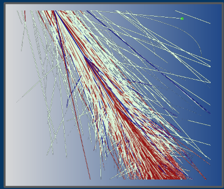
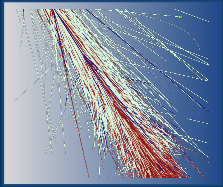


CORSIKA Output Format

Ralf Ulrich, 24.9.2020

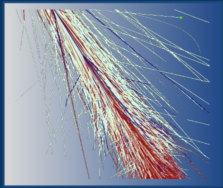
Note



playground branch: **output_format_testing**

- Added: **ASCII, ROOT, parquet, HDF5** output writers based on **ObservationPlane**
- removed all std::cout and std::cerr
- extended **vertical_EAS** to generate multiple showers

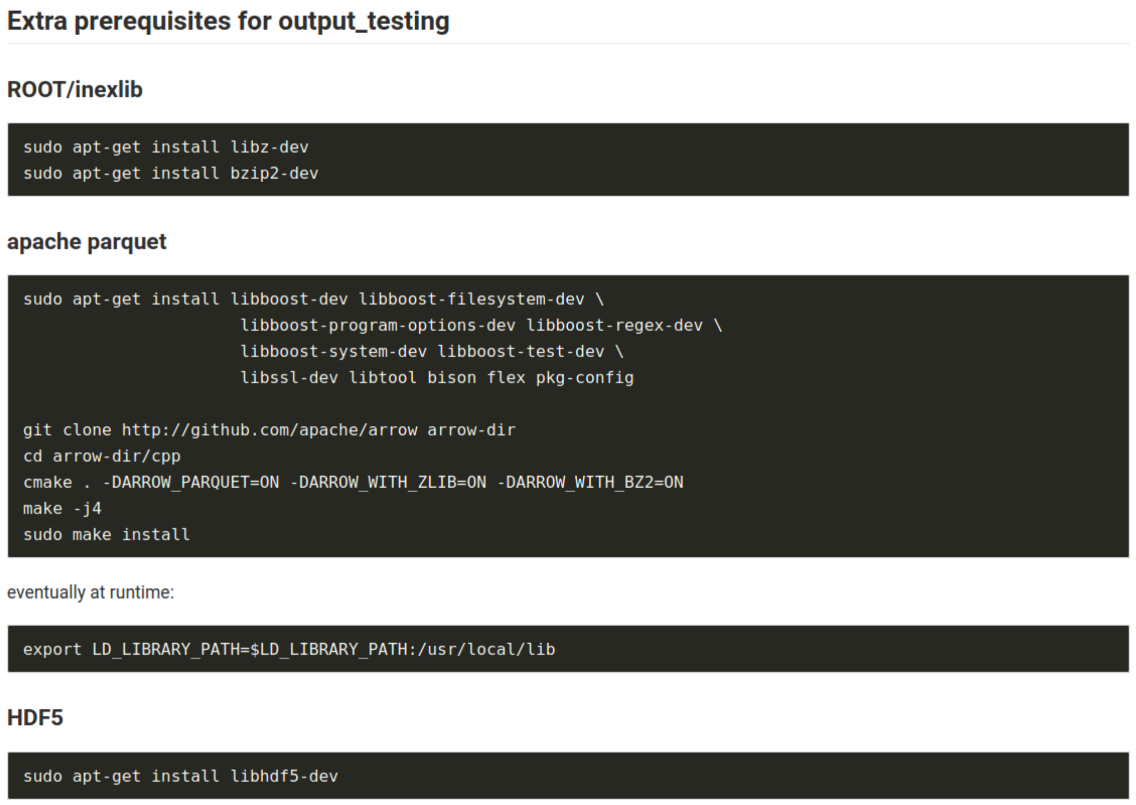
More prerequisites



README.md

This will work on a typical Ubuntu system.

Feedback on other systems Welcome.



Extra prerequisites for output_testing

ROOT/inexlib

```
sudo apt-get install libz-dev
sudo apt-get install bzip2-dev
```

apache parquet

```
sudo apt-get install libboost-dev libboost-filesystem-dev \
                     libboost-program-options-dev libboost-regex-dev \
                     libboost-system-dev libboost-test-dev \
                     libssl-dev libtool bison flex pkg-config

git clone http://github.com/apache/arrow arrow-dir
cd arrow-dir/cpp
cmake . -DARROW_PARQUET=ON -DARROW_WITH_ZLIB=ON -DARROW_WITH_BZ2=ON
make -j4
sudo make install
```

