TUTORIAL MUONTOMOGRAPHY - v.1.2

Este tutorial e o kit da simulação estão disponíveis em:
https://wiki-lip.lip.pt/Projectos/MuonTomography/MuonTutorial
Para entrar em contacto: pmmt@uevora.pt

CONTEÚDO

A aplicação MuTomo, aqui apresentada, corre em GEANT4 e está desenhada para simular a trajetória de partículas, neste caso de muões. Este tutorial considera que a instalação e a execução da aplicação MuTomo será feita em máquinas do servidor Pauli e que os utilizadores possuem credenciais de acesso a este servidor e uma diretoria de trabalho. Essa diretoria poderá ou não ficar localizada na Lstore, que de uma forma ou de outra, será necessário ter também uma diretoria na Lstore, pois devido à uma maior capacidade de armazenamento é para lá que deverão ser enviados os resultados das simulações executadas na farm. Os utilizadores do grupo LIP Cosmo poderão usar o servidor Auger, exclusivo do grupo, correndo tudo da mesma forma como em Pauli. As máquinas destes servidores já possuem o GEANT4 e o ROOT instalados.

Este tutorial guia o utilizador através de uma simulação teste interativa e da submissão de uma simulação teste na farm, que servem não só para dar a conhecer os procedimentos envolvidos como também para configurar a aplicação para a primeira utilização e posteriores. Se surgirem erros durante o processo, poderão usar o contacto apresentado na primeira página para obter informações.

0. Acesso ao Servidor

0.1. Numa janela shell aceder ao servidor do LIP usando o login pessoal:

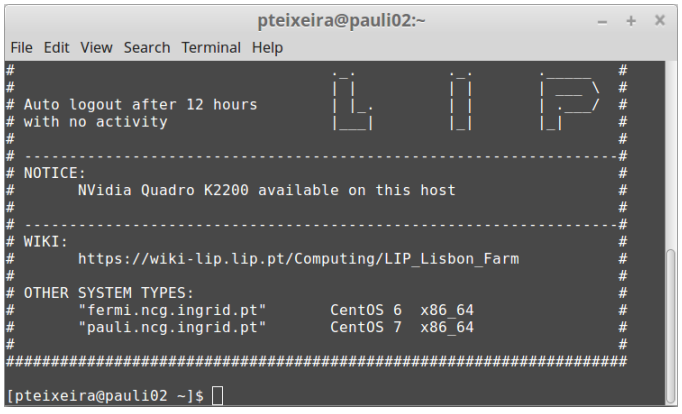
```
$ ssh -X username@lnlip01.lip.pt
```

Usando o próprio username e introduzindo a password solicitada.
O acesso pode ser feito com outros endereços, "lnlip01.lip.pt" aparece aqui como exemplo.

0.2. Aceder ao servidor do Pauli (ou Auger) usando o login pessoal:

```
$ ssh -X username@pauli.ncg.ingrid.pt
ou
$ ssh -X username@auger.ncg.ingrid.pt
```

Usando o próprio username e introduzindo a password solicitada



Janela shell do Linux com início de sessão no servidor Pauli

1. Instalação da Aplicação MuTomo

1.1. Copiar o MuonTomography.tar para a diretoria de trabalho. Se for necessário copiar o ficheiro do computador para o servidor, abrir uma outra janela shell na localização onde se encontra o ficheiro no computador e introduzir:

```
$ scp MuonTomography.tar username@pauli.ncg.ingrid.pt:/home/cosmo/pteixeira/.
```

Substituir pelo próprio username

Substituir aqui pelo caminho da diretoria de trabalho para onde vai ser copiado o ficheiro

O caminho de uma diretoria pode ser consultado introduzindo dentro da própria diretoria o comando:

```
$ pwd
```

1.2. Na janela iniciada no servidor Pauli e com o ficheiro MuonTomography.tar dentro da diretoria de trabalho, extrair a pasta MuonTomography com o seguinte comando:

```
$ tar xvf MuonTomography.tar
```

1.3. Entrar na pasta MuonTomography e criar diretoria build:

```
$ cd MuonTomography
$ mkdir build
```

Usando o comando **ls** ou **ll** é possível listar o conteúdo da diretoria e ver que contém o seguinte:

```
0setup.gdml   analysis        include           read_gdml.mac   setprocesses.mac   tofarm
1blank.gdml   build           lip-conf.sh       README.md       source.mac         vis.mac
1BMine.gdml   CMakeLists.txt   materials.gdml    read_step.mac   src                write_gdml.mac
2SphDen.gdml  data            MuonTomography.cc ROOT            test.gdml
3Sphs.gdml    History         MuTomo.mac.in     run.mac         testprocesses.mac
```

O conteúdo desta pasta será explicado em maior detalhe no ponto 5.2.

1.4. Carregar os ambientes do GEANT e do ROOT:

```
$ source lip-conf.sh
```

→ Este script carrega os módulos GEANT4 10.6.0, ROOT 6.18/04 e cmake 3.11.2

(devido a uma incompatibilidade detetada, não é garantido que a aplicação MuTomo funcione em Pauli com outros ambientes de GEANT e ROOT)

1.5. Entrar na pasta build e correr cmake:

```
$ cd build
$ cmake ..
```

→ A configuração termina com sucesso se apresentar no fim o seguinte:
-- Configuring done
-- Generating done
-- Build files have been written to: [current path to build directory]

```
$ make
```

→ A instalação estará completa quando indicar 100%

Nesta etapa o conteúdo da pasta build deverá ser o seguinte:

```
0setup.gdml       3Sphs.gdml           dict.cxx           libEventROOT.so  MuTomo.mac.in  setprocesses.mac
1blank.gdml       CMakeCache.txt       Hits.txt           Makefile         read_gdml.mac  source.mac
1blank_mine.gdml  CMakeFiles           libdict_rdict.pcm  materials.gdml   read_step.mac  vis.mac
2SphDen.gdml      cmake_install.cmake  libdict.rootmap    MuTomo           run.mac        write_gdml.mac
```

Após isto a instalação da aplicação MuTomo deverá estar pronta a executar uma simulação. As simulações são executadas dentro da pasta build.

Sempre que for necessário atualizar ou fazer um reset à pasta build, basta voltar a executar o comando **make**. Isto significa que os ficheiros necessários para correr a aplicação MuTomo serão novamente copiados da pasta anterior (MuonTomography) e todas as alterações inseridas nos ficheiros dentro da

pasta build serão perdidas. Por isso mesmo, todas as alterações importantes devem ser inseridas nos ficheiros correspondentes na pasta MuonTomography, para serem copiados para a pasta build, no caso de ser necessário atualizar a pasta.
Contudo, sempre que se iniciar novo login no servidor e houver necessidade de executar de novo os comandos **cmake** e **make**, os módulos do ambiente de GEANT e de ROOT têm de ser carregados novamente usando o comando **source lip-conf.sh**.

2. Simulação Teste

Os ficheiros utilizados neste teste já contêm parâmetros pré-definidos para uma pequena simulação. Estes parâmetros poderão ser posteriormente alterados e isso será explicado mais adiante no ponto 5.
De igual forma, como foi dito no ponto anterior, cada vez que se entre de novo no servidor, usando o login pessoal, e se queira executar uma simulação no modo interativo, isto é, usando manualmente a aplicação MuTomo dentro da pasta build, como demonstrado a seguir, é necessário carregar sempre o ambiente de GEANT e de ROOT com o comando:

```
$ source lip-conf.sh
```

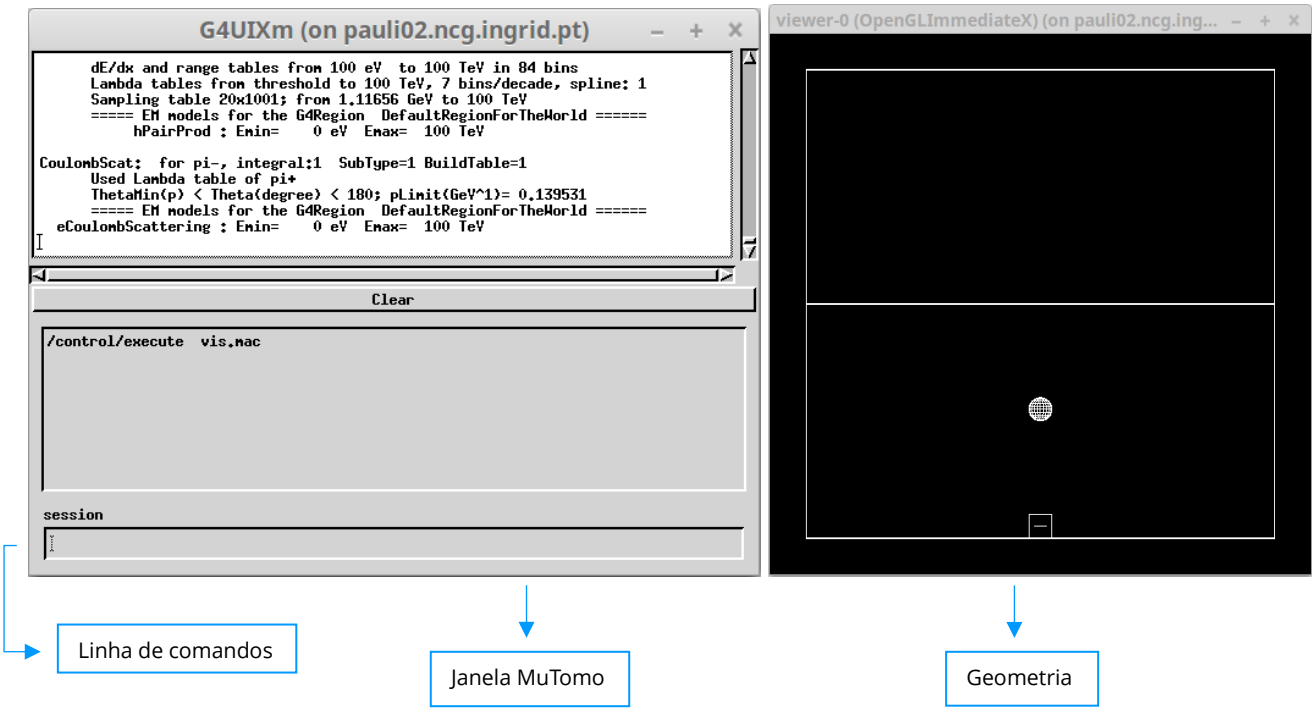
2.1. Para verificar se ficou tudo bem configurado, vamos visualizar a geometria da simulação, começando por inserir o comando:

```
$ MuTomo
```

→ Quando inserido sozinho, este comando abre a aplicação MuTomo numa nova janela

2.2. Na janela MuTomo executar a macro de visualização com o comando:

```
$ /control/execute vis.mac
```



O MuTomo irá executar o comando e abrir uma outra janela apresentando a geometria da simulação (neste caso, esta geometria está definida no script 2SphDen.sh) que contém uma esfera no meio de um volume retangular de terra, cujo nível da superfície é a linha que divide o quadrado exterior, e um detetor em baixo (pequena linha) dentro de galeria subterrânea (quadrado). A geometria está representada em 3D, mas por causa do ângulo de visualização escolhido na macro vis.mac, aparenta estar representada em 2D. É possível mudar o ângulo de visualização, como indicado mais à frente no ponto 5.2.1.. Depois de verificar que o visualizador abre, podem fechar-se estas duas janelas e passar para o passo seguinte.

2.3. De volta à pasta build, inserir o comando:

```
$ MuTomo run.mac
```

Isto irá correr a macro run.mac que tem definidos os parâmetros para uma pequena simulação com 10 milhões de muões injetados no volume da terra. Após a execução desta macro, uma série de informações relacionadas com a simulação será apresentada, terminando com:

```
### Run 0 start.
```

Nesse momento a simulação estará a ser executada e será necessário aguardar. O programa demora à volta de 36 segundos a simular 1 milhão de muões. Portanto, para simular os 10 milhões de muões irá demorar cerca de 5/6 minutos. Quando terminar irá apresentar as seguintes mensagens:

```
ROOT file Tree.root saved and closed.
Running time : User=348.730000s Real=361.182767s Sys=4.900000s
Graphics systems deleted.
Visualization Manager deleting...
```

Quando terminar, um ficheiro Tree.root será criado na pasta build com a informação da simulação executada. Na verdade, o ficheiro Tree.root também foi criado no passo anterior, na execução de vis.mac, mas não houve injeção de muões e por isso o ficheiro foi ignorado. A execução de run.mac reescreveu o ficheiro Tree.root anterior com a informação desta simulação de teste.

