How to Use MC++

MC++: C++ interface to event generators

Two basic classes:

- **MCparticle: contains particle 4-vector and pointers to:**
  - parent vertex
  - daughter vertex
  - particle type information
- **MCvtx: contains vertex 4-vector (x, y, z and ct) and lists of pointers to:**
  - parent particles
  - daughter particles

The event is a linked chain of MCparticle and MCvtx objects.
Partons and particles are treated as the same objects.

Additional container classes:

- MCevtInfo: general event information
- MCevt: container of all event information-
  MCevtInfo + related MCparticle and MCvtx objects
- MCptype: static particle type information (id, charge, lifetime).

There is also an option to make jets using a simple cone algorithm. The resulting jets are in a list of MCjet objects stored in MCevt.

The basic library is mcpp. It must be linked with isajet as it makes use of isajet utility functions. From the basic classes we derive DØ classes to be able to use mcpp with edm and d0om.
The derived classes are in d0_mcpp.

In mcpp there are in addition a number of event generator classes and a number of utility classes for manipulating particle and vertex lists.
See mcpp/doc/mcpp_user_guide.ps for more details.

Simple program to generate and write ISAJET events

```
#include  "mcpp/MCgenIsajet.hpp"

//       program to generate and write only


int main(){


  //  Isajet generator


  MCgenIsajet isagen;


  // generate and write events
  // writes to standard ISAJET file (not d0om!)


   isagen.genAndwrite(done);
  return 0;
}
```

Program will request a decay table and an ISAJET command file.
Note: ISAJET does not generate vertices

Program to read an ISAJET file and generate an MC++ event:

```
#include "mcpp/MCreadIsajet.hpp"
#include "mcpp/MCgetEvent.hpp"
#include "mcpp/MCgenInterPoint.hpp"

int main(){

  bool done=false;
  int i=0;


  //  reader of Isajet events from file


  MCreadIsajet isaread;


  //  define interaction region


  MCgenInterPoint gip(xoff,yoff,zoff,sigx,sigy,sigz,xsl,ysl);


  // read ISAJET file and generate MC++ event


  MCgetEvent mcevent(&isaread, &gip);
```

```
  do {

    i++;
    bool ok=true;


    //  generate an MC++ structure


    MCevt *ptevt=mcevent.generate(ok,done);

    if(!ok){
      cout<<" FAILED ON EVENT "<<i;
      return i;
    }
    delete ptevt;
  }while(!done);

  return 0;
}
```

If instead of reading an Isajet file you want to generate the events in the same job just replace MCreadIsajet with MCgenIsajet. Given a pointer to MCevt (ptevt) the particle and vertex information is accessible as an STL container of constant pointers ordered by decreasing p_T. At present the only STL container is a list but there will soon be a release where it can be also be a vector or a set.

```
list<const MCparticle*> listp;
ptevt->getParticles(listp);

list<const MCvtx*> listv;
ptevt->getVertices(listv);
```

listp is an ordered list in descending order of $\mathrm{p_T}$. To find the highest $\mathrm{p_T}$ electron or positron in the list:

```
list<const MCparticle*>::const_iterator jp;
bool found=false;
jp=listp.begin();
while (!found && jp !=listp.end()){
   if(abs((*jp)->isaId()) == 12) found=true;
   else ++jp;
}
```

One can get a filtered list by

```
#include "SomeFilter.hpp"
```

SomeFilter.hpp must inherit `mcpp/MCparticleFilter.hpp`

```
SomeFilter filter;
ptevt->getParticles(&filter,listp);
```

You can filter listp again and store the result in a particle container MCparticleList.

```
AnotherFilter filter2;
MCparticleList(&filter2,listp) mcpl;
list<const MCparticle*> listp2;
mcpl->getParticles(listp2);
```

The point of using MCparticleList is that you also get additional methods (such as adding or subtracting) for manipulating lists. The methods assume the particles are ordered in p_T. Most methods in MCparticleList have static counterparts so you do not have to create an MCparticleList if you don't need it.

ChargedLeptonFilter
StableParticleFilter
FinalPartonFilter
ShortLivedFilter
DisplacedVtxFilter (for MCvtx)

Example: How to find the stable particles from a W

```
#include "mcpp/StableParticleFilter.hpp"
#include "mcpp/MCparticleList.hpp"

   list<const MCparticle*>::const iterator jp;
   bool found=false;
   while(!found && jp !=listp.end()){
     if(abs((*jp)->isaId()) == 80) found=true; // find W
     else ++jp;
   }

  // get vertex W points to

  MCvtx *vtxptr=ptr->vtxend();

  // get all W descendants

  list<const MCparticle*> listw;
  vtxptr->all_descendants(listw);
```

```
// get a filter
// (should be called in a constructor outside event loop)
// and make a container of filtered list

StableParticleFilter stfilter;
MCparticleList wstable(&stfilter, listw);

// print the contents of list in wstable

wstable.print(cout);

// get list of stable particles in W

list<const MCparticle*> lstablew;
wstable.getParticles(lstablew);
```

To generate events in edm/d0om format use
bin/IRIX6-KCC_3_3/MCisagen.x
It needs from d0_mcpp/rcp :
runMCisagen.rcp, mcvertex.rcp
MCNewEvent.rcp, MCDumpEvent.rcp, WriteEvent.rcp

Executable is made with a package in official framework. Sources for d0_mcpp packages are in d0_mcpp/src/package
The generated event information is in
MCKineChunk

To get a pointer (ptevt) to MCevt

```
#include  "d0_mcpp/MCKineChunk.hpp"
#include  "emd/TKey.hpp"
#include  "edm/THandle.hpp"

  TKey<MCKineChunk> mcKey;
  THandle<MCKineChunk> mcchunk=mcKey.find(event);
  const MCevt* ptevt=mcchunk->ptrMCevt();
```

Given the pointer you can proceed as with standard MC++ but you must not delete ptevt as it is managed by Event class.

At present full implementation exists only for ISAJET. However, there is a class MCgenIsaStdHep.cpp that shows implementation for ISAJET using Std Hep format. It is trivial to implement another class for any other generator that has an Std Hep interface.