Fermilab

Managed by Fermi Research Alliance, LLC for the U.S. Department of Energy Office of Science

Introduction to LArSoft code and work environment

Saba Sehrish

art/LArSoft Course

(08/03/15-08/07/15)

Goals

- Goals of this session:
  - Learn about
    - LArSoft repositories and code organization
    - mrb
    - LArSoft work environment
    - how to get, build and run LArSoft code
  - Checkout, build, run AnalysisExample in larexamples
- Ultimate goal is to contribute code to LArSoft

LArSoft Repositories and Description

LArSoft code lives in a set of git repositories hosted at Fermilab

Name	Description
larcore	Low level utilities and functions e.g. Geometry services
lardata	Data products and other common data structures
larevt	Low level algorithm code that use data products
larsim	Simulation code
larreco	Primary reconstruction
larana	Secondary reconstruction/analysis e.g. PID
lareventdisplay	LArSoft based event display
larpandora	LArSoft interface to the pandora reconstruction package
larexamples	Placeholder for examples

1. All are publically accessible at: http://cdcvs.fnal.gov/projects/<repository name>
2. For read/write access: ssh://p-<repository name>@cdcvs.fnal.gov/cvs/projects/<repository name> (requires valid kerberos ticket)

LArSoft Products

The build procedure creates and installs a product from each repository

Product	Description
larcore	Low level utilities and functions e.g. Geometry services
lardata	Data products and other common data structures
larevt	Low level algorithm code that use data products
larsim	Simulation code
larreco	Primary reconstruction
larana	Secondary reconstruction/analysis e.g. PID
lareventdisplay	LArSoft based event display
larpandora	LArSoft interface to the pandora reconstruction package
larexamples	Placeholder for examples

Each product is entirely self-contained, aside from dependencies

larsoft Product

- A LArSoft release is a consistent set of LArSoft products built from tagged versions of code in the repositories
  - Implicitly includes corresponding versions of all external dependencies used to build it.
  - Each release of LArSoft has a release page on Scisoft http://scisoft.fnal.gov/scisoft/bundles/larsoft/<version>/larsoft-<version>.html
- `larsoft`
  - A larsoft umbrella product binds it all together to give it one version, one setup command:

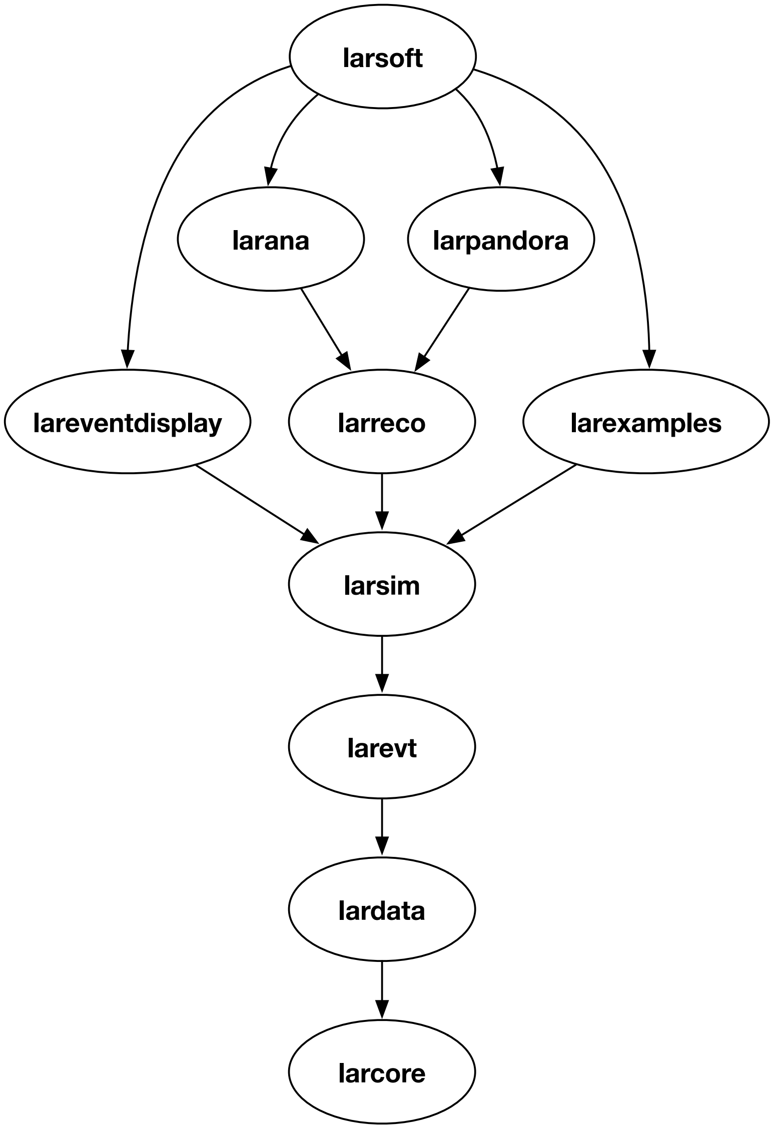
    ```
    setup larsoft v04_18_00 -q …
    ```
- `larsoft_data`
  - A ups product (not a repository)
  - a place for large configuration files

larsoft v04.16.00

Product	Version
larcore	v04.13.00
lardata	v04.11.00
larevt	v04.08.06
larsim	v04.08.03
larreco	v04.12.00
larana	v04.08.00
lareventdisplay	v04.06.00
larpandora	v04.04.16
larexamples	v04.04.16
larsoft_data	v0.04.00

https://cdcvs.fnal.gov/redmine/projects/larsoft/wiki/LArSoft_release_list

LArSoft Products Dependencies



How to look for a LArSoft product version in a specific LArSoft release?