2.4. A análise do ficheiro Tree.root é feita na pasta Analysis (localizada dentro da pasta MuonTomography), por isso, primeiro é necessário copiá-lo para lá e depois entrar nessa pasta:

```
$ cp Tree.root ../analysis/.
$ cd ../analysis/
```

```
DumpEvent.C  mergeFiles.sh  mergeTrees.sh  PlotHits.C  PlotMutomo.C  Status.key  Tree.root
```

2.5. Usando o ROOT, executa-se o script DumpEvent.C para ler os eventos registados em Tree.root:

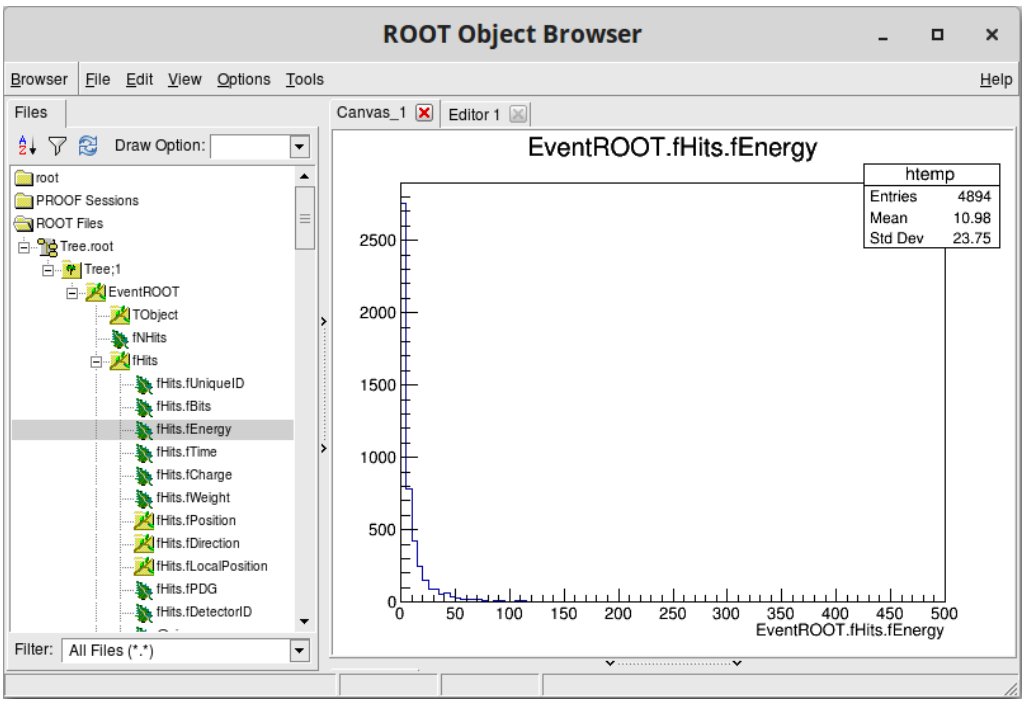
```
$ root -l DumpEvent.C
```

O DumpEvent.C irá criar um ficheiro Hits.txt, na pasta Analysis, para onde serão copiadas as coordenadas das direções x e y de todos os muões que atravessaram o detetor.

2.6. Aproveitando que se está dentro do ROOT, é possível explorar o ficheiro Tree.root para ver outras informações sobre os muões da simulação, fazendo o seguinte:

```
root [1] TFile *file0 = TFile::Open("Tree.root")
root [2] TBrowser *TB = new TBrowser()
```

Este comando abre a janela de exploração



Esta é a janela de exploração do Tree.root mostrando, por exemplo, a energia (em GeV) dos muões detetados na simulação.

Para sair do ROOT e voltar à pasta Analysis, inserir:

```
root [3] .q
```

Ao ver o conteúdo da pasta Analysis, o ficheiro Hits.txt recém-criado já aparece listado com os restantes ficheiros.

```
DumpEvent.C  Hits.txt  mergeFiles.sh  mergeTrees.sh  PlotHits.C  PlotMutomo.C  Status.key  Tree.root
```

Com as coordenadas das direções x e y contidas no ficheiro Hits.txt é possível desenhar um histograma 2D com a representação espacial das deteções.

2.7. Para representar o histograma usa-se PlotHits.C, mas primeiro é necessário abri-lo e inserir a localização atual do ficheiro Hits.txt. Uma forma de fazer isso é usar um editor de texto, como por exemplo o Emacs, que será usado neste tutorial, o qual se já não estiver presente no sistema, pode ser instalado.

Nota: no caso de se usar Emacs, dependendo da versão, poderá ocorrer uma incompatibilidade com o ambiente ROOT carregado (ROOT 6.18/04), devido a este usar uma versão de Python diferente do

sistema CentOS 7 onde está a correr o servidor. Nesse caso uma forma de contornar o problema é usar uma outra janela shell ligada ao servidor Pauli, onde ainda não se carregou os ambientes de GEANT e de ROOT e usá-la para abrir os ficheiros com o Emacs. Isto só será necessário em momentos como este, em que é preciso carregar os ambientes para correr uma simulação em modo interativo e introduzir alterações nos ficheiros.

Assim, abre-se PlotHits.C para alterar a localização usando:

```
$ emacs PlotHits.C
```

File Edit Options Buffers Tools C++ Help

Save Undo

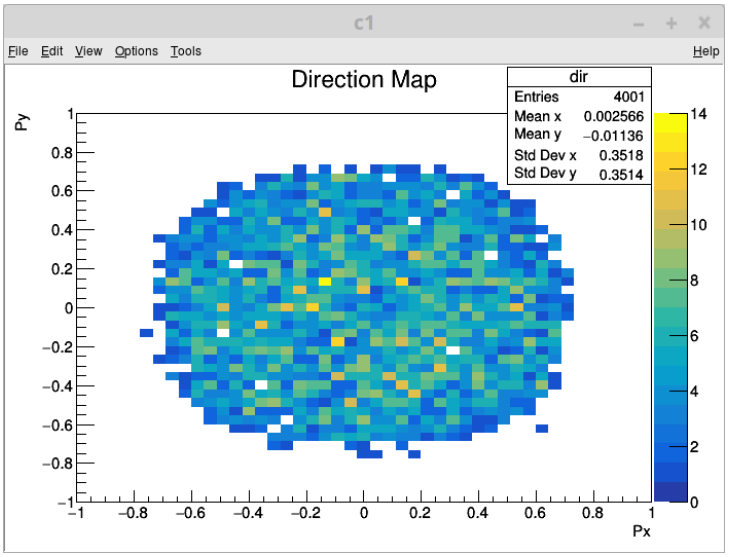
```
void PlotHits()
{
        int NbinDir=45;
        TH2F *dir=new TH2F("dir","Direction Map; Px; Py",NbinDir,-1.,1.,NbinDir,-1.,1.);

        float px,py;
        int Nmuons = 0;

        TTree *T=new TTree("ntuple","data from ascii file");
        Nmuons=T->ReadFile(Form("/home/cosmo/pteixeira/MuonTomography/analysis/Hits.txt"),"px:py");

        T->Project("dir","py:px");

        cout<<"Read "<<Nmuons<<" muons"<<endl;
        new TCanvas; dir->Draw("colz");
        gStyle->SetPalette(55); // set color map to rainbow

        TFile *keep=new TFile(Form("Hits.root"),"recreate");
        dir->Write();
        keep->Close();
}
```

Alterar para a localização atual de Hits.txt, na pasta Analysis

-:--- **PlotHits.C** All L24 (C++/l Abbrev)

2.8. Agora o ficheiro Hits.txt será detetado e poderá ser representado graficamente num histograma 2D, executando PlotHits.C através do ROOT:

```
$ root -l PlotHits.C
```



O histograma resultante apresenta um mapa das direções, numa malha de 45x45 bins, dos muões que atravessaram o detetor. Para esta simulação teste, o terreno possui uma densidade de 2.6 g/cm^3 e a esfera foi preenchida por ouro, com densidade 19,3 g/cm^3. Contudo, para a área do terreno simulado, a passagem de 10 milhões de muões corresponde a uma exposição de cerca de 3 minutos em tempo de observação real, que mesmo com a grande diferença entre as densidades da esfera e do terreno, não é suficiente para distinguir claramente a esfera. Para auxiliar neste processo, usa-se a farm, onde é possível correr várias simulações ao mesmo tempo, em várias máquinas, juntando todos os resultados no final.

Após executar PlotHits.C, o histograma será guardado num novo ficheiro designado Hits.root criado na pasta Analysis.

```
DumpEvent.C  Hits.root  Hits.txt  mergeFiles.sh  mergeTrees.sh  PlotHits.C  PlotMutomo.C  Status.key  Tree.root
```

Este ficheiro é útil para guardar o resultado das simulações, ocupando pouco espaço. Poderá ser aberto novamente com os comandos:

```
$ root -l Hits.root
root [1] dir->Draw("colz");
```

3. Configurar Submissão para a Farm

Para correr estas simulações na farm é necessário usar uma diretoria na Lstore, devido à sua maior capacidade para armazenar a grande quantidade de ficheiros que podem ser gerados. Dentro dessa diretoria é preciso criar duas pastas, uma que irá receber os scripts de cada simulação que for enviada para a farm, assim como os ficheiros de erro e de output correspondentes, e outra pasta onde serão depositados os ficheiros com as coordenadas das direções x e y das deteções de cada simulação.

3.1. Dentro da diretoria escolhida na lstore, criar as seguintes pastas:

```
$ mkdir dump_dir
$ mkdir runscripts
```

Agora é preciso ligar a localização destas pastas aos scripts relacionados com a submissão das simulações à farm.

3.2. Voltando à diretoria MuonTomography, os ficheiros usados para submeter à farm estão na pasta tofarm:

```
$ cd tofarm
```

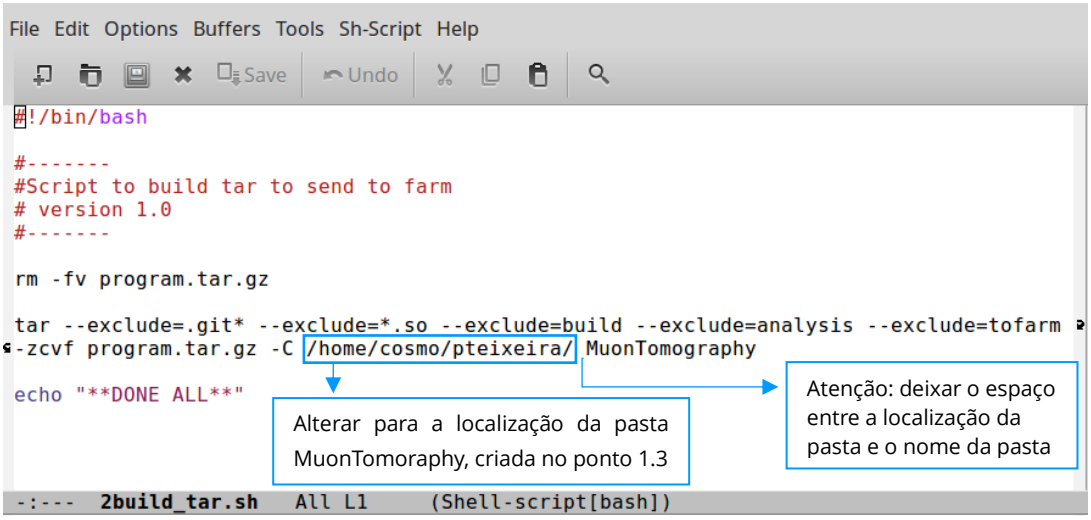
```
1status.sh  2build_tar.sh  3apagar.sh  5offl_submit.sh  RunStandalone.sh  steerStandalone.sh
```

Usando o editor de texto para abrir os ficheiros, será então necessário atualizar algumas localizações de diretorias.

