How to document software packages

Offline computing tutorial Maintained by Wenaus
Last modified Thu Oct 7 11:17:14 CEST 1999

Package documentation resides in the CVS repository together with the code. There are two levels of documentation for each package:

1. A README file in each package directory with a brief description of what the package is for, what subsystem domain it resides in, who the authors are, and a brief reverse chronological history of revisions to the package. That is, the README file should give a concise picture of the purpose of the package and its history and status.

2. The `doc` subdirectory under the package directory which contains reliable information about performance and usage of the package. The main page should be in `index.html`. The main page may contain little more than links pointing to the doc resident elsewhere; it's up to you. The important thing is that the page guide the user through detailed documentation enabling them to understand and use the package.

```
PackageDirectory
 |-----README
 |-----doc
        |---index.html
```

The documentation is most easily accessed via the software guide linked from the computing homepage. README files and doc directories for each package are linked there.

Documenting CVS commits

When you commit a change to the CVS repository you must supply a comment as to what the change is all about. It is important that you make the comment descriptive enough to be useful. If one line is all you need you can just use `-m "comment"` on the commit command line, but you can also enter more extended comments by leaving off the -m option. You will be put into an editor in which you can enter as descriptive a comment as you like; when you save+exit the editor your comment will be saved and the commit processed. The editor used follows the CVSEDITOR environment variable if it is set, otherwise the EDITOR environment variable (eg. `setenv CVSEDITOR "emacs -nw"`).

If you have changed multiple files for multiple reasons you can also ensure that the proper comments are associated with the proper files by committing the files separately, eg.

```
cvs commit -m "Comment 1" file1
cvs commit -m "Comment 2" file2 file3
```

Using ROOT documentation tools (V. Fine)

You can use the StMaker::MakeDoc method to produce documentation for your Maker classes.

Calling this method one can produce an HTML documentation of his/her Maker class. I want to stress the automatically produced HTML documentation is not to replace other form of docs rather to facilitate it. See for example http://www.star.bnl.gov/STAR/html/comp_l/train/root/Star2C++tut.html.

I'd like mention that bfc.C macro has a built-in parameter to produce the documentation for all classes and StMakers involved. For example calling root4star 'bfc.c(1,"y1a MakeDoc")' will produce the HTML documentation for all classes involved for "y1a" reconstruction. To get the HTML documentation for some particular maker one has to add ONE single line in his / her macro

```
->MakeDoc();
```

This documentation is available to review via the STAR software guide by clicking the name of the source or header file.

To achieve the best result one has to follow a few very simple rules. Those can be seen in the template St_TLA_Maker. I would advise people to compare the link above with the source of that class to see how one provide

- short class description
- each method description
- pictures and figures
- can keep the Web link and provide different fonts.

The ROOT html class can be used to create documentation for ROOT macros or any "plain text file". See for example SubDetectorView.C.html.

From my experience creating a good looking HTML doc even if it is generated automatically needs a lot of effort and several circles of the documentation "debugging". No tool can make your comments good looking if good comments are not written.