Ni hao

你好, Fun4All!

Part-I

G. Nukazuka (RIKEN/RBRC)

About this talk

This talk presents how to run your analysis codes in the Fun4All framework. Audiences are asked to download/run/change some codes, so you need to have a BNL account.

Hands-on Program

1. Downloading the sample codes
2. Checking your environmental variables
3. Running the minimal code (Fun4All_minimum.C)
4. Making/compiling your analysis module
5. Modifying your environmental variables to include your analysis module
6. Modifying and running the sample code (Fun4All_minimum_2.C)

what/who
why
where
when
how

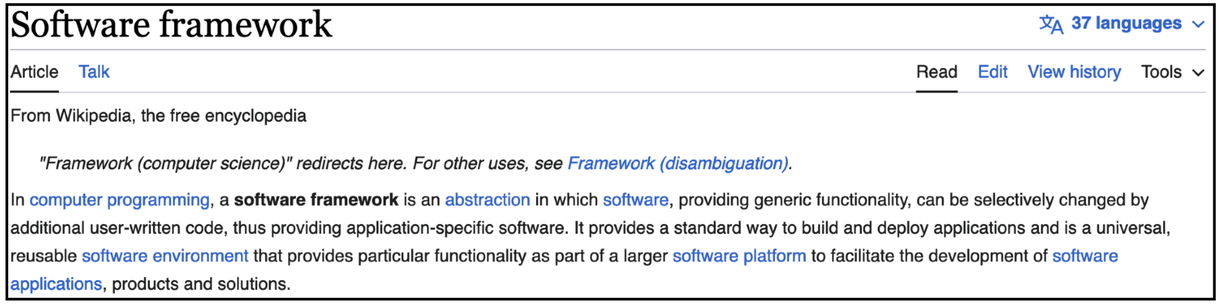
What is Fun4All?

An analysis framework originally developed for the PHENIX experiment

what/who
why
where
when
how

What is Fun4All?

An analysis framework originally developed for the PHENIX experiment



Software framework

37 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

"Framework (computer science)" redirects here. For other uses, see Framework (disambiguation).

In computer programming, a **software framework** is an abstraction in which software, providing generic functionality, can be selectively changed by additional user-written code, thus providing application-specific software. It provides a standard way to build and deploy applications and is a universal, reusable software environment that provides particular functionality as part of a larger software platform to facilitate the development of software applications, products and solutions.

軟體框架

37种语言

条目 讨论 汉 漢 不转换 阅读 编辑 查看历史 工具

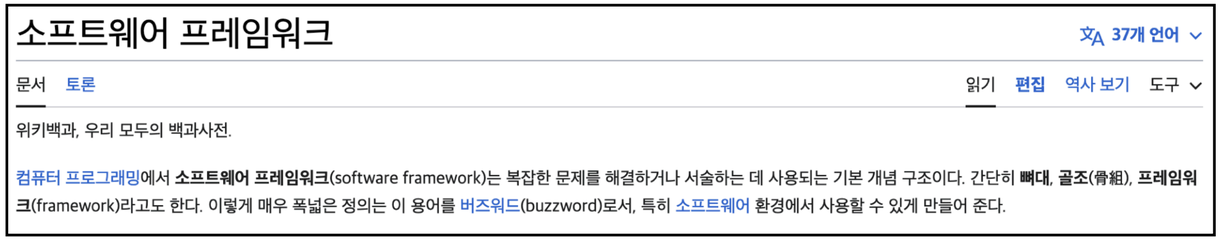
维基百科，自由的百科全书

此條目**沒有列出任何參考或來源**。 *(2016年8月3日)*
維基百科所有的內容都應該可供查證。请协助補充可靠来源以改善这篇条目。无法查证的內容可能會因為異議提出而被移除。

軟體框架（software framework），通常指的是為了實現某個業界標準或完成特定基本任務的軟體組件規範，也指為了實現某個軟體組件規範時，提供規範所要求之基礎功能的軟體產品。

框架的功能類似於基礎設施，與具體的軟體應用無關，但是提供並實現最為基礎的軟體架構和體系。軟體開發者通常依據特定的框架實現更為複雜的商業運用和業務邏輯。這樣的軟體應用可以在支持同一種框架的軟體系統中運行。

簡而言之，框架就是制定一套規範或者規則（思想），大家（程序员）在該規範或者規則（思想）下工作。或者說使用別人搭好的舞台來做編剧和表演。



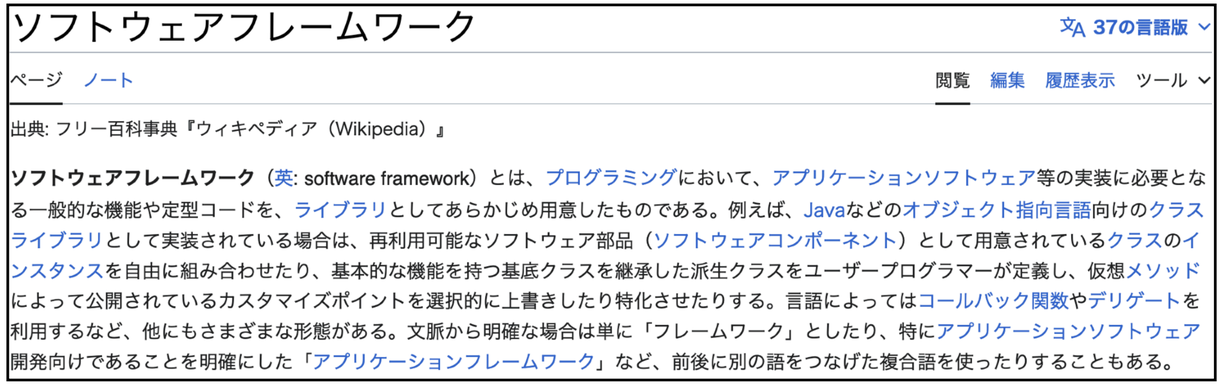
소프트웨어 프레임워크

37개 언어

문서 토론 읽기 편집 역사 보기 도구

위키백과, 우리 모두의 백과사전.

컴퓨터 프로그래밍에서 **소프트웨어 프레임워크**(software framework)는 복잡한 문제를 해결하거나 서술하는 데 사용되는 기본 개념 구조이다. 간단히 **뼈대**, **골조**(骨組), **프레임워크**(framework)라고도 한다. 이렇게 매우 폭넓은 정의는 이 용어를 버즈워드(buzzword)로서, 특히 소프트웨어 환경에서 사용할 수 있게 만들어 준다.



ソフトウェアフレームワーク

37の言語版

ページ ノート 閲覧 編集 履歴表示 ツール

出典: フリー百科事典『ウィキペディア（Wikipedia）』

ソフトウェアフレームワーク（英: software framework）とは、プログラミングにおいて、アプリケーションソフトウェア等の実装に必要となる一般的な機能や定型コードを、ライブラリとしてあらかじめ用意したものである。例えば、Javaなどのオブジェクト指向言語向けのクラスライブラリとして実装されている場合は、再利用可能なソフトウェア部品（ソフトウェアコンポーネント）として用意されているクラスのインスタンスを自由に組み合わせたり、基本的な機能を持つ基底クラスを継承した派生クラスをユーザープログラマーが定義し、仮想メソッドによって公開されているカスタマイズポイントを選択的に上書きしたり特化させたりする。言語によってはコールバック関数やデリゲートを利用するなど、他にもさまざまな形態がある。文脈から明確な場合は単に「フレームワーク」としたり、特にアプリケーションソフトウェア開発向けであることを明確にした「アプリケーションフレームワーク」など、前後に別の語をつなげた複合語を使ったりすることもある。

Wikipedia

what/who
why
where
when
how

What is Fun4All?