3.3. Abrir e editar 2build_tar.sh:

```
$ emacs 2build_tar.sh
```

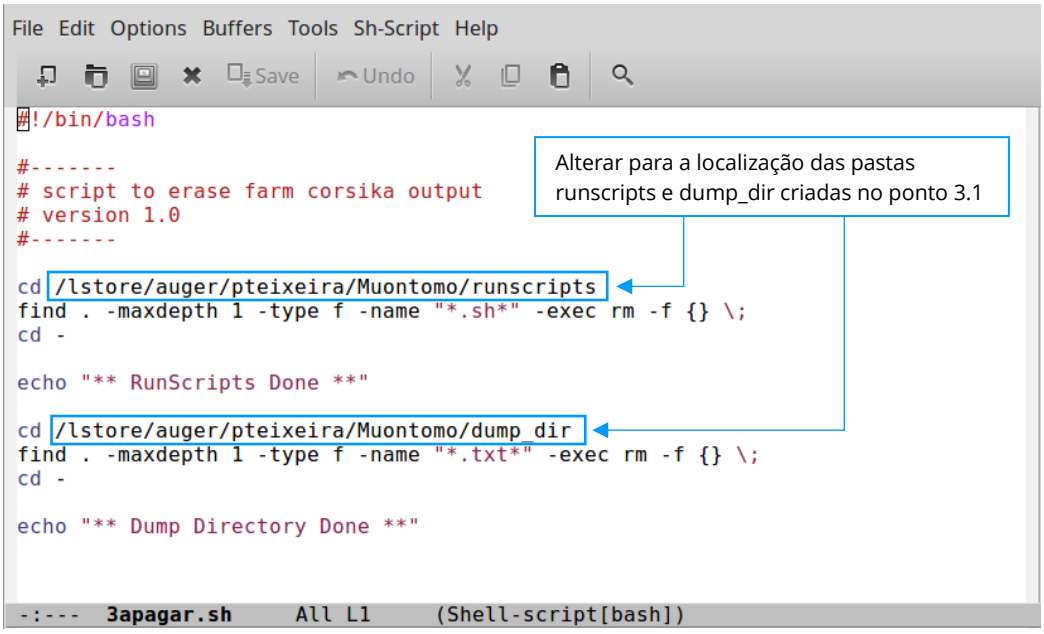
Este script compacta todos os ficheiros necessários para serem enviados para a farm e cria a pasta compactada program.tar.gz dentro da pasta tofarm.



3.4. Abrir e editar 3apagar.sh:

```
$ emacs 3apagar.sh
```

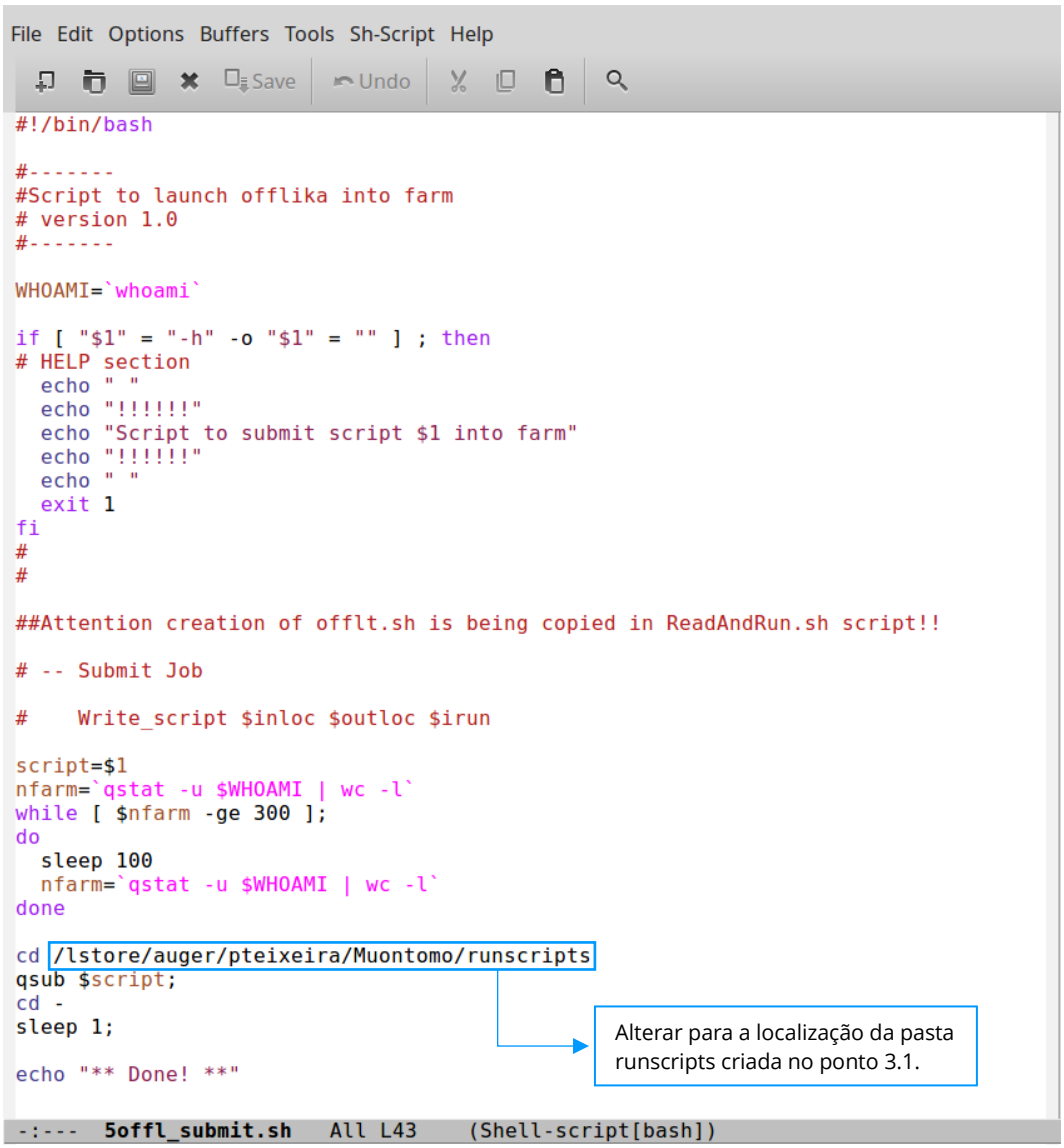
Este script serve para eliminar o conteúdo das pastas dump_dir e runscripts depois de já se ter retirado a informação que se pretendia e já não se precisarem dos ficheiros. Convém esvaziar estas pastas antes de se correrem novas simulações para os dados não se misturarem.



3.5. Abrir e editar 5offl_submit.sh:

```
$ emacs 5offl_submit.sh
```

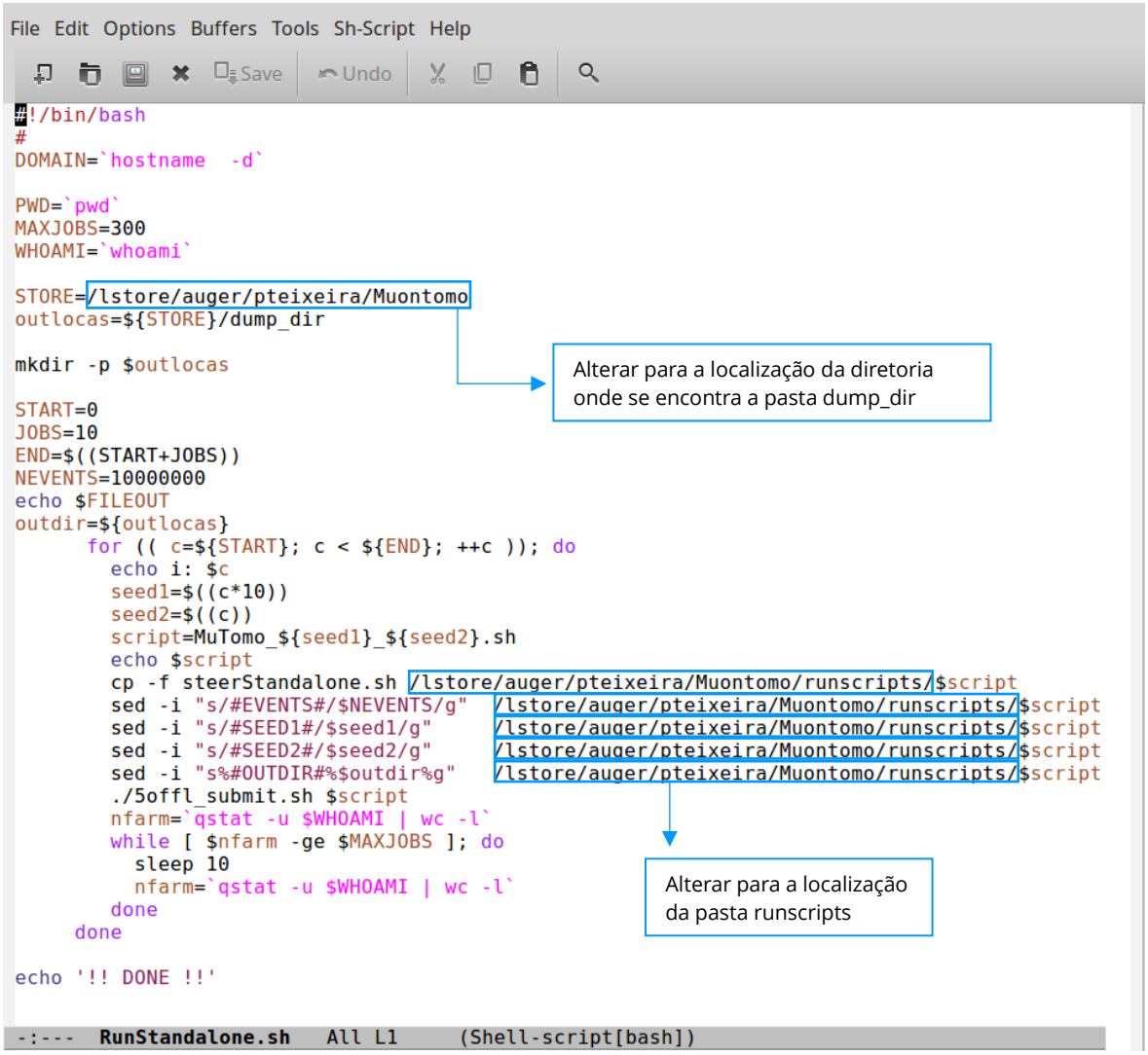
Este é o script que lança os jobs com as simulações para a farm em modo offline.



3.6. Abrir e editar RunStandalone.sh:

```
$ emacs RunStandalone.sh
```

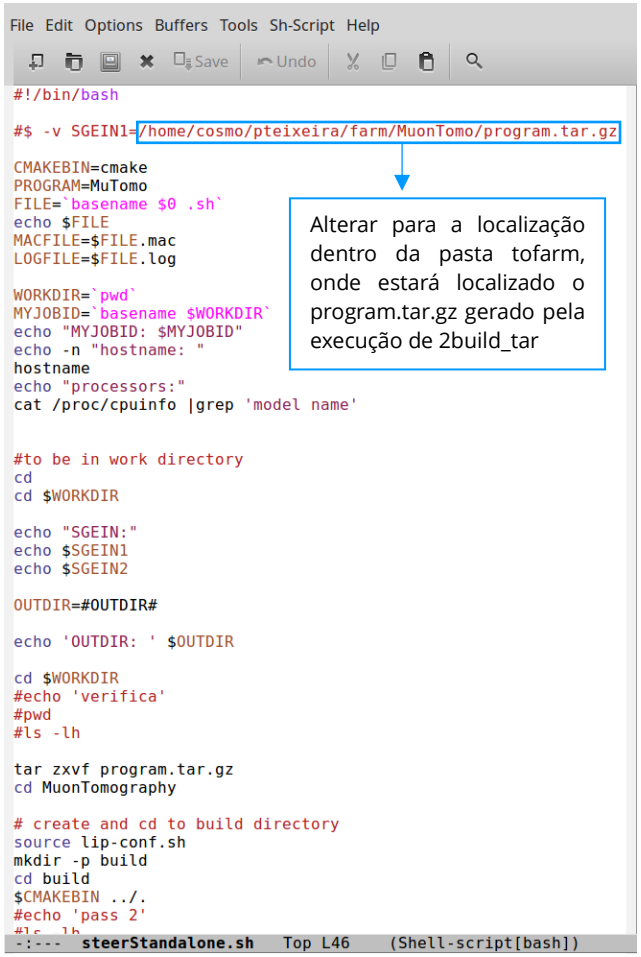
Este é o script que se executa para submeter os jobs com as simulações para a farm.

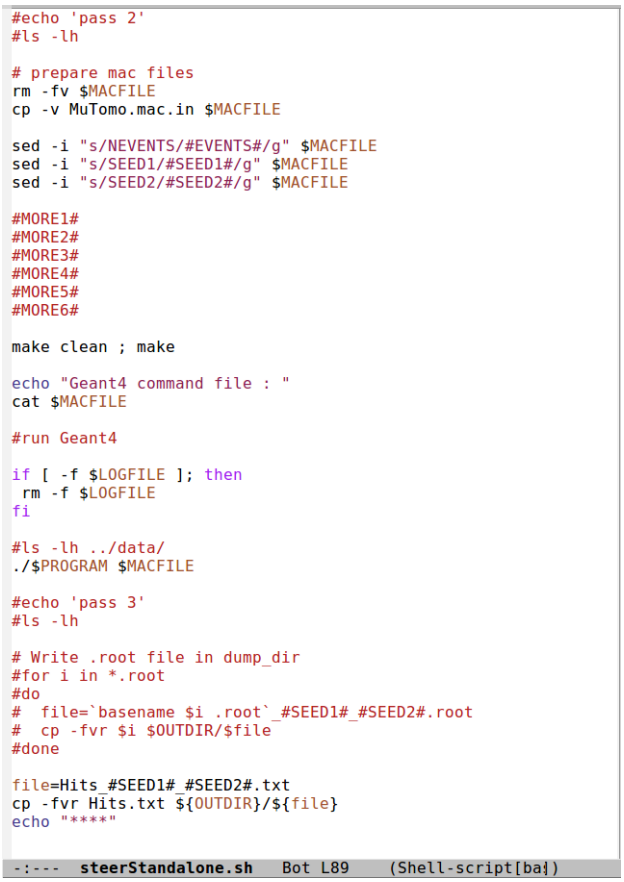


3.7. Abrir e editar steerStandalone.sh:

```
$ emacs steerStandalone.sh
```

Este script processa as simulações dentro da farm e copia para a pasta dump_dir os ficheiros com as coordenadas das deteções.





