Table of Contents

Totem command line parser

Using Totem command line parser (totemCLparser.py) one may pass arguments to a configuration file directly from a command line, e.g.:

```
cmsRun test_cfg.py +nevents=1 +beta=2p5 +energy=3500
```

This CL parser defines several common options such as nevents, energy, beta, help etc. so user of this parser doesn't need to define those options in each configuration. This parser uses python modul argparse (see http://code.google.com/p/argparse/) .

The Problem

The problem is that if we try to pass arguments to a configuration file (test_cfg.py) from command line like:

```
cmsRun test_cfg.py arg1 arg2 ...
```

then all arguments arg1, arg2, ... are passed by cmsRun to the configuration file test_cfg.py but only if they don't have a dash ('-') in prefix, e.g. -x or --xxx are not allowed, because options with a dash prefix are recognized by cmsRun and not by test_cfg.py. So we can't do e.g. :

```
cmsRun test_cfg.py -nevents=10
```

which would be realy nice... The arg1, arg2, etc. may be accesed in test_cfg.py using sys.argv:

```
import sys

 for arg in sys.argv:
     print arg, "\n"
```

but then we have to do all the arguments handling on our own which is really tedious...

Elegant Solution

Using purely python modul argparse (see http://code.google.com/p/argparse/) we may avoid many problems because it is posible to configure also prefix '+' for arg options instead of just dash ('-'). This aproach is used by totemCLparser which predefine some options such as nevents, energy, optics, help (i.e., options which are common for many users). So it is possible from command line to do something like:

```
cmsRun -p test_cfg.py +nevents=10 +energy=3500 +beta=2.5
```

these +options are correctly passed to the test_cfg.py as options and possible -options are passed to the cmsRun. Definition of the first +option in totemCLparser.py is just:

```
parser.add_argument(
    "+nevents",                 # option name(s) used from command line
    dest     = 'nevents',     # one common name
    default = -1,                # dafault value
    type     = int,               # support for types int,float,complex...
    action  = "store",
    help     = "number of events to be processed"
)
```

so it is really easy to specify default value for an option, type, define help and much more (all one needs...).

Help for all options is automatically generated and may be printed to a screen also with usage etc - as it is common in unix world...

For details and possible options see documentation of argparse http://code.google.com/p/argparse/, mainly the pdf which is there: http://argparse.googlecode.com/files/argparse-1.1.pdf

How to use totemCLparser

In a configuration file put:

```
# import the totem CL parser
from Configuration.TotemCommon.totemCLParser import parser

# add new option if you want
parser.add_argument(
    "+nplanes",
    dest    = 'nplanes',
    default = 3,
    type    = int,
    action  = "store",
    help    = "minimum number of planes needed for fitting"
)

# default values of some options are not defined (energy, beta and so on) so it is necessery to d
# (totemCLparser sets default values of all the options which defines to "None" so user is forced
# one may set here also default value for "nplanes" argument
parser.set_defaults(
      nevents =-1,
      beta       = "2p5",
      energy   = 3500
      #nplanes = 3
)

# here the parser reads and sets values passed from command line (or just sets the default values
args = parser.parser_args()

# we may print all arguments and their values handled by the parser
# print "Parsed arguments = ", args

...
# now, we may set the number of events
process.maxEvents = cms.untracked.PSet(
    input = cms.untracked.int32(args.nevents)
)

# print value of  newly defined argument "nplanes" (or do something else with nplanes)
print 'Number of planes', args.nplanes
...
```

To see help for our defined options (and options predefined by totemCLparser) just use +h (or ++help) option

```
cmsRun test_cfg.py +h
```

and we obtain something like:

```
usage: test_cfg.py [+nevents NEVENTS] [+beta BETA]  [+energy ENERGY] [+h] [+nplanes NPLANES]  cfg

positional arguments:
  cfgfilename           cfg file name; this positional argument has to be  defined to suppres one
```

```
optional arguments:
  +nevents NEVENTS      number of events to be processed, as default all
                                events from a source are processed
  +beta BETA                  beta* - optics
  +energy ENERGY        energy
  +h, ++help                  show this help message and exit

  +nplanes NPLANES      minimum number of planes needed  for  fitting

%MSG-s ConfigFileReadError:  25-Mar-2010 19:09:24 CET pre-events
Problem with configuration file test_cfg.py
---- Configuration BEGIN
python encountered the error: exceptions.SystemExit 0
---- Configuration END


%MSG
```

-- JiriProchazka - 25-Mar-2010

This topic: TOTEM > TotemCommandLineParser
Topic revision: r4 - 2010-04-14 - JiriProchazka