Installation and Compilation of the Software

March 23, 2023

1 Container

The purpose of using containers in this context is to have full control over the dependant software for Mu3e. In the HEP environment there was some issues with the event display as it needed a package called Cairo for some plotting function. If you are working on your own laptop that runs a linux based operating system then this is largely irrelevant. There are some fundamental issues with Centos and how it runs so for the container that was run here, opensuse was used. There is a definition file that can be used to do all of these actions at once. This appeared not to work in my environment but is always worth a try, as it means the software and it's relevant dependencies are installed in one go with no manual involvement. I will leave a link to the file on the wiki. It should be noted in the following recipe, some of the double dashes that indicate an option in the command line format to a single long dash, I will attempt to fix this.

Find below the commands required to set up a container that runs the software:

- singularity build –sandbox mu3e docker://opensuse/leap
  - This pulls an image from the docker hub and constructs a sandbox, from which a shell can be run. This shell is isolated from the rest of the environment which is why dependant software can be installed without sudo commands safely.
- singularity shell –fakeroot –writable mu3e
  - This opens a shell based on the newly built directory, this runs opensuse and this is where we work.
- cd /
  - This takes us to the directory.
- zypper –non-interactive update
  - This just updates any pre-installed libraries.
- zypper –non-interactive install libboost_filesystem1_66_0-devel libboost_program_options1_66_0-devel libboost_system1_66_0-devel libexpat-devel cfitsio-devel cmake eigen3-devel gcc9-c++ gcc-fortran git git-svn gsl-devel gzip libxerces-c-devel libXmu-devel libXpm-devel libXft-devel libxml2-devel libopenssl-devel libpng12-0 libfftw3-3 libgsl23 libX11-devel libXext-devel openjpeg-devel patch tar tbb-devel uuid-devel vim wget gtkmm3-devel fmt-devel Packagekit-gtk3-module libcanberra-gtk3-module pango-devel libpangomm-1_4-1 fontconfig
  - This installs all of the minor dependant software required.
- zypper –non-interactive si boost
- zypper clean
- wget https://www.python.org/ftp/python/3.11.2/Python-3.11.2.tgz
  - We now install python.

- tar -zxvf Python-3.11.2.tgz
  - Unzip
- cd Python-3.11.2
- ./configure
- make
- make altinstall
- cd ..
- rm Python-3.11.2.tgz
- wget https://root.cern/download/root_v6.28.00.source.tar.gz
  - Now Root
- tar -zxvf root_v6.28.00.source.tar.gz
- rm root_v6.28.00.source.tar.gz
- mkdir build-root
- mkdir install-root
- cd build-root
- export CC=/usr/bin/gcc-9
  - This path might be different for other people, but the initial forward slash should just take you to the opensuse directory, not your home directory. If it does then the fix is to add the path to the opensuse directory to the front of it.
- export CXX=/usr/bin/g++-9
  - Again if the issue arrises, do the same thing.
- cmake -DCMAKE_INSTALL_PREFIX=../install-root \
- -DCMAKE_CXX_STANDARD=17 -DCMAKE_BUILD_TYPE=Release \
- -DPython3_ROOT_DIR=/Python-3.11.2 -DPYTHON_EXECUTABLE=/usr/bin/python3 \
- -DLLVM_CXX_STD=c++17 -Dxrootd=OFF \
- ../root-6.28.00
  - The options here give paths to various dependencies, the options are not double dashed, the backslash keeps all of the cmake options on the same command.
- make -j$(nproc)
- make install
- cd /
- rm -rf build root/*
- rm -r build-root
- rm -rf root-6.28.00/*
- rm -r root-6.28.00
- zypper clean

- git clone https://gitlab.cern.ch/geant4/geant4.git
- mkdir geant4-build
- cd geant4-build
- cmake -DCMAKE_INSTALL_PREFIX=/geant4-install -DGEANT4_INSTALL_DATA=ON \
- -DGEANT4_USE_QT=OFF \
- -DGEANT4_USE_OPENGL_X11=OFF \
- ../geant4
  - The options here give paths to various dependencies, the options are not double dashed, the backslash keeps all of the cmake options on the same command.
- make -j$(nproc)
- make install
- rm -rf /geant-build/
- rm -rf /geant4/
- zypper clean
- git clone git@bitbucket.org:mu3e/mu3e.git
  - This clones the mu3e software into the directory, this is for the dev branch I think. Just change this to whatever branch you want to clone.
- cd mu3e
- git submodule update –init –recursive
- mkdir build
- mkdir installation
- cd build
- cmake -DCMAKE_C_COMPILER=/usr/bin/gcc-9 -DCMAKE_CXX_STANDARD=17 \
- -DCMAKE_CXX_COMPILER=/usr/bin/g++-9 \
- -DCMAKE_INSTALL_PREFIX=../installation -DMU3E_TRIREC_DISPLAY=ON ..
  - Remember the two dots at the end otherwise it will compile wrong.

  make -j$(nproc) make install

That completes the install, the final step is to use the shell script provided to set up the terminal with the mu3e environment. I have provided the shell script below, place this in the run directory of the mu3e environment found at /mu3e/run. Everything should now be set up, any problems, let me know.