An analysis framework originally developed for the PHENIX experiment

ROOT: analyzing petabytes of data, scientifically.

An open-source data analysis framework used by high energy physics and others.

ℹ Learn more | ↓ Install v6.28/06

what/who
why
where
when
how

Why do we use Fun4All?

- Fun4All has a successful history.
- Fun4All has useful features.
- Other sPHENIX members use it.
- **Only analysis results done with Fun4All can be published from sPHENIX.**

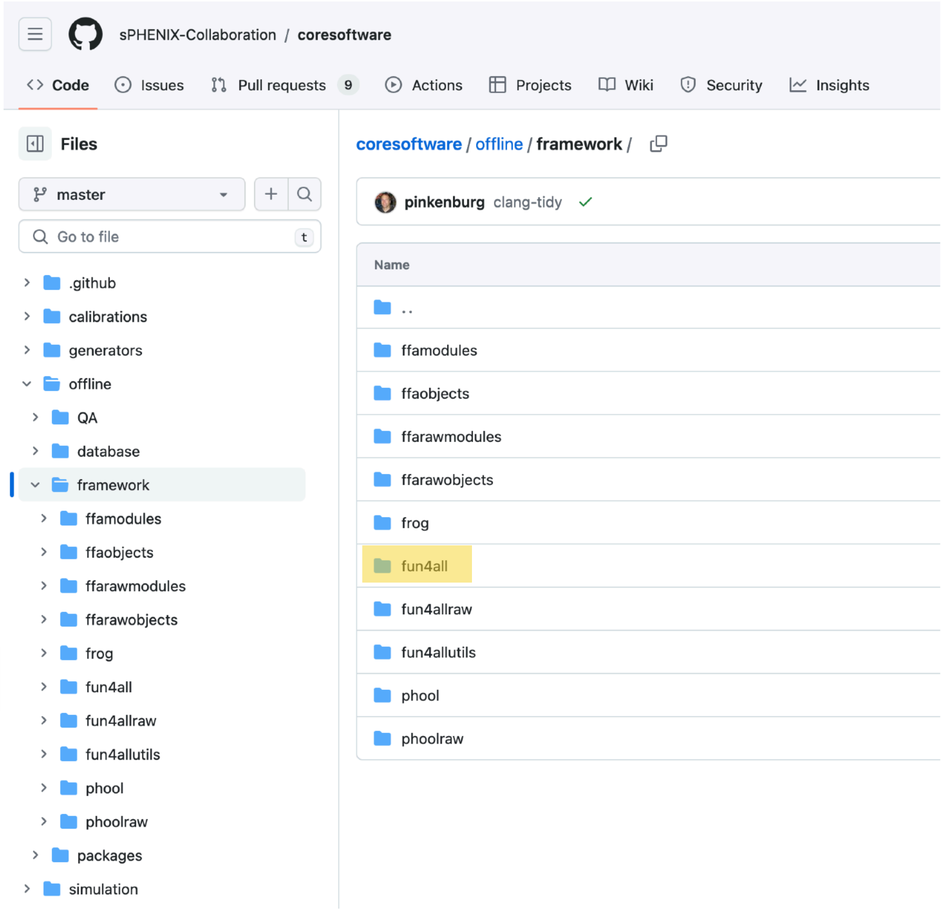
what/who
why
where
when
how

Where is Fun4All?

- You can find it on GitHub: https://github.com/sPHENIX-Collaboration/coresoftware
- You can use it in the SDCC servers.

Steps to set up Fun4All in SDCC

1. Log in to the SDCC gateway machine:
   ```
   $ ssh {username}@ssh.sdcc.bnl.gov
   ```
2. Log in to the SDCC servers:
   ```
   $ ssh {username}@sphnx{num}sdcc.bnl.gov
   ```
   {num}: 01 — 08
3. Execute the setting shell script:
   ```
   $ source /opt/sphenix/core/bin/sphenix_setup.sh
   ```



what/who
why
where
when
how

When should we start using Fun4All?

- In the test bench, we use FEM/FEM-IB system for INTT operation.
  It writes results to a TTree. So we just need ROOT.
- In the commissioning, we got evt files from RCDAQ. Our decoder generated ROOT files containing a TTree. Only ROOT is necessary for the analysis.
- The sPHENIX decoder, which will be released in a perfect performance soon, reads evt files and outputs DST files (other formats are possible technically).

Now is a good time to migrate from ROOT to Fun4All.

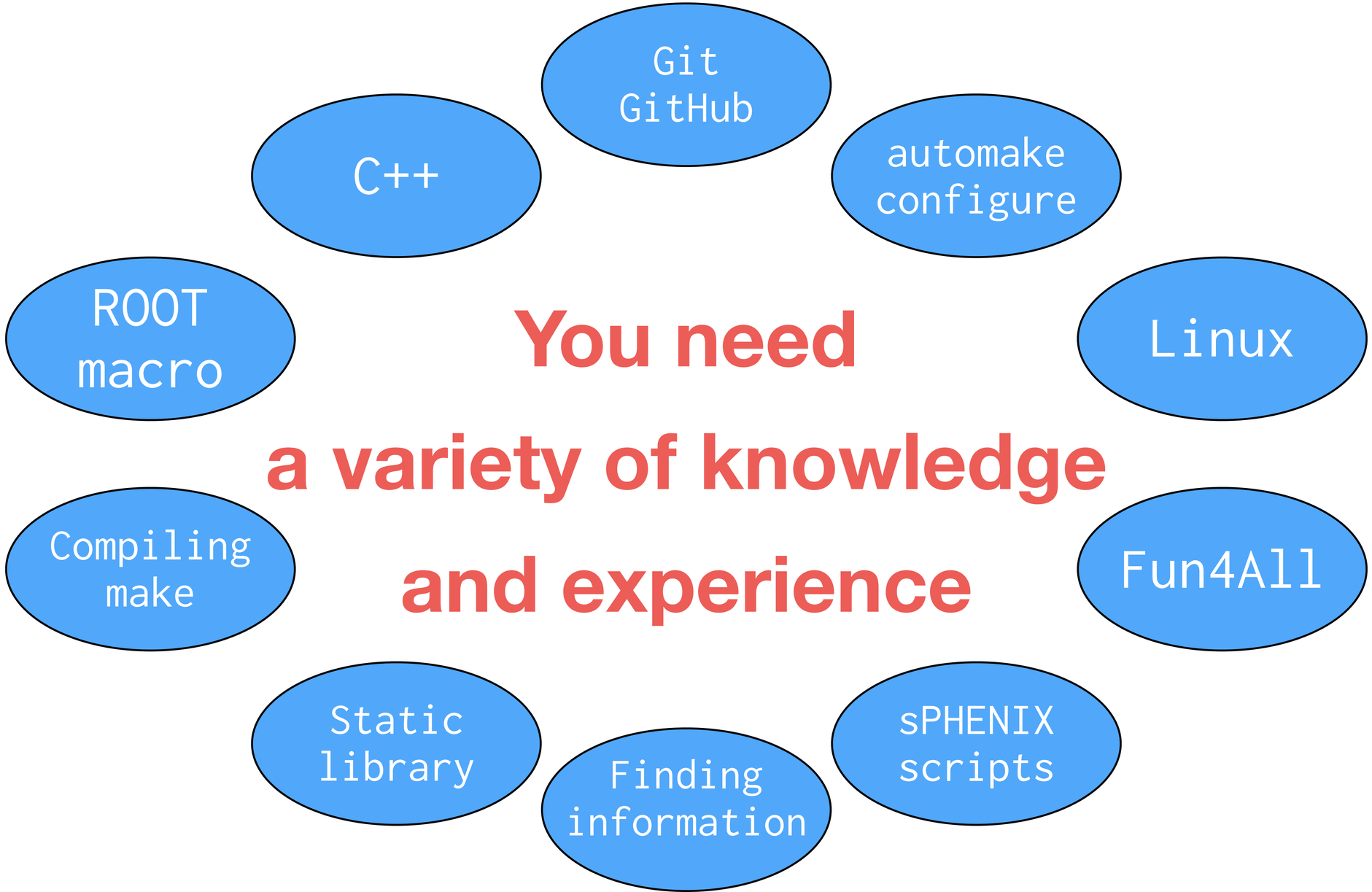
How?

