A Version Management System for GlueX

Mark M. Ito

Jefferson Lab

October 9, 2015

Outline

1. Introduction
2. The Packages
3. The Directory Structure
4. Scripts to Implement the VMS: the `BUILD_SCRIPTS` Directory
5. Setting Up the Shell Environment
6. The Build System
7. The Versioning System
8. The Prerequisites System
9. The gluex_install System

Introduction

There are three fundamental areas of concern that make up all software systems. They are:

1. a directory structure
2. a build system
3. a version management system

They are all related, aspects of one affects aspects of each of the others.

The Packages

1. build_scripts: scripts to manage building and the shell environment
2. Xerces-C: for reading XML files
3. CERNLIB: to support GEANT 3 simulations
4. GEANT4: simulation engine
5. ROOT: general purpose HENP toolkit
6. EVIO: CODA format data handling library
7. CCDB: Calibration Constants Database
8. JANA: event-based analysis framework
9. HDDS: detector geometry specification library
10. sim-recon: simulation and reconstruction for GlueX

- multiple versions (releases) of each of these
- more than one version of a package may be built
- packages depend on one or several others

The Directory Structure

```
$GLUEX_TOP
|-- build_scripts
|-- cernlib
|   `-- 2005
|-- hdds
|   |-- hdds-3.1
|   `-- hdds-3.2
|-- jana
|   |-- jana_0.7.2
|   `-- jana_0.7.3
|-- root
|   |-- root_5.34.04
|   `-- root_5.34.26
|-- sim-recon
|   |-- sim-recon-1.2.0
|   `-- sim-recon-1.3.0
`-- xerces-c
    |-- xerces-c-3.1.1
    `-- xerces-c-3.1.2
```

Scripts to Implement the VMS: the BUILD_SCRIPTS Directory

All scripts and makefiles to support VMS are found the in the build_scripts directory. At JLab, the full path is

/group/halld/Software/build_scripts.

The directory can also cloned from the Git repository at GitHub. The URL is

https://github.com/jeffersonlab/build_scripts.

Many of the scripts and makefiles require the environment variable BUILD_SCRIPTS to be defined.

Setting Up the Shell Environment

- Environment setting needed for building and using
- Both Bourne shell and C shell supported

Low-Level Environment Set-Up: gluex_env.(c)sh

- The gluex_env.(c)sh script will define all environment variables
- Input: home directories of each package

Package	Home Directory Variable
Xerces-C	XERCESCROOT
CERNLIB	CERN
Geant4	GEANT4_HOME
ROOT	ROOTSYS
EVIO	EVIOROOT
CCDB	CCDB_HOME
JANA	JANA_HOME
HDDS	HDDS_HOME
sim-recon	HALLD_HOME

- gluex_env.(c)sh takes care of all the other variables.
  - For example, XERCES_INCLUDE is defined as $XERCESCROOT/include.
  - Directories are added to the PATH, LD_LIBRARY_PATH, and PYTHONPATH

Pre-Defined Home Variables and Defaults

- If defined in advance, key variables will be respected.
- If not default provided.
  - `GLUEX_TOP`, default = `/usr/local/gluex`
  - `BUILD_SCRIPTS`, default = `$GLUEX_TOP/build_scripts`
  - Home directories, default = `$GLUEX_TOP/<package>/prod`
- See example below

Consistency Checking

- Each home directory is checked for a prerequisites version file
- Files list versions of prerequisite packages used at build time
- The build-time version are checked against the environment versions
- Warnings are printed on mismatches

Cleaning the Environment: `gluex_env_clean.(c)sh`

- `gluex_env.(c)sh` is sensitive to definitions hanging around in the environment
- Undo script: `gluex_env_clean.(c)sh`

High-Level Example: Custom Script

- The build of sim-recon in the user's home directory will be used.
- All other packages be set up to use with their default builds under `/home/gluex/gluex_top`.
- Note that `GLUEX_TOP` and `BUILD_SCRIPTS` are defined explicitly

```
export GLUEX_TOP=/home/gluex/gluex_top
export BUILD_SCRIPTS=$GLUEX_TOP/build_scripts
export HALLD_HOME=/home/username/sim-recon
source $BUILD_SCRIPTS/gluex_env.sh
```

High-Level Example: Use of the Versioning System

- Package version information is contained in the XML file `my_versions.xml`
- Alternate combinations of package versions can be tried by making alternate versions of the version file.

```
export GLUEX_TOP=/home/gluex/gluex_top
export BUILD_SCRIPTS=$GLUEX_TOP/build_scripts
source $BUILD_SCRIPTS/gluex_env_version.sh \
    /home/username/my_versions.xml
```

Default Environment at JLab

Pre-packaged scripts exist, they use the versioning system, JLab-specific for bash:

```
source /group/halld/Software/build_scripts/gluex_env_jlab.sh
```

for tcsh:

```
source /group/halld/Software/build_scripts/gluex_env_jlab.csh
```

The Build System

- Each of the packages have their own native build system
- Each build system has its own set of details to master.

