FNAL CD/D0/CDF Run II Code Management Working Group

Jim Bellinger, Walter Brown, Glenn Cooper, Flavia Donno-Raffaelli, Lynn Garren, Herb Greelee, Robert Harris, Alan Joncheere, Qizhong Li-Demarteau, Don Petravick, Ruth Pordes, and Elizabeth Sexton-Kennedy

Formation

- ✓ Formed in ’96 with a change to :
  - ✓ isolate common core requirements
  - ✓ propose common tools and projects to create tools
- ✓ First collaborative group of its kind devoted to Run II needs
- ✓ Some fraction of the group must also participate in implementation

Motivation

- ✓ CDF and D0 Run I systems were based of VMS only products
- ✓ VMS is not a supported Operating System for Run II
- ✓ A collaborative effort would reduce the total manpower devoted to replacing the Run I systems

Areas to Address

- ✓ Source version control systems
- ✓ Binary building tools
- ✓ Management of releases
- ✓ Distribution and Installation

* Communication and control of changes
* Testing and bug tracking

Requirements Summary

- ✓ Work on supported platforms
- ✓ All tools must be freely available
- ✓ Naturally handle contribution to the project from a large number of collaborators
- ✓ Multiple version of the project software must be supportable

Choice of Tool Set

- ✓ CVS in client server mode using the SDSS cvsh shell
- ✓ SoftRelTools
  - ✓ gmake
  - ✓ GNU's "cpp -M" for dependency generation
  - ✓ Bourne shell written utility scripts
- ✓ UPS for enviroment support
- ✓ UPS and UPP for distribution and installation

Why CVS?

- ✓ Version control of files in a hierarchical tree structure
- ✓ Transparent remote access and remote file distribution
- ✓ Optional hooks for testing and notification of new code versions
- ✓ CVS is a standard in wide use by the HEP community

CVS in SDSS Configuration

- ✓ User foo issues standard CVS commands
- ✓ .rhosts file determines if access is accepted
- ✓ cvsuser runs a restricted shell that can only do CVS commands, it starts the server
- ✓ If remote access will modify the repository a check_access script will be called by the standard commit_info

Advantages of SDSS-CVS

- ✓ Remote and local access are handled the same way
- ✓ Access control is not handled by the local file system. Users can not accidentally change file protections
- ✓ Users are forced to use CVS to change the repository
- ✓ Write access is granted on a module by module basis
- ✓ The whole collaboration can check out files while only a few can check in

Why SoftRelTools?

- ✓ Asynchronous development of packages and releases
- ✓ The knowledge of which package version belongs to which release is contained in the file system
- ✓ Only releases contain binaries
- ✓ Developers are encouraged to make their own test releases...they do some integration work themselves
- ✓ Make file complexity is hidden and centralized

SoftRelTools Directories

- ✓ Two main subtrees pack. and rel.
- ✓ There are multiple package instances
- ✓ There are multiple release instances
- ✓ Package area contains source only
- ✓ Symbolic links in the release instance -> package sources are not duplicated
- ✓ include/ , and lib/ serve as collection points for -I and -L directives
- ✓ bin/ and man/ are added to appropriate path variables when release is setup

SoftRelTools Development

- ✓ Developers modifying a package first making a test release
- ✓ addpkg is uses to check the package into this release
- ✓ *depend* finds other packages that need to be linked to when include files in the developers package are changed
- ✓ *lnkpkg* makes the needed links
- ✓ -I and -L directives search the local test release first then the base release

Unix Product Support

- UPS allows the developer of a package to modify the environment of the package user for proper functioning
  - Modify path variables
  - Define compile/run-time variables
- users: "setup <package> <version>"
- UPS requires a root directory
- Optionally a script can be sourced when the product is "setup"

SoftRelTools use of UPS

- ✓ Each *project* (a release is an instance of a *project)* and each package within a project is a ups product
- ✓ Developers can define additional variables for run-time use ... these will be defined when the project is "setup"
- ✓ tools used by the project *(e.g. g++)* are also ups products which will be setup when the project is "setup"
- ✓ Path variables are set to point to the bin/ and man/ areas of the release

Using a SoftRelTools Project

- ✓ After a project is setup:
  - ✓ Users can use any executable or man page contained in the release
  - ✓ Make files for linking to a release library are also provided. Paths to the library and include areas are defined by environment variables
  - ✓ Source code can be browsed using environment variables

Distribution

- Now:
  - makeDist script builds tar files and readies the release for UPD
  - pull_<project> uses the files created by makeDist to pull a coherent release
- Future:
  - UPD and UPS are being redesigned
  - In the future they will add versatility and function in order to automate the distribution