Manuel Barbera Asin january 2005

Automatic Generation of DecFiles options for DC06

The goal is to have the tool generate the options for all Event Type in as much a generic and flexible way as possible, so that no or minimum changes are required when a new event type is introduced.

This script should produce an option file called GSDCTNXU.opts where GSDCTNXU is the value for the keyword "EventType". It should produce too sql and txt tables (in the *doc* directory) containing information to be used by the Bookkeeping database in the first case and by the EventTypeSvc in the second (only the format is different).

The script is based in the previous one from J. Closier, 21 April 2004.

The code (see appendix 1)

The script searches several information on the dec files (some keywords) to act according to them. If it finds a specific keyword it will add some lines in options file, and if it finds another keyword it will add others.

First, there are several functions declared to make easier the comprehension an use of the code.

The keywords are categorized as mandatory, they must be present for the options file be produced, or optional, they may or not be present. If a optional keyword is present additional pieces are written in the corresponding options file.

Then, the script verifies the mandatory keywords are presents and correct. If some errors are finded, the script sends an error message and stop the process for this dec file.

For the optional keywords, the scripts searches them, and if it finds them it adds the corresponding lines to the options file.

The scripts check also if there are optionals keywords without values or with incorrects values. In these cases, it sends a warning message and continues the process (because they are optionals and the program must go on).

There are also several lines which must be added depending on the value of EventType keyword. For that the value of GSDCTNXU must be checked.

Anyway, there are some cases they have not already an specific solution. These are the cases where G=5 or G=6.

The functions

OptionsValue(filename,word)
It is a simple function to search the value of a keyword (word) in a specific file (filename)

writeOptions(filename,word)
This function inserts a string in a file. It is very useful because we must write a lot of different lines in the options files.

HeaderOptions(filename)
This is the function who writes the header of the options files.

writeCleanParam(filename)
If the script has the clean argument but it does not find the keyword Clean in the corresponding dec file or its value is empty, the script modifies the dec file by adding the Clean keyword with Yes as value.

writeBkkTable(evttypeid,desc,nickname,cleanvalue)
This function creates the file table_event.txt in the doc directory to generate the entry in the ORACLE database.

writeSQLTable(evttypeid,desc,nickname,cleanvalue)
This function creates the file table_event.sql in the doc directory to generate the entry in the ORACLE database.

run_create(clean)
It is the function who take the responsability to create the options file corresponding to an specific dec file.

run_loop()
This function search all the dec files in the dkfiles directory to create all the options files

usage()
This function shows the correct use of the script

And finally there is the main program code. It just controls the number of arguments to call after the corresponding functions.

How to use it

To use the script, we just must execute it. By default, the script produce the options file for all the dec files in the *dkfiles* directory, but when a dec file name is passed as argument (only the name without the .dec extension), it creates the options file for this dec file.

Examples:

```
$ create_options minbias
```

We can also call the script with a dec file name and the clean particle (it is not possible with all the dec files, only with corresponding to signal events).

```
$ create_options minbias clean
```

it will generate an error because minbias is not a

```
$ create_options Bd_pp clean
```

