke each other. They have methods such as beginJob, endJob, beginRun, endSubrun etc. There is a method to process an event, called analyze(art::Event const& event). There are 5 types: analyzer, producer and filter modules, deriving from art/Framework/Core/EDAnalyzer, EDProducer and EDFilter; and then there are source and output modules.

About art data products: the TTree Events in a root file containing art event data has TBranches with names of the form <data type>\_<fhicl file module label>\_\_<process name>. (Unless there are multiple instances of the same process). For an art::Assn, the <data type> is the names of the two data products, concatenated, even if the association carries a 3rd data element with it, followed by a concatenated voidart::Assns.

These are large files – about 1M per event at this point – and they get larger as generation goes through the steps of GENIE/Helper, readout simulation and reconstruction as all these tasks are art producer modules; they add art data products to the ::art::Event but cannot remove anything. But you cannot rm the intermediate files and just keep the reco job output ☹.

When creating a class that will be an art data product, you need to modify the classes.h to include the new header. Then in the classes\_def.xml file, add 3 lines such as

<class name="std::vector < std::pair<float,float> > "/>

<class name="gar::rec::TrackIoniz" ClassVersion="10" >

</class>

The 1st line is there because I made a vector of pairs as a data member of this class, so ROOT has to make a dictionary entry for that.

And then, lower down, add something like

<class name="std::vector<gar::rec::TrackIoniz >” />

and

<class name="art::Wrapper< std::vector<gar::rec::TrackIoniz > >" />

Finally, a set of six lines for any associations that will be needed. E.g.

<class name="std::pair< art::Ptr<gar::rec::TrackIoniz>, art::Ptr<gar::rec::Track> >" />

<class name="std::pair< art::Ptr<gar::rec::Track>, art::Ptr<gar::rec::TrackIoniz> >" />

<class name="art::Assns<gar::rec::TrackIoniz, gar::rec::Track>" />

<class name="art::Assns<gar::rec::Track, gar::rec::TrackIoniz>" />

<class name="art::Wrapper<art::Assns<gar::rec::TrackIoniz, gar::rec::Track> >" />

<class name="art::Wrapper<art::Assns<gar::rec::Track, gar::rec::TrackIoniz> >" />

But wait. There is more. Build with mrb i -j4, or some other j. Watch the build fail, saying

INFO: adding version info for class 'gar::rec::TrackIoniz':<version ClassVersion="10" checksum="2108826658"/>

WARNING: classes\_def.xml files have been updated: rebuild dictionaries.

make[2]: \*\*\* [garsoft/ReconstructionDataProducts/ReconstructionDataProducts\_dict\_checked] Error 2

make[1]: \*\*\* [garsoft/ReconstructionDataProducts/CMakeFiles/checkClassVersion\_ReconstructionDataProducts.dir/all] Error 2

make: \*\*\* [all] Error 2

Then build again. What will happen on the 2nd build is that a field of the form <version ClassVersion="10" checksum="2108826658"/> will be put into the <class name="gar::rec::TrackIoniz" ClassVersion="10" >, appearing between that <class …> and the following </class> delimiters.

To prevent a field named IDnumber that exists in a data products class from appearing on the output data product, put <field name=”IDnumber” transient=”true”/> at the same level as the <version … /> xml nodes.

Here is an example of what has to go into classes\_def.xml if you want to build a 3-way association; in this case between a gar::rec::Vertex and a gar::rec::Track, with a gar::rec::TrackEnd as the extra data. TrackEnd is a typedef to int, but the dictionary-maker does not know that.

<class name="std::pair< art::Ptr<gar::rec::Vertex>,

art::Ptr<gar::rec::Track> >" />

<class name="std::pair< art::Ptr<gar::rec::Track>,

art::Ptr<gar::rec::Vertex> >" />

<class name="art::Assns<gar::rec::Vertex, gar::rec::Track>" />

<class name="art::Assns<gar::rec::Track, gar::rec::Vertex>" />

<class name="art::Wrapper<art::Assns<gar::rec::Vertex,

gar::rec::Track> >" />

<class name="art::Wrapper<art::Assns<gar::rec::Track,

gar::rec::Vertex> >" />

<class name="art::Assns<gar::rec::Vertex, gar::rec::Track,

gar::rec::TrackEnd>" />

<class name="art::Assns<gar::rec::Track, gar::rec::Vertex,

gar::rec::TrackEnd>" />

<class name="art::Wrapper<art::Assns<gar::rec::Vertex,

gar::rec::Track, gar::rec::TrackEnd> >" />

<class name="art::Wrapper<art::Assns<gar::rec::Track,

gar::rec::Vertex, gar::rec::TrackEnd> >" />

<class name="std::pair< art::Ptr<gar::rec::Vertex>, art::Ptr<gar::rec::Track> >" />

