Getting started with core development

Paul Gessinger
CERN
2022-09-28

Dependencies

Dependencies

- (Prerequisites not explained: clone the repository somehwere (say `$SOURCE`), do `git submodule init` and `git submodule update`)
  - You need GIT LFS for the OpenDataDetector repository
- Minimal dependencies: Boost & Eigen for core
- Developing in the Examples environments requires additional dependencies
  - ROOT, tbb, dd4hep, Geant4, edm4hep, HepMC3, Pythia8 if all enabled
- Local development:
  - Build dependencies yourself
  - Use docker with an image used by our CI
  - Use LCG + custom updated DD4hep version
  - Use key4hep stack install
  - Use pre-built dependencies (for macOS)

Dependencies: Python + ROOT + DD4hep

Attention

ROOT + DD4hep **strictly requires** the Python version to be **exactly** identical to the one they were built with! If you get errors with a mismatching Python version, you will have to acquire this exact version!

Docker

```
$ docker pull ghcr.io/acts-project/ubuntu2004
$ docker run --rm -it -v/src:/src -w/src ghcr.io/acts-project/ubuntu2004 bash
$ # dependencies available in docker
```

CentOS7 with CVMFS e.g. lxplus

```
$ source /cvmfs/sw.hsf.org/key4hep/setup.sh
```

[1] You don't need docker in this scenario, just CentOS7 + CVMFS

macOS: x86

- Prerequisites (requires homebrew)

```
$ brew install cmake eigen boost ninja
```

- Can build with dependency tarball from CI:
  https://acts.web.cern.ch/ci/macOS/deps.1e4136f.tar.gz
- Needs to be extracted to `/usr/local/acts`

```
$ mkdir -p /usr/local/acts
$ tar -xf macOS_arm_deps.tar.gz -C /usr/local/acts
```

- Add to CMake command:

```
$ -DCMAKE_PREFIX_PATH="/usr/local/acts"
```

macOS: ARM

- Prerequisites (requires homebrew)

```
$ brew install cmake eigen boost ninja
```

  - Currently have no CI and no central tarball
  - I created a tarball from my local dependencies that **should** work: https://pagessin.web.cern.ch/pagessin/macOS_arm_deps.tar.gz
  - Needs to be extracted to `/opt/hep`
  - In this case: need ROOT from homebrew as well: `brew install root`

```
$ mkdir -p /opt/hep
$ tar -xf macOS_arm_deps.tar.gz -C /opt/hep
```

- Add to CMake command:

```
$ -DCMAKE_PREFIX_PATH="/opt/hep/dd4hep/v01-21;"\
                      "/opt/hep/geant4/11.0.0/;"\
                      "/opt/hep/hepmc3/3.2.1;"\
                      "/opt/hep/pythia8/307"
```

CMake step and unit tests

```
$ cd $SOURCE
$ cmake -S . -B build -GNinja \
-DACTS_BUILD_PLUGIN_DD4HEP=ON \
-DACTS_BUILD_EXAMPLES=ON \
-DACTS_BUILD_EXAMPLES_PYTHON_BINDINGS=ON \
-DACTS_BUILD_EXAMPLES_DD4HEP=ON \
-DACTS_BUILD_UNITTESTS=ON \
-DACTS_BUILD_ODD=ON
$ cmake --build build
$ # run the unit tests for basic checks:
$ cd build
$ ctest -j$(nproc)
```

```
# more tests...
197/210 Test #206: TGeoTrd2Conversion ..........   Passed
198/210 Test #133: BoundingBox .................   Passed
199/210 Test #207: TGeoTubeConversion ..........   Passed
200/210 Test #209: TGeoParser ..................   Passed
201/210 Test #208: TGeoLayerBuilder ............   Passed
202/210 Test #115: SurfaceLocalToGlobalRoundtrip   Passed
203/210 Test  #13: TransformBoundToFree ........   Passed
204/210 Test #141: KDTree ......................   Passed
205/210 Test  #14: TransformFreeToBound ........   Passed
206/210 Test #158: ImpactPointEstimator ........   Passed
207/210 Test   #5: BoundTrackParameters ........   Passed
208/210 Test #152: Subspace ....................   Passed
209/210 Test   #8: FreeTrackParameters .........   Passed
210/210 Test   #7: CurvilinearTrackParameters ...   Passed

100% tests passed, 0 tests failed out of 210

Total Test time (real) =   9.70 sec
```

Python level tests

- This is possible on all platforms
- Currently not as straightforward on macOS due to complex geometry lookup path handling
- Will focus on the lxplus path from above to give you a rough idea

```
$ cd $SOURCE
$ export LD_LIBRARY_PATH=$SOURCE/build/thirdparty/OpenDataDetector/factory:$LD_LIBRARY_PATH
$ python -m venv actsvenv
$ source actsvenv/bin/activate
$ pip3 install -r Examples/Python/tests/requirements.txt
$ source $SOURCE/build/python/setup.sh
$ export ROOT_HASH_CHECKS=on
$ export ROOT_HASH_FILE=$SOURCE/hashes.txt
$ pytest
```