- List of LArSoft products (after you setup `larsoft`)
  - `ups active`
    - Will show you all the products that are setup including `larsoft` and other products that got setup
    - You can look at the `larsoft` products and version numbers
- List of `larsoft` dependencies (`larsoft` doesn't need to be setup)
  - `ups depend larsoft -v <version> -q <qualifiers>`
- http://scisoft.fnal.gov/
  - LArSoft Distribution
    - Version
      - Manifest e.g. http://scisoft.fnal.gov/scisoft/bundles/larsoft/v04_18_00/manifest/

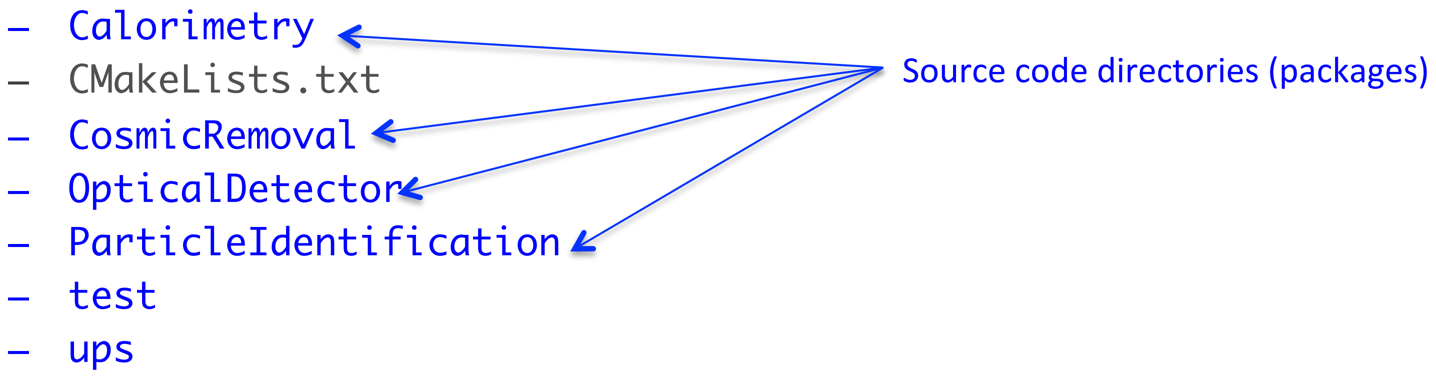
LArSoft Distributions

- LArSoft releases and nightly builds are uploaded to
  1. `/grid/fermiapp/products/larsoft`
     - Can be used on fermigrid
     - Can't run jobs on non-fermigrid OSG nodes
     - Normally used in development setup
  2. `/cvmfs/fermilab.opensciencegrid.org/products/larsoft/`
     - CernVM File System (cvmfs)
     - An http-based, network file system
       - Appears to applications as a file system mounted on a local disk
       - A repository server that store copies of all files to be distributed OSG maintains one at Indiana for use with OSG grid sites (Oasis)
         - Files written to a single repository
         - Can be read in "100,000's of locations"
     - Can run jobs on all OSG nodes (Eurogrid as well)

Code organization within a repository

- Each repository has a similar organization:
  - A number of source code directories => packages
  - `test` – unit and integration tests organized by source directory
  - `ups` – config files, dependency lists, …
  - `CMakeLists.txt`

- For example: `ls -l larana`

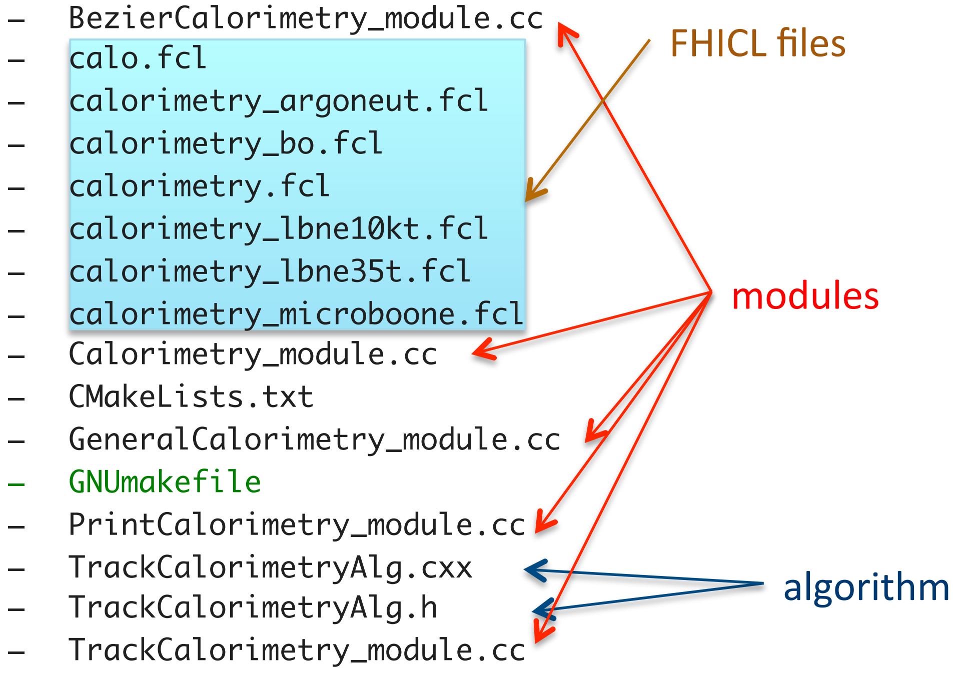


- Calorimetry
- CMakeLists.txt
- CosmicRemoval
- OpticalDetector
- ParticleIdentification
- test
- ups

Source code directories (packages)

Inside a Package Directory

- Source code files
  - Headers and sources
    - Modules, Algorithms, Services
- CMakeLists.txt
- FHICL files – configuration language used by *art*
- For example, `ls -l Calorimetry`



- BezierCalorimetry_module.cc
- calo.fcl
- calorimetry_argoneut.fcl
- calorimetry_bo.fcl
- calorimetry.fcl
- calorimetry_lbne10kt.fcl
- calorimetry_lbne35t.fcl
- calorimetry_microboone.fcl
- Calorimetry_module.cc
- CMakeLists.txt
- GeneralCalorimetry_module.cc
- GNUmakefile
- PrintCalorimetry_module.cc
- TrackCalorimetryAlg.cxx
- TrackCalorimetryAlg.h
- TrackCalorimetry_module.cc

`CMakeLists.txt`

- `CMakeLists.txt`
  - The file `CMakeLists.txt` is the file used by the build system (`cmake`) to learn what steps it should do.
  - There is a `CMakeLists.txt` in every directory/subdirectory; each contains additional instructions for the build system.
- The top level `CMakeLists.txt includes`
  - minimum version of `cmake`
  - `project()` for `mrb`
  - `include()` for additional macros
  - `find_ups_product()` for external dependencies
    - Checks if the product with at least the specified version is setup
  - `add_subdirectory()` for all the subdirectories