This is the question!

what/who
why
where
when
how

Why is Fun4All difficult?

Why is Fun4All difficult?



What can we start with?

A minimum program is good to start with.

In the case of C++:

What can we start with?

A minimum program is good to start with.

In the case of C++:

```
[genki 18:29:48 fun4all_tutorial] $ /bin/cat cpp_minimum.cc
int main(){}
[genki 18:29:49 fun4all_tutorial] $ g++ cpp_minimum.cc
[genki 18:29:56 fun4all_tutorial] $ ./a.out
```

It's useless, I know.

In the case of a ROOT macro:

```
[genki 18:31:41 fun4all_tutorial] $ /bin/cat root_minimum.cc
void root_minimum(){}
[genki 18:31:45 fun4all_tutorial] $ root root_minimum.cc
root [0]
Processing root_minimum.cc...
root [1] .q
```

It's also useless.

What can we start with?

In the case of Fun4All:

```
1  #include <fun4all/Fun4AllServer.h>
2
3  R__LOAD_LIBRARY(libfun4all.so)
4
5  int Fun4All_minimum()
6  {
7
8    Fun4AllServer *se = Fun4AllServer::instance();
9
10   se->run( 1 );
11   se->End();
12   delete se;
13
14   gSystem->Exit(0);
15   return 0;
16 }
```

Fun4All_minimum.C

terminal

```
[nukazuka@sphnx04 22:38:47 tutorial] $  root -q -b Fun4All_minimum.C

Processing Fun4All_minimum.C...
Fun4AllServer::setRun(): could not get timestamp for run  0, using t
ics(0) timestamp: Wed Dec 31 19:00:00 1969
----------------------------------------

List of Nodes in Fun4AllServer:
Node Tree under TopNode TOP
TOP (PHCompositeNode)/
   DST (PHCompositeNode)/
   RUN (PHCompositeNode)/
   PAR (PHCompositeNode)/
```

This is a ROOT macro.

What can we start with?

You can get the sample codes: https://github.com/nukazuka/INTT_Fun4All_Tutorial

Get them by

```
$ git clone git@github.com:nukazuka/INTT_Fun4All_Tutorial.git
```

in any directory

```
[genki 17:55:14 fun4all_tutorial] $ git clone git@github.com:nukazuka/INTT_Fun4All_Tutorial.git
Cloning into 'INTT_Fun4All_Tutorial'...
X11 forwarding request failed on channel 0
remote: Enumerating objects: 50, done.
remote: Counting objects: 100% (50/50), done.
remote: Compressing objects: 100% (33/33), done.
remote: Total 50 (delta 8), reused 47 (delta 8), pack-reused 0
Receiving objects: 100% (50/50), 443.36 KiB | 417.00 KiB/s, done.
Resolving deltas: 100% (8/8), done.
```

```
[nukazuka@sphnx04 21:26:54 work_now] $ ls -1
Fun4All_Intt_cosmics.C
Fun4All_minimum_2.C
Fun4All_minimum_3.C
Fun4All_minimum.C
header_common.h
README.md
sample_module_1
sample_module_2
sample_module_3
```

What can we start with?

```
1  #include <fun4all/Fun4AllServer.h>
2
3  R__LOAD_LIBRARY(libfun4all.so)
4
5  int Fun4All_minimum()
6  {
7
8    Fun4AllServer *se = Fun4AllServer::instance();
9
10   se->run( 1 );
11   se->End();
12   delete se;
13
14   gSystem->Exit(0);
15   return 0;
16 }
```

Fun4All_minimum.C

This is a ROOT macro.

Let’s see the sample code line by line.

What can we start with?

```
1 #include <fun4all/Fun4AllServer.h>
```

include statement to include fun4all/Fun4Allserver.h

To find the file

1. Check the environment variable ROOT_INCLUDE_PATH:

```
$ echo $ROOT_INCLUDE_PATH
```

You need to execute a shell script provided by sPHENIX to set up analysis environment:

```
$ source /opt/sphenix/core/bin/sphenix_setup.sh
```

```
[nukazuka@sphnx04 22:38:53 tutorial] $ echo $ROOT_INCLUDE_PATH
./:/sphenix/tg/tg01/commissioning/INTT/repositories/tutorials/AnaTutorial/install/include:/sphenix/tg/tg01/commissioning/INTT/repositories/tutorials/AnaTutorial/install/include
/anatutorial:/sphenix/tg/tg01/commissioning/INTT/work/genki/repos/INTT/general_codes/hachiya/F4AInttRead/install/include:/sphenix/tg/tg01/commissioning/INTT/work/genki/repos/IN
TT/general_codes/hachiya/F4AInttRead/install/include/inttread:/sphenix/tg/tg01/commissioning/INTT/work/genki/repos/INTT/general_codes/genki/Fun4All_codes/install/include:/sphen
ix/tg/tg01/commissioning/INTT/work/genki/repos/INTT/general_codes/genki/Fun4All_codes/install/include/inttanalysiscosmic:/sphenix/tg/tg01/commissioning/INTT/work/genki/repos/co
resoftware/simulation/g4simulation/g4intt/install/include:/sphenix/tg/tg01/commissioning/INTT/work/genki/repos/coresoftware/simulation/g4simulation/g4intt/install/include/g4int
t:/sphenix/tg/tg01/commissioning/INTT/repositories/libraries/include:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include:/cvmfs/sphenix.sdcc.bnl.gov/gcc-
12.1.0/release/release_ana/ana.382/include:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/ffarawobjects:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/relea
se/release_ana/ana.382/include/JSON:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/half:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/a
na.382/include/torch:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/g4detectors:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/i
nclude/eventplane:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/kineto:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/g
4decayer:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/phfield:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/LHAPDF:/c
vmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/c10:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/oneapi:/cvmfs/sphenix.sd
cc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/DDCond:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/g4tracking:/cvmfs/sphenix.sdcc.bnl.g
ov/gcc-12.1.0/release/release_ana/ana.382/include/litecaloeval:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/g4intt:/cvmfs/sphenix.sdcc.bnl.gov/gcc
-12.1.0/release/release_ana/ana.382/include/phool:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/boost:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/releas
e/release_ana/ana.382/include/Pythia8Plugins:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/calib_emc_pi0:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/rel
ease/release_ana/ana.382/include/ffaobjects:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/EvtGenBase:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release
/release_ana/ana.382/include/flowafterburner:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/google:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/re
```

It’s not human-readable. Paths are separated by “:”. Let’s make it better.

