mpdroot refactoring

Version 0.1 – 2025. 01. 24

Refactoring Timeline

Discussion on the topic of refactoring until the March release of mpdroot (25.03.25). Early march opening milestone+issues in git with detailed description of changes. Implementation during april 2025 (after MPD meeting), exept for 6, 7, 8, and 9 which can be done on an ongoing basis. Affecting whole code base, it is better to do whole refactoring in one large update rather than to split it in several months as any class renaming will break all classes depending on it.

Expected Impacts

Negative: Since this refactoring concerns (almost) all libraries and classes names, all macros will stop working. We support only 2 macros (`runReco.C`, `runMC.C`) and this should have not large impact on us. It is expected, that opening old root files will produce warnings on load. Naturally old user's macros will stop working we can prepare short guide for those willing to read.
Positive: We will be able to reuse same file names for the same type of tasks on multiple detectors. We will be able to build (link) against latest FairRoot (18.8.2 or 19.0.0). We should get rid of problems when multiple loading of the same library in ROOT unloads most of other libraries. We will fix problem of build+installation when during build headers are looked for in subdirectories but during runtime mpdroot expects libraries to be in the main directory thus forcing to copy libraries during install to two different locations (manually). Unification of codebase allows us further to force programming house style.

Contents

1 Libraries Renaming

All library names will start with Mpd as is a standard in Linux. Following list is only about the names of libraries, not the classes they provide.
Renaming concerns 25 out of current 59 libraries.

name	
current	new
MpdBase	
MpdDst	
MpdMiniEvent	
MpdField	
Passive	MpdPassive
MpdPid	
Bbc	MpdBbc
Emc	MpdEmc
Etof	MpdEtof
Ffd	MpdFfd
Bmd	MpdBmd
Mcord	MpdMcord
Lusi	MpdLusi
multi	MpdMulti
Sts	MpdSts
Tof	MpdTof
tpc	MpdTpc
tpcAlignment	MpdTpcAlignment
tpcClusterHitFinder	MpdTpcClusterHitFinder
tpcDigitizer	MpdTpcDigitizer
tpcFairTpc	MpdTpcFairTpc
tpcGeometry	MpdTpcGeometry
tpcPid	MpdTpcPid
MpdTpcActsTracker	
Zdc	MpdZdc
MpdDielectrons	
MpdCentralityAll	
MpdFlowEventPlane	
MpdPIDAll	
MpdEventPlaneAll	
MpdFsiTools	
MpdFemtoMaker	
MpdFemtoMakerUser	
MpdCumulantAnalysis	

name	
current	new
MpdFluctPt	
MpdPtMultAnalysis	
MpdPtAnalysis	
MpdPTNFluctCorr	
MpdGlobalPolarization	
Hyperons	MpdHyperons
MpdPhysics	
NicaMpdCuts	MpdCuts
NicaMpdFormat	MpdFormat
NicaMpdHelper	MpdHelper
NicaMpdTasks	MpdTasks
MpdNuclei	
MpdPairGGTracks	
MpdPairGLambdaTracks	
MpdPairKKTracks	
MpdPairPiKTracks	
MpdPairPiKsTracks	
MpdPairPiLambdaTracks	
MpdPairPiPiTracks	
MpdPairPKTracks	
MpdPhotons	
Kalman	MpdKalman
LHETrack	MpdLHETrack
MpdGenFactory	MpdGeneratorFactory
MpdGen	MpdGenerator
MpdGeneralGenerator	
UniGenFormat	MpdUniGenFormat
MpdMcDst	MpdMCDst
MpdMCStack	
DbUtils	MpdDbUtils
UniCommon	MpdUniCommon
UniDb	MpdUniDb
EventDisplay	MpdEventDisplay
QA	MpdQA

2 Library Merging

We have many small libraries providing mpdroot's functionality leading to complicated inter-library dependencies. It would be beneficial to merge these to more complex libraries. For instance MpdBase, MpdField, MpdPassive, and MpdPid could create together MpdCore library. MpdMiniEvent as a stand-alone library should be be a part of Core. MpdDst being dependent of large portions of MpdRoot should not be a part of Core as well. By the same logic we can talk about MpdTpc containing functionality s.a. MpdTpc, MpdTpcAlignment, MpdTpcClusterHitFinder, MpdTpcDigitizer, MpdTpcFairTpc (why Fair and 2× Tpc?), MpdTpcGeometry, and MpdTpcPid.

