Considerations on package management for CEPCSW

Tao Lin

2020 年 3 月 30 日

Outline

Package

- A project is divided into a lot of packages.
  - BESIII offline: about 355 packages, JUNO offline: about 120 packages
- Each package is belong to a specific category:
  - Core, Detector, Event, Utilities, Database
  - Generator, Simulation, Reconstruction
  - Analysis
- A package contains a list of source files, which can be organized as following:
  - Header directory: the public header files, which will be used by other packages. For an example, a service interace.
  - Source directory: both internal header files and source files.
  - Script or configuration directory
- A package can produce several different types of libaries:
  - Module: a specific shared library, with all symbol resolved, can be loaded dynamically. It should not be linked by others.
  - Library: a common shared library, which will be linked by others.

Examples of Module and Library

Athena (ATLAS)

Module: DetectorDescription/GeometryDBSvc

- GeometryDBSvc/IGeometryDBSvc.h
- share
- src
- CMakeLists.txt

Library: DetectorDescription/Identifier

- Identifier
- share
- src
- CMakeLists.txt

Category

- Core: Framework related.
- Detector: Detector description, geometry service related.
- Event: event data model related. Wrapper on edm4hep/plcio.
- Utilities: common tools. Such as timer.
- Database: Database related.
- Generator: physics generators.
- Simulation: detector simulation, digitization.
- Reconstruction
  - Vertex
  - Tracking
  - Calo
- Analysis

Different organizations

Both need additional utilities, as our project consists a lot of packages.

A large repo

- All packages are in one git repo.
- Easy to manage.
- If there are a lot of packages, time consuming to build them.

A number of small repos

- Each package is in its own git repo.
- Don't need to checkout the complete project.
- If there are too many repos, difficult to manage them.

Do we need a utility `Git-CEPC`?

I have created `Git-BOSS` before, to help BESIII developers migrate their developing environment to Git.

```
$ source /afs/ihep.ac.cn/bes3/offline/ExternalLib/\
 SLC6/contrib/git/setup.sh
$ git boss initwork myworkarea
$ cd myworkarea
$ git boss listpkgs
$ git boss addpkg Analysis/Physics/RhopiAlg
```

The magic is Sparse Checkout. My tool is a wrapper to edit file `.git/info/sparse-checkout`.
See source code:
`http://code.ihep.ac.cn/lintao/git-boss/-/blob/master/git-boss`

Summary

- It is necessary to share the same convention on the package management.
- I prefer a large repo, but with addition utilities to help users.