CMakeLists.txt

- In the `CMakeLists.txt` of subdirectories
  - Use `art_make` to build all the modules, all the services and one shared library
  - Use `simple_plugin` to build modules and services with different set of dependencies
  - Use the following to install headers, fhicl and sources

    ```
    install_headers()
    install_fhicl()
    install_source()
    ```

    Several LArSoft packages also use

    ```
    install_scripts()
    install_gdml()
    ```

Example `CMakeLists.txt`

```
# use cmake 2.8 or later
cmake_minimum_required (VERSION 2.8)
project(larana)
…
find_ups_product( larcore v1_00_00 )
…
# macros for dictionary and simple_plugin
include(ArtDictionary)
include(ArtMake)
include(BuildPlugins)
# source
add_subdirectory(Calorimetry)
add_subdirectory(OpticalDetector)  …

# tests   & ups
add_subdirectory(test)

# ups - table and config files
add_subdirectory(ups)

# packaging utility
include(UseCPack)
```

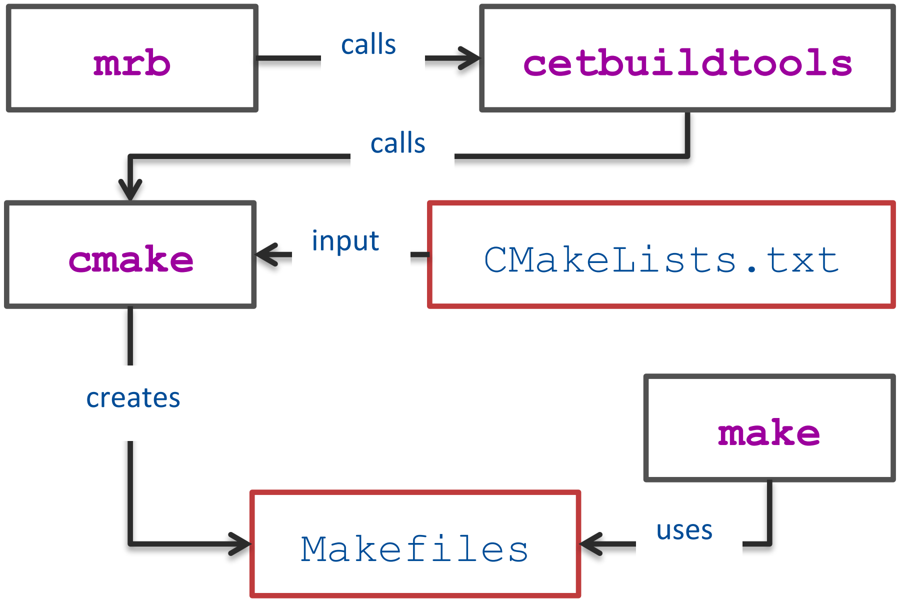
`CMakeLists.txt` of `larana`

```
include_directories ( ${PROJECT_SOURCE_DIR} )
art_make( BASENAME_ONLY
          LIBRARY_NAME Calorimetry
          LIB_LIBRARIES Filters
                        RecoBase
                        Geometry
                        Geometry_service
…
                        ${MF_MESSAGELOGGER}
                        ${MF_UTILITIES}
                        …
MODULE_LIBRARIES
                        Calorimetry
                        Filters ...
                        ${MF_UTILITIES}
                        ${FHICLCPP}
                        ${CETLIB}
                        ${ROOT_BASIC_LIB_LIST}
         )
install_headers()
install_fhicl()
install_source()
```

`CMakeLists.txt` of `larana/Calorimetry`

Build Systems Review

- `make` is the standard build tool that determines dependencies, build order, and issues the commands.
  - `make` uses `Makefile(s)` for configuration and construction.
- `cmake` is a tool with a simpler configuration language that will write all of the `Makefile(s)` for us.
- `cetbuildtools` are convenience macros for `cmake` (used by *art* framework).
- `mrb` for convenience to simplify the building of multiple products pulled from separate repositories.



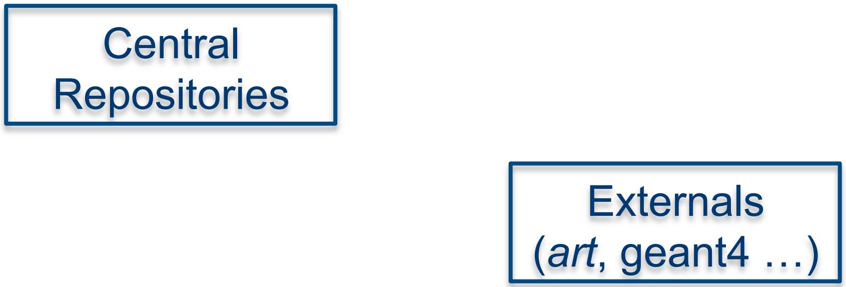
mrb – Quick Guide

- `mrb` – multi-repository build System
- The purpose is to simplify the building of multiple products pulled from separate repositories
- `setup mrb`
- `mrb --help / mrb -h` will display a list of all commands that are available with a brief description
- `mrb <command> --help` will display help on a particular mrb command, e.g. `mrb newDev -h`

mrb -h

```
Usage /products/larsoft/mrb/v1_04_05/bin/mrb (newDev | gitCheckout | svnCheckout | mrbsetenv |
build | install | test | makePackage | mrbslp |
           zapBuild | newProduct | changelog | updateDepsCM | updateDepsPV | checkDeps | pullDeps
| makeDeps ) [-h for help]"

  Tools ( for help on tool, do "/products/larsoft/mrb/v1_04_05/bin/mrb <tool> -h" )

   newDev (n)                  Start a new development area
   gitCheckout (g)             Clone a git repository
   svnCheckout (svn)           Checkout from a svn repository
   build (b)                   Run buildtool
   install (i)                 Run buildtool with install
   test (t)                    Run buildtool with tests
   makePackage (mp)            Make distribution tarballs
   zapBuild (z)                Delete everything in your build area
   newProduct (p)              Create a new product from scratch
   changelog (c)               Display a changelog for a package
   updateDepsCM (uc)           Update the master CMakeLists.txt file
   updateDepsPV (uv)           Update a product version in product_deps
   updateSource                Update all svn or git code in MRB_SOURCE
   makeDeps (md)               Build or update a header level dependency list
   checkDeps (cd)              Check for missing build packages
   pullDeps (pd)               Pull missing build packages into MRB_SOURCE

  Aliases ( we use aliases for these commands because they must be sourced )

   mrbsetenv                   Setup a development environment
                               (source $MRB_DIR/bin/mrbSetEnv)
   mrbslp                      Setup all products installed in the working localProducts_XXX
directory
                               (source $MRB_DIR/bin/setup_local_products)
```