What can we start with?

```
1 #include <fun4all/Fun4AllServer.h>
```

include statement to include fun4all/Fun4Allserver.h

To find the file

1. Check the environment variable ROOT_INCLUDE_PATH:
   ```
   $ echo $ROOT_INCLUDE_PATH
   ```
2. To separate the paths: Log in to the SDCC servers:
   ```
   $ echo $ROOT_INCLUDE_PATH  | sed -e  “s/:/\n/g”
   ```
   sed command replaces : to \n.

```
[nukazuka@sphnx04 22:48:37 tutorial] $ sed_path  $ROOT_INCLUDE_PATH
./
/sphenix/tg/tg01/commissioning/INTT/repositories/tutorials/AnaTutorial/install/include
/sphenix/tg/tg01/commissioning/INTT/repositories/tutorials/AnaTutorial/install/include/anatutorial
/sphenix/tg/tg01/commissioning/INTT/work/genki/repos/INTT/general_codes/hachiya/F4AInttRead/install/include
/sphenix/tg/tg01/commissioning/INTT/work/genki/repos/INTT/general_codes/hachiya/F4AInttRead/install/include/inttread
/sphenix/tg/tg01/commissioning/INTT/work/genki/repos/INTT/general_codes/genki/Fun4All_codes/install/include
/sphenix/tg/tg01/commissioning/INTT/work/genki/repos/INTT/general_codes/genki/Fun4All_codes/install/include/inttanalysiscosmic
/sphenix/tg/tg01/commissioning/INTT/work/genki/repos/coresoftware/simulation/g4simulation/g4intt/install/include
/sphenix/tg/tg01/commissioning/INTT/work/genki/repos/coresoftware/simulation/g4simulation/g4intt/install/include/g4intt
/sphenix/tg/tg01/commissioning/INTT/repositories/libraries/include
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/ffarawobjects
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/JSON
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/half
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/torch
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/g4detectors
```

Much better! Let’s find paths which have a certain word.

What can we start with?

```
1 #include <fun4all/Fun4AllServer.h>
```

include statement to include fun4all/Fun4Allserver.h

To find the file

1. Check the environment variable ROOT_INCLUDE_PATH:
   ```
   $ echo $ROOT_INCLUDE_PATH
   ```
2. To separate the paths: Log in to the SDCC servers:
   ```
   $ echo $ROOT_INCLUDE_PATH  | sed -e  “s/:/\n/g”
   ```
   sed command replaces : to \n.
3. Select paths which contain fun4all
   ```
   $ echo $ROOT_INCLUDE_PATH  | sed -e "s/:/\n/g" | grep fun4all
   ```

```
[nukazuka@sphnx04 22:49:13 tutorial] $ sed_path  $ROOT_INCLUDE_PATH | grep fun4all
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/fun4all
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/fun4allutils
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/fun4allraw
```

← this one!

What can we start with?

```
1 #include <fun4all/Fun4AllServer.h>
```

include statement to include fun4all/Fun4Allserver.h

To find the file

1. Check the environment variable ROOT_INCLUDE_PATH:
   ```
   $ echo $ROOT_INCLUDE_PATH
   ```
2. To separate the paths: Log in to the SDCC servers:
   ```
   $ echo $ROOT_INCLUDE_PATH  | sed -e  “s/:/\n/g”
   ```
   sed command replaces : to \n.
3. Select paths which contain fun4all
   ```
   $ echo $ROOT_INCLUDE_PATH  | sed -e "s/:/\n/g" | grep fun4all
   ```
4. Confirmation **Note: The path depends on your environment**
   ```
   $ ls /cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/fun4all/Fun4AllServer.h
   ```

Try them

```
[nukazuka@sphnx04 22:51:51 tutorial] $ ls /cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/fun4all/Fun4AllServer.h
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/fun4all/Fun4AllServer.h
```

What can we start with?

Rtypes.h

◆R__LOAD_LIBRARY

#define R__LOAD_LIBRARY (LIBRARY)

Definition at line 467 of file Rtypes.h.

```
1 #include <fun4all/Fun4AllServer.h>
2
3 R__LOAD_LIBRARY(libfun4all.so)
```

Learn C language more if you don't know.

R__LOAD_LIBRARY is a function-like macro defined in ROOT to load a library.
A shared library libfun4all.so is loaded. Where is it?

1. Check the environment variable LD_LIBRARY_PATH:
   `$ echo $LD_LIBRARY_PATH`

```
[nukazuka@sphnx04 23:12:14 tutorial] $ echo $LD_LIBRARY_PATH
/sphenix/tg/tg01/commissioning/INTT/repositories/tutorials/AnaTutorial/install/lib:/sphenix/tg/tg01/commissioning/INTT/work/genki/repos/INTT/general_codes/hachiya/F4AInttRead/install/lib:/sphenix/tg/tg01/commissioning/INTT/work/genki/repos/INTT/general_codes/genki/Fun4All_codes/install/lib:/sphenix/tg/tg01/commissioning/INTT/work/genki/repos/coresoftware/simulation/g4simulation/g4intt/install/lib:/sphenix/tg/tg01/commissioning/INTT/repositories/libraries/lib:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/opt/sphenix/core/gcc/12.1.0-57c96/x86_64-centos7/lib:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/opt/sphenix/core/gcc/12.1.0-57c96/x86_64-centos7/lib64:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/opt/sphenix/core/binutils/2.37-355ed/x86_64-centos7/lib:.:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/lib64:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/lib:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/opt/sphenix/utils/lib64:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/opt/sphenix/utils/lib:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/opt/sphenix/core/lib:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/opt/sphenix/core/root-6.26.06.p01/lib:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/opt/sphenix/core/geant4.10.07.p04/lib64:/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/opt/sphenix/core/lhapdf-5.9.1/lib:/afs/rhic.bnl.gov/app/insure-7.5.5/lib:/usr/local/lib64:/usr/lib64
```

It's not human-readable again. Let's do the same.

What can we start with?

Rtypes.h

◆ R__LOAD_LIBRARY

#define R__LOAD_LIBRARY (LIBRARY)

Definition at line 467 of file Rtypes.h.

```
1 #include <fun4all/Fun4AllServer.h>
2
3 R__LOAD_LIBRARY(libfun4all.so)
```

R__LOAD_LIBRARY is a function-like macro defined in ROOT to load a library.
A shared library libfun4all.so is loaded. Where is it?

1. Check the environment variable LD_LIBRARY_PATH:
   `$ echo $LD_LIBRARY_PATH`
2. Replce : to \n (or something else you like)
   `$ echo $LD_LIBRARY_PATH | sed -e “s/:/\n/g”`

```
[nukazuka@sphnx04 23:12:15 tutorial] $ echo $LD_LIBRARY_PATH | sed -e "s/:/\n/g"
/sphenix/tg/tg01/commissioning/INTT/repositories/tutorials/AnaTutorial/install/lib
/sphenix/tg/tg01/commissioning/INTT/work/genki/repos/INTT/general_codes/hachiya/F4AInttRead/install/lib
/sphenix/tg/tg01/commissioning/INTT/work/genki/repos/INTT/general_codes/genki/Fun4All_codes/install/lib
/sphenix/tg/tg01/commissioning/INTT/work/genki/repos/coresoftware/simulation/g4simulation/g4intt/install/lib
/sphenix/tg/tg01/commissioning/INTT/repositories/libraries/lib
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/opt/sphenix/core/gcc/12.1.0-57c96/x86_64-centos7/lib
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/opt/sphenix/core/gcc/12.1.0-57c96/x86_64-centos7/lib64
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/opt/sphenix/core/binutils/2.37-355ed/x86_64-centos7/lib
.
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/lib64
```

It’s better but still not clear…
Let’s search the file.

What can we start with?

```
1 #include <fun4all/Fun4AllServer.h>
2
3 R__LOAD_LIBRARY(libfun4all.so)
```

R__LOAD_LIBRARY is a function-like macro defined in ROOT to load a library.
A shared library libfun4all.so is loaded. Where is it?

