Using PRESPEC_Go4 with version control system

August 22, 2011

This manual is a brief instruction to use Go4 the online/offline analysis tool for the PRESPEC experiments at GSI. The prespec_Go4 code is based on the frs_Go4 code wich has been used previously by the FRS group, in addition to it the implementation for EUROBALL, LYCCA, HECTOR and FINGER detectors has been included. (For more information about Go4 itself see Ref. [1])

It is very important to point out that this code is just a starting version of the analysis, which will allow you to read, sort and extract the data (lmd files) in ROOT trees with a basic analysis, after this you can develop your own analysis methods. However since this is a tool that can be use for everybody in the PRESPEC collaboration if you notice that you can improve the code you are welcome to do it by using the version control system provided.

1 Version Control System

For the version control system there are several tools, to keep it in the easiest way possible, the platform *Mercurial (hg)* [2] was chosen. "It is a fast, lightweight source control management system designed for easy and efficient handling of very large distributed project" [3].

If you are a new user of Go4 you just need to copy the files that are located at the GSI computers, this process is called to *clone* a repository. If you already have a version of the code download the repository and apply your modifications to the files in the extracted folder. The repository is the latest version of **prespec_Go4** used at the end of the November 2010 experiment.

Here is the starting procedure of how to use the repository:

1. Install *Mercurial*.
   You need to have mercurial in your computer, usually it comes with the standard versions of linux. It can be downloaded from http://mercurial.selenic.com/.
   It will be useful to have also *kdiff3* to handle the conflicts later.

2. Clone the repository in the latest version (Currently number 164). The symbol $ indicates that is a command line.
   From GSI computers:
   $ hg -v clone ssh://prespec@lxg1030//u/prespec/prespec_Go4
   From outside GSI:
   $ hg -v clone ssh://prespec@lxi012.gsi.de:22//u/prespec/prespec_Go4
   For the password please contact: E.Merchan@gsi.de
   This process creates a folder named **prespec_Go4**. Now you can start to work. Go to the new folder:
   $ cd prespec_Go4

3. Inside the folder create a file ∼/.hgrc that contains:
   [ui]
   username = USER <e-mail@something.com>
   USER is your user name that will be the tag to identify the changes that you made to the code. It is possible to set an e-mail account to contact you in case of conflicts.

4. Now you need to define the environment variable EDITOR in your /.bashrc or ∼/.bash_profile. You can use for example nano, vim or emacs as you prefer. Just add:
   export HGEDITOR="yourEditor"

5. Then you can start to make changes. As *Mercurial* is a non central version control system any change that you want to save is not necessary upload to the central repository. Your local version control is used with the command:
   $ hg commit -m '*description of your changes*'
   It is very important that the descriptions of every committed version are well done, and very well described, this would help to keep track of the implementations you made.

   If you use the command without the "-m" it will open the HGEDITOR you already set, this allows to write a well detailed description of the changes made.

6. Before upload your changes or any time you want, it is useful to check if there were any other changes in the central repository. This can be made with a simple command line:
   $ hg pull
   This action downloads all the changes performed to the central repository after you clone it, but it does not merge them. To merge all the changes and update your local files:

$ hg update
This will merge your changes with those from the repository, if there is any conflict there are some tools as *kdiff3* which helps to deal with the conflicts. Then if everything works fine load the changes to the central repository:
$ hg push
password: S62aturne
This is a simple command because *Mercurial* remembers were it was extracted from.

After pushing a version to the general repository please remember to send a message [4], then after checking if the new implementation works the final update will be done.

There are many other helpful commands that allows to have a good control of your files. A brief list of the commands you may need are listed below:

$ hg help	Write all *Mercurial* commands.
$ hg log -l N	Write the N last comments of the previous versions.
$ hg status	Show changed files in the working directory.
$ hg parents	Show the parents of the working directory or revision.
$ hg merge	Merge working directory with another revision.
$ hg add	Add the specified files on the next commit.
$ hg rm	Remove the specified files on the next commit but not from the total repository.

There are multiple options to work with *Mercurial* here were described the commands that allow you to start working, an extended user guide can be found in Ref. [3]. Any comment or question that you have about how to work with the repository or how to solve conflicts please contact E.Merchan [4].

2 Running prespec_Go4

Go4 in general is organized in steps, in this particular case five steps are implemented unpacking, sorting, calibration, analysis and user. The processing scheme is shown in Fig. 1, the scheme shows that the steps are done in a sequence, from each step a ROOT tree can be written. However to implement the LYCCA preliminary analysis we have used the User step from were the outputs of each step

can be read. The variables to be written in the ROOT tree of an specific step are declared in the corresponding XXXEvent.h file, those variables with a commentary (//!) are omitted in the tree.