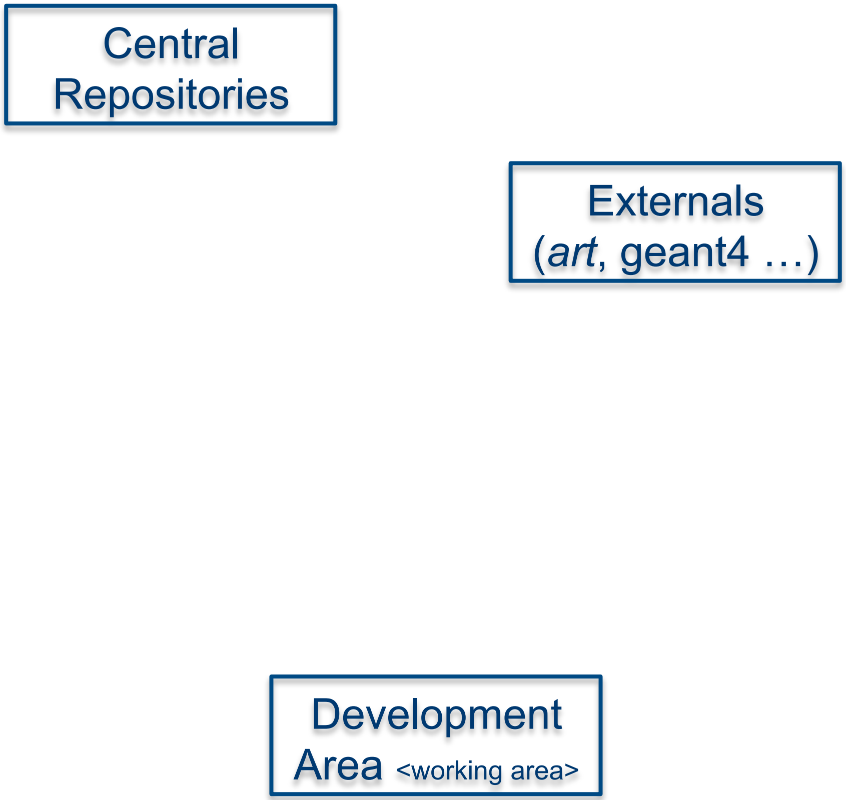
LArSoft Work Environment



Central Repositories

Externals (*art*, geant4 …)

LArSoft Work Environment

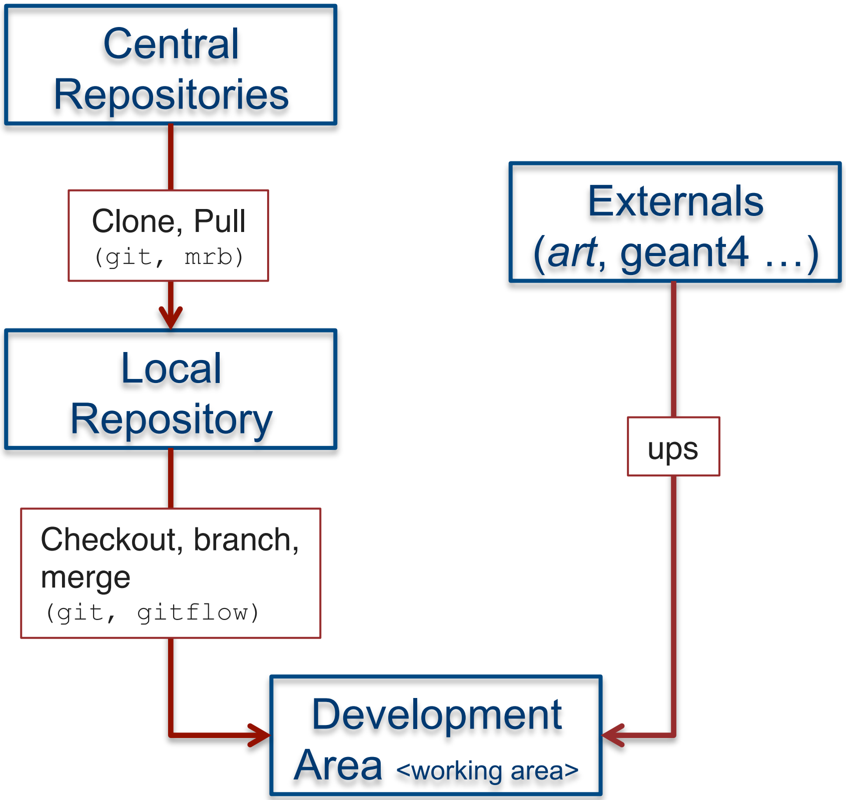


Central Repositories

Externals (*art*, geant4 …)