1. Check the environment variable LD_LIBRARY_PATH:
   `$ echo $LD_LIBRARY_PATH`
2. Replce : to \n (or something else you like):
   `$ echo $LD_LIBRARY_PATH | sed -e “s/:/\n/g”`
3. Search libfun4all.so:
   `$ echo $LD_LIBRARY_PATH | sed -e "s/:/\n/g" | xargs -I {} find {} -name “libfun4all.so"`

```
[nukazuka@sphnx04 23:14:14 tutorial] $ echo $LD_LIBRARY_PATH | sed -e "s/:/\n/g" | xargs -I {} find {} -name "libfun4all.so"
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/lib/libfun4all.so
```

What can we start with?

HANDS ON!
#2

```
1 #include <fun4all/Fun4AllServer.h>
2
3 R__LOAD_LIBRARY(libfun4all.so)
```

R__LOAD_LIBRARY is a function-like macro defined in ROOT to load a library.
A shared library libfun4all.so is loaded. Where is it?

Try them

1. Check the environment variable LD_LIBRARY_PATH:
   `$ echo $LD_LIBRARY_PATH`
2. Replce : to \n (or something else you like):
   `$ echo $LD_LIBRARY_PATH | sed -e “s/:/\n/g”`
3. Search libfun4all.so:
   `$ echo $LD_LIBRARY_PATH | sed -e "s/:/\n/g" | xargs -I {} find {} -name “libfun4all.so"`

Another way I could come up:

```
$ for dir in `echo $LD_LIBRARY_PATH | sed -e "s/:/\n/g"` ; do find $dir -name "libfun4all.so" ; done
```

What can we start with?

```
1 #include <fun4all/Fun4AllServer.h>
2
3 R__LOAD_LIBRARY(libfun4all.so)
4
5 int Fun4All_minimum()
6 {
7
8   Fun4AllServer *se = Fun4AllServer::instance();
```

Including the header file and loading the shared library are for here.

A pointer of an instance of the Fun4AllServer class is assigned to “se”.

What can we start with?

```
1  #include <fun4all/Fun4AllServer.h>
2
3  R__LOAD_LIBRARY(libfun4all.so)
4
5  int Fun4All_minimum()
6  {
7
8    Fun4AllServer *se = Fun4AllServer::instance();
9
10   se->run( 1 );   ← Running analysis processes for the given number of events.
11   se->End();      ← Some processes are launched at the end of event-by-event processes.
12   delete se;      ← Just delete it.
13
14   gSystem->Exit(0); ← Just do it.
15   return 0;         ← Just do it.
16 }
```

Including the header file and loading the shared library are for here.

This super simple macro takes no input file and outputs nothing. 1 event is processed.

What can we start with?

```
[nukazuka@sphnx04 04:58:09 INTT_Fun4All_Tutorial] $ root -q -b Fun4All_minimum.C

Processing Fun4All_minimum.C...
Fun4AllServer::setRun(): could not get timestamp for run  0, using tics(0) timestamp: Wed Dec 31 19:00:00 1969
-----------------------------------

List of Nodes in Fun4AllServer:
Node Tree under TopNode TOP
TOP (PHCompositeNode)/
   DST (PHCompositeNode)/
   RUN (PHCompositeNode)/
   PAR (PHCompositeNode)/
```

Execute Fun4All_minimum.C.

Practical example

It depends on what you want to do. For example:

- inputting raw file(s)
- inputting DST file(s)
- Monte-Carlo as an input

- running someone’s analysis codes
- running your analysis codes

- Outputting results to DST file(s)
- Outputting results to histograms/TTrees

This super simple macro takes no input file and outputs nothing. 1 event is processed.

Practical example

It depends on what you want to do. For example:

- inputting raw file(s)
- inputting DST file(s)
- Monte-Carlo as an input

- running someone's analysis codes
- running your analysis codes

- Outputting results to DST file(s)
- Outputting results to histograms/TTrees

Let's try a simple case.

This super simple macro takes no input file and outputs nothing. 1 event is processed.

Practical example

```
 1 #include <G4_Input.C>
 2
 3 #include <ffamodules/FlagHandler.h>
 4 #include <ffamodules/HeadReco.h>
 5 #include <ffamodules/SyncReco.h>
 6 #include <ffamodules/CDBInterface.h>
 7
 8 #include <fun4all/Fun4AllDstOutputManager.h>
 9 #include <fun4all/Fun4AllOutputManager.h>
10 #include <fun4all/Fun4AllServer.h>
11
12 #include <phool/PHRandomSeed.h>
13 #include <phool/recoConsts.h>
14
15 R__LOAD_LIBRARY(libfun4all.so)
16
17 #include <tutorial.h>
18 R__LOAD_LIBRARY( libtutorial.so )
19
20 int Fun4All_minimum_2(
21                       int nEvents = 1, //5,
22                       const string &inputFile = "https://www.phenix.bnl.gov/WWW/publish/phnxbld/sPHENIX/files/sPHENIX_G4Hits_sHijing_9-11fm_00000_00010.root",
23                       const int skip = 0
24                       )
25 {
26
27   Fun4AllServer *se = Fun4AllServer::instance();
28
29   INPUTREADHITS::filename[0] = inputFile;
30   InputInit();
31   InputRegister();
32
33   tutorial* analysis_module = new tutorial( "name" );
34   se->registerSubsystem( analysis_module );
35
36   se->skip(skip);
37   se->run(nEvents);
38   se->End();
39   delete se;
40
41   gSystem->Exit(0);
42   return 0;
43 }
-UU-:---  F1  Fun4All_minimum_2.C   All   L2     (C++//l Abbrev) ----------
```

Fun4All_minimum_2.C

Analysis module

You need to write your analysis codes for a certain class.
The class is called “analysis module”. Analysis modules need to

- inherit the SubsysReco class (class inheritance)
- implement functions in the SubsysReco (polymorphism)
- be registered to Fun4AllServer by Fun4AllServer::registerSubsystem

Analysis module

You need to write your analysis codes for a certain class.
The class is called “analysis module”. Analysis modules need to

- inherit the SubsysReco class (class inheritance)
- implement functions in the SubsysReco (polymorphism)
- be registered to Fun4AllServer by Fun4AllServer::registerSubsystem

You can learn class inheritance (継承, 継承, 상속)

and polymorphism (多态, ポリモーフィズム, 다형성)

in C++ textbooks. It’s not easy to understand them without taking time to learn.

Analysis module

The standard way to implement the class, add it to the ROOT macro, and run it is

1. generating a template by CreateSubsysRecoModule.pl
   ```
   $ CreateSubsysRecoModule.pl [name_of_the_module] [options]
   ```
   Joseph's minimum example is also a good start.
2. generating the configuration files by autogen.sh
   ```
   $ autogen.sh --prefix=[install_path]
   ```
3. implementing the header file (*.h) and the source file (*.cc) by yourself.
4. compiling the analysis module by make command
   ```
   $ make
   ```
5. installing the library (*.so) and the header file (*.h)
   ```
   $ make install
   ```
6. setting your LD_LIBRARY_PATH and ROOT_INCLUDE_PATH
   (here is a little bit complicated. The explanation is given later.)
7. adding an include statement and R__LOAD_LIBRARY macro to your ROOT macro.
   (It's also given later.)