4. Submeter Simulações para a Farm

Com a configuração terminada, pode-se fazer a submissão de várias simulações com a mesma geometria da simulação teste usada no ponto 2, neste caso, um terreno com uma esfera de ouro no interior, para usar como comparação.

4.1. Dentro da pasta tofarm, compactar a pasta MuonTomography executando 2build_tar.sh:

```
$ 2build_tar.sh
```

Isto irá criar o ficheiro program.tar.gz dentro da pasta tofarm, contendo todos os ficheiros necessários para correr as simulações na farm.

4.2. Submeter os jobs com as simulações executando RunStandalone.sh:

```
$ RunStandalone.sh
```

Este script já tem inserido a informação para submeter 20 jobs com 10 milhões de eventos, ou injeções de muões. Na janela shell irá ser apresentada a submissão dos jobs, um de cada vez, até todos serem submetidos.

4.3. Para verificar o estado da execução dos jobs na farm, usar:

```
$ 1status.sh
```

Este script verifica o estado da submissão na farm e indica quantos jobs estão em espera e em quantos estão em execução, apresentando a seguinte mensagem:

```
Jobs running...
20
Jobs waiting...
0
***
```

Quando apresentar 0 em ambas as mensagens, a execução terminou.

4.4. Consultar a past dump_dir e verificar se contém 20 ficheiros Hists_#_#.txt, onde o simbolo # representa a numeração de identificação e também as seeds com que foram submetidos à farm.
Se os 20 ficheiros estiverem presentes, a execução correu com sucesso. Se não, consultar os ficheiros de erro (.e) na pasta runscripts e verificar o que sucedeu. A pasta runscripts terá sempre um ficheiro de erro (.e) e outro de output (.o) para cada job submetido para a farm, quer tenho sido executado com sucesso ou não. Os ficheiros de output contêm as mesmas informações sobre a simulação como foram apresentadas no ponto 2.3, após a execução de run.mac.

Quando uma pasta tem muitos ficheiros, uma forma útil de verificar a quantidade é usar um comando que devolve o número de ficheiros contidos nessa pasta, por exemplo:

```
$ ls -l | grep ^- | wc -l
```

4.5. Após ter os 20 ficheiros Hists_#_#.txt. com as coordenadas das deteções, só falta agora juntá-las todas. Para isso é necessário voltar à pasta Analysis e correr o script mergeFiles, indicando a localização da pasta dump_dir, onde se encontram os ficheiros a juntar:

```
$ mergeFiles.sh path_to_dump_dir
```

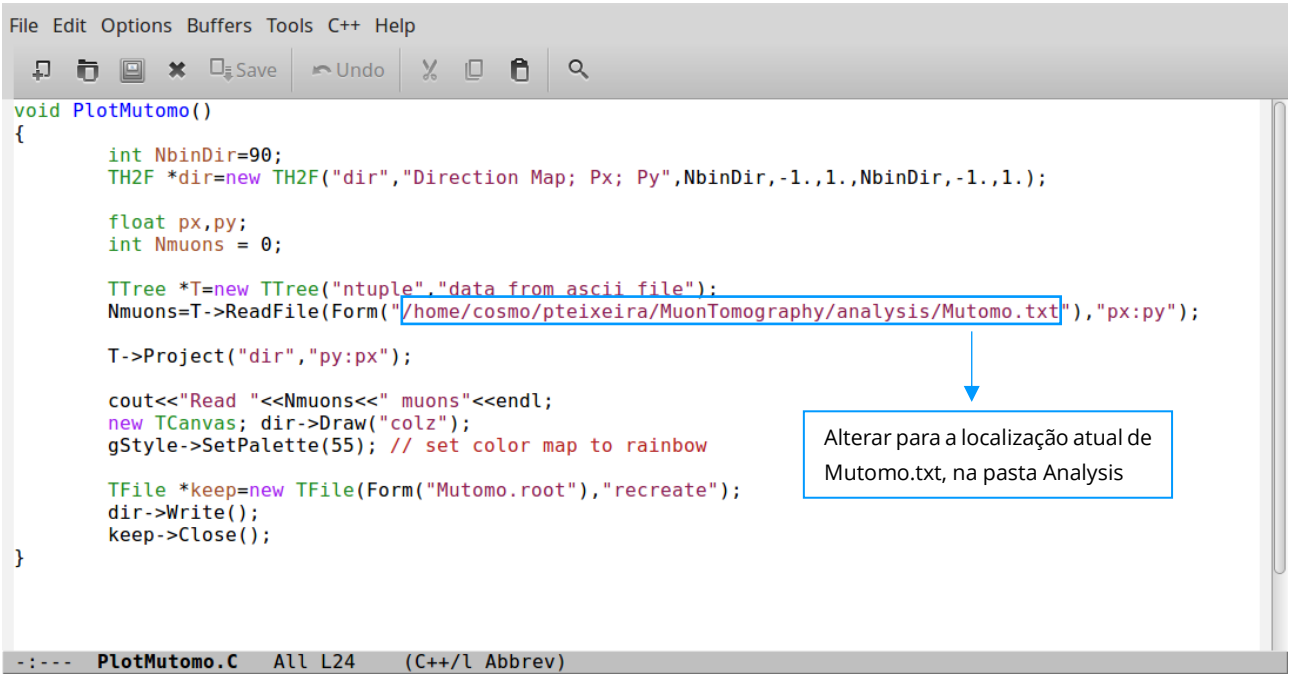
→ Substituir path_to_dump_dir pela localização da pasta dump_dir

Isto irá criar o ficheiro Mutomo.txt que contém as coordenadas juntas dos 20 ficheiros com coordenadas que resultaram das simulações.

```
DumpEvent.C  Hits.txt       mergeTrees.sh  PlotHits.C    Status.key
Hits.root    mergeFiles.sh  Mutomo.txt     PlotMutomo.C  Tree.root
```

4.6. Agora já é possível criar um histograma com toda a informação correndo PlotMutomo.C, mas tal como no ponto 2.7, também é necessário atualizar a localização do ficheiro Mutomo.txt, por isso:

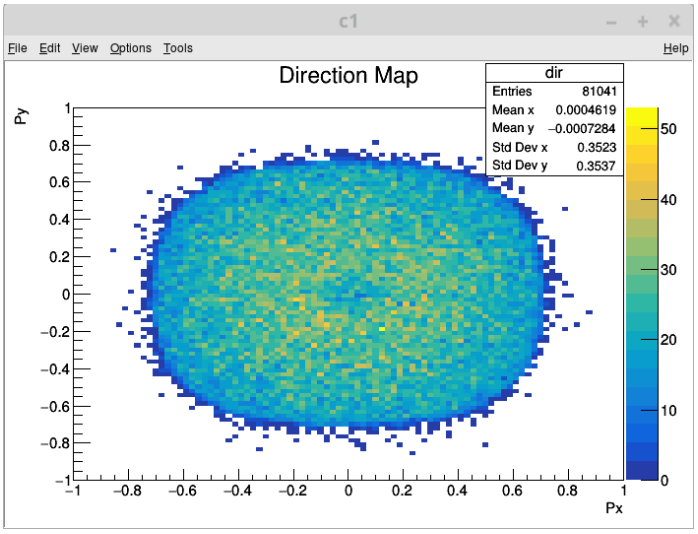
```
$ emacs PlotMutomo.C
```



4.7. Agora o ficheiro Mutomo.txt será detetado e poderá ser representado graficamente num histograma 2D, executando PlotMutomo.C através do ROOT:

```
$ root -l PlotMutomo.C
```

O resultado é um histograma com mais informação que a primeira tentativa que por isso usa uma malha de 90x90 bins, onde a presença da esfera está visivelmente mais clara no centro. Elementos mais densos do que o meio envolvente irão ser identificados por um fluxo menor de muões vindos da sua direção, pois requerem muões com mais energia para serem atravessados e, estatisticamente, os muões com energias mais elevadas são menos frequentes, como foi visível na curva da energia dos muões injetados, apresentada no ponto 2.6. Pelo contrário, elementos menos densos que o meio envolvente deixarão passar mais muões e por isso o fluxo será maior na sua direção.



Como foi referido no ponto 2.8, para a área da geometria usada nesta simulação, os 10 milhões de muões simulados correspondiam a uma exposição de cerca de 3 minutos em tempo de observação real. Ao juntar-se a informação de 20 simulações, cada uma com 10 milhões de eventos, obtém-se o resultado de uma simulação com 200 milhões de eventos, dos quais apenas pouco mais de 80 mil muões atravessaram o detetor com 1 m de largura, colocado 19 metros abaixo da superfície do terreno. Esta quantidade de muões, para a área simulada, equivale a uma exposição de cerca de 1 hora em tempo de observação real, o que devido à grande diferença entre as densidades do ouro e do terreno é suficiente para visualizar de forma mais clara a esfera. Para outros materiais ou geometrias mais complexas, o tempo de exposição, traduzido na quantidade de muões injetados, terá de ser muito maior para se poderem distinguir os materiais/volumes diferentes.

Como anteriormente, após executar PlotMutomo.C, o histograma será guardado num novo ficheiro designado Mutomo.root criado na pasta Analysis.

```
DumpEvent.C  Hits.txt       mergeTrees.sh  MuTomo.txt    PlotMutomo.C    Tree.root
Hits.root    mergeFiles.sh  Mutomo.root    PlotHits.C    Status.key
```

Poderá ser aberto novamente com os comandos:

```
$ root -l Mutomo.root
root [1] dir->Draw("colz");
```

5. Parâmetros e Definições

Os pontos anteriores deste tutorial serviram para passar a ideia base do funcionamento e execução da aplicação MuTomo, usando como exemplo uma geometria que representa um terreno com o detetor no interior de uma mina. A maioria dos parâmetros e dados inseridos podem ser personalizados, por isso, esta secção aponta para mais detalhes de alguns dos ficheiros envolvidos nas simulações, indicando alguns locais nas linhas de código onde de pode pretender fazer alterações.

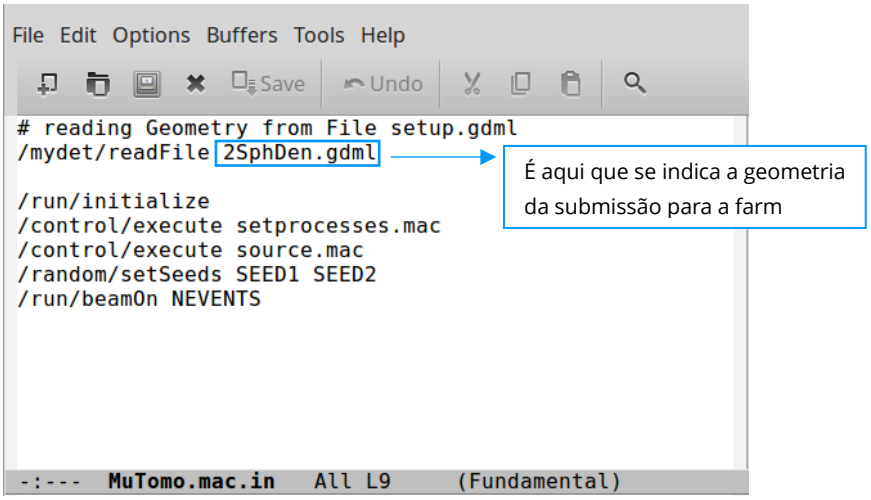
Após se fazerem alterações num ficheiro, é por vezes criada uma cópia dele mesmo antes da alteração, que mantém o mesmo nome mas com um til (~) acrescentado no fim. Pode ser eliminado, mas também é útil se desejarmos voltar atrás na alteração.

5.1. Processo de Submissão para a Farm

Este primeiro ponto irá resumir e expandir a informação sobre os parâmetros a ter em conta cada vez que se submete uma série de simulações para a farm.

