# Общая структура програмного обеспечения детектора Супер Ц-Тау фабрики

Д. Максимов

2 июля 2018

# Содержание

# Структура и состав програмного обеспечения

# Структура и состав програмного обеспечения

- Всё ПО собрано в набор проектов
- Проект состоит из набора пакетов, для сборки своих пакетов могут использоваться пакеты из другого проекта (базового)
- Для сборки используется cmake 3.11 и gcc 7.3
- Имеются следующие проекты
  - HEPCommon
  - SCTauSimExternals
  - GAUDI
  - SCTauSim
  - ...

  + проект WorkDir

# HEPCommon: список пакетов

- AIDA — 3.2.1
- Boost — 1.66
- CLHEP — 2.4.0.4
- CppUnit — 1.12.1
- Eigen — 3.3.4
- FastJet — 3.3.1
- HepPDT — 3.04.01
- Rangev3 — 0.3.5
- ROOT — 6.13.02
- TBB — 2018_U3
- cppgsl — b07383ea

# SCTauSimExternals: список пакетов

- EvtGen — R01-07-00 + патч для поддержки gcc7
- Geant4 — 10.04.p01
- HepMC3 — 3.0.0
- Photos — 3.61
- Pythia8 — 235
- Tauola — 1.1.6c
- FCC/dag — v0.1
- FCC/podio — v0.8
- FCC/fcc-edm — v0.5.2

# SCTauSim: список пакетов

- Control — ядро фреймворка
  - AuroraAlg
  - AuroraBase
  - AuroraCommon
  - AuroraKernel — запланирован
  - AuroraServices
- External — интерфейсные пакеты для подключения внешних библиотек
- Event — манипуляции данными события
  - Conversion — EDM $\leftrightarrow$ HepMC
  - PodioIO — чтение и запись данных в формате Podio
- Generation — генерация первичных событий (подробнее далее)
- Tools — общие вспомогательные инструменты
  - PathResolver — ищет файлы по путям поиска
- Simulation — запланировано в следующем релизе
- DetectorDescription — запланировано в следующем релизе

# SCTauSim/Generation: список пакетов (1)

- GenAlgs — алгоритмы (в терминах Gaudi) генерации событий и связанные с ними
  - GenAlg — общий алгоритм генерации, посредством вызовов IHepMCProviderTool
  - GenMerge — генерация путём слияния данных из нескольких источников (EDM)
  - HepMCFileWriter — запись данных HepMC в текстовый файл
- Generators — генераторы событий в формате HepMC (реализация IHepMCProviderTool)
  - EvtGen_i — вызов EvtGen
  - EvtGenModels — дополнительные модели к EvtGen, могут быть подключены через EvtGen_i
  - HepMC_In —чтение событий из текстового файла
  - ParticleGun — генерация тестовых частиц
  - Pythia8_i — прямой вызов Pythia8

# SCTauSim/Generation: список пакетов (2)

- GenExamples — набор примеров
- GenInterfaces — интерфейсы
  - IEvtGenModel
  - IHepMCFileReaderTool
  - IHepMCMergeTool
  - IHepMCProviderTool
  - IParticleGunTool
  - IVertexSmearingTool
  - Units — определение единиц измерения в моделировании
- GenTools
  - EvtGenExternal — связка EvtGen с внешними моделями (Photos, Pythia8, Tauola)
  - HepMCInterfaces — связка HepMC с Photos, Pythia8 и Tauola
  - HepMCMerge — слияние событий в формате HepMC
  - PileUp — генераторы числа PileUp событий, сейчас только ConstPileUp
  - Vertex — манипуляции с вершинами, сейчас только равномерная размазка

# Рабочий процесс

# Репозиторий

Центральный репозиторий https://git.inp.nsk.su/sctau/aurora

Перед началом работы нужно сделать свою копию

# Настройка среды

Среда для работы подготовлена на машинах `stark` и `proxima`.

```
# Настройка самой базовой среды,
# данную команду необходимо выполнять каждый раз при входе
$ setupSCTAU
```