<class name="std::pair< art::Ptr<gar::rec::Track>, art::Ptr<gar::rec::Vertex> >" />

<class name="art::Assns<gar::rec::Vertex, gar::rec::Track>" />

<class name="art::Assns<gar::rec::Track, gar::rec::Vertex>" />

<class name="art::Wrapper<art::Assns<gar::rec::Vertex, gar::rec::Track> >" />

<class name="art::Wrapper<art::Assns<gar::rec::Track, gar::rec::Vertex> >" />

<class name="art::Assns<gar::rec::Vertex, gar::rec::Track, bool>" />

<class name="art::Assns<gar::rec::Track, gar::rec::Vertex, bool>" />

<class name="art::Wrapper<art::Assns<gar::rec::Vertex, gar::rec::Track, bool> >" />

<class name="art::Wrapper<art::Assns<gar::rec::Track, gar::rec::Vertex, bool> >" />

<class name="art::Wrapper<art::Assns<gar::rec::Track, gar::rec::Vertex, bool> >" />

### The build system

When doing a fresh install, one has to know that all the dependent products exist in the correct COMPILER:BUILDTYPE combo. For example, when I created an area in Nov 2018, I could use e15:debug; but by Jan 2019, we had moved to art v3\_00\_00 which only exists in e17:debug. Or one might use e17:prof, because e17:opt does not exist! Evidently all builds are -pg as there is no speed penalty there.

Some useful environmental names:

LD\_LIBRARY\_PATH: the load library path; where art looks for modules.

FHICL\_FILE\_PATH: where art looks for fcl files.

PRODUCTS: where ups searches for products, including your local ones.

FW\_SEARCH\_PATH: for finding frameworks. Whatever they are.

NUTOOLS\_DIR: There’s probably something useful in there.

MRB\_PROJECT, MRB\_TOP, etc: From the install step.

Although you might think that everything is being built based on what you see on the screen, in fact mrb uses CMake uses make, and that is not the case. Most of those message are telling you that mrb realized it does not have to do anything for this, that or the other part of the system.

The mrb command actually does the build. The mrbsetenv command, which you have in GArSoftsetup.sh, sets up this area to build a local, you-are-the-only-customer ups product called garsoft. The mrbslp command then does a ups setup command for this product.

To “make clean”, so to speak, use mrb z. This will delete the contents of $MRB\_BUILDDIR, and you may also mrb zd to delete the localProducts distribution too, which is necessary so you won't set up the old one when you log in and build a new one.

What happens under the hood here is that your local copy of the git repository is in …/srcs/garsoft and mrb builds in a temporary directory of sorts, which is the …/build\_slf6.x86\_64 directory. If that is successful, it installs by copying the right bits and pieces to the area which serves as the definition of your own, personal ups products, …/localProducts\_garsoft\_develop\_e17\_debug.

Perforce, the files you would want to edit are your copy of the git repository, the …/srcs/garsoft area. If you need a source file for the debugger, you could take it from the localProducts area, if it is there; or from the …/srcs/garsoft area because it would be the same as long as the build worked. Tom says that allinia should be able to fine the source from the build area, so if you get an error message saying “no source” then there is no point in looking for the source code in the build area.

Perforce again, in …/srcs/garsoft there is a directory ups, which contains information about the ups information for this you-are-the-only-customer product. For example, …/srcs/garsoft/ups/product\_deps tells me that building GArSoft depends on v2.12.10 of genie\_xsec and genie\_phyopt.

CMake: In our CMakeLists.txt files, we use the art\_make macro. The LIBRARY\_NAME is the name of the pair of libraries that you are making. The MODULE\_LIBRARIES are what you link to in order to make the library of art modules. The LIB\_LIBRARIES are what you link to in order to make the library of all the other stuff in that directory. The MODULE\_LIBRARIES should refer to the very same library name, meaning that the MODULE\_LIBRARIES link to the LIB\_LIBRARIES in the same area. There is only one LIBRARY\_NAME field, at most, in each CMakeLists.txt file, so there must be some convention about the names such that the 2 are related and there is some knowable name if there is not a LIBRARY\_NAME field.

If there is no module, tool, service or whatever in a library area, then you do not get told that there is no --.so file created. Instead, if some other library has a declared dependency on that area, it tells you that it could not find that library.