5.1.1. Na pasta MuonTomography, abrir o ficheiro MuonTomo.mac.in e alterar a geometria a ser executada, quando necessário.

```
$ emacs MuonTomo.mac.in
```



Este passo é necessário na primeira vez que se usar uma geometria diferente da indicada atualmente no ficheiro, e subsequentemente se se se correrem simulações com geometrias diferentes. Caso contrário, o processo de submissão começará sempre com o ponto seguinte.

5.1.2. Já dentro da pasta tofarm, se necessário, limpar as pastas dump_dir e runscripts do conteúdo relativo a simulações anteriores:

```
$ emacs 3apagar.sh
```

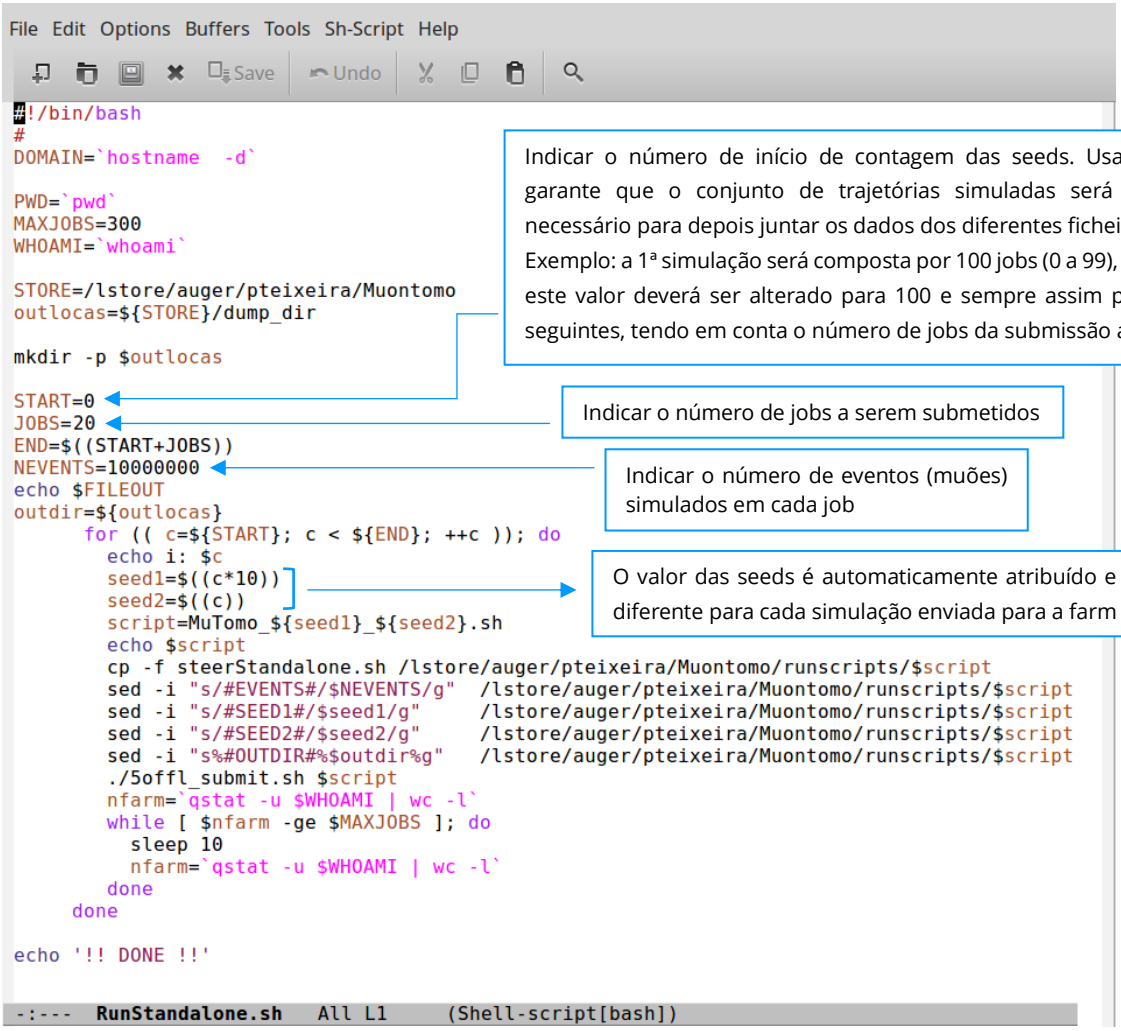
5.1.3. Executar 2build_tar.sh para criar o program.tar.gz:

```
$ emacs 2build_tar.sh
```

Este passo é necessário sempre que se altere algum ficheiro dentro da pasta MuonTomography, como, por exemplo, alterar uma dimensão ou um material no ficheiro da geometria.

5.1.4. Abrir RunStandalone.sh e inserir os dados da simulação:

```
$ emacs RunStandalone.sh
```



5.1.5. Depois executa-se RunStandalone.sh, mas para uma submissão com muitos jobs, poderá ser útil deixar a submissão ocorrer em background. Para isso existe um truque, além da execução simples:

$ **RunStandalone.sh**

ou

$ **nohup ./RunStandalone.sh &**

Se se pretender fechar a janela shell onde foi executada a submissão (ou até desligar o computador), é necessário usar este comando para que a submissão continue a ser executada. E diferente do caso simples, o registo da submissão será gravado num ficheiro com o nome nohup.out, criado na própria pasta

Acrescentar o símbolo & a seguir a qualquer comando, permite que a linha de comandos continue ativa. Algo útil como em casos que se usa o editor de texto para abrir um ficheiro e se pretende mantê-lo aberto enquanto se continua e usar a linha de comandos

5.1.6. Após isto, repetem-se os passos 4.3 a 4.7 (saltando o 4.6), sobre o tratamento dos dados.

5.2. Conteúdo da Pasta MuonTomography

Esta é a diretoria principal e contém a maioria dos ficheiros e pastas necessários para correr e executar as simulações.

```
0setup.gdml   analysis        include            read_gdml.mac  setprocesses.mac  tofarm
1blank.gdml   build           lip-conf.sh        README.md      source.mac        vis.mac
1BMine.gdml   CMakeLists.txt  materials.gdml     read_step.mac  src               write_gdml.mac
2SphDen.gdml  data            MuonTomography.cc  ROOT           test.gdml
3Sphs.gdml    History         MuTomo.mac.in      run.mac        testprocesses.mac
```

5.2.1. Geometrias

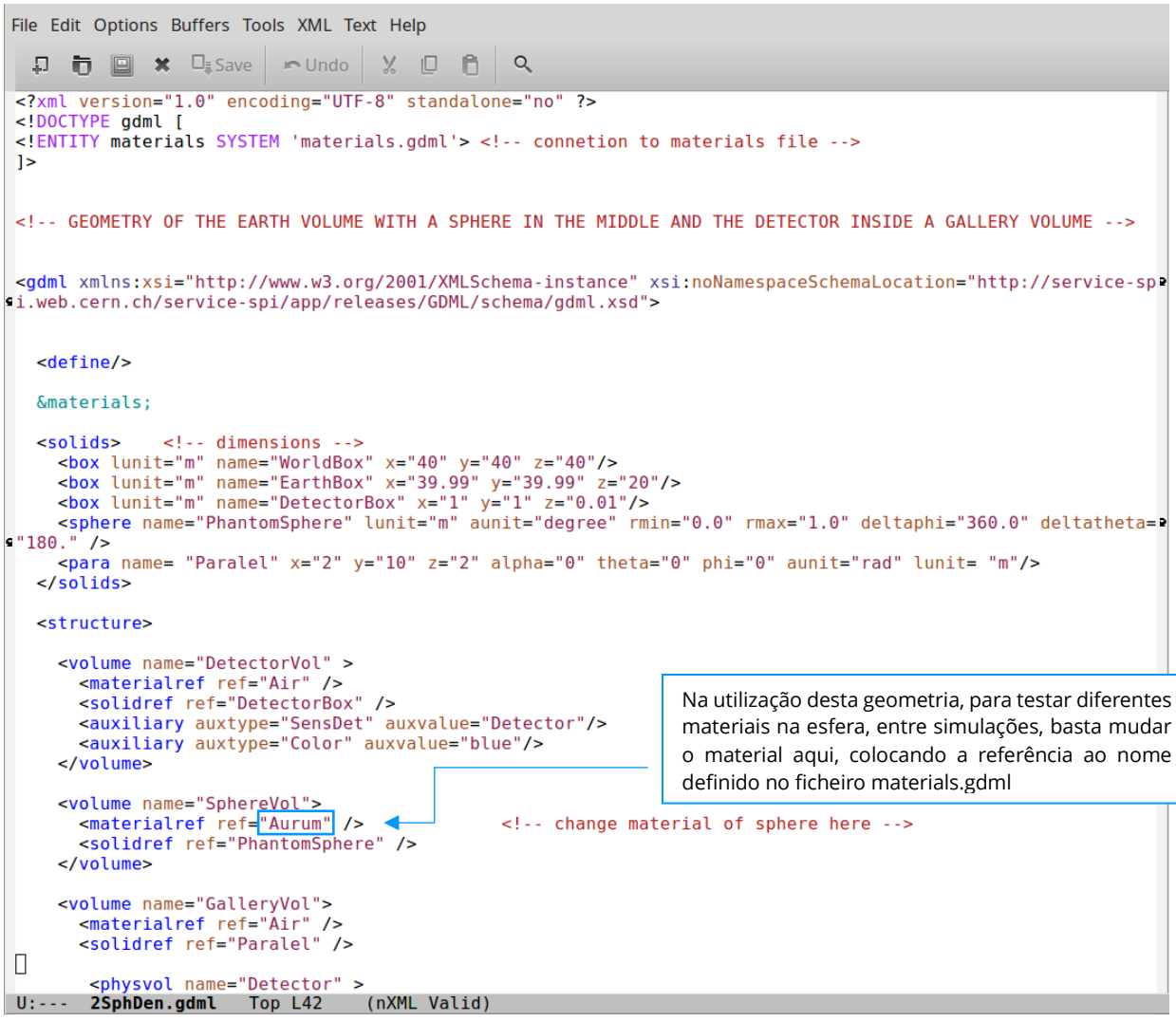
As geometrias são os ficheiros na primeira coluna da esquerda, na imagem anterior, cujos nomes começam por 0 até 3 e terminam em .gdml. Cada um dos ficheiros é uma abordagem com características diferentes ao cenário do terreno com a mina subterrânea.

Consultando cada um dos ficheiros gdml é possível ler as seguintes descrições em cada um:
0setup.gdml - first test geometry with lake, ellipsoid and sphere, gallery and detector inside earth volume
1blank.gdml – blank geometry of the earth volume with only a detector inside
1BMine.gdml – blank geometry of the earth volume with the detector inside a gallery volume
2SphDen.gdml – geometry of the earth volume with a sphere in the middle and the detector inside a gallery volume
3Sphs.gdml – geometry of the earth volume with 3 spheres in the middle and the detector inside a gallery volume

Analisando, por exemplo, o código de 2SphDen.gdml, usado na simulação teste:

```
$ emacs 2SphDen.gdml
```

A intensão desta geometria é estudar o tempo de exposição necessário para a esfera ser detetada, mudando o material da esfera entre simulações para testar diferentes densidades.



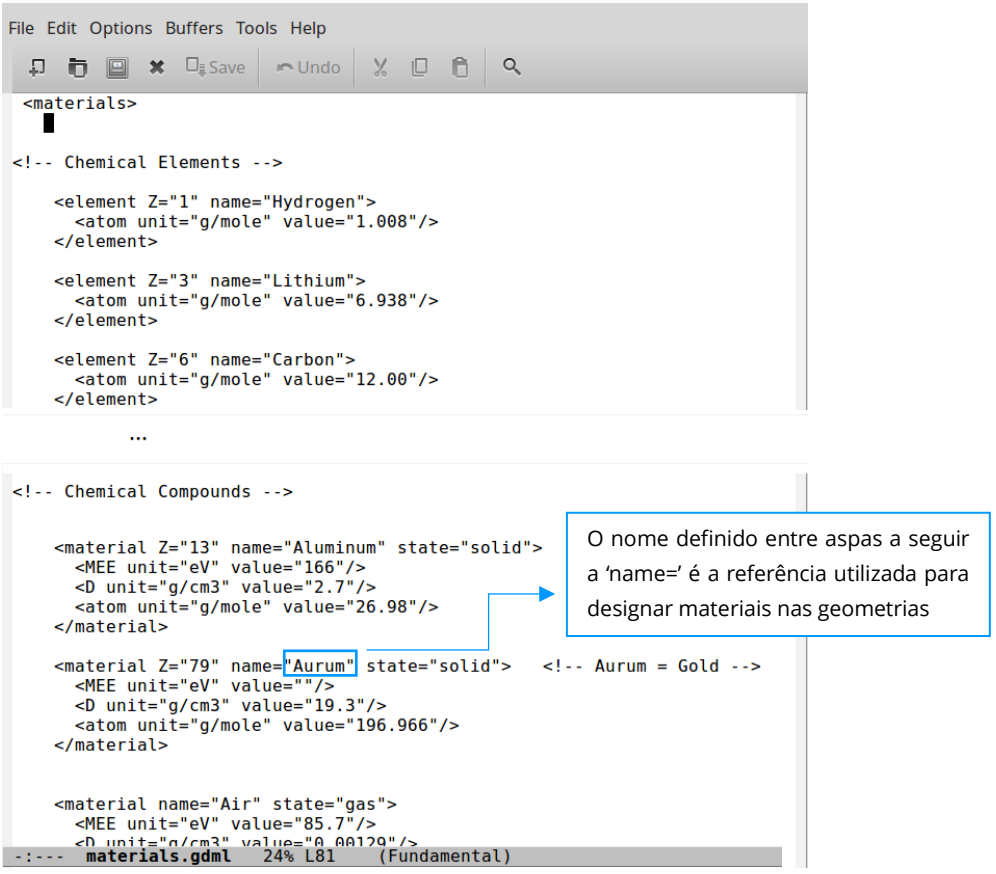
Num ficheiro de geometria como este, começa-se por definir as dimensões dos sólidos e depois definem-se as estruturas. A caixa que contém a geometria (WorldBox) tem dimensões 40x40x40m, por sua vez o terreno (EarthBox) tem uma área de 39,99x39,99 m e 20 m de profundidade. Os outros sólidos representados são uma esfera (PhantomSphere), com 1 m de raio, um paralelepípedo (Paralel) que representa a galeria e um plano sensível à passagem de partículas (DetetorBox).