3 Directory Structure Change

The main reason to the directory structure change are 3 problems. No possibility to use the same (header) file name multiple times since headers are copied in single directory during the install and problem with proper including of headers inside sub-directories. This leads to the necessity during the install to keep 2 copies of headers. One inside the main folder and second one inside the proper sub-directory. It became a common approach that includes of libraries are (at least) inside the directory with the project name. To solve all 3 problems we will move headers into external directory called `include` (see Fig. 1) that will mirror the structure of include directory after install (e.g., `include/Mpd/Core/Base`) and inside cpp we

will use full paths to the headers (e.g., `#include "Mpd/Core/Base/Base.h"`). Source files will be placed in a `src` directory.

4 Examples and Macros

Moving source files to the directories `include` and `src` (see Section 3) allows us to unify approach on Examples and Macros. We can now put them to the corresponding subdirectories `examples` and `macros` with similar directory structure as of the source files. Question is, whether we want this and if yes, on which condition (build switch) macros and/or examples should be installed during build/installation and to what place.

5 Classes Renaming

In classes naming, we should remove Mpd from classes names. Instead, all classes have to be contained within the Mpd namespace (capital M, lowercase letters p and d). Files also may not contain Mpd in their names. We will introduce also "second level" namespaces, e.g., Mpd::Tpc::SomeClass. This will allow re-using SomeClass name for multiple detectors. Especially, if this class does the same but for another detector. Since we include headers with full path, we won't have clashes when using multiple headers of the same name. List of all namespaces will be kept in separate file/documentation.

6 Virtual Classes

Instead of using preposition Abstract we will use (letter) I as in Interface. This is more common in the programming world and it will make names of classes and files containing them less verbal (`AbstractXXX.h` → `IXXX.h`).

7 Guard Rails

While `#infdef HEADER_INCLUDED` works, it is probably better to switch to `#pragma once` which is more fool-proof and does not lead to errors if two headers use same guard rail. While this pragma is not part of a C++ standard, it is supported by all major compilers and there are no plans to remove it.

8 Removing cout, cerr

Even though it is part of our coding style, there are still many parts, where we use cout and cerr to print out information to the screen. All such occurences should be replaced by usage of FairLogger as is stated in the MPDroot Coding Convention (`https://mpdroot.jinr.ru/mpdroot-naming-convention/`). This will allow to control the level of information provided during runs without the necessity to recompile MpdRoot each time.

9 ClassImp

ClassImp has been deprecated in ROOT and should be removed from all source files.

10 mpdPassive

It is very similar to ExPassive library from FairRoot, there have been some modifications to the source code (probably in FairRoot, rather than in MpdRoot). New files (`FairCradle.(cxx|h)`) have been added but they don't seem to be used anywhere. It is recommended to either rename all files (and contained classes) removing Fair from their names and embedding classes into Mpd namespace or to remove Passive and build FairRoot with examples enabled. Why isn't FairCradle used anywhere? Or is it in some macros which we don't have? In our macros (`geometry_stage1.C`) is CRADLE derived from FairMagnet and not from (our) FairCradle.

Current source files structure

- /mpdroot
  - core
    - mpdBase
    - mpdDst
      - MpdMiniEvent
    - mpdField
    - mpdPassive
    - mpdPid
  - detectors
    - bbc
    - emc
    - etof
    - ffd
    - legacy
    - lusi
    - multi
    - sts
    - tof
    - tpc
    - zdc
  - physics
  - reconstruction
    - tracking
      - kalman
      - lheTrack
  - simulation
    - generators
      - genFactory
      - mpdGen
      - mpdGeneralGenerator
      - unigenFormat
  - tools
    - database
      - dbUtils
      - uniCommon
      - uniDb
    - eventDisplay
    - tdd
      - QA

Proposed files structure

- /mpdroot
  - Core
    - include (.h)
      - Mpd
        - Core
          - Base
          - Dst
          - MiniEvent
          - Field
          - Passive
          - Pid
    - src (.cpp)
      - Base
      - Dst
      - MiniEvent
      - Field
      - Passive
      - Pid
  - Detectors
    - include
      - Mpd
        - Detectors
          - bbc
          - emc
          - etof
          - ffd
          - lusi
          - multi
          - sts
          - tof
          - tpc
            - Alignment
              - Muons
            - ClusterHitFinder
              - Fast
              - HitProducer
              - mlem
            - ...
          - zdc
    - legacy
    - src (.cpp)
      - bbc
      - emc
      - tof
      - ffd
      - lusi
      - multi
      - sts
      - tof
      - tpc...
      - zdc

