# コンパイル

中村輝石

20210609

# コンパイルしたくない？

- メリット　速くなる
- デメリット　めんどい
- いずれみんな思う「速さが足りない！」

# コンパイルとは

- コンパイルとは
  - 人間が読めるプログラム（マクロとか）を、機械が読める実行ファイルに変えること
  - マクロでは1行ごとにコンパイルしてるような感じで動いているので、遅い

- Makefileについて（今回は省略）
  - 複数のc++のファイルをまとめてコンパイルしたりできる
  - Makefileというのファイルにどの順番にコンパイルするのか指定する。いずれみんな使うと思う
  - Makefileを自動で作るcmakeというのもあり、最近はこれが多い（Geant4など大規模なプログラムはだいたい使ってる）

- 今回は簡単な方法をやってみる

# サンプルマクロ

- ROOTのマクロとして動かしてみよう（hoge1.C）
- aaaが表示されますね

```
{
  cout<<"aaa"<<endl;
}
```

# コンパイル

- マクロを少しいじってhoge2.Cに
- g++コマンドでコンパイル
- jikkouという実行ファイルができる
  - -o オプションを省略すると、実行ファイルはデフォルトの「a.out」になる
- 実行すると、aaaと表示される
- 本当は最後に「return 0;」とかが要る（サボった）

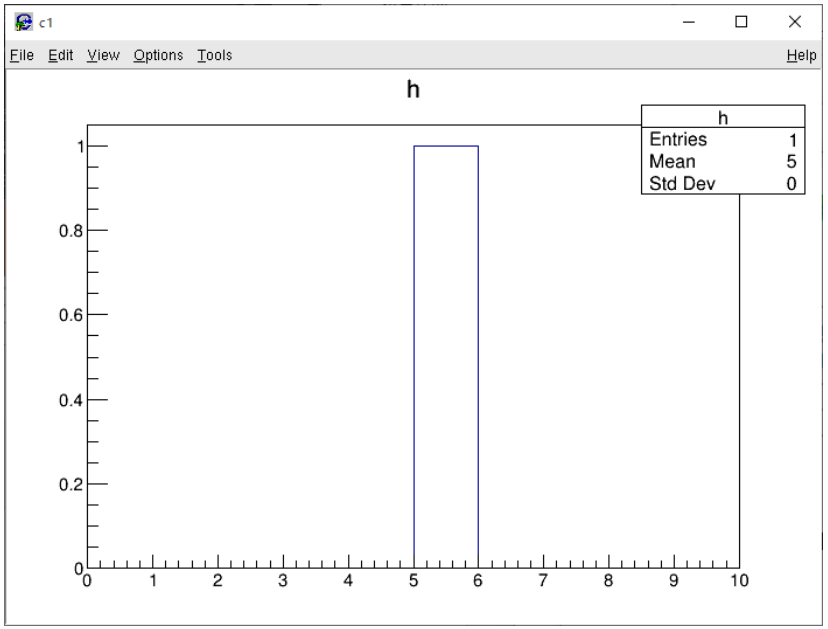
```
#include <iostream>
using namespace std;

int main(){
  cout<<"aaa"<<endl;
}
```

```
$ g++ hoge2.C -o jikkou
$
$ ls
hoge2.C jikkou
$
$ ./jikkou
aaa
$
```

# ROOTのクラスを使いたい

- root -l hoge3.C
- これをもとにやる

```
{
  TH1D *h = new TH1D("h","h",10,0,10);
  h->Fill(5);
  h->Draw();
}
```



# ROOTのクラス入りコンパイル

- g++ -o aho hoge4.C
- 「TH1なんて知らんよ」、と怒られる
- TH1がどんな関数なのか、g++は知らないので、教えてあげる必要がある

```
#include "TH1.h"
int main(){
  TH1D *h = new TH1D("h","h",10,0,10);
  h->Fill(5);
  h->Draw();
}
```

```
kiseki@dell-xps13:~/test$ g++ hoge4.C
hoge4.C:1:10: fatal error: TH1.h: そのようなファイルやディレクトリはありません
 #include ”TH1.h”
          ^~~~~~~

compilation terminated.
```

# root-config

- 便利コマンド
  - root-config --cflags
  - root-config --libs

```
kiseki@dell-xps13:~/test$ root-config --cflags
-pthread -std=c++11 -m64 -I/home/kiseki/src/root/include
```

```
kiseki@dell-xps13:~/test$ root-config --libs
-L/home/kiseki/src/root/lib -lCore -lImt -lRIO -lNet -lHist
 -lGraf -lGraf3d -lGpad -lROOTVecOps -lTree -lTreePlayer -l
Rint -lPostscript -lMatrix -lPhysics -lMathCore -lThread -l
MultiProc -lROOTDataFrame -pthread -lm -ldl -rdynamic
```

- ROOTインストール時に、このコマンドも入ってるハズ

- ライブラリのヘッダファイルへのpathとかを自動で表示してくれるコマンド
- g++の引数にこれらをいれるとg++がROOTのクラスをコンパイルできるようになる

# ROOTのクラス入りコンパイル（再）

- シングルクォーテーションで囲った部分はコマンドが展開される（例えば、「echo `ls`」とかいう風にできる）
- 順番が大事

```
$ g++ `root-config --cflags` hoge4.C `root-config --libs` -o aho
```

ここにソースコード

これはどこでもいい（たぶん）

- ただし、このサンプルは実行してもつまらない

```
$ ./aho
```

```
kiseki@dell-xps13:~/test$ ./aho
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
```

# せめて絵を残したい

- hoge5.C のようにすれば画像ができる
- （もっと簡単な方法があるかもしれない）
- できた絵は「eog c.png」とかで見てみよう

```
#include "TH1.h"
#include "TCanvas.h"
int main(){
  TCanvas *c = new TCanvas("c","c",800,600);
  TH1D *h = new TH1D("h","h",10,0,10);
  h->Fill(5);
  h->Draw();
  c->Print("c.png");
}
```

# とはいえ実行時に画面が出したい

- TRint を使い、hoge6.C のようにする
  - TApplicationでも似たようなことができる
- root起動時の状態になり、画像がでる

```
#include "TH1.h"
#include "TRint.h"
int main(){
  TRint app("app", 0, 0);

  TH1D *h = new TH1D("h","h",10,0,10);
  h->Fill(5);
  h->Draw();

  app.Run();
}
```

# まとめ

- コンパイル
  - g++ で実行ファイルを作る
  - マクロより速い

- キーワード
  - Makefile、cmake、root-config、TRint、TApplication
  - 凝りたくなったら、調べるか、その辺のYMAP会員に聞こう