Cada volume é definido em três etapas: sólido geométrico (forma e dimensões), volume lógico (material de preenchimento) e volume físico (posição). Dependendo da configuração que se pretende, existem lugares específicos para definir cada uma destas etapas no código do ficheiro gdml. O Manual de GDML é um bom começo para aprender mais sobre como definir geometrias e está disponível em:

http://gdml.web.cern.ch/GDML/doc/GDMLmanual.pdf

Todos os materiais utilizados nos volumes físicos das geometrias estão definidos em materials.gdml.

5.2.2. Materials.gdml



Este ficheiro contém a lista de todos os elementos e compostos químicos utilizados para preencher os volumes definidos nas geometrias. Neste ficheiro podem ser adicionados quaisquer elementos e compostos químicos necessários. Podem ser contruídos de raiz ou podem ser consultados na lista Geant4 Material Database, acessível aqui:

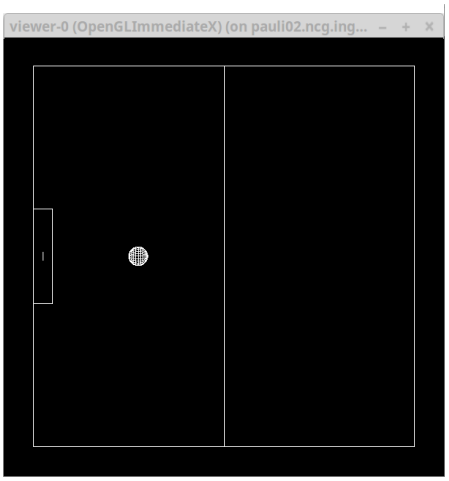
http://geant4-userdoc.web.cern.ch/geant4-userdoc/UsersGuides/ForApplicationDeveloper/BackupVersions/V9.3/html/apas09.html

5.2.3. Vis.mac

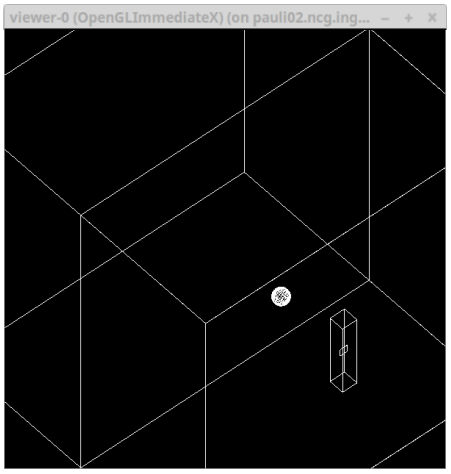
Este ficheiro define os parâmetros de uma simulação para ser executada em modo interativo com a aplicação MuTomo, como efetuado nos pontos 2.1 e 2.2. Nos ficheiros do tipo .mac, as linhas de código precedidas por # são ignoradas, pois são consideradas como comentários.

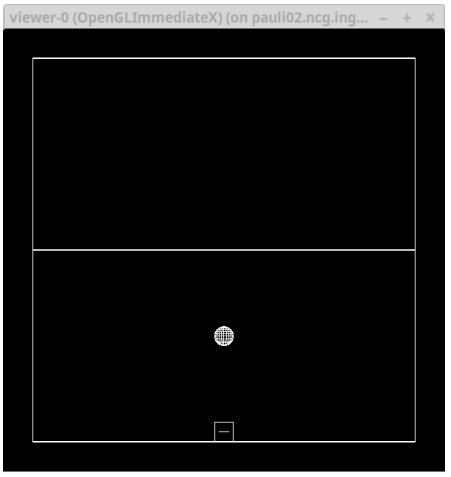
```
File  Edit  Options  Buffers  Tools  Help
Save  Undo
##################################################
# Visualization of detector geometry and events
##################################################

# reading Geometry from File
/mydet/readFile 2SphDen.gdml
/run/initialize

# create empty scene
#/vis/scene/create
#/run/initialize


# Use this open statement to create an OpenGL view:
#/vis/open OGL 600x600-0+0
/vis/open OGLIX 600x600-0+0

# Use this open statement to create a .prim file suitable for
# viewing in DAWN:
#/vis/open DAWNFILE

# Use this open statement to create a .heprep file suitable for
# viewing in HepRApp:
#/vis/open HepRepFile

# Use this open statement to create a .wrl file suitable for
# viewing in a VRML viewer:
#/vis/open VRML2FILE

# Disable auto refresh and quieten vis messages whilst scene and
# trajectories are established:
/vis/viewer/set/autoRefresh false
/vis/verbose errors
```

Aqui indica-se o nome do ficheiro com geometria que será usada na simulação




```
# Draw geometry:
/vis/drawVolume


# Specify view angle:
# 0. 90./0. 0. > gives a view from the bottom/top of the sphere
# 89.9 90. > detector facing N
# 90. 0 > detector facing E
# 90. 180. > detector facing W
# 90. 90. > detector facing NW
# 60. 60. > detector facing NE - 3D view
# 240. 240. > detector facing SE - 3D view
# 200. 240. > detector facing S/SE - 3D view
# 240. 270. > detector facing S/up
# 270. 270. > detector facing SE
# 89.9 270. > detector facing S
#Theta rotates view in N-S direction
#Phi rotates view in E-W direction and in N-S sideways (?)
/vis/viewer/set/viewpointThetaPhi 89.9 270.


# Specify zoom value:
/vis/viewer/zoom 1.5

# Specify style (surface or wireframe):
#/vis/viewer/set/style wireframe

# Draw coordinate axes:
#/vis/scene/add/axes 0 0 0 1 m

# Draw smooth trajectories at end of event, showing trajectory points
# as markers 2 pixels wide:
/vis/scene/add/trajectories smooth
/vis/modeling/trajectories/create/drawByCharge
#/vis/modeling/trajectories/drawByCharge-0/default/setDrawStepPts true
/vis/modeling/trajectories/drawByCharge-0/default/setStepPtsSize 2
# (if too many tracks cause core dump => /tracking/storeTrajectory 0)

# Draw hits at end of event:
/vis/scene/add/hits
-:**-  vis.mac        23% L42     (Fundamental)
```

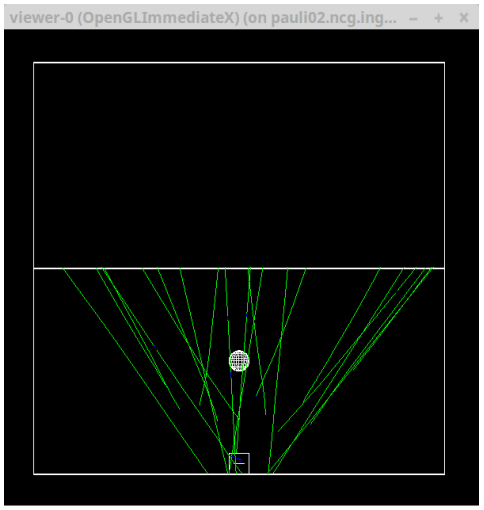




Os ângulos theta e phi definem a orientação da visualização da geometria. Alguns pares de valores encontram-se listados com a respetiva indicação da orientação

```
# Filter all particles except:
/vis/filtering/trajectories/create/particleFilter
/vis/filtering/trajectories/particleFilter-0/add gamma
/vis/filtering/trajectories/particleFilter-0/add nu_mu
/vis/filtering/trajectories/particleFilter-0/add anti_nu_e

# To invert the above, drawing all particles except the ones mentioned above,
# we keep the above lines and add:
/vis/filtering/trajectories/particleFilter-0/invert true
#
# Many other options are available with /vis/modeling and /vis/filtering.
# For example, to select colour by particle ID:
/vis/modeling/trajectories/create/drawByParticleID
/vis/modeling/trajectories/drawByParticleID-0/set geantino red
/vis/modeling/trajectories/drawByParticleID-0/set mu- green
/vis/modeling/trajectories/drawByParticleID-0/set e- blue
/vis/modeling/trajectories/drawByParticleID-0/set nu_mu yellow
/vis/modeling/trajectories/drawByParticleID-0/set anti_nu_e orange
#/vis/modeling/trajectories/drawByParticleID-0/set mu+ green
#/vis/modeling/trajectories/drawByParticleID-0/set e+ blue
#/vis/modeling/trajectories/drawByParticleID-0/set anti_nu_mu yellow
#/vis/modeling/trajectories/drawByParticleID-0/set nu_e orange


# To superimpose all of the events from a given run:
/vis/scene/endOfEventAction accumulate -1

# Re-establish auto refreshing and verbosity:
/vis/viewer/set/autoRefresh true
/vis/verbose warnings

# For file-based drivers, use this to create an empty detector view:
#/vis/viewer/flush

#hard coded position and direction for gun
#/gun/position 0 0 40 m
#/gun/direction  0 0 -1

# define source through a .mac
/control/execute source.mac
/control/execute setprocesses.mac

/random/setSeeds 0 0

#if you want make a run with particle
#/tracking/verbose 1
/run/beamOn 0


-:**-  vis.mac        Bot L128    (Fundam
```

Trajetórias de partículas diferentes podem ser diferenciadas por cores. A trajetória dos muões está definida para verde

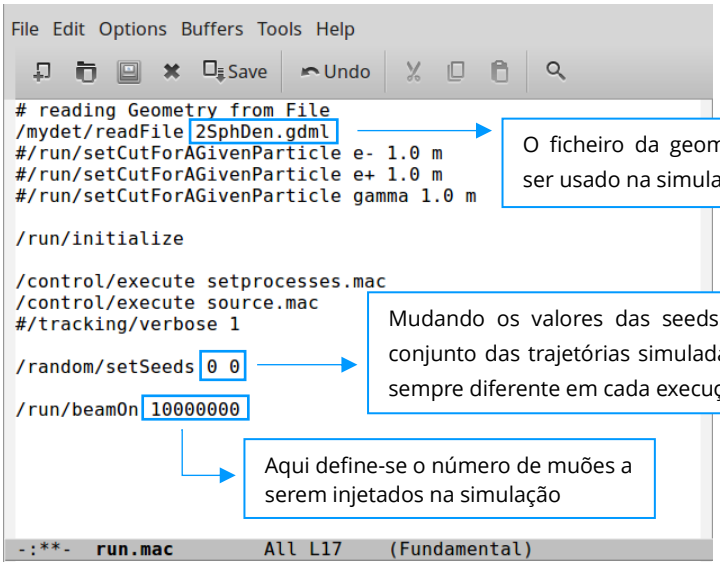


Os parâmetros do fluxo dos muões estão definidos nestes dois ficheiros

Aqui o beamOn igual a zero significa que não houve injeção de muões, pois apenas se pretendia visualizar a geometria. Alterando este algarismo e introduzindo o valor que se desejar, quando o vis.mac voltar a ser executado, a simulação irá representar as trajetórias dos muões com linhas verdes (como visto na acima). A representação está definida para que só sejam representadas as trajetórias cuja sua extensão passa dentro de um determinado raio à volta do detetor (como indicado no ponto 5.2.7)

5.2.4. Run.mac

Este ficheiro define os parâmetros de uma simulação para ser executada interactivamente com a aplicação MuTomo, como efetuado no ponto 2.3.