... structure continued

- Physics
- Reconstruction
  - include (.h)
    - Mpd
      - Reconstruction
        - Tracking
          - Kalman
          - lheTrack
  - src (.cpp)
    - Tracking
      - Kalman
      - lheTrack
- Simulation
  - include (.h)
    - Mpd
      - Simulation
        - Generators
          - GenFactory
          - Gen
          - GeneralGenerator
          - UnigenFormat
  - src (.cpp)
    - Generators
      - GenFactory
      - Gen
      - GeneralGenerator
      - UnigenFormat
- Tools
  - Documentation
  - include (.h)
    - Mpd
      - Tools
        - Database
          - DbUtils
          - UniComon
          - UniDb
        - EventDisplay
        - tdd
          - QA
  - src (.cpp)
    - Database
      - DbUtils
      - UniCommon
      - UniDb
    - EventDisplay
    - tdd
      - QA

Figure 1: Current and proposed directory structure

11 CMake + New Library Versioning

We will switch to target based builds, looking at FairRoot, it is recommended to call the targets `Mpd::${target}`. It is also possible to use the form `MpdRoot::${target}`. So to link to our libraries, we would use, e.g., `set(DEPENDENCIES Mpd::Base)` instead of just Base. This way we know, we are linking Mpd::Base and not FairRoot::Base or ROOT::Base. Switching to this approach would also force us to modify whole build system and allow us to switch to FairRoot 19.0.0. During this rebuild, we could start our own library versioning since in current the mpdroot libraries number is connected only to the FairRoot version without any information on mpdroot development progress (version).

12 MpdGeneratorType

Inside the file `simulation/generators/genFactory/MpdGeneratorType.h` there is an MpdGeneratorType enum class. enum EGenerators located inside `macros/common/runMC.C` is virtually copy of it. It should be removed and replaced by the one from `MpdGeneratorType.h`. Also in `MpdPid.cxx` we have generators, this time identified by TString, it would be preferable to switch to enum class as well to have a single place for generators identification.

13 Remove Dead Detectors

This is recurring topic. Detectors `bbc`, `etof`, `ffd`, and `sts` have not been touched in (at least) 2 years. Question is if they are still legit or can be (re)moved to the legacy folder and removed from build.

14 Copyright Notice

We don't have a unified approach to the copyright notice at the beginning of (all!) files. We should come with a proper license text and add it to all files. Question is what type of license our code should use (LGPL v3 as in FairROOT or maybe MPL/2.0 as in ACTS)? Does JINR have answer to such question? Personally I like the FairRoot and the ACTS license messages the best. After the license text can come line with the author of individual file if we wan't to show programmer's credits (and imply his/hers responsibility).
FairRoot:

```
/********************************************************************************
 *    Copyright (C) 2014 GSI Helmholtzzentrum fuer Schwerionenforschung GmbH    *
 *                                                                               *
 *              This software is distributed under the terms of the              *
 *              GNU Lesser General Public Licence (LGPL) version 3,              *
 *                  copied verbatim in the file "LICENSE"                        *
 ********************************************************************************/
```

ROOT:

```
/*************************************************************************
 * Copyright (C) 1995-2000, Rene Brun and Fons Rademakers.               *
 * All rights reserved.                                                  *
 *                                                                       *
 * For the licensing terms see $ROOTSYS/LICENSE.                         *
 * For the list of contributors see $ROOTSYS/README/CREDITS.             *
 *************************************************************************/
```

ACTS:

```
// This file is part of the ACTS project.
//
// Copyright (C) 2016 CERN for the benefit of the ACTS project
//
// This Source Code Form is subject to the terms of the Mozilla Public
// License, v. 2.0. If a copy of the MPL was not distributed with this
// file, You can obtain one at https://mozilla.org/MPL/2.0/.
```

```
//
// ********************************************************************
// * License and Disclaimer                                           *
// *                                                                  *
// * The  Geant4 software  is  copyright of the Copyright Holders  of *
// * the Geant4 Collaboration.  It is provided  under  the terms  and *
// * conditions of the Geant4 Software License,  included in the file *
// * LICENSE and available at  http://cern.ch/geant4/license .  These *
// * include a list of copyright holders.                             *
// *                                                                  *
// * Neither the authors of this software system, nor their employing *
// * institutes,nor the agencies providing financial support for this *
// * work  make  any representation or  warranty, express or implied, *
// * regarding  this  software system or assume any liability for its *
// * use.  Please see the license in the file  LICENSE  and URL above *
// * for the full disclaimer and the limitation of liability.         *
// *                                                                  *
// * This  code  implementation is the result of  the  scientific and *
// * technical work of the GEANT4 collaboration.                      *
// * By using,  copying,  modifying or  distributing the software (or *
// * any work based  on the software)  you  agree  to acknowledge its *
// * use  in  resulting  scientific  publications,  and indicate your *
// * acceptance of all terms of the Geant4 Software license.          *
// ********************************************************************
//
```