The Makefiles

- Makefiles invoke the native package-specific build system
- There is a "package makefile" for each package (*e. g.*, `Makefile_jana, Makefile_sim-recon`).
- Invoking make with a package makefile will build that package in the current working directory.
- Package makefiles can be used stand-alone

The Top-Level: Makefile_all

```
gluex: env xerces_build cernlib_build root_build clhep_build \
       evio_build ccdb_build jana_build hdds_build \
       sim-recon_build
```

- builds all the packages necessary for using GlueX software
- puts them in the standard directory structure
- Makefile_all should always be invoked from the $GLUEX_TOP directory

Individual Package Targets

- individual package targets (*e. g.*, `evio_build` and `hdds_build`) use the corresponding package makefile
- directories are created to give the standard directory structure
- each individual package build target can be invoked directly
- note: CERNLIB is non-standard

The Package Makefiles

- package makefiles are sensitive to version defining environment variables

```
setenv JANA_VERSION 0.7.2
setenv SIM_RECON_VERSION 1.2.0
setenv HDDS_VERSION 3.2
```

- each makefile knows how to fetch a particular version
- Some packages can be checked out from a source code repository
- Instead of a version variable, a URL variable can be used

```
setenv HDDS_URL https://github.com/jeffersonlab/hdds
```

- others: `JANA_URL`, `SIM_RECON_URL`, and `CCDB_URL`
- Special Variable for `Makefile_sim-recon`

```
make -f $BUILD_SCRIPTS/Makefile_sim-recon \
   SIM_RECON_SCONS_OPTIONS='SHOWBUILDS=1'
```

Adding New Package Builds to an Existing Tree

There are two ways:
1) Use the Top-Level `Makefile_all`

- set up the version variables
- invoke `Makefile_all`

```
cd $GLUEX_TOP
make -f $BUILD_SCRIPTS/Makefile_all gluex
```

2) Use Individual Package Makefile Directly

```
cd $GLUEX_TOP/sim-recon
make -f $BUILD_SCRIPTS/Makefile_sim-recon SIM_RECON_VERSION=1
```

The Versioning System

Version File Format: an XML

```
<gversions>
<package name="jana" version="0.7.3"/>
<package name="sim-recon" version="1.4.0"/>
<package name="hdds" version="3.3"/>
<package name="cernlib" version="2005" word_length="64-bit"/
<package name="xerces-c" version="3.1.1"/>
<package name="clhep" version="2.0.4.5"/>
<package name="root" version="5.34.26"/>
<package name="ccdb" version="1.05"/>
<package name="evio" version="4.3.1"/>
</gversions>
```

Element and Attributes

One type of element, the `package`. Attributes are:

name: The name of the software package.

version: The version number of the package.

url: A URL to be used to checkout (Subversion) or clone (Git) the code. The URL should point to an appropriate repository.

branch: When using a Git repository, the branch to be checked out.

dirtag: A string (directory tag) to be appended to the standard directory name of the package when it is built.

home: Force the location of the package home directory when setting up the environment.

Setting Up the Environment with a Version File

```
$BUILD_SCRIPTS/version.pl version_1.7.xml
```

assume GLUEX_TOP is /home/gluex/gluex_top, then

```
setenv JANA_VERSION 0.7.3;
setenv JANA_HOME \
    /home/gluex/gluex_top/jana/jana_0.7.3/Linux_RHEL7-x86_64-gcc4.8
setenv SIM_RECON_VERSION 1.4.0;
setenv HALLD_HOME /home/gluex/gluex_top/sim-recon/sim-recon-1.4.0;
setenv HDDS_VERSION 3.3;
setenv HDDS_HOME /home/gluex/gluex_top/hdds/hdds-3.3;
...
```

Since you would want these commands applied to the current shell,

```
eval `$BUILD_SCRIPTS/version.pl version_1.7.xml`
```

- following this step, invoke `gluex_env.(c)sh` to complete the set-up.
- `gluex_env_version.(c)sh` combines use of `version.pl` and `gluex_env.(c)sh` (example above)

Specifying Alternate Source Code Sources with a Version File

To clone sim-recon and checkout the branch `test_stuff`, use

```
<package name="sim-recon"
         url="https://github.com/jeffersonlab/sim-recon"
         branch="test_stuff"/>
```

Directory Tags

The `dirtag` attribute: distinguish different builds with different prerequisite packages.

```
<package name="hdds" version="3.3" dirtag="xerces_test"\>
```

`version.pl` adds a variable to the environment:

```
setenv HDDS_DIRTAG xerces_test
```

`Makefile_hdds` produces a directory named `hdds-3.3^xerces_test`

Specifying Alternate Home Directory Locations

A package may lie outside of the standard directory structure.

```
<package name="sim-recon" home="/home/my/sim-recon"/>
```

will cause `version.pl` to generate

```
export HALLD_HOME=/home/my/sim-recon
```

Useful for creating an environment for use; when building it you are on your own.

The Prerequisites System

- At build time, a version xml file is created in the home directory of a package if that package has dependencies on others in the system.
- Example: $HALLD_HOME will have the file `sim-recon_prereqs_version.xml`

```
<gversion version="1.0"
><package name="evio" version="4.3.1"
 /><package name="cernlib" version="2005"
 /><package name="root" version="5.34.26"
 /><package name="jana" version="0.7.3"
 /><package name="hdds" version="3.3"
 /><package name="ccdb" version="1.05"
 /></gversion
>
```

How the Prerequisite File is Used

- At set-up time...
- check only if a version file with prerequisites exists
- each package in version file is checked for version consistency with correspoding home variable
- version from the home directory comes from:
  1. **Tar File**: parsed from home directory name
  2. **Subversion Checkout**: `svn info` command is used to get the URL info
  3. **Git Clone**: not yet implemented!

If a version mismatch is found, a warning is written to the screen.

The gluex_install System

- Described at a previous collaboration meeting
- Uses the VMS
- Installs all required distribution-provided packages

Tested on:

Distribution	**Package Type**
CentOS	RedHat
Scientific Linux	RedHat
Ubuntu	Debian
Fedora	RedHat
LinuxMint	Debian
openSUSE	RedHat
RedHat Enterprise	RedHat

Summary

- Automation of builds of GlueX-related software
- Defines a standard directory structure
- System for setting up the environment for using the software
- Version combinations summarized succinctly in an XML file
- XML file can be used to drive both building and environment set-up
- Version consistency checking mechanism provided
- Includes features for custom builds of user-specified packages