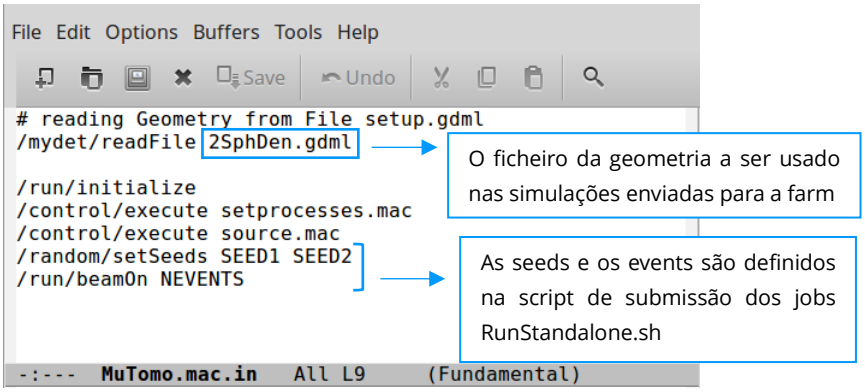
O ficheiro da geometria a ser usado na simulação

Mudando os valores das seeds garante-se que o conjunto das trajetórias simuladas dos muões será sempre diferente em cada execução interativa

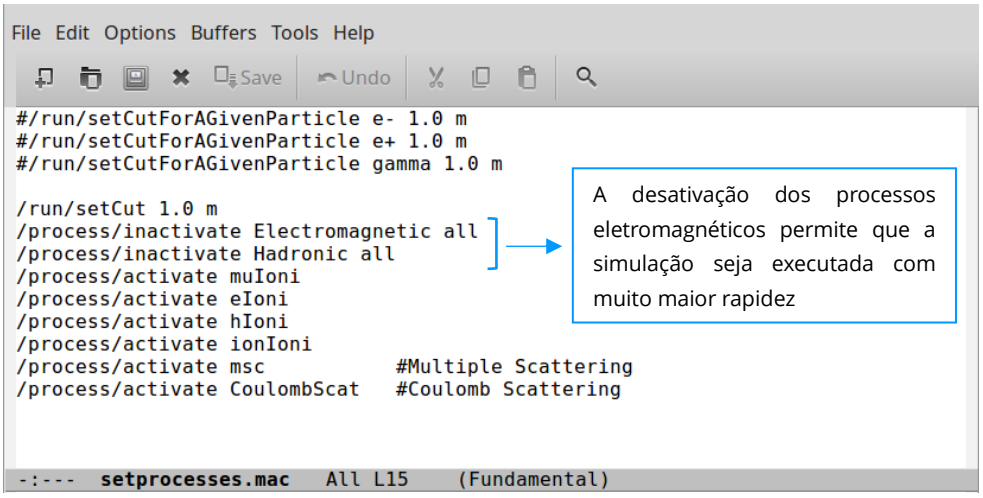
Aqui define-se o número de muões a serem injetados na simulação

5.2.5. MuTomo.mac.in, Setprocesses.mac e Source

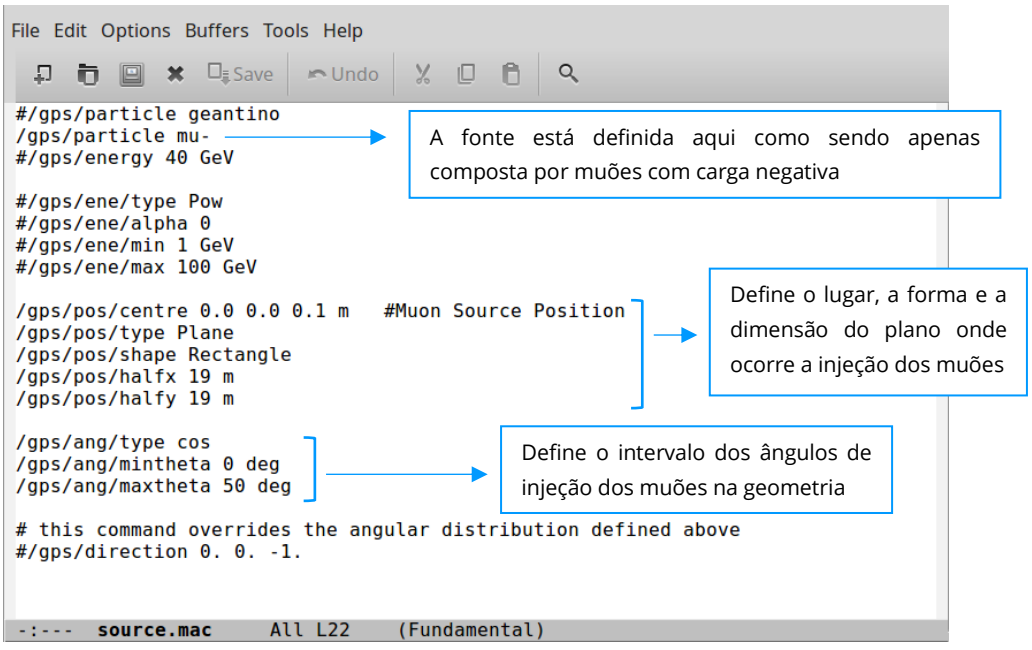
O ficheiro MuTomo.mac.in estabelece os parâmetros das simulações que são submetidas para a farm.



Setprocesses.mac define os processos de interação das partículas que estarão ativos ou dasativos.

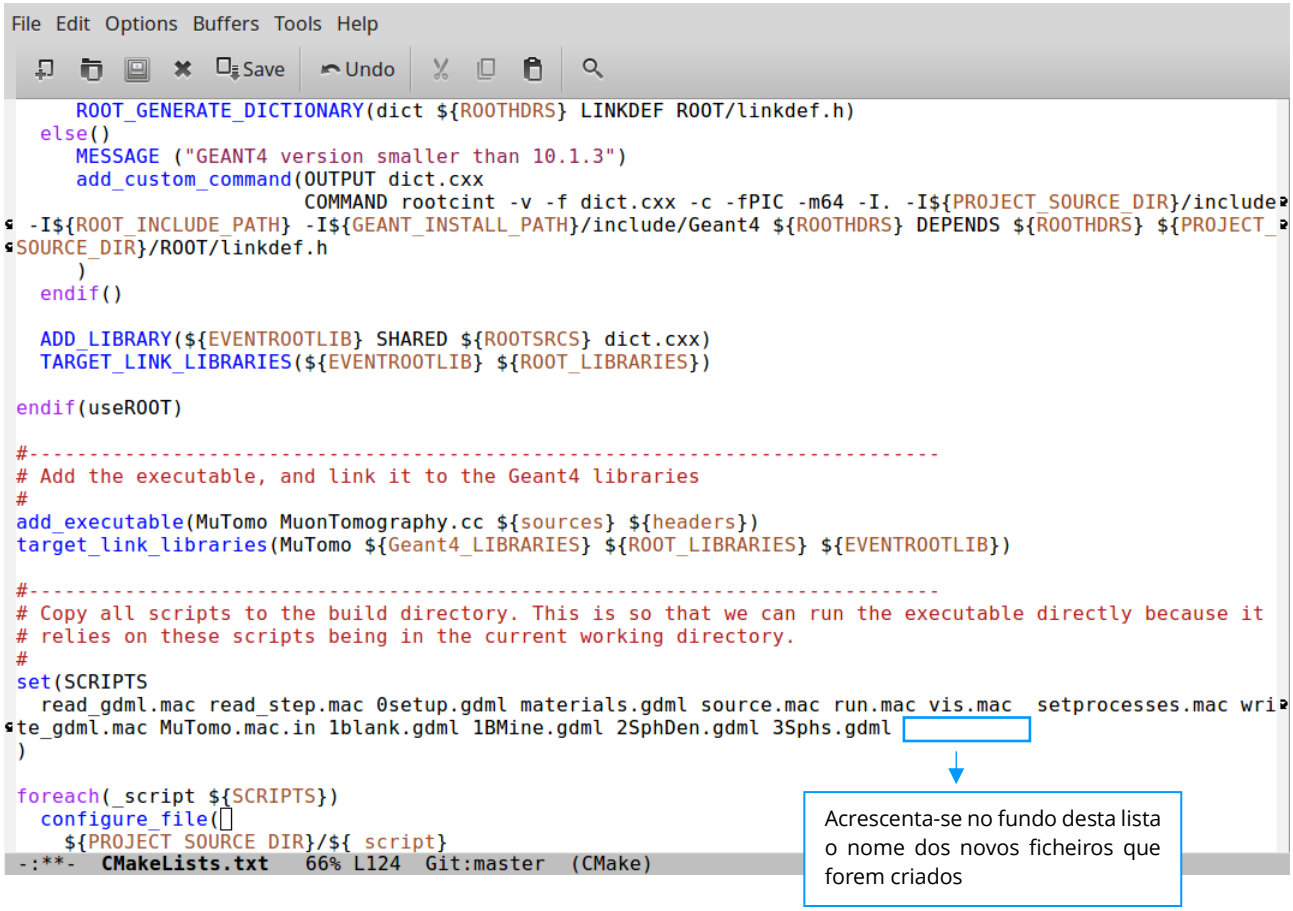


O ficheiro source.mac define os parâmetros da fonte de muões a ser injetada durante a simulação.



5.2.6. CMakeList.txt

Este ficheiro contém a lista dos vários ficheiros que são copiados da pasta MuonTomography para a pasta build, quando é executado o comando make, como mostrado no ponto 1.5. Quando que se cria um novo ficheiro, como um ficheiro de geometria, por exemplo, ou se elimina algum dos ficheiros que aparecem listados, é necessário acrescentar o nome do novo ficheiro à lista, ou apagar o nome, se for esse o caso. Acaso se esqueça de atualizar a lista dos ficheiros a copiar com o nome de um ficheiro novo, o resultado é que este não será copiado quando o comando make for executado. No caso de um ficheiro ser eliminado e o seu nome continuar na lista, quando o comando make for executado, este irá retornar um erro avisando do ficheiro que está em falta. Este erro resolve-se ao atualizar a lista, como indicado na imagem seguinte.



5.2.7. Pastas Src e Include

O conteúdo destas duas pastas é complementar uma da outra. Estes ficheiros são as engrenagens que fazem correr a simulação, por isso uma exploração através deles é recomendada. Alguns deles contêm comentários para auxiliar a compreensão.

SRC

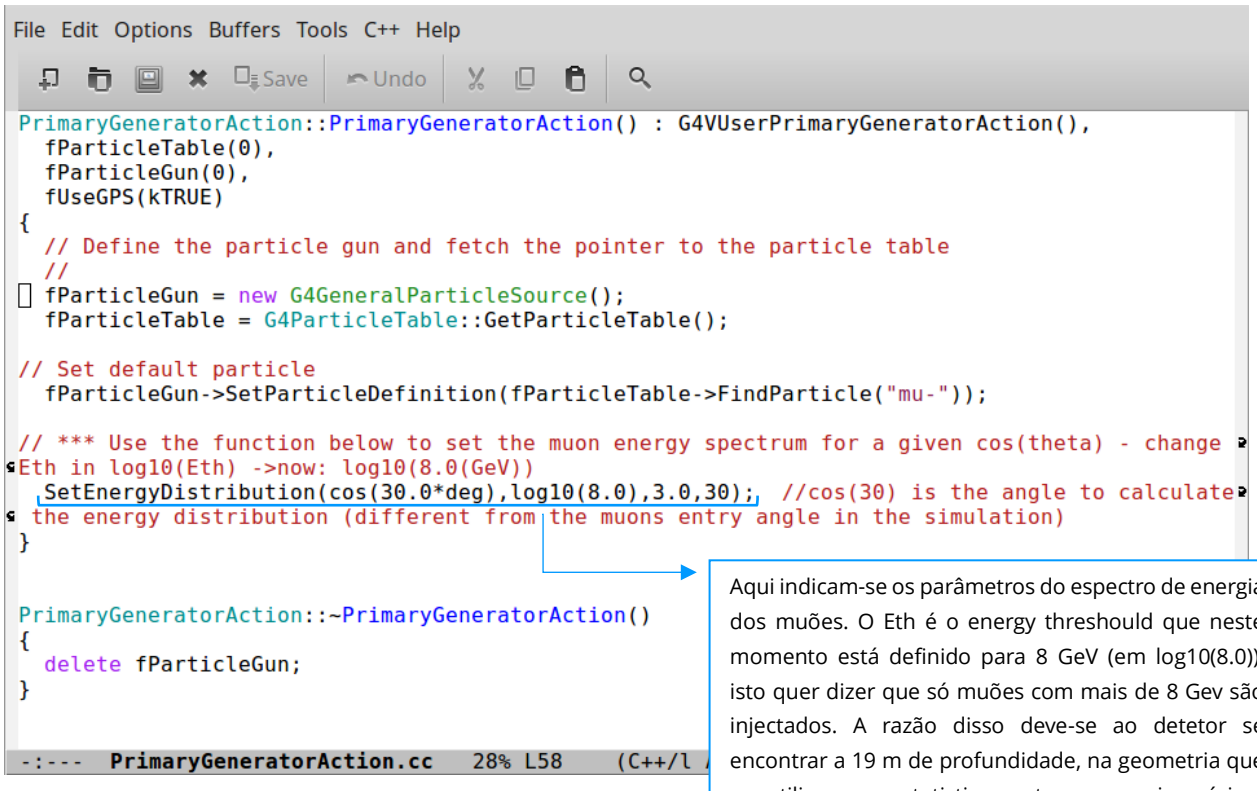
```
DetectorConstruction.cc  Hit.cc                     RunAction.cc
DetectorMessenger.cc     PhysicsList.cc             SensitiveDetector.cc
EventAction.cc           PrimaryGeneratorAction.cc  TrackingAction.cc
```

INCLUDE

```
DetectorConstruction.hh  Hit.hh                     RunAction.hh
DetectorMessenger.hh     PhysicsList.hh             SensitiveDetector.hh
EventAction.hh           PrimaryGeneratorAction.hh  TrackingAction.hh
```

Todavia, por ser pertinente para o objetivo deste tutorial, vale a pena explorar aqui o ficheiro PrimaryGeneratorAction.cc e TrackingAction.cc.

