Steps to Install LUMA in Ubuntu 20.04

Follow these steps to install and configure LUMA in your Ubuntu System. This document will guide you seamlessly through the whole installation process of LUMA and its dependencies. You need to install and configure the dependencies first to ensure the proper functioning of LUMA.

Dependencies

- **LAPACK and BLAS**

  For LAPACK and BLAS we need to have g++ and gfortan compilers installed. If g++ and gfortran compilers are not installed in your system, install them using the following commands:

  > g++

  ```
  sudo apt install build-essential
  ```

  > gfortran

  ```
  sudo apt-get install gfortran
  ```

  Once these compilers are installed, LAPACK and BLAS can be installed.

  > BLAS

- Download the latest version of **BLAS**
- Open a terminal and go to the directory where you have it saved and type-
- `tar -xvf blas-{version}.tgz # unzip the blas source files`
- `cd BLAS-{version}/`
- `make`
- `mv blas_LINUX.a libblas.a`
- `mv *.a /usr/local/lib # move the blas lib to the library you will be including at compile`

  > LAPACK

- Download the latest version of **LAPACK**
- Open a terminal and go to the directory where you have it saved and type-
- **tar -xvf lapack-{version}.tar.gz** # unzip the blas source files
- **cd lapack-{version}/**
- **cp make.inc.example make.inc** # use example make as make
- **make**
- **cp *.a /usr/local/lib**

Now that the libraries have been built, and are stored in path/to/lib, the short example code given below can be compiled and tested. If correctly installed there should not be any error.

Code: filename: test.cpp

```
#include <iostream>
#include <vector>
using namespace std;

extern "C" void dgesv_( int *n, int *nrhs, double  *a, int *lda, int *ipiv,
double *b, int *lbd, int *info  );

int main() {
    int SIZE = 3;
    int nrhs = 1; // one column in b
    int lda = SIZE;
    int ldb = SIZE;
    std::vector<int> i_piv(SIZE, 0);  // pivot column vector
    int info;
    std::vector<double> A(SIZE*SIZE, 0); // sq mat with 0's
    A = {5, 2, 8, 9, 7, 2, 10, 3, 4};
    std::vector<double> b(SIZE);
    b = {22, 13, 17};

    dgesv_( &SIZE, &nrhs, &*A.begin(), &lda, &*i_piv.begin(), &*b.begin(), &ldb,
&info );
    cout << "I think it is working!!" << endl;
    return 0;
}
```

- **g++ test.cpp -L/usr/local/lib -llapack -lblas -lgfortran** # compiles the code
- **./a.out** # runs the code