# Подготовка рабочей директории

```
# создадим рабочую директорию
$ mkdir workarea
$ cd workarea

# Директории для сборки и запуска
$ mkdir build run

# Подготовка рабочей директории (делается один раз)
$ git sctau init-workdir \
>     ssh://git@git.inp.nsk.su/sctau/aurora.git
$ cd aurora
```

# Подготовка рабочей директории

```
# Получение обновлений с головного репозитория
# нужно делать периодически при длительном
# существовании рабочей директории и
# существенных изменениях в головном
$ git fetch upstream

# Подготовка рабочей тематической ветки,
# название стоит выбирать осмысленным
$ git checkout -b MyDevelopmentBranch upstream/0.1 --no-track

# Добавим пакеты из репозитория
$ git sctau addpkg GenExamples

# Выберем релиз и его версию, в которой будем работать
$ asetup SCTauSim,0.1.0
```

# Сборка и запуск

```
# Сборка
$ cd ../build/
$ cmake ../aurora/Projects/WorkDir
$ make

# Настройка локального окружения
$ source x86_64-slc7-gcc7-opt/setup.sh

# Запуск
$ cd ../run
$ ctaurun.py GenExamples/evtgen.py
```

# Сохранение изменений и их отправка на сервер

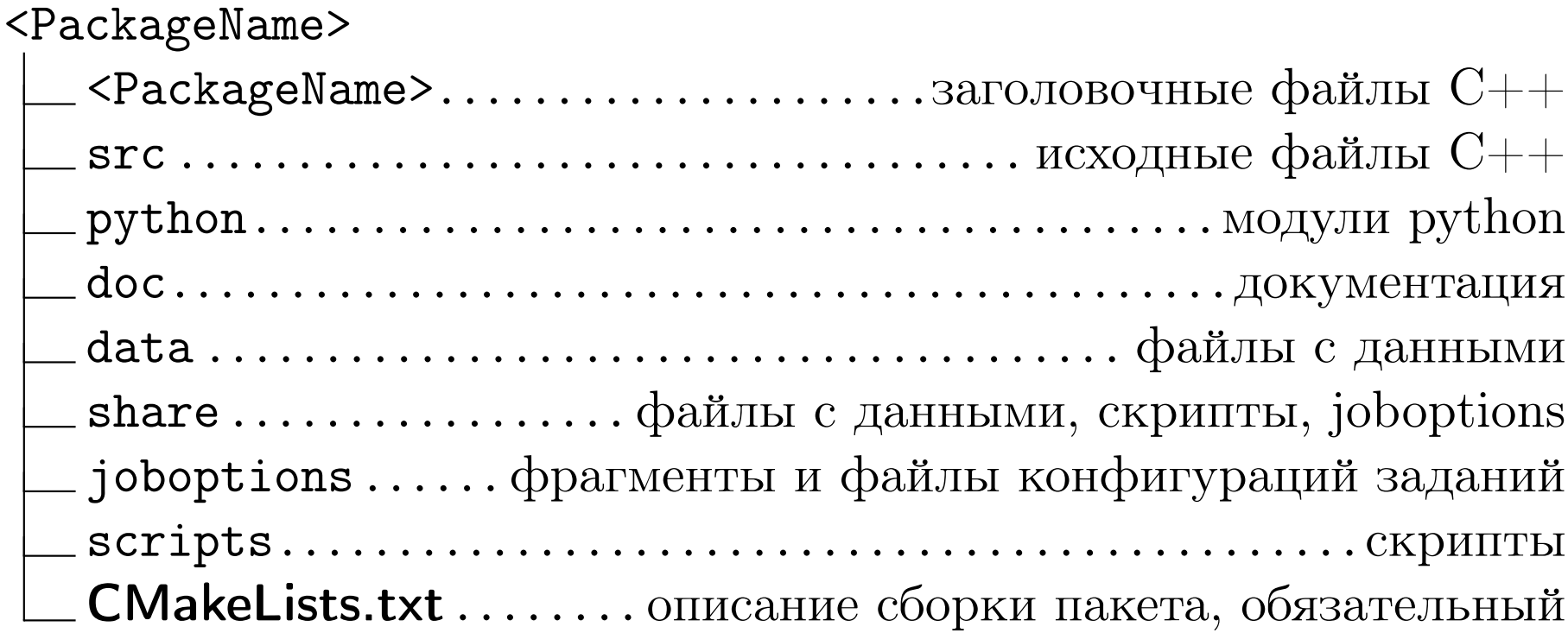
```
$ git add ...

$ git commit -m '...'

$ git push
```