### Baffling git shit

⋅ Todefine which repository I am accessing,

cd $GArSRC

git remote set-url origin ssh://p-garsoft@cdcvs.fnal.gov/cvs/projects/garsoft-garsoft

⋅ To be prudent!

git diff file

– to compare your staging area (red) against your working

area (green) using more or less. With any luck, your

staging area matches your repository. Might match the

remote repository, might not.

git status – to compare staging area against local area, working area

against staging area, and to find out what files are not being

gitted.

git commit -a --dry-run

git fetch; git merge

– this is the same as git pull

When doing diff in redmine, it seems that the new one is the version marked in green.

To remove a file not just from your local area but also from the index file and hence, after commit, from the repository,

git rm file

If you have moved a file from one location to another in the directory tree, you need to (perhaps) git add the new directory, git add the file in the new location, and git rm the file from the old location.

To push into the repository, first cd $GArSRC. Then do a git status to figure out what you need to do, then git add to the staging area, then git commit to the local repository, then git push to the project repository:  
 cd $GArSRC  
 git status  
 git add <filenames>

git commit -m "Add a comment"

git pull

git push <remote> <branch>

Where I don’t actually need to do <remote> <branch> - presumably because my develop branch is a “tracking branch” and so knows that I am talking to origin develop. The -m comment needs a double quote. There is an error message about email not being sent, which is just fine. One can skip the git add by using -a in the git command step. The git pull is there on Tom’s advice, to be super-certain that nobody has messed with the repository since you pulled this version; that step will throw an error of some sort if that has happened. It might say:

CONFLICT (content): Merge conflict in < filename >

Automatic merge failed; fix conflicts and then commit the result.

In which case, go into that file and look for <<<. The 1st part, labeled HEAD is your code, and the 2nd part, between the === and the hash code is what’s in the repository. Fix it up and then redo the git add and git commit steps. Then do the git push.

At the end of the day you might not feel like pushing to the repository. Do MyBack and git status. Because you only change your local repository when you push to the remote one, git status is a list of the files that you have changed since your last push. It is of course, not a list of the files that other people have changed and pushed to the remote repository recently. Then, if a git pull changes one of these files, you’ll see that in the message from the git pull and have the old version on hand.

At the end of it all, look at the ‘revisions’ section of the repository from the GArSoft Redmine page. It takes a while to be updated; 30 minutes in one case from my git push to it being fully updated.

The error message

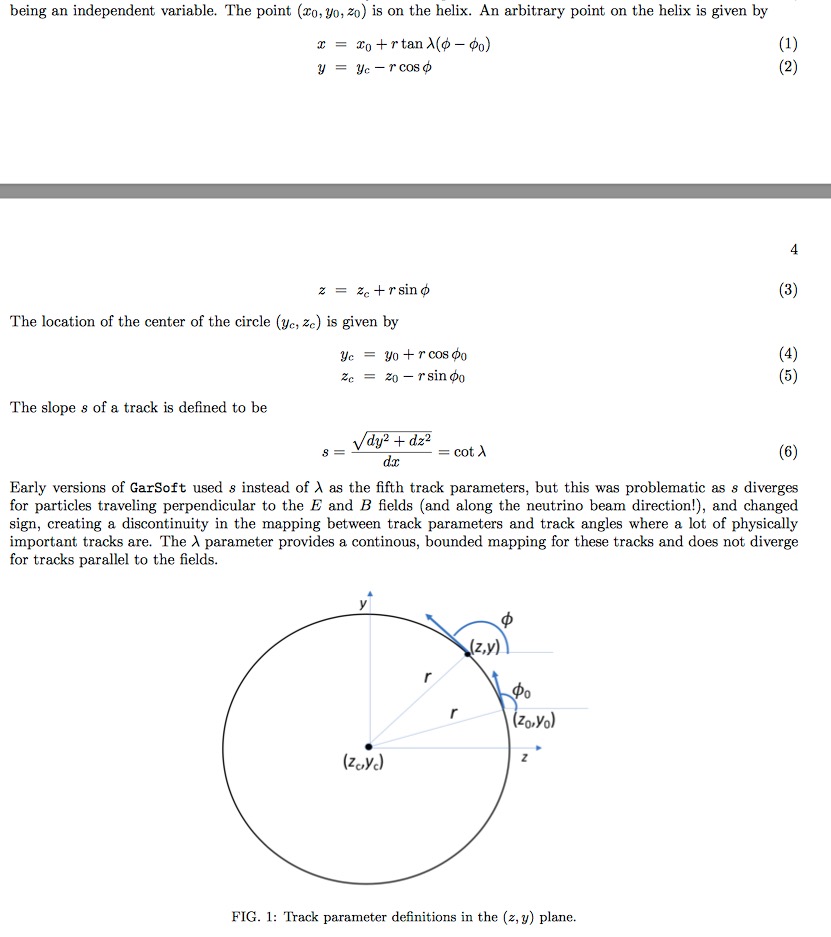
fatal: Not a git repository (or any parent up to mount point /dune)