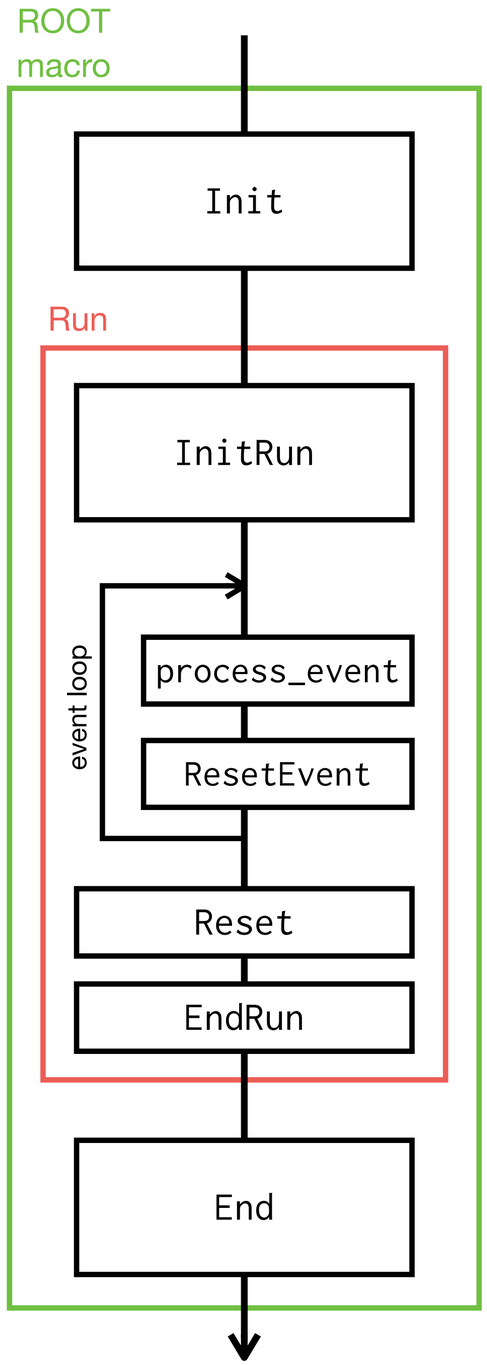
SybsysReco class [Github]

Code Blame 73 lines (58 loc) · 2.06 KB

```
1   // Tell emacs that this is a C++ source
2   //  -*- C++ -*-.
3   #ifndef FUN4ALL_SUBSYSRECO_H
4   #define FUN4ALL_SUBSYSRECO_H
5
6   #include "Fun4AllBase.h"
7
8   #include <string>
9
10  class PHCompositeNode;
11
12  /** Base class for all reconstruction and analysis modules to be
13   *  used under the Fun4All framework.
14   *
15   *  If you write a reconstruction/analysis module, you must derive
16   *  from this base class and you have to implement this class methods.
17   *  None of these are strictly required as far as C++ is concerned, but as
18   *  far as your job is concerned, at least process_event(), to do the
19   *  job, and InitRun(), to initialize, should be implemented.
20   *
21   */
```

The only header is in Fun4All. The actual behavior of functions should be implemented in your inheriting class by yourself (polymorphism). The class itself is not too complicated.

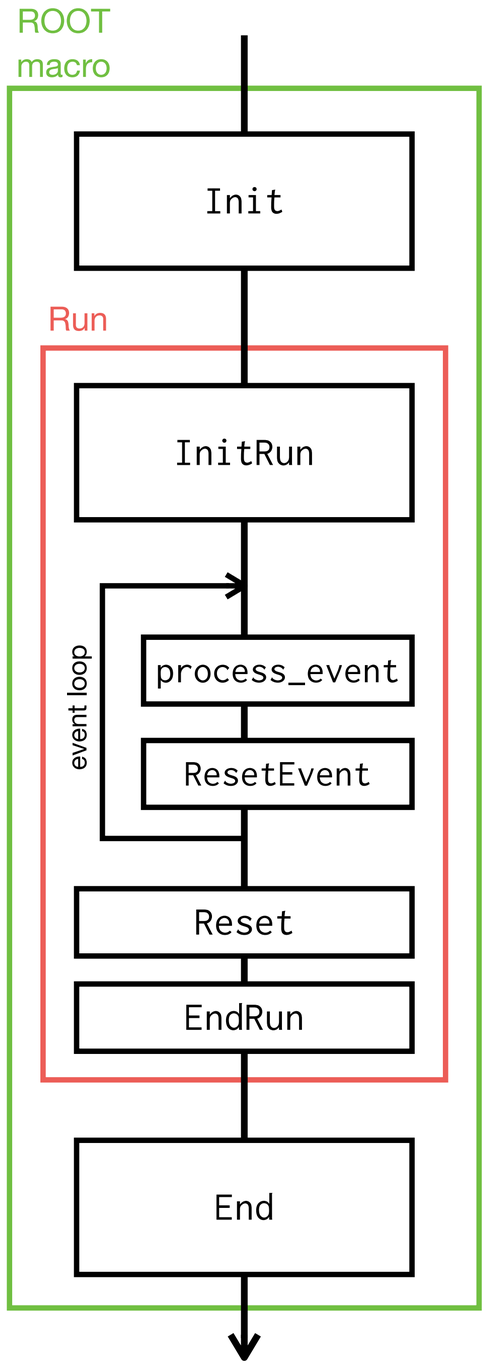
```
23  class SubsysReco : public Fun4AllBase
24  {
25   public:
26    /** dtor.
27        Does nothing as this is a base class only.
28    */
29    ~SubsysReco() override {}
30
31    /// Called at the end of all processing.
32    virtual int End(PHCompositeNode * /*topNode*/) { return 0; }
33
34    /// Called at the end of each run.
35    virtual int EndRun(const int /*runnumber*/) { return 0; }
36
37    /** Called during initialization.
38        Typically this is where you can book histograms, and e.g.
39        register them to Fun4AllServer (so they can be output to file
40        using Fun4AllServer::dumpHistos() method).
41     */
42    virtual int Init(PHCompositeNode * /*topNode*/) { return 0; }
43
44    /** Called for first event when run number is known.
45        Typically this is where you may want to fetch data from
46        database, because you know the run number.
47     */
48    virtual int InitRun(PHCompositeNode * /*topNode*/) { return 0; }
49
50    /** Called for each event.
51        This is where you do the real work.
52    */
53    virtual int process_event(PHCompositeNode * /*topNode*/) { return 0; }
54
55    /// Reset.
56    virtual int Reset(PHCompositeNode * /*topNode*/) { return 0; }
57
58    /// Clean up after each event.
59    virtual int ResetEvent(PHCompositeNode * /*topNode*/) { return 0; }
60
61    void Print(const std::string & /*what*/ = "ALL") const override {}
62
63   protected:
64    /** ctor.
65        @param name is the reference used inside the Fun4AllServer
66    */
67    SubsysReco(const std::string &name = "NONAME")
68      : Fun4AllBase(name)
69    {
70    }
71  };
72
73  #endif
```



SybsysReco/Your analysis module class [Github]

The functions to be executed by Fun4AllServer take PHCompositNode* as an argument.
For example: `int process_event(PHCompositeNode *)`

SybsysReco/Your analysis module class [Github]



The functions to be executed by Fun4AllServer take PHCompositNode* as an argument.
For example: `int process_event(PHCompositeNode *)`

Preparation for the run. For example, making histograms.

The main part of your analysis

Finalization. Writing histogram objects to output files, etc.

Making your own analysis module

1. Generating a template by CreateSubsysRecoModule.pl

```
$ CreateSubsysRecoModule.pl [name_of_the_module] [options]
```

```
[nukazuka@sphnx04 00:47:17 module] $ CreateSubsysRecoModule.pl  tutorial --all
[nukazuka@sphnx04 00:47:20 module] $ ls -1
autogen.sh
configure.ac
Makefile.am
tutorial.cc
tutorial.h
```

2. Generating the configuration files by autogen.sh

```
$ autogen.sh --prefix=[install_path]
```

```
[nukazuka@sphnx04 00:47:22 module] $ mkdir install
[nukazuka@sphnx04 00:48:25 module] $ ./autogen.sh --prefix=$PWD/install
libtoolize: putting auxiliary files in `.'.
libtoolize: linking file `./ltmain.sh'
libtoolize: Consider adding `AC_CONFIG_MACRO_DIR([m4])' to configure.ac and
libtoolize: rerunning libtoolize, to keep the correct libtool macros in-tree.
libtoolize: Consider adding `-I m4' to ACLOCAL_AMFLAGS in Makefile.am.
configure.ac:7: installing './config.guess'
configure.ac:7: installing './config.sub'
configure.ac:4: installing './install-sh'
configure.ac:4: installing './missing'
Makefile.am: installing './depcomp'
```

The install directory is arbitral.

