How to Use PRIMEX_MY

PrimEx Note 21

Mark Ito
Thomas Jefferson National Accelerator Center

January 22, 2004

1 Introduction

In a collaborative software development environment, it is important that individual developers share as much of the body of code as possible at any given point time. This maximizes that chances that new changes will be compatible with the code of others. Also, it is important that any differences in a developer's code versus some standard version be clearly identifiable. If that is the case, then understanding differences in results from two versions of a program is much easier. Both of these considerations favor an approach where an individual developer has a small set of code under his direct control (or at least as small as possible) and relies on a public version for the bulk of the code set. Once the private code is functional, it can be incorporated in the next standard version and removed from private directories.

This approach also is compatible with experimental code, code that is meant to be used once to try new ideas and then thrown away. Experimenting with code can be done by modifying and building the small fraction of the total code necessary; there is no need to build the entire code set in order to test out the change of a single line.

This note describes features of the PrimEx make scheme that facilitate a source code management practice that keeps private and/or experimental code to a minimum, while still maintaining the ability of a developer to change anything in the entire code set for his or her own purposes.

2 Environment Variables

The PrimEx make scheme depends on a number of environment variables to do its work. These are set up (at JLab) by `/group/primex/scripts/primex_jlab` (see the usage example in the next section). For this discussion, three of them are especially important:

`PRIMEX_BUILD` This variable chooses the public build that will be used to build against, *i. e.*, the place from where most of the code will come. This variable is special in that its value affects others set by `primex_jlab`. If this variable is changed, `primex_jlab` must be re-source'd to reset the variables which depend on `PRIMEX_BUILD`. The default value at JLab is /group/primex/builds/prod, the "production build". This default will only be used by `primex_jlab` if there is no other pre-existing definition. In other words, the pre-existing definition will be respected.

`PRIMEX_MY` This variable chooses where the privately built object libraries and binaries will be placed. Libraries are placed in `$PRIMEX_MY/$OSNAME/lib` and binaries are placed in `$PRIMEX_MY/$OSNAME/bin`. `OSNAME` is assumed to be set by the shell. The default value for `PRIMEX_MY` is

```
/work/hallb/primex/pdisk/$USER
```

If a value of `PRIMEX_MY` is set before sourcing `primex_jlab`, that pre-existing value will be respected.

`PRIMEX_INCLUDE` This variable indicates the directory to use in the include path for compilation. By default, it points to `$PRIMEX_BUILD/src/libraries/include`. The user can set this to any directory if that default is not desired. `primex_jlab` resets the value of this variable to the default. Private definitions must be made after sourcing `primex_jlab`. When using a private definition of `PRIMEX_INCLUDE` the user must exercise care that all libraries being used were compiled with compatible versions of the include files.

3 An Example

Let's assume that we want to make a change in the `hycal` library and test it out its effect when running `prim_ana`. Let us further assume that your username is `janed`. There are five steps:

1. Set up the environment.

```
source /group/primex/scripts/primex_jlab
```

Your `PRIMEX_MY` now points to `/work/hallb/primex/pdisk/janed`.

2. Get the code.

Assuming that you have already created a directory in your home area, called `/home/janed/primex`,

```
> cd /home/janed/primex
> cvs checkout src/libraries/hycal src/programs/prim_ana
cvs checkout: Updating src/libraries/hycal
U src/libraries/hycal/HYCALCLUSTER_GetHycalClusters.cc
U src/libraries/hycal/HYCALHIT_GetHycalHits.cc
...
cvs checkout: Updating src/programs/prim_ana
U src/programs/prim_ana/Makefile
U src/programs/prim_ana/hist_book.cc
...
```

This creates the directory `/home/janed/primex/src` with the indicated sub-directories.

3. Make the modification.

What you do here is up to you. Presumably you will want to edit some of the code you just checked out.

4. Build the code

```
> cd /home/janed/primex/src/libraries/hycal
> make
```

This creates the file `libhycal.a` in `/work/hallb/primex/pdisk/janed/lib/Linux`.

```
> cd /home/janed/primex/src/programs/prim_ana
> make
```

This creates the file `prim_ana` in `/work/hallb/primex/pdisk/janed/bin/Linux`. This version of `prim_ana` uses your private version of `libhycal.a` and not the one in the production build.

5. Run it.

```
> $PRIMEX_MY/bin/Linux/prim_ana
```

$Id: primex_my.tex,v 1.10 2004/01/22 14:51:28 marki Exp $