Stopping at filesystem boundary (GIT\_DISCOVERY\_ACROSS\_FILESYSTEM not set).

is due to not being in $GArSRC when you type your git command.

### Coordinates, conventions:

Right handed coordinate system, z is the direction of the beam, y is straight up. The magnetic field is in the x direction: .



The track parameters are said to be: (*y*0, *z*0, *ρ*, *ϕ*0, *λ*; *x*) where the order reflects their appearance in the code; (*x*, *y*0, *z*0) are a point on the helix (not a center as determined at drift position *x*) and are in cm; *ρ* = 1/*r* is curvature of the track at that point in cm⁻¹, *ϕ*0 is the parameterizing angle at (*x*, *y*0, *z*0) as seen in the transverse plane w.r.t. the *z* axis, oriented to be 0 for a track in the *z* direction and +π/2 for a track in the *y* direction in the transverse plane, and λ is the angle of the track with respect to the transverse plane (equivalently, the angle with respect to the HV membrane). The sign of *ρ* is the same as the sign of *r*, and it does *not* give the charge; that also needs some concept of *dϕ*/*dt*, i.e. which end of the track the particle started at and which end it ended at. (But a track from −*z* to +*z* that curves upwards has positive charge.) The *x* parameter is a separate variable not in the track parameters, i.e. the full set at the beginning of the track is fTrackParametersBegin[5], fXBeg. The forward fit consumes TPCClusters in order of increasing *x*; the backwards fit, in order of decreasing *x*.

The notation for the Kalman filter is from the 2 Wikipedia articles on them, for which there are copies in the same directory as this file.

From the GENIE manual, it uses “ℏ = c = 1, so almost every simulated quantity is expressed in powers of [GeV]. Exceptions are the event vertex in the detector coordinate system (in SI units) and particle positions in the hit nucleus coordinate system (in fm).” In GENIE, there is a variable for the position of the initial nucleus in the lab frame and then subsequent particle/vertex locations are relative to this location. However, when GENIE’s output, GHepRecord, is converted into an *art* structure, the positions in the lab frame are needed. In cm and ns, I guess.

GEANT uses MeV as default units but for some reason LArSoft, and of a consequence, GArSoft, prefers GeV.

### The event display:

About Tom’s event display: art -c Toms\_evd.fcl <reco.root file>.

CNTL SHIFT + and CNTL - to get bigger and smaller. Rodent will rotate, no pan yet. Reco colors are assigned randomly and reco tracks have a little arrow on their end to indicate direction of some sort. For MC Truth tracks, light dashed blue is a ν, solid purple is a proton, red for an electron, blue for a muon, magenta for a pion (?) and little dots for a neutron or photon. Yellow dots are vertices. Green octohedrals are entire ECAL clusters.

### What is the geometry, anyway?

The geometry, as of mid-2019, is in …/srcs/garsoft/Geometry/gdml/ND\_Strawman\_Concept\_v09.gdml.

This geometry is now obsolete, but some of the following is still useful for the various geometries now being studied.

The Cu is 2mm thick, and the scintillator is 5mm thick; from the C/H ratio it looks like polystyrene. The first 8 layers also have 1mm thick layers of FR4 epoxy-fiberglass. These are for the pad readout regions, and the rest of the device is strip readout. Some stats:

2mm of Cu is 0.1393 of an X0; 5mm of polystyrene are 0.0121of an X0. Then X0 of the ECAL is 6.61cm. For λHAD,, the numbers are 0.0132 λ, 0.0065 λ and 51.17cm.

MIP loss in passing through ECAL: 0.2cm(12.57 MeV/cm) + 0.5cm(2.052 MeV/cm) = 3.54 MeV per layer; and for 80 layers, 283 MeV.

Density: Cu is 8.96 g/cm3; polystyrene is 1.06 g/cm3. Areal density per layer is thus ( 0.2cm(8.96g/cm3) + 0.5cm(1.06g/cm3) ) = (1.792 g/cm2 + 0.530 g/cm2) = 2.322 g/cm2. That is by weight 772m~~º~~ Cu, and by volume 286m~~º~~ Cu.