Development Area <working area>

LArSoft Work Environment


Central Repositories

Clone, Pull
`(git, mrb)`

Local Repository

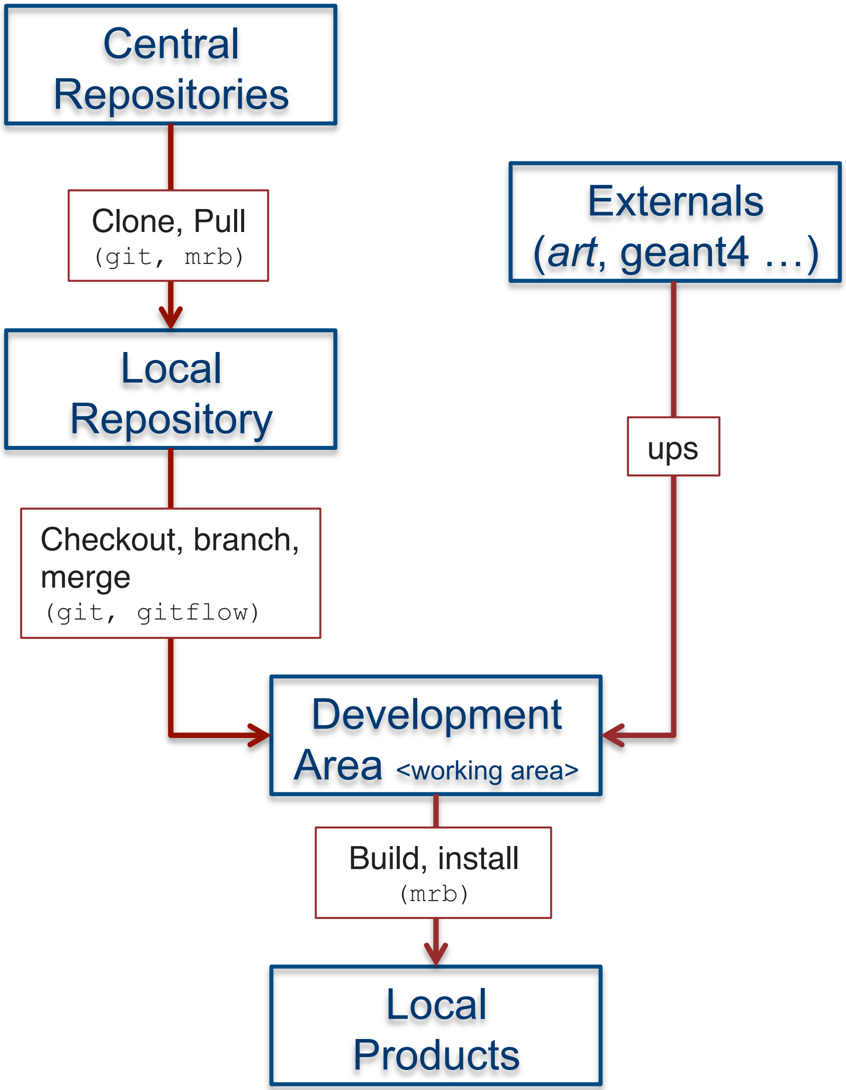
Checkout, branch, merge
`(git, gitflow)`

Externals (*art*, geant4 …)

ups

Development Area <working area>

LArSoft Work Environment



Central Repositories

Clone, Pull
(git, mrb)

Local Repository

Checkout, branch, merge
(git, gitflow)

Externals
(*art*, geant4 …)