Making your own analysis module

```
[nukazuka@sphnx04 00:49:01 module] $ ls
aclocal.m4  autom4te.cache  config.log     config.sub  configure.ac  install     libtool     Makefile     Makefile.in  tutorial.cc
autogen.sh  config.guess    config.status  configure   depcomp       install-sh  ltmain.sh   Makefile.am  missing      tutorial.h
```

4. compiling the analysis module by make

   `$ make`

```
[nukazuka@sphnx04 00:50:28 module] $ make
echo "//*** this is a generated file. Do not commit, do not edit" > testexternals.cc
echo "int main()" >> testexternals.cc
echo "{" >> testexternals.cc
echo "  return 0;" >> testexternals.cc
echo "}" >> testexternals.cc
make  all-am
make[1]: Entering directory `/direct/sphenix+tg+tg01/commissioning/INTT/work/genki/analysis/tutorial/module'
/bin/sh ./libtool  --tag=CXX   --mode=compile /cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/opt/sphenix/core/gcc/12.1.0-57c96/x86_64-ce
AGE_TARNAME=\"tutorial\" -DPACKAGE_VERSION=\"1.00\" -DPACKAGE_STRING=\"tutorial\ 1.00\" -DPACKAGE_BUGREPORT=\"\" -DPACKAGE_URL=\
STDC_HEADERS=1 -DHAVE_SYS_TYPES_H=1 -DHAVE_SYS_STAT_H=1 -DHAVE_STDLIB_H=1 -DHAVE_STRING_H=1 -DHAVE_MEMORY_H=1 -DHAVE_STRINGS_H=1
ISTD_H=1 -DHAVE_DLFCN_H=1 -DLT_OBJDIR=\".libs/\" -I.  -I/sphenix/u/nukazuka/work_now/module/install/include -I/cvmfs/sphenix.sdc
/include -isystem/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/opt/sphenix/core/root-6.26.06.p01/include   -g -O2 -std=c++17 -Wall -Wer
po -c -o tutorial.lo tutorial.cc
libtool: compile:  /cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/opt/sphenix/core/gcc/12.1.0-57c96/x86_64-centos7/bin/g++ -DPACKAGE_NAM
```

   To check whether compiling was done successfully: `$ echo $?`

5. installing the library (*.so) and the header file (*.h)

   `$ make install`

Making your own analysis module

5. installing the library (*.so) and the header file (*.h)

```
$ make install
```

```
[nukazuka@sphnx04 03:28:25 module] $ make install
make  install-am
make[1]: Entering directory `/direct/sphenix+tg+tg01/commissioning/INTT/work/genki/analysis/tutorial/module'
make[2]: Entering directory `/direct/sphenix+tg+tg01/commissioning/INTT/work/genki/analysis/tutorial/module'
 /usr/bin/mkdir -p '/sphenix/u/nukazuka/work_now/module/install/lib'
 /bin/sh ./libtool   --mode=install /usr/bin/install -c   libtutorial.la '/sphenix/u/nukazuka/work_now/module/i
libtool: install: /usr/bin/install -c .libs/libtutorial.so.0.0.0 /sphenix/u/nukazuka/work_now/module/install/li
libtool: install: (cd /sphenix/u/nukazuka/work_now/module/install/lib && { ln -s -f libtutorial.so.0.0.0 libtut
0.0 libtutorial.so.0; }; })
```

```
[nukazuka@sphnx04 03:28:32 module] $ ls install
include  lib
[nukazuka@sphnx04 03:30:12 module] $ tree install
install
├── include
│   └── tutorial
│       └── tutorial.h
└── lib
    ├── libtutorial.la
    ├── libtutorial.so -> libtutorial.so.0.0.0
    ├── libtutorial.so.0 -> libtutorial.so.0.0.0
    └── libtutorial.so.0.0.0

3 directories, 5 files
```

You need to inform the path to this directory to ROOT to use them.

Analysis module

in a new directory.
Give original name to the module.

The standard way to implement the class, add it to the ROOT macro, and run it is

1. generating a template by CreateSubsysRecoModule.pl
   ```
   $ CreateSubsysRecoModule.pl [name_of_the_module] [options]
   ```
   Joseph's minimum example is also a good start.
2. generating the configuration files by autogen.sh
   ```
   $ autogen.sh --prefix=[install_path]
   ```
3. implementing the header file (*.h) and the source file (*.cc) by yourself.
4. compiling the analysis module by make command
   ```
   $ make
   ```
5. installing the library (*.so) and the header file (*.h)
   ```
   $ make install
   ```

Try them

Any directory is OK.
If you have no preference, make and use
$PWD/install

HANDS ON!
#4

Making your own analysis module

6. setting your `LD_LIBRARY_PATH` and `ROOT_INCLUDE_PATH`

   `LD_LIBRARY_PATH`: an environmental variable generally used in Linux to find libraries

   `ROOT_INCLUDE_PATH`: an environmental variable introduced by ROOT to find header files

   The basic setup of them is done by `/opt/sphenix/core/bin/sphenix_setup.sh`.

```
[nukazuka@sphnx04 03:41:23 ~] $ sed_path $ROOT_INCLUDE_PATH
./
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.387/include
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.387/include/g4mbd
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.387/include/ffarawobjects
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.387/include/JSON
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.387/include/half
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.387/include/torch
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.387/include/g4detectors
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.387/include/eventplane
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.387/include/kineto
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.387/include/g4decayer
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.387/include/phfield
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.387/include/LHAPDF
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.387/include/c10
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.387/include/oneapi
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.387/include/DDCond
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.387/include/g4tracking
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.387/include/litecaloeval
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.387/include/g4intt
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.387/include/phool
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.387/include/boost
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.387/include/Pythia8Plugins
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.387/include/calib_emc_pi0
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.387/include/hijingflipafterburner
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.387/include/ffaobjects
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.387/include/EvtGenBase
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.387/include/flowafterburner
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.387/include/google
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.387/include/g4tpc
```

```
[nukazuka@sphnx04 03:41:14 ~] $ sed_path $LD_LIBRARY_PATH
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/opt/sphenix/core/gcc/12.1.0-57c96/x86_64-centos7/lib
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/opt/sphenix/core/gcc/12.1.0-57c96/x86_64-centos7/lib64
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/opt/sphenix/core/binutils/2.37-355ed/x86_64-centos7/lib
.
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.387/lib64
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.387/lib
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/opt/sphenix/utils/lib64
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/opt/sphenix/utils/lib
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/opt/sphenix/core/lib
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/opt/sphenix/core/root-6.26.06.p01/lib
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/opt/sphenix/core/geant4.10.07.p04/lib64
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/opt/sphenix/core/lhapdf-5.9.1/lib
/afs/rhic.bnl.gov/app/insure-7.5.5/lib
/usr/local/lib64
/usr/lib64
```

Files in the paths in the variables can be used with only the file name.