Python tests

```
============================= test session starts ==============================
platform linux -- Python 3.9.0, pytest-7.0.1, pluggy-1.0.0
rootdir: /scratch/pagessin/dev/acts, configfile: pytest.ini, testpaths: Examples/Python/tests
plugins: pytest_check-1.0.4, xdist-2.5.0, forked-1.4.0, rerunfailures-10.2
collected 225 items

Examples/Python/tests/test_algorithms.py .......................ss.          [ 12%]
Examples/Python/tests/test_base.py ..........                                [ 16%]
Examples/Python/tests/test_detectors.py .s...                                [ 18%]
Examples/Python/tests/test_examples.py sssssssssssssssssssssssssssssssssssss [ 35%]
Examples/Python/tests/test_magnetic_field.py .....                           [ 37%]
Examples/Python/tests/test_material.py ....                                  [ 39%]
Examples/Python/tests/test_propagation.py ......                             [ 42%]
Examples/Python/tests/test_reader.py ............s.........sssssss           [ 55%]
Examples/Python/tests/test_writer.py ......FFFFFFFFFFFFF..................... [ 73%]
.............................................sss.sssssssss                    [100%]

=================================== FAILURES ===================================
```

Python tests

```
---------------------------- RootHashAssertionErrors -----------------------------
The ROOT files produced by tests have changed since the last recorded reference.
This can be be expected if e.g. the underlying algorithm changed, or it can be a test failure symptom.
Please manually check the output files listed below and make sure that their content is correct.
If it is, you can update the test reference file Examples/Python/tests/root_file_hashes.txt with the n
See https://acts.readthedocs.io/en/latest/examples/python_bindings.html#root-file-hash-regression-chec

test_root_prop_step_writer[configPosConstructor]__prop_steps.root: 6ad8738725ca41d1751efd30f13fc1b45df
test_root_prop_step_writer[configKwConstructor]__prop_steps.root: 6ad8738725ca41d1751efd30f13fc1b45df7
test_root_prop_step_writer[kwargsConstructor]__prop_steps.root: 6ad8738725ca41d1751efd30f13fc1b45df77a
test_root_particle_writer[configPosConstructor]__particles.root: 7d2c8cce6f491c22ce149b526866bdfa8795c
test_root_particle_writer[configKwConstructor]__particles.root: 7d2c8cce6f491c22ce149b526866bdfa8795cf
test_root_particle_writer[kwargsConstructor]__particles.root: 7d2c8cce6f491c22ce149b526866bdfa8795cfac
test_root_meas_writer__meas.root: 5c7a9c196b92937ddaebf34646a5ffa12d323116883069053dc6fe1ae6de4d961
test_root_simhits_writer[configPosConstructor]__meas.root: 0843ba98210e7c64092592953e2987c836e310409f1
test_root_simhits_writer[configKwConstructor]__meas.root: 0843ba98210e7c64092592953e2987c836e310409f11
test_root_simhits_writer[kwargsConstructor]__meas.root: 0843ba98210e7c64092592953e2987c836e310409f116e
test_root_clusters_writer[configPosConstructor]__clusters.root: 7e452af7243d282dd0a8f5aa2844e150ef4436
test_root_clusters_writer[configKwConstructor]__clusters.root: 7e452af7243d282dd0a8f5aa2844e150ef44364
test_root_clusters_writer[kwargsConstructor]__clusters.root: 7e452af7243d282dd0a8f5aa2844e150ef4436498
============================= short test summary info ==============================
FAILED Examples/Python/tests/test_writer.py::test_root_prop_step_writer[configPosConstructor]
```

Python tests

- You can also select individual tests like

```
$ pytest -k ckf
```

- Removing skipped tests:

```
$ source /path/to/root/thisroot.sh
$ source /path/to/g4/geant4.sh
$ source /path/to/dd4hep/thisdd4hep.sh
```

- updated hashes (and adding all dependencies)

Python tests

```
================================== test session starts ==================================
platform linux -- Python 3.9.0, pytest-7.0.1, pluggy-1.0.0
rootdir: /scratch/pagessin/dev/acts, configfile: pytest.ini, testpaths: Examples/Python/tests
plugins: pytest_check-1.0.4, xdist-2.5.0, forked-1.4.0, rerunfailures-10.2
.......................................................................... [ 36%]
.......................................................................... [ 74%]
................................................                           [100%]
============================== 225 passed in 71.22s (0:01:11) ==============================
```

Conclusion

- These steps should put you in a good position to develop in the examples context
- Unit tests and python level tests should help you developing without worrying about regressions
- Adding example components to the python bindings makes them testable in the examples infrastructure
- If you run into issues: ping us on our general mattermost channel

Questions?