The LYCCA analysis subroutine was written by Robert Hoischen [5], it handles the data from the DSSD wall, DSSD target, stop and start plastic scintillators and diamond, but not the CsI detectors. You have to be careful with how the variables for the DSSD wall and CsI detectors are arranged, it is not the same. It calculates variables such as positions, multiplicities, energy loss, time of flight and velocity. To see all the possible outputs and a brief explanation of them take a look to the LYCCAEventDataClass.h file.

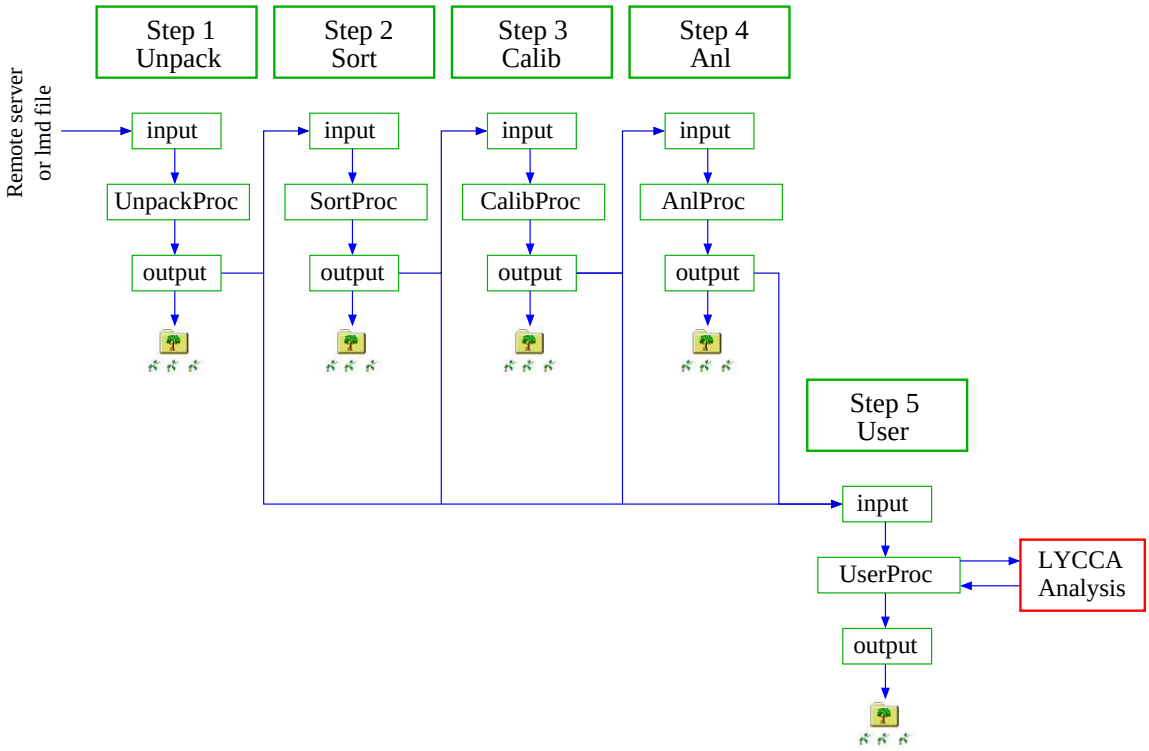

Figure 1: Scheme of the processing steps in prespec_Go4.

Once you have download the code you have to check if you have the appropriate versions of ROOT, Qt and Go4. If you are running at GSI it is enough to type
$. go4login qt3
this command currently enables the ROOT version 527-06, Qt version v3.3.7 and Go4 version v4.04.02. Then just:
$make If you succeed to compile it, check if you have the proper magnetic field values for the FRS setting and the proper Ge and LYCCA calibration files in the setup.C file. Finally as any Go4 you can run it in the graphic or in the

console mode. For the graphic mode just type *go4*, it will display the GUI. For more detailed information see the Go4 manual [6]. Once you know if the data are process in the right way you can enable Go4 to write the ROOT trees by changing to TRUE the SetStoreEnabled function in each step you want to save in the TFRSAnalysis.cxx file, then run as many files you require creating a list file with extension *lml* and run it in the console mode as:
$./MainUserAnalysis XXX.lml

If you need more help or have any question do not hesitate to ask, good luck with your analysis.

References

[1] http://www-win.gsi.de/go4/.

[2] http://mercurial.selenic.com/.

[3] http://hgbook.red-bean.com/.

[4] Edana Merchan, E.Merchan@gsi.de.

[5] Robert Hoischen, robert.hoischen@nuclear.lu.se.

[6] http://www-linux.gsi.de/ go4/go4V04/manuals/Go4introV4.pdf.