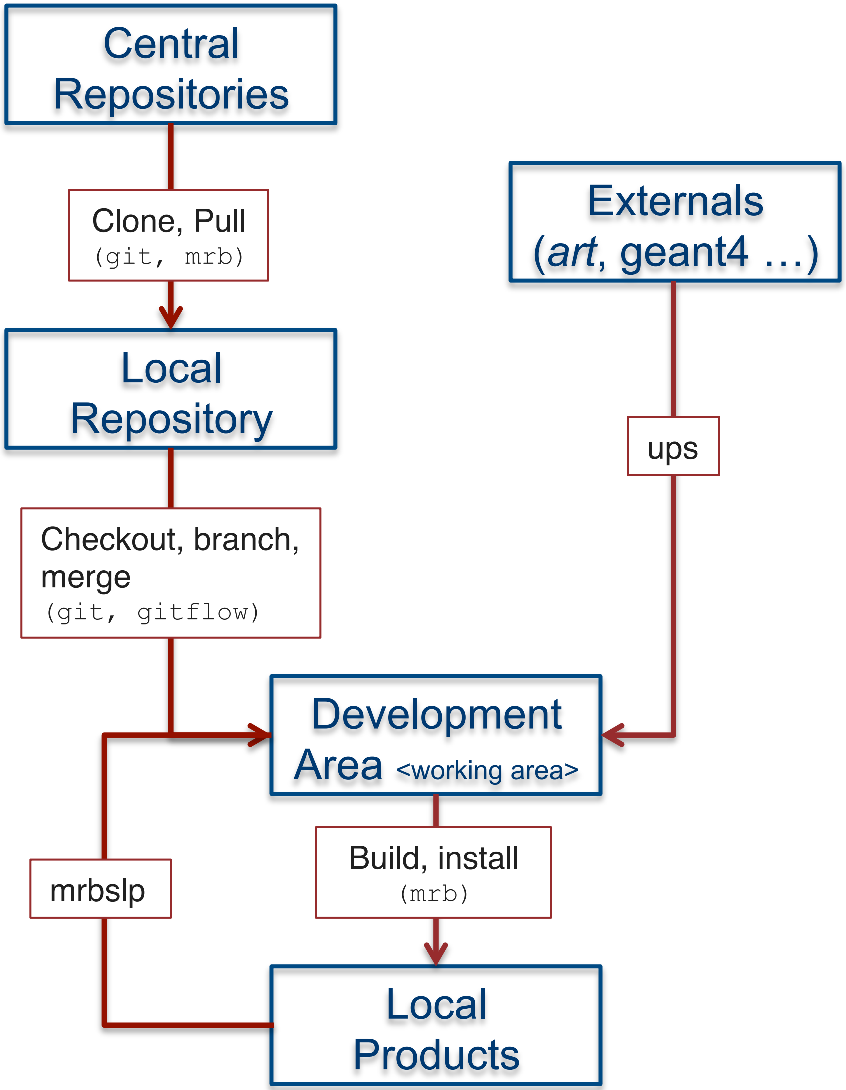
ups

Development Area <working area>

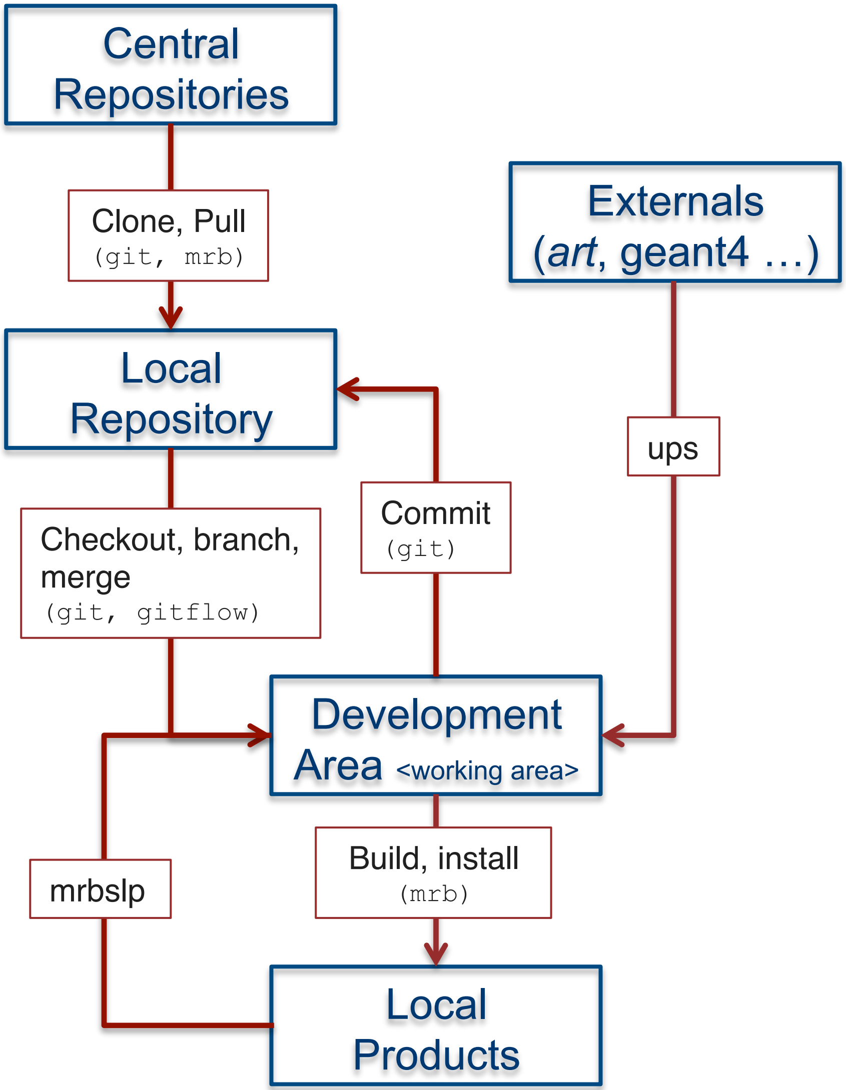
Build, install
(mrb)

Local Products

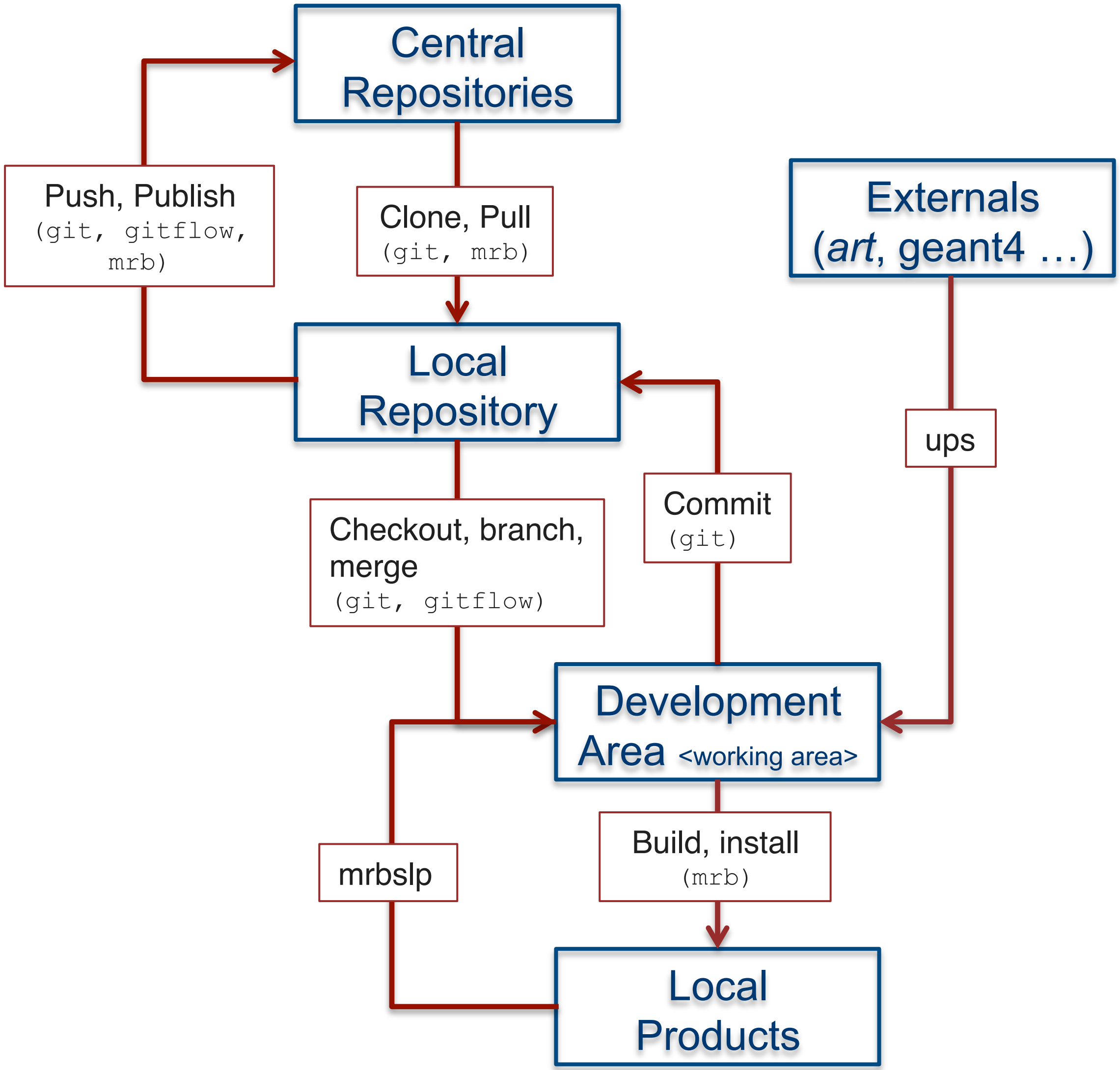
LArSoft Work Environment



LArSoft Work Environment



LArSoft Work Environment



Central Repositories

Push, Publish (git, gitflow, mrb)