Here, sed_path command is defined by Genki:

```
function sed_path ()
{
    echo $@ | sed -e "s/:/\n/g"
}
```

Making your own analysis module

6. setting your LD_LIBRARY_PATH and ROOT_INCLUDE_PATH

   To add paths to the variables, you can run /opt/sphenix/core/bin/setup_local.sh, for example,

```
 $ source /opt/sphenix/core/bin/setup_local.sh /sphenix/tg/tg01/commissioning/INTT/work/genki/repos/coresoftware/simulation/g4simulation/g4intt/install
```

```
[nukazuka@sphnx04 03:49:44 ~] $ sed_path $LD_LIBRARY_PATH
/sphenix/tg/tg01/commissioning/INTT/work/genki/repos/coresoftware/simulation/g4simulation/g4intt/install/lib
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/opt/sphenix/core/gcc/12.1.0-57c96/x86_64-centos7/lib
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/opt/sphenix/core/gcc/12.1.0-57c96/x86_64-centos7/lib64
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/opt/sphenix/core/binutils/2.37-355ed/x86_64-centos7/lib
.
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/r
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/r
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/o
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/o
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/o
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/o
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/o
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/o
/afs/rhic.bnl.gov/app/insure-7.5.5/lib
/usr/local/lib64
/usr/lib64
```

```
[nukazuka@sphnx04 03:49:51 ~] $ sed_path $ROOT_INCLUDE_PATH
./
/sphenix/tg/tg01/commissioning/INTT/work/genki/repos/coresoftware/simulation/g4simulation/g4intt/install/include
/sphenix/tg/tg01/commissioning/INTT/work/genki/repos/coresoftware/simulation/g4simulation/g4intt/install/include/g4intt
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/ffarawobjects
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/JSON
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/half
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/torch
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/g4detectors
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/eventplane
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/kineto
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/g4decayer
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/phfield
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/LHAPDF
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/c10
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/oneapi
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/DDCond
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/g4tracking
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/include/litecaloeval
```

Making your own analysis module

6. setting your `LD_LIBRARY_PATH` and `ROOT_INCLUDE_PATH`

   Typing the command every time is trouble. You should write it to ${HOME}/.bashrc, so the command is executed just after login. For example,

```
|test.sh x| +
 1 #!/bin_bash
 2
 3 source /opt/sphenix/core/bin/sphenix_setup.sh
 4 source /opt/sphenix/core/bin/setup_local.sh /sphenix/tg/tg01/commissioning/INTT/wo\
   rk/genki/repos/coresoftware/simulation/g4simulation/g4intt/install
```

   You can add multiple paths as arguments with separation with a space.

   To do it in a more user-friendly way, you can use my script:
   `/sphenix/tg/tg01/commissioning/INTT/repositories/libraries/intt_setup_v2.sh`
   An explanation of this script can be found in the backup slide (planned).

Making your own analysis module

7. adding an include statement and `R__LOAD_LIBRARY` macro to your ROOT macro.

```
 1 #include <G4_Input.C>
 2
 3 #include <ffamodules/FlagHandler.h>
 4 #include <ffamodules/HeadReco.h>
 5 #include <ffamodules/SyncReco.h>
 6 #include <ffamodules/CDBInterface.h>
 7
 8 #include <fun4all/Fun4AllDstOutputManager.h>
 9 #include <fun4all/Fun4AllOutputManager.h>
10 #include <fun4all/Fun4AllServer.h>
11
12 #include <phool/PHRandomSeed.h>
13 #include <phool/recoConsts.h>
14
15 R__LOAD_LIBRARY(libfun4all.so)
16
17 #include <tutorial.h>
18 R__LOAD_LIBRARY( libtutorial.so )
19
20 int Fun4All_minimum_2(
21                       int nEvents = 1, //5,
22                       const string &inputFile = "https://www.phenix.bnl.gov/WWW/publish/phnxbld/sPHENIX/files/sPHENIX_G4Hits_sHijing_9-11fm_00000_00010.root",
23                       const int skip = 0
24                       )
25 {
26
27   Fun4AllServer *se = Fun4AllServer::instance();
28
29   INPUTREADHITS::filename[0] = inputFile;
30   InputInit();
31   InputRegister();
32
33   tutorial* analysis_module = new tutorial( "name" );
34   se->registerSubsystem( analysis_module );
35
36   se->skip(skip);
37   se->run(nEvents);
38   se->End();
39   delete se;
40
41   gSystem->Exit(0);
42   return 0;
43 }
-UU-:---  F1  Fun4All_minimum_2.C   All   L2     (C++//l Abbrev) ----------
```

```
17 #include <tutorial.h>
18 R__LOAD_LIBRARY( libtutorial.so )
```

```
33   tutorial* analysis_module = new tutorial( "name" );
34   se->registerSubsystem( analysis_module );
```

Making your own analysis module

7. adding an include statement and `R__LOAD_LIBRARY` macro to your ROOT macro. and execute it!

```
[nukazuka@sphnx04 04:13:30 work_now] $ root -q -b Fun4All_minimum_2.C

Processing Fun4All_minimum_2.C...
tutorial::tutorial(const std::string &name) Calling ctor
tutorial::Init(PHCompositeNode *topNode) Initializing
Fun4AllServer::setRun(): could not get timestamp for run  0, using tics(0) timestamp: Wed Dec 31 19:00:00 1969
tutorial::InitRun(PHCompositeNode *topNode) Initializing for Run XXX
-----------------------------------

List of Nodes in Fun4AllServer:
Node Tree under TopNode TOP
TOP (PHCompositeNode)/
   DST (PHCompositeNode)/
   RUN (PHCompositeNode)/
   PAR (PHCompositeNode)/


tutorial::process_event(PHCompositeNode *topNode) Processing Event
tutorial::ResetEvent(PHCompositeNode *topNode) Resetting internal structures, prepare for next event
tutorial::EndRun(const int runnumber) Ending Run for Run 0
tutorial::End(PHCompositeNode *topNode) This is the End...
tutorial::Reset(PHCompositeNode *topNode) being Reset
tutorial::~tutorial() Calling dtor
```

Analysis module

6. setting your `LD_LIBRARY_PATH` and `ROOT_INCLUDE_PATH`
7. adding an include statement and `R__LOAD_LIBRARY` macro to your ROOT macro. Change the ROOT macro accordingly, then run it!

Try them

```
#include <G4_Input.C>

#include <ffamodules/FlagHandler.h>
#include <ffamodules/HeadReco.h>
#include <ffamodules/SyncReco.h>
#include <ffamodules/CDBInterface.h>

#include <fun4all/Fun4AllDstOutputManager.h>
#include <fun4all/Fun4AllOutputManager.h>
#include <fun4all/Fun4AllServer.h>

#include <phool/PHRandomSeed.h>
#include <phool/recoConsts.h>

R__LOAD_LIBRARY(libfun4all.so)

#include <tutorial.h>
R__LOAD_LIBRARY( libtutorial.so )

int Fun4All_minimum_2(
                      int nEvents = 1, //5,
                      const string &inputFile = "https://www.phenix.bnl.gov/WWW/publish/phnxbld/sPHENIX/files/sPHENIX_G4Hits_sHijing_9-11fm_00000_00010.root",
                      const int skip = 0
                      )
{

  Fun4AllServer *se = Fun4AllServer::instance();

  INPUTREADHITS::filename[0] = inputFile;
  InputInit();
  InputRegister();

  tutorial* analysis_module = new tutorial( "name" );
  se->registerSubsystem( analysis_module );

  se->skip(skip);
  se->run(nEvents);
  se->End();
  delete se;

  gSystem->Exit(0);
  return 0;
}
-UU-:---  F1  Fun4All_minim...
```

```
17 #include <tutorial.h>
18 R__LOAD_LIBRARY( libtutorial.so )
```

```
33   tutorial* analysis_module = new tutorial( "name" );
34   se->registerSubsystem( analysis_module );
```

HANDS ON!
#5, #6

Note: The name of the header file and the class depends on you.