Appendix 1

```
#!/usr/bin/env python2.2
#
#  Create_options.py :
#  Manuel Barbera Asin, 14 November 2005
#
#  creates options files from the decay files
#

import sys,os,re, string
import time,fileinput

#
#  return the value of the correponding string
#
def OptionsValue(filename,word):
    fd = open(filename)
    fdlines = fd.readlines()
    for fdline in fdlines:
        if fdline.find(word) != -1:
            return fdline.split()[2]

#
#  write the options file
#
def writeOptions(filename,word):
    fd = open(filename,"a+")
    fd.write(word)
    fd.close()

#
#  write the header of the options file
#
def HeaderOptions(filename):
    fd = open(filename,"w")
    fd.write("// file  "+OptionsName+".opts generated: "+time.strftime("%a, %d %b %Y %
H:%M:%S", time.localtime())+"\n")
    fd.write("//\n")
    fd.write("// Event Type:"+OptionsName+"\n")
    fd.write("//\n")
    fd.write("// ASCII decay Descriptor: "+AsciiName+"\n")
    fd.write("//\n")
    fd.close()



#
# add the line(# Clean: Yes) in the Decay file
#
def writeCleanParam(filename):
    print "The Clean parameter is added in the Decay file ",filename,"\n"
    for line in fileinput.input(filename,inplace=1):
        print line[:-1]
        if line.find("EventType:") != -1:
            print "#\n# Clean: Yes"

    os.system("cvs -n update "+filename)

#
# write the file to create the entry in the ORACLE database
#
def writeBkkTable(evttypeid,desc,nickname,cleanvalue):
    TableName = os.environ["DECFILESROOT"]+"/doc/table_event.txt"
    if not os.path.exists(TableName):
        os.system("touch "+TableName)
        line = "EventTypeID | NickName | Description\n"
        writeOptions(TableName,line)

    insert_event = "true"
    for line in fileinput.input(TableName,inplace=1):
        print line[:-1]
        if line.find(evttypeid) != -1:
            insert_event = "false"

    if insert_event == "true":
        if cleanvalue == "true":
            line = evttypeid+" | "+nickname+" | "+desc+" (clean)\n"
        else:
            line = evttypeid+" | "+nickname+" | "+desc+"\n"

        writeOptions(TableName,line)
```

```
#
# write the file to create the entry in the ORACLE database
#
def writeSQLTable(evttypeid,desc,nickname,cleanvalue):
   TableName = os.environ["DECFILESROOT"]+"/doc/table_event.sql"
   if not os.path.exists(TableName):
      os.system("touch "+TableName)

   insert_event = "true"
   for line in fileinput.input(TableName,inplace=1):
      print line[:-1]
      if line.find(evttypeid) != -1:
         insert_event = "false"

   if insert_event == "true":
      if cleanvalue == "true":
         line = "svc.addEvtType("+evttypeid+",'"+nickname+"','"+desc+"','clean')\n"
      else:
         line = "svc.addEvtType("+evttypeid+",'"+nickname+"','"+desc+"','')\n"

      writeOptions(TableName,line)



#
#  create an options file corresponding to a single Decay file
#
def run_create(clean):
   global optionsdir,OptionsName,AsciiName, decdir

   if clean == "true":
      title = DecayName+" clean"
   else:
      title = DecayName

   print "Creation of options file for Decay ",title

#  Build the name of the dkfiles
   decdir = os.environ["DECFILESROOT"]+"/dkfiles/"+DecayName+".dec"

# check if the Decay files exists
   if not os.path.exists(decdir):
      print "The file"+decdir+" does not exist"
```

```
        sys.exit(2)

# Get the equivalent eventtype
# convention add +5 to the last for a clean event..
    OptionsName = OptionsValue(decdir,"EventType")

#check if the options file already exist and do not overwrite it
    optionsdir = optionsdirroot+OptionsName+".opts"
    if os.path.exists(optionsdir):
        print " *****  The file "+optionsdir+" exists. CHECK it ****"
        print " ***** to overwrite it, you should remove it first\n"
        writeBkkTable(OptionsName,AsciiName,OptionsNick,clean)
        writeSQLTable(OptionsName,AsciiName,OptionsNick,clean)
        sys.exit(2)

# EventType must have 8 digits ( GSDCTNXU, G<>0 )
# Check if the EventType is correct
    if OptionsName is None:
        print "ERROR: The EventType is not correct.\nIt must have some value"
        print "The options file for "+DecayName+".dec file can't be generated"
        sys.exit(3)
    # The EventType mustn't start by zero
    if OptionsName[0] == '0':
        print "ERROR: The EventType is not correct.\nIt mustn't start by 0"
        print "The options file for "+DecayName+".dec file can't be generated"
        sys.exit(3)
    # The EventType must have at least 8 digits
    if len(OptionsName) < 8:
        print "ERROR: The EventType is not correct.\nIt must have at least 8 digits"
        print "The options file for "+DecayName+".dec file can't be generated"
        sys.exit(3)

    # Check Type
    if OptionsName[0] == '3':
        if OptionsName[1:] == '0000000' or OptionsName[1:] == '00000000':
                # MinimumBias
                sample = "MinimumBias"
    elif int(OptionsName[0]) in (1, 2):
        if OptionsName[1:] == '0000000' or OptionsName[1:] == '00000000':
                # Inclusive
                sample = "Inclusive"
        elif int(OptionsName[0]) == 1 and int(OptionsName[1]) in (1, 2, 3, 5):
            # SignalRepeatedHadronization
            sample = "SignalRepeatedHadronization"
        elif int(OptionsName[0]) == 2 and int(OptionsName[1]) in (1, 2, 3, 4):
            # SignalPlain
            sample = "SignalPlain"
        elif int(OptionsName[0]) == 1 and int(OptionsName[1]) == 4:
           # SignalForcedFragmentation
           sample = "SignalForcedFragmentation"
        else:
           sample = "otherTreatment"
    elif int(OptionsName[0]) == 4 and int(OptionsName[1]) in (0, 1, 2):
        sample = "Special"
    else:
        sample = "otherTreatment"

    # Check Clean Event
    if clean == "true":
        if sample in ('SignalRepeatedHadronization', 'SignalPlain',
'SignalForcedFragmentation'):
#           oldbit = OptionsName[len(OptionsName)-1]
#           newbit=int(oldbit) + 5
#           OptionsName = OptionsName[:-1]+str(newbit)
           OptionsName = OptionsName[:-1]+'5'
           checkClean = OptionsValue(decdir,"Clean:")
           if checkClean != "Yes":
              writeCleanParam(decdir)
        else:
           clean = 'false'
           print "ERROR: You are trying to do Clean with a wrong file type.\nThe file must
be from a Signal type.\n"
           sys.exit(3)

# Check if the Descriptor is correct
# the cheking is a little bit different, we must to split by :
    fd = open(decdir)
    for fdline in fd:
        if fdline.find("Descriptor") != -1:
           AsciiName = string.strip(fdline.split(":")[1])
#    AsciiName = OptionsValue(decdir,"Descriptor")[:-1]
    if AsciiName is None:
        print "ERROR: The Descriptor is not correct.\nIt must have some value"
        print "The options file for "+DecayName+".dec file can't be generated"
        sys.exit(3)
    # The Descriptor must be smaller than 256 characters
    if len(AsciiName) > 255:
        print "ERROR: The Descriptor is not correct.\nIt is too long. It cannot be longer than
256 characters"
        print "The options file for "+DecayName+".dec file can't be generated"
        sys.exit(3)

# Check if the Nickname is the correct
    OptionsNick = OptionsValue(decdir,"NickName")
    if OptionsNick != DecayName:
        print "WARNING: The nickname "+OptionsNick+" is not equal to the
"+DecayName+".dec"
    # The NickName cannot be longer than 256 characters
    if len(OptionsNick) > 255:
        print "ERROR: The NickName is not correct.\nIt is too long. It cannot be longer
than 256 characters"
        print "The options file for "+DecayName+".dec file can't be generated"
        sys.exit(3)

# Check if Production keyword is correct
    ProductionValue = OptionsValue(decdir,"Production")
    if ProductionValue is None:
        print "The Production value is not correct.\nIt must have some value"
        print "The options file for "+DecayName+".dec file can't be generated"
        sys.exit(3)

# get the first digit of the eventtype
    AB = OptionsName[0:2]

    HeaderOptions(optionsdir)
# Higgs files
    if AB == "40":
        str = "#include \"$DECFILESROOT/options/Higgs.opts\";\n"
        writeOptions(optionsdir,str)

# Check if exists Include keyword
    incl = OptionsValue(decdir,"Include")
    if incl != None:
        if len(incl) == 1:
            print "WARNING: There is an Include keyword with no value in "+decdir
        str = "#include \"$DECFILESROOT/options/"+string.strip(incl)+".opts\";\n"
        writeOptions(optionsdir,str)

# Write lines in the options file
    line = EvtGenMain+"EventType = "+OptionsName+";\n"
    writeOptions(optionsdir,line)


# Mandatory lines to write -----------------------------------------

    str = "Generation.SampleGenerationTool = \""+string.strip(sample)+"\";\n"
    writeOptions(optionsdir,str)
    str = "Generation."+string.strip(sample)+".ProductionTool = \""+string.strip
(ProductionValue)+"Production\";\n"
    writeOptions(optionsdir,str)
    str = "ToolsSvc.EvtGenDecay.UserDecayFile =
\"$DECFILESROOT/dkfiles/"+DecayName+".dec\";\n"