Clone, Pull (git, mrb)

Externals (*art*, geant4 …)

Local Repository

ups

Commit (git)

Checkout, branch, merge (git, gitflow)

Development Area <working area>

mrbslp

Build, install (mrb)

Local Products

To build and run LArSoft

- Initial setup for the working environment (fresh login)

- Create a working area

- Check out, build and install a package from LArSoft code

LArSoft Initial Setup (on `alcourse.fnal.gov`)

1. Run the setup script

   ```
   source /products/larsoft_setup.sh
   ```

   This script does the following:

   - `source /products/setup`
   - `setup git`
   - `setup gitflow`
   - `setup mrb`
   - `export MRB_PROJECT=larsoft`

2. For experiment specific setup, use the following instead

   ```
   source <experiment specific script>
   ```

Create a new Development Area

- To create a new development area

  ```
  mkdir <working_dir>
  cd  <working_dir>
  ```

  `mrb newDev` or `mrb n`

  If `larsoft` product is not set up then specify the version and qualifiers, otherwise you will get an error message

  ```
  mrb newDev -v <version> -q <qualifiers>
  ```

- The above command will create the following subdirectories in the `<working_dir>`
  1. A build directory: `build_<flvr>`,
  2. A source directory: `srcs`,
  3. A local products directory `localProducts_<MRB_PROJECT>_<version>_<qualifiers>`

  We refer to the local products dir as `<localProdDir>`

Setup ups localProducts (Example output)

4. Setup local products

```
source localProducts_larsoft_v04_18_00_e7_prof/setup
```

```
MRB_PROJECT=larsoft
MRB_PROJECT_VERSION=v04_18_00
MRB_QUALS=e7:prof
MRB_TOP=/home/ssehrish/my_larsoft
MRB_SOURCE=/home/ssehrish/my_larsoft/srcs
MRB_BUILDDIR=/home/ssehrish/my_larsoft/build_slf6.x86_64
MRB_INSTALL=/home/ssehrish/my_larsoft/
localProducts_larsoft_v04_18_00_e7_prof

PRODUCTS=/home/ssehrish/my_larsoft/
localProducts_larsoft_v04_18_00_e7_prof:/products
```

Using `mrb newDev` with Options

- `mrb newDev -p`
  - just make the products area (checks src, build are already there)
- `mrb newDev -f`
  - use a non-empty directory anyway
- Use –T and –S options with `mrb newDev`, -T specifies where to put the build and local products directories and –S for source code
  - `mkdir <working_dir>`
  - `mkdir <source_dir>`
  - `mrb newDev -T <working_dir> -S <source_dir>`
- This creates
  - `<working_dir>/build_<flvr>`
  - `<working_dir>/localProducts_<MRB_PROJECT>_<version>_<qualifiers>`

Setup work environment for an existing working area from fresh login

- The generic steps are as follows:
  - Set up `ups`
  - Make sure you have `gitflow`, `git` and `mrb`
  - Set `$MRB_PROJECT`
  - `source <localProdDir>/setup`
- For example on `alcourse.fnal.gov,` use the following:
  - `source /products/setup_larsoft.sh`
  - `cd <working_dir>`
  - `source ./localProducts_XXX/setup`

Checkout code from <lar repo>

- Checkout the repository you want to work with after doing the initial set up

```
cd $MRB_SOURCE
mrb gitCheckout larexamples
```

will get the `larexamples` from current development head. If for some reason, you want larexamples with a different version of larsoft, use `–t` with `mrb g`

```
mrb g -t LARSOFT_SUITE_v04_14_00 larexamples
```

Build the code

- After doing initial set up

```
cd $MRB_BUILDDIR
```

- Setup development environment

```
mrbsetenv
```

- Run buildtool

`mrb b (build) [-jN]` #N is number of parallel build streams

- To get rid of what you just built:
  - `mrb z (zapBuild)`
  - `mrbsetenv`

Set up your code to run

- Run buildtool with install

  ```
  mrb i (install)
  ```
  #all commands must be run in the same shell

  – After everything is built and installed,

  ```
  cd ${MRB_TOP}
  mrbslp
  ```

    - `mrbslp` will setup all products installed in the working localProduct directory. (‘slp’ stands for setup local products.)

- Run lar job from larexamples :

```
lar -c AnalysisExample.fcl -s /home/larsoft/
course_data/AnalysisExampleInput.root
```

Running LArSoft with OSX (Mavericks or Yosemite Only)

1. Using “`pullproducts`”
   - a shell script for pulling down the LArSoft code
     - Can pull binary installations and/or source code
     - For example: (http://scisoft.fnal.gov/scisoft/bundles/larsoft/v04_18_00/larsoft-v04_18_00.html)
   - If pulling only source code then must build but process is straightforward
2. CVMFS
   - Key elements:
     - Install OSXFuse
     - Install and configure the CVMFS client

   https://cdcvs.fnal.gov/redmine/projects/larsoft/wiki/LArSoft_cvmfs_page

   - Once done with this, follow setup instructions for LArSoft!
   - Another resource:

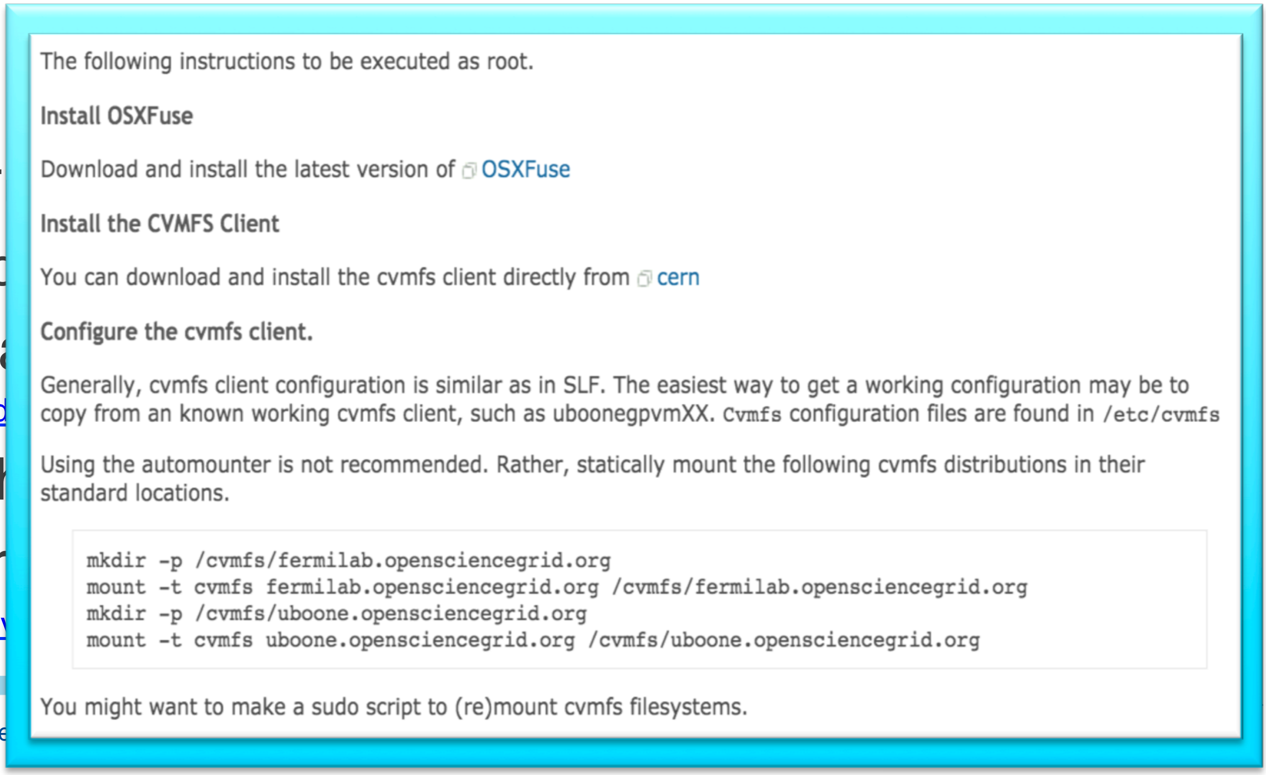
     http://gm2-docdb.fnal.gov/cgi-bin/RetrieveFile?docid=2459&filename=macDevelopment.pdf&version=2

Running LArSoft with OSX (Mavericks or Yosemite Only)

1. Using “`pullproducts`”
   - a shell script for pulling down the LArSoft code
     - Can pull binary installations and/or source code
     - For example: (http://scisoft.fnal.gov/scisoft/bundles/larsoft/v04_18_00/larsoft-v04_18_00.html)
   - If pulling only source code then must build but process is straightforward
2. CVMFS
   - Key elements:
     - Install OSXF
     - Install and co
   - MicroBooNE ha

     https://cdcvs.fnal.gov/red
   - Once done wit
   - Another resour

     http://gm2-docdb.fnal.gov



The following instructions to be executed as root.

Install OSXFuse

Download and install the latest version of OSXFuse

Install the CVMFS Client

You can download and install the cvmfs client directly from cern

Configure the cvmfs client.

Generally, cvmfs client configuration is similar as in SLF. The easiest way to get a working configuration may be to copy from an known working cvmfs client, such as uboonegpvmXX. `Cvmfs` configuration files are found in `/etc/cvmfs`

Using the automounter is not recommended. Rather, statically mount the following cvmfs distributions in their standard locations.

```
mkdir -p /cvmfs/fermilab.opensciencegrid.org
mount -t cvmfs fermilab.opensciencegrid.org /cvmfs/fermilab.opensciencegrid.org
mkdir -p /cvmfs/uboone.opensciencegrid.org
mount -t cvmfs uboone.opensciencegrid.org /cvmfs/uboone.opensciencegrid.org
```

You might want to make a sudo script to (re)mount cvmfs filesystems.

Exercise: Analysis Example in `larexamples`

- Checkout `larexamples`
  - Use HEAD of develop branch i.e. use `mrb g` without options
- Build and run AnalysisExample
- Input file is in "`/home/larsoft/course_data/AnalysisExampleInput.root`"
- You will see the following output file
  - `AnalysisExample.root`
- Use ROOT to browse through the output file

Analysis Example - Output

If your setup and build process was correct, expect to see a similar output.

```
….
TrigReport ---------- Event  Summary ------------
TrigReport Events total = 100 passed = 100 failed = 0

TrigReport ------ Modules in End-Path: end_path ------------
TrigReport  Trig Bit#    Visited     Passed     Failed      Error Name
TrigReport     0    0        100        100          0          0 AnalysisExample

TimeReport ---------- Time  Summary ---[sec]----
TimeReport CPU = 1.402724 Real = 1.417784

=============================================================================================
=============================
TimeTracker printout (sec)                    Min          Avg          Max        Median          RMS        nEvts
=============================================================================================
=============================
Full event                          0.0116195     0.0140445     0.085295     0.0130024    0.00733716      100
----------------------------------------------------------------------------------------------------------------
end_path:AnalysisExample:AnalysisExample     0.0115531     0.0139673     0.0841771     0.0129362    0.00723513      100
=============================================================================================
=============================

Art has completed and will exit with status 0.
```

Follow these instructions to finish exercise

```
1) mkdir my_larsoft
2) source /products/larsoft_setup.sh
3) cd my_larsoft/
4) mrb newDev -v v04_18_00 -q e7:debug
5) source
   localProducts_larsoft_v04_18_00_e7_debug/
   setup
6) cd $MRB_SOURCE
7) mrb g larexamples
```

The commands in red are not needed if you are returning to the development area from a fresh login

More instructions

```
8) cd $MRB_BUILDDIR
9) mrbsetenv
10)mrb b -jN
11)mrb i -jN #optional
12)mrbslp    #optional
13)lar -c AnalysisExample.fcl -s /home/larsoft/
   course_data/AnalysisExampleInput.root
14)root AnalysisExample.root
  –  Tbrowser* b = new Tbrowser(“Browser”, _file0);
```

Summary

- LArSoft Code
  - Repositories
  - Products
  - Dependencies
  - Packages
- LArSoft build tools
  - `mrb`
- LArSoft work environment
  - Setup working area
  - Checkout code
  - Build and run code