По достижении результата создаётся Merge request из интерфейса GitLab

# Устройство пакета



# Пример CMakeLists.txt (1)

```
################################################################
# Package: EvtGen_i
################################################################
# Declare the package name:
sctau_subdir(EvtGen_i)

# Declare the package's dependencies:
sctau_depends_on_subdirs( PUBLIC
                          Control/AuroraCommon
                          Control/AuroraAlg
                          External/HepMC
                          External/EvtGen
                          Generation/GenInterfaces
                          PRIVATE
                          Generation/GenTools/EvtGenExternal
                          Tools/PathResolver )
```

# Пример CMakeLists.txt (2)

```
# Component(s) in the package:
sctau_add_component( EvtGen_i
                     src/*.cpp
                     PUBLIC_HEADERS EvtGen_i
                     INCLUDE_DIRS  AuroraAlg HepMC EvtGen
                     PRIVATE_LINK_LIBRARIES AuroraAlg
                       EvtGenExternal EvtGen HepMC
                       PathResolver )

sctau_install_data( share/* )
```

# Доступные cmake функции (1)

- sctau_subdir — имя пакета
- sctau_depends_on_subdirs — зависимости пакета
- sctau_add_library — декларация собираемой библиотеки в пакете
- sctau_add_component — как библиотека, плюс ещё генерируется конфигурация для joboptions
  не может линковаться с другими библиотеками и компонентами
- sctau_add_executable — декларация исполняемого файла
- sctau_add_alias — определение псевдонима для исполняемого файла или скрипта (для удобства)
- sctau_add_test — определение unit-тестов в пакете
- sctau_add_dictionary — создание ROOT (Reflex) словаря в виде отдельной библиотеки
- sctau_add_root_dictionary — создание ROOT (CINT) словаря в виде исходного текста для включения в библиотеку

# Доступные cmake функции (2)

- sctau_install_headers — установка заголовочных файлов, полезно для чисто интерфейсных пакетов
- sctau_install_python_modules — установка python модулей
- sctau_install_data — установка файлов с данными
- sctau_install_joboptions — установка установка конфигураций задания
- sctau_install_docs — установка документации
- sctau_install_xmls — установка XML файлов
- sctau_install_scripts — установка исполняемых скриптов
- sctau_install_runtime — установка файлов требуемых во время выполнения, но не попадающих ни в одну из категорий выше

# Дальнейшее развитие

# План релизов

- 0.1 — Настройка и отладка общей инфраструктуры. Начало генерации первичных событий
- 0.2 — Интеграция DD4HEP, Papas, Geant4, доработка модели данных события
- 0.3 — ...

# Вопросы для обсуждения

- Организация процедуры обработки Merge request'ов
- Реализация автоматического тестирования, отдельных пакетов и релизов в целом
- Доработка модели событий, ревизия fcc-edm, уход от пространства имён fcc
- Физическая валидация кода моделирования и её техническая реализация
- Система пакетных заданий
- Стандарты кодирования
  - Исключить абсолютные пути из исходных текстов
- SSO для ScTau Wiki