O ficheiro PrimaryGeneratorAction.cc contém os parâmetros que definem a distribuição da energia e do fluxo de muões injetado nas simulações.



...

```
//     event->AddPrimaryVertex(fVertex);
// }


void PrimaryGeneratorAction::SetEnergyDistribution(G4double Costh,G4double logEmin,G4double logEmax,G4int Nbins)
 //define energy distribution, parameters are indicated above in ***
{
  G4SPSEneDistribution *eneDist = fParticleGun->GetCurrentSource()->GetEneDist();
  eneDist->SetEnergyDisType("Arb");


  G4ThreeVector EnergyHisto;
  G4double energy, prob;

  G4cout << "Generating spectrum for costh = " <<  Costh << G4endl;

  G4double logE;
  G4double delta = (logEmax-logEmin)/float(Nbins);

  G4double norm = dNdEdOmegadtdA(pow(10.0,logEmin)*GeV,Costh);

  G4cout << norm << G4endl;

  for (int ibin=0;ibin<=Nbins;++ibin) {      // bin size

    logE = logEmin + ibin*delta;
    energy = pow(10.0,logE)*GeV;
    prob = dNdEdOmegadtdA(energy,Costh)/norm;

    if (prob > 0.0) {                              //energy probability
      EnergyHisto = G4ThreeVector(energy,prob,0.0);
      eneDist->ArbEnergyHisto(EnergyHisto);
      G4cout <<  EnergyHisto.x()/GeV << " (GeV)     " << EnergyHisto.y() <<  G4endl;
    }
  }
-:---  PrimaryGeneratorAction.cc   80% L184   (C++/l Abbrev)
```

A distribuição da energia está definida nesta parte do código e usa os parâmetros indicados na imagem anterior

...

```
      EnergyHisto = G4ThreeVector(energy,prob,0.0);
      eneDist->ArbEnergyHisto(EnergyHisto);
      G4cout <<  EnergyHisto.x()/GeV << " (GeV)     " << EnergyHisto.y() <<  G4endl;
    }
  }

  eneDist->ArbInterpolate("Exp");
}


// Muon Flux Distribution Equation (at sea level)

G4double PrimaryGeneratorAction::dNdEdOmegadtdA(G4double energy, G4double costh)
{
 //units of a*b*c : cm-2 s-1 sr-1
  //double costh=cos(TMath::DegToRad()*theta);

  const G4double E = energy/GeV;

  double a=18/(E*costh+145);
  double b=pow(E+2.7/costh,-2.7);
  double c=(E+5)/(E+5/costh);

  //  double d=pow(p,2.7);
  //conversion from cm^-2 to  m^-2    *1E4
  //conversion from sr^-1   to  deg^-2    *3E-4
  //conversion from sr^-1   to  (0.001*0.001) sr^-1    *1E-6
  //conversion from s^-1    to day^-1 *86400

  return a*b*c*1E4;//*3E-4;  // return units  m-2 s-1 sr-1
}



-:---  PrimaryGeneratorAction.cc   Bot L224   (C++/l Abbrev)
```

Na última parte do código está definida a equação da distribuição do fluxo de muões

O ficheiro TrackingAction.cc define os parâmetros das trajetórias dos muões que serão seguidas durante a simulação, ou seja, só são contabilizados os muões que passam no interior de um determinado raio à sua volta. que se devem adequar de acordo com a geometria usada nas simulações.

File Edit Options Buffers Tools C++ Help

Save Undo

```
{;}


void TrackingAction::PreUserTrackingAction(const G4Track* aTrack)
{
  // Decision is taken to track a primary only if it will hit the detector, allowing for multiple
 scattering

  // Select only for primary particles
  if (aTrack->GetParentID() != 0) {
    return;
  }

  // Check if primary track will hit the detector

  G4PhysicalVolumeStore * store = G4PhysicalVolumeStore::GetInstance();
  G4VPhysicalVolume * physDet = store->GetVolume("Detector");
  G4VPhysicalVolume * physEarth = store->GetVolume("Earth");

  G4Box * box  = dynamic_cast<G4Box*>(physDet->GetLogicalVolume()->GetSolid());
  G4double x = box->GetXHalfLength();
  G4double y = box->GetYHalfLength();
  G4double detRadius = sqrt(x*x + y*y);

  G4ThreeVector tmp = physDet->GetObjectTranslation() /*+ physEarth->GetObjectTranslation()*/ ;
//  HARDCODED detector z. To be CHECKED !!!
  G4ThreeVector detPosition = G4ThreeVector(tmp.x(),tmp.y(),-19.0*m);

  G4double kineticE = aTrack->GetKineticEnergy();

  const G4ParticleDefinition * partDef = aTrack->GetParticleDefinition();
```

...

```
// Compute multiple scattering
  const G4double depth = abs(detPosition.z());
  const G4double slantDepth = abs(depth/cos(dir.theta()));

  G4double theta0;
  if (charge  == 0.0) {
    theta0 = 0.0;
  }
  else {
    theta0 = 13.6*MeV/betapc * abs(charge) * sqrt(slantDepth/X0) * (1.0 + 0.038*log(slantDepth/X0
*charge2/beta2));
  }

  // Deviation due to multiple scattering; TO BE CHECKED !!! In any case for now R0 yileds typica
lly < 1 m, so no big impact considering
  // the safety factor of 5 x detRadius
  const G4double R0 = slantDepth * tan(theta0);

  if ( (newpos-detPosition).perp() > 3.0*(detRadius+R0) ) {      //factor 3 (but above is safety
factor 5(?))
    fpTrackingManager->GetTrack()->SetTrackStatus(fStopAndKill);
  }
  else {
    // if ( 180.*deg - dir.theta() > 45.0*deg)
    // G4cout << (180.0*deg-dir.theta())/deg << "    " << pos.perp()/m << G4endl;
  }

  // G4cout << X0/cm << "    " << slantDepth/m << "   " << mass/MeV << "    " << beta2 << "    " << t
heta0*180.0/CLHEP::pi << "    " << R0/m << G4endl;
  // G4cout << "Radius at detector before/after MS :  " << (newpos-detPosition).perp()/m  << "
" <<  R0/m << G4endl;

  return;
}


void TrackingAction::PostUserTrackingAction(const G4Track*)
{;}
```

-:--- **TrackingAction.cc** Bot L74 (C++/l Abbrev)

Aqui indica-se a posição do detetor e assinala o ponto onde as trajetórias contabilizadas irão convergir. Este valor é a coordenada "z" do sólido "WorldBox" onde se situa o detetor e deve ser alterado de acordo com a geometria que estiver a ser usada nas simulações

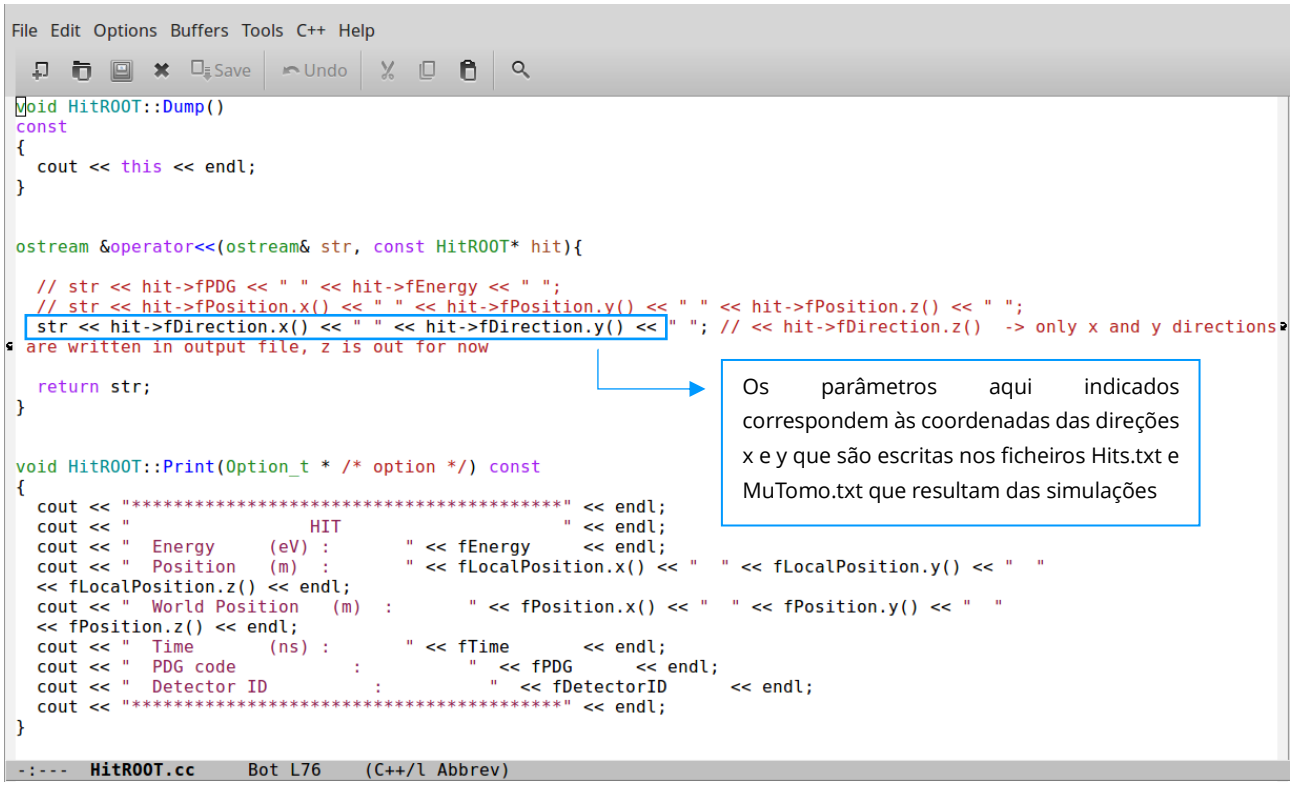
Nesta parte do código está definido o raio à volta do detetor dentro do qual só as trajetórias que aqui passem serão seguidas. Neste caso, o fator de multiplicação do raio está definido para 3

5.2.8. Pasta ROOT

Os ficheiros contidos nesta pasta também são responsáveis por operar partes das simulações, e, como nos casos anteriores, valem a pena uma exploração.

```
EventROOT.cc  EventROOT.hh  HitROOT.cc  HitROOT.hh  linkdef.h  Write2ROOT.cc  Write2ROOT.hh
```

O ficheiro HitROOT.cc merece ser explorado aqui pois nele é definido o que é escrito nos ficheiros que resultam das simulações, nomeadamente Hits.txt e Mutomo.txt, obtidos nos pontos 2.5 e 4.5 respetivamente, e essa informação pode ser alterada como for conveniente.



6. Manuais Online GEANT4

A página da internet do GEANT4 disponibiliza vários manuais de utilização, divididos em diferentes tópicos, para consulta e download. Toda a documentação está disponível em:

https://geant4.web.cern.ch/support/user_documentation

LABORATÓRIO DE INSTRUMENTAÇÃO
E FÍSICA EXPERIMENTAL DE PARTÍCULAS
partículas e tecnologia

UNIVERSIDADE
DE ÉVORA

Centro Ciência Viva do Lousal
Mina de Ciência

Instituto de Ciências da Terra
Institute of Earth Sciences