mkLDD

February 19, 2009

1 Bug fix

1. UserHook/Hist: In k90whist1.f, you will find a line

   `write(fno, '(a,a)') '#k ', h%c%id(3:klena(h%c%id))`

   The 3 above is a simple typo and corrected to be 1. The effect so far is not serious. However, when we analyze a -t.hist histogram, we utilize that line to get the layer number where histogram was taken.

   After correcting, in Hist and mkLDD, do

   make clean; make

2. In UserHook/mkLDD. If your interface.f contains

   ```
   c            specify bin or ascii output
                call kwhistso( binw )
   ```

   move them to just before the "return" in the following lines.

   ```
         return
   c     ********************************** hook for Beginning of  1 event
   c     *  All system-level initialization for 1 event generation has been
   c     *  eneded at this moment.
   c     *  After this is executed, event generation starts.
   c     *
         entry xBgEvent
   ```

   and If you find similar "call" in your interface1.f, it may be removed (you may keep it and remove the one in interface.f. Having both is actually no problem.)

   The effect so far: you might have empty "-r.hist" files.

3. In UserHook/mkLDD/Zprivate.f, we see a line

   `character*128 basefilename, basefilename2, filename`

   In some environment, 128 may be short. Maybe 192 to 256 is enough.

2 Processing -t.hist file

2.1 splitHisto.sh

You already know this command. It is the quick way to have a look at the histograms. However, it is not suited to do a systematic work for the histograms.

2.2 Step by step examination of the histograms

Before doing a systematic processing of the histograms for later use, we may look at some of the histograms and examine how the fitting is good or not. Each step is explained in separate figures. Before doing so, you have to "make".

- `make -f getBasicHistoInfo.mk`
- `make -f procTime.mk`

After making a fitting routine, explained in the next section, you may consult the separate figures.

2.3 Making a fitting program with CERN minute

The fitting routine (binary executable) is available as
`/TAMCDB/F/src/Minuit/Util/timeFit/timeFitPCLinuxIFC`
or
`/TAMCDB/F/src/Minuit/Util/timeFit/timeFitPCLinuxIFC64`
at tasim501/502/599. So if you are lazy, you may use it directory. However, if you want to do it somewhere other than tasim's, you have to get the source and compile it yourself.

- You may copy the whole set of Minuite:

  `cp␣-r␣/TAMCDB/F/src/Minuite␣Cosmos/UserHook/`

  This example makes a copy in UserHook of Cosmos. Although it is not a direct application of Cosmos, it is treated like the case of Hist routines.

- Before "make", you have two choices:
  - Change `COSMOSROOT` to `COSMOSTOP` in Makefile in "Minuite", "code" and "unix" directories. Do the same also for timeFit.mk in `Util/timeFit`.
  - Or if you are lazy, you may temporarily
    `setenv COSMOSROOT $COSMOSTOP`
    or
    `export COSMOSROOT=$COSMOSTOP`
    but you have to remember this fact.

  Then, do "make".

  In some environment, you may have to modify the definition of function name in unix/intrac.c (to capital letter, with or without _ etc)[1]

- The library of minuit, libminuit.a, will be created in

  `$COSMOSTOP/lib/$ARCH/`

- Go to Util, and

  `make -f timeFit.mk`

  This will create a fitting routine timeFit linked to `timefit$ARCH`.

[1]This will be found when you make an executable at timeFit; the linker will compline if you need to modify it.

3 Making data for time correction automatically

For a large set of -t.hist files, you cannot do the business outlined above. One single command,

```
./procAllLDD.sh
```

will make *.time files needed to make a correction of arrival times in FDD shower. The command eventually make a list of coefficients used in a formula

$$T10 = ar^{b+c\log r} \tag{1}$$

in a file with .time extension for each -t.hist. See separate figures. This will take a long time for, say, 1000 showers. So do it by sge job, or go to cpu idle tasimxxx (you can find it by qhost command) and issue the command.