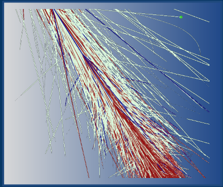
eventually at runtime:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib
```

HDF5

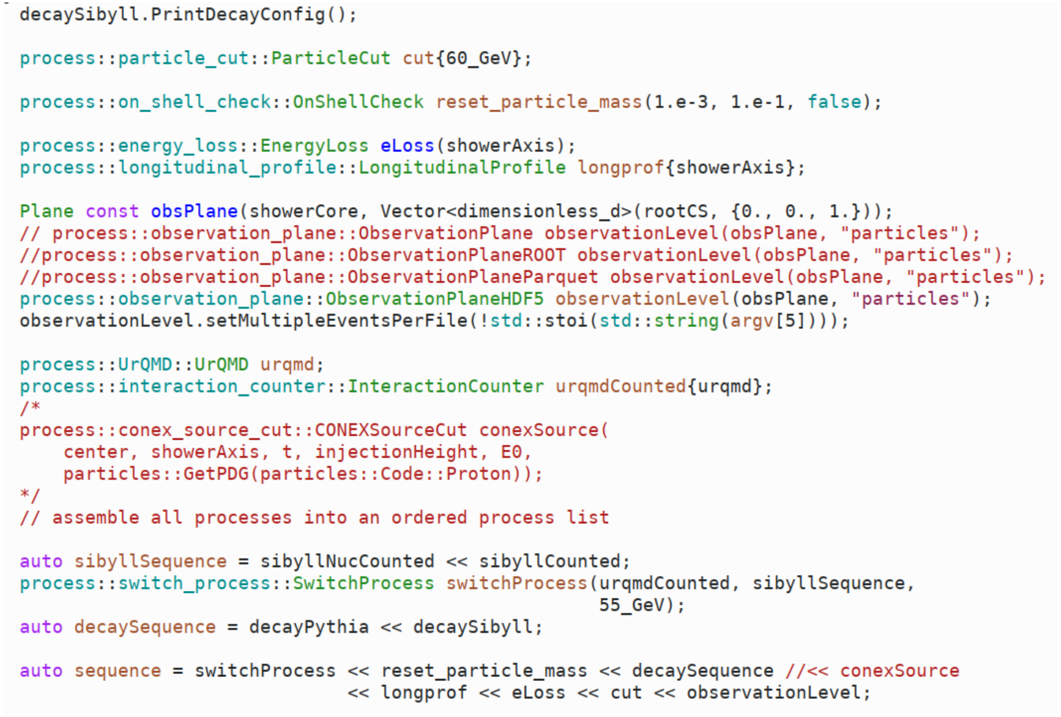
```
sudo apt-get install libhdf5-dev
```

How to change output



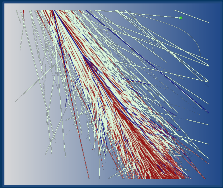
Edit:

vertical_EAS.cc



```
decaySibyll.PrintDecayConfig();

process::particle_cut::ParticleCut cut{60_GeV};

process::on_shell_check::OnShellCheck reset_particle_mass(1.e-3, 1.e-1, false);

process::energy_loss::EnergyLoss eLoss(showerAxis);
process::longitudinal_profile::LongitudinalProfile longprof{showerAxis};

Plane const obsPlane(showerCore, Vector<dimensionless_d>(rootCS, {0., 0., 1.}));
// process::observation_plane::ObservationPlane observationLevel(obsPlane, "particles");
//process::observation_plane::ObservationPlaneROOT observationLevel(obsPlane, "particles");
//process::observation_plane::ObservationPlaneParquet observationLevel(obsPlane, "particles");
process::observation_plane::ObservationPlaneHDF5 observationLevel(obsPlane, "particles");
observationLevel.setMultipleEventsPerFile(!std::stoi(std::string(argv[5])));

process::UrQMD::UrQMD urqmd;
process::interaction_counter::InteractionCounter urqmdCounted{urqmd};
/*
process::conex_source_cut::CONEXSourceCut conexSource(
    center, showerAxis, t, injectionHeight, E0,
    particles::GetPDG(particles::Code::Proton));
*/
// assemble all processes into an ordered process list

auto sibyllSequence = sibyllNucCounted << sibyllCounted;
process::switch_process::SwitchProcess switchProcess(urqmdCounted, sibyllSequence,
                                                     55_GeV);
auto decaySequence = decayPythia << decaySibyll;

auto sequence = switchProcess << reset_particle_mass << decaySequence //<< conexSource
                              << longprof << eLoss << cut << observationLevel;
```

Needed work



Task list:

- Look at code and implementation in **Processes/ObservationPlane**
- Add further interesting output options
- Make some real simulations for comparison
  - High energy, large number of particles, maybe Fe at 1e18eV?
  - Low energy, large number of showers, maybe 1e6 x p at 1e13eV
- Profiling
- Timing; **WIP Adde benchmark processor and dummy processors** !253
- Analysis code example: python? With performance study!