    writeOptions(optionsdir,str)

# Optional lines depending of existing keywords ---------------------------------------------

# Check if exists DecayEngine keyword
    DecayEngineValue = OptionsValue(decdir,"DecayEngine")
    if DecayEngineValue != None:
        if len(string.strip(DecayEngineValue)) == 0:
            print "WARNING: The Decay Keyword in "+decdir+" has no value\n"
        str = "Generation.DecayTool = \""+string.strip(DecayEngineValue)
+"Decay\";\nGeneration."+string.strip(sample)+".DecayTool = \""+string.strip
(DecayEngineValue)+"Decay\";\n"
        writeOptions(optionsdir,str)

# Check if exists cuts keyword
    CutsValue = OptionsValue(decdir,"Cuts")
    if CutsValue != None:
        if len(string.strip(CutsValue)) == 0:
            print "WARNING: The Cuts Keyword in "+decdir+" has no value\n"
        str = "Generation."+string.strip(sample)+".CutTool = \""+string.strip(CutsValue)
+"\";\n"
        writeOptions(optionsdir,str)

# Check if exists FullEventCuts keyword
    FullEventCutsValue = OptionsValue(decdir,"FullEventCuts")
    if FullEventCutsValue != None:
        if len(string.strip(FullEventCutsValue)) == 0:
            print "WARNING: The FullEventCuts Keyword in "+decdir+" has no value\n"
        str = "Generation.FullGenEventCutTool = \""+string.strip(FullEventCutsValue)
+"\";\n"
        writeOptions(optionsdir,str)

# Generation.SAMPLE.GENERATOR.InclusivePIDList
# if Inclusive
    if sample == 'Inclusive':
        if OptionsName[0] == '1':
                list = '521, -521, 511, -511, 531, -531, 541, -541, 5122, -5122, 5222, -5222,
5212, -5212, 5112, -5112, 5312, -5312, 5322, -5322, 5332, -5332, 5132, -5132, 5232,
-5232'
        elif OptionsName[0] == '2':
                list = '421, -421, 411, -411, 431, -431, ...'
        str = "Generation.Inclusive."+string.strip(ProductionValue)+".InclusivePIDList =
{"+list+"};\n"
        writeOptions(optionsdir,str)
# if Type Signal
    else:
        listing = {'11':'511,-511','12':'521,-521','13':'531,-531','14':'541,-541','15':'5122,-
5122','19':'521, -521, 511, -511, 531, -531, 541, -541, 5122, -5122, 5332, -5332, 5132,
-5132, 5232, -5232','21':'411,-411','22':'421,-421','23':'431,-431','24':'443,-433','25':'4122,-
4122','40':'25,-25'}
        if listing.has_key(AB):
            str = "Generation."+string.strip(sample)+"."+string.strip(ProductionValue)
+".SignalPIDList = {"+listing[AB]+"};\n"
            writeOptions(optionsdir,str)

# write Clean lines
    if clean == "true":
        str = "Generation."+sample+".Clean = true;\n"
        str = str + "GeneratorToG4.HepMCEventLocation =
\"/Event/Gen/SignalDecayTree\";\n"
        writeOptions(optionsdir,str)

#  test if a Decay could have a clean event type
    if clean == "false":
        checkClean = OptionsValue(decdir,"Clean:")
        if checkClean == "Yes":
            commandline = command+" "+DecayName+" clean"
            os.system(commandline)
    writeBkkTable(OptionsName,AsciiName,OptionsNick,clean)
    writeSQLTable(OptionsName,AsciiName,OptionsNick,clean)

# Check if exists ParticleTable keyword
    arg = OptionsValue(decdir,"ParticleTable:")
    if arg != None:
        if len(string.strip(arg)) == 0:
            print "WARNING: The Keyword ParticleTable has no value\n"
        str = "ParticlePropertySvc.ParticlePropertiesFile = 
\"$PARAMFILESROOT/data/ParticleTable,"+string.strip(arg)+".txt\";"
        writeOptions(optionsdir,str)

#
#  loop in the DKFILES directory to generate the options file
#
def run_loop():
    files = os.listdir(os.environ["DECFILESROOT"]+"/dkfiles/")
    for f in files:
        res = re.search('.dec',f)
        if res is not None:
            basefile = f.split('.')[0]
            commandline = command+" "+basefile
            os.system(commandline)



#
# give the usage of the command
#
def usage():
    print "This command should be used with the name of the decay file\n"
    print "create_options.py DECAY_NAME\n\n"
    return 0

#
#  Main
#

loop = "false"
clean = "false"
command = sys.argv[0]
EvtGenMain = "Generation."
```

```
optionsdirroot = os.environ["DECFILESROOT"]+"/options/"

# test number of argument
if len(sys.argv) > 3:
   usage()
   sys.exit(2)
else:
   if len(sys.argv) == 1:
      loop = "true"
   elif len(sys.argv) == 2:
      DecayName = sys.argv[1]
   else:
      DecayName = sys.argv[1]
      if sys.argv[2].lower() == "clean":
         clean = "true"
      else:
         print "Option "+sys.argv[2]+" unknown\n"
         sys.exit(2)


if loop == "true":
   run_loop()
   sys.exit(0)
else:
   run_create(clean)
```