Volume: The inner octagon of the barrel has apothem a = 278.45 cm, corresponding to a side s = 230.68 cm, and a length of ℓ = 2(385.0 cm) = 770 cm. The area of an octagon is 2(1 + √2) s2.

Figure Cu, polystyrene and Ar have the same cross sections per nucleon ( equivalently, per ton) for the following: 422 metric tons times [1.50, 0.50] x106 [CC, NC] events per standard year is [633, 211] x106 / (56% 31.6 x106) = [35.8, 11.9] [CC, NC] events per spill.

Pad sizes and ROC sizes for the TPC:

The central ROC has 6 × 6 mm pads, arranged in a rectangular layout. The inner ROC has 4 × 7.5 mm pads, arranged in a radial way, and with trapezoidal deformation to project back to the center of the TPC as one moves out towards the sector boundaries. For the inner OROC, 6 × 10 mm, and for the outer OROC 6 × 15 mm. NOMINAL radii of these different ROCs are available as Geometry methods GetIROCInnerRadius(), GetIROCOuterRadius(), GetOROCInnerRadius(), GetOROCOuterRadius() and the inner/outer OROC boundary is at GetOROCPadHeightChangeRadius().

### About GENIE and its little Helper

GENIEGen\_module.cc is a thin layer over the nutools class evgb::GENIEHelper. It is an art producer module, i.e. its interface conforms to the demands of art/Framework/Core/EDProducer.h. GENIEHelper is a not-so-thin layer over GENIE itself which calls the generator event by event in GENIEHelper::Sample(…). Also in that method, GENIEHelper calls PackMCTruth(…) and PackGTruth(…) to unpack the genie::EventRecord into simb::MCTruth and simb::GTruth objects that live in GENIEGen::produce(::art::Event& evt). Then, in some way still obscure to me, GENIEGen::produce(::art::Event& evt) will put vectors of this info into the large root file that it produces. The small file that it produces contains histograms made in GENIEGen::beginJob() and filled in GENIEGen::produce(::art::Event& evt).

UserPhysicsOptions.xml is in a ups product called genie\_xsec; there is also a ups product called genie\_phyopt as well. These are not in $LD\_LIBRARY\_PATH,probably because they contain data of various sorts needed by GENIE, but not any executable libraries. Specifically, UserPhysicsOptions.xml exists in 2 locations from /cvmfs/larsoft…genie\_xsec/v2\_12\_10/NULL. One is DefaultPlusMECWithNC/data and the other is DefaultPlusValenciaMEC/data. The selection between these two directories to get UserPhysicsOptions.xml appears to be made with the environmental variable$GXMLPATH although the $GENIEXSECPATH and $GENIE/config (for v2.12.10) seem to be relevant somehow.

In $GENIE/config are the xml files that control what GENIE does. They are documented in Gabe & Tomasz’ MINERvA 101 talks in 2014 and 2015. If, in Tom’s GENIEGen.fcl, you change the parameter EventGeneratorList, this changes what process GENIE looks up in $GENIE/config/EventGeneratorListAssembler.xml – good options are Default (for a reasonable mix of all interactions), CCCOH for charged-current coherent π± production, and NCCOH for neutral-current coherent π? production.

Evidently, flux files are copied by GENIEHelper to /var/tmp/ifdh\_6200\_### where ### is the process pid. Now, /var/tmp/ has a garbage collection process that collects items that are not garbage, because if the system runs out of space there it stops the system. It seems that a cet::exception& gets thrown - I’m not sure where because I don’t think root knows about cet::exception - and gets caught in art::Worker::runWorker, at line 593.  (v3.0.0 of *art*).  In the 2015 art/LArSoft course materials, I read that *art* will attempt a graceful shutdown. (<https://indico.fnal.gov/event/9928/session/2/material/3/2.pdf>, pg 9) In principle. In practice, the output files are not closed correctly and the effort in the job is lost.

This is a problem when running GENIE to generate even a modest number of CCCOH events; I am seeing that this particular type of event is produced at the rate of one every 12, 13 seconds on dunegpvm07 for example. However, I try changing the field FluxCopyMethod from IFDH to DIRECT in GENIECCOH.fcl – and this does seem to do the trick. No, it doesn’t. FluxCopyMethod DIRECT means that you read directly off Laura’s pnfs space and there is a possibility of the process going into a “D” state, meaning uninterruptible sleep because it’s pnfs!