- **VTK**

First we need to install **cmake** and **ccmake.**

> cmake

```
sudo apt-get install cmake
```

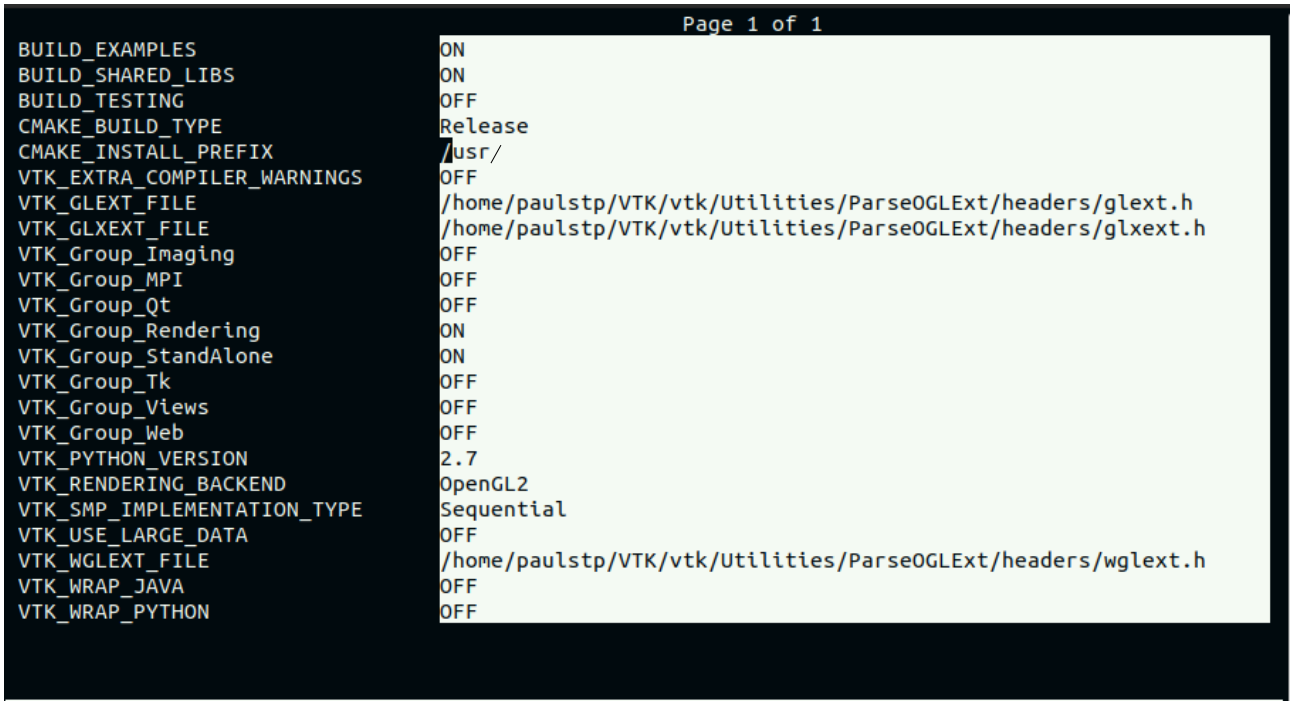
> ccmake

```
sudo apt-get install cmake-curses-gui
```

Create a directory in the Home with name "VTK". Open terminal in that directory and type:

- `git clone https://gitlab.kitware.com/vtk/vtk.git`
- `cd vtk`
- `git checkout v8.2.0 # use this version as it is stable`
- `mkdir build`
- `cd build`
- `ccmake ..`

use the following confiuration, then save and generate:



- `make`
- `make install # if raises permission issue use 'sudo'`
- `ldconfig # if raises permission issue use 'sudo'`

To verify if VTK is build and installed properly follow this thread: Test VTK

HDF5

First we need to install **openmi**

> openmpi

```
sudo apt-get install -y openmpi-bin
```

> hdf5

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

After this open **.profile** from "home" and add the following path:

```
export HDF5_HOME=/usr/lib/x86_64-linux-gnu/hdf5/openmpi
```

The dependencies are now installed. We can go ahead and configure LUMA.

LUMA

Go to the directory where you want to install LUMA and open terminal and type:

- `git clone https://github.com/cfdemons/LUMA.git`

Luma source is now installed in in this directory.

Installing a Test Case

LUMA relies on a number of input files to define a particular case. As parameters are a mixture of compile-time and run-time variables, it is advisable to incrementally recompile parts of LUMA when setting up a new case.

The input files are:

> definitions.h

This file specifies the compile-time parameters required for LUMA to run a particular case. It must be copied to the **`inc`** directory, overwriting any previous version. LUMA must be recompiled after changing this file.

> geometry.config

This file specifies the configuration of any geometrical objects to be inserted into the flow simulation. It is read by LUMA at initialisation if LUMA was built with the **`L_GEOMETRY_FILE`** flag enabled in the **`definitions.h`** file. This file can be changed and LUMA re-run without the need to recompile. It must be copied into the **`input`** directory.

To run a case:

Once you have put the required files in the specied folders, open terminal in the LUMA source directory and type:

- `make`
- `mpirun -np NPROCS ./LUMA # NPROCS = number of processors in which you want to run LUMA; for more than 1 processor uncomment #define L_BUILD_FOR_MPI in definitions.h file`
- `./LUMA # can also be used instead if only 1 processor is used`

> Output file:
The output id written in hdf5 file in .h5 format inside a folder created in the main LUMA source directory. This .h5 file can be easily viewed by a tool named **Panoply.**

Panoply requires Java Development Kit (JDK); make sure you have that installed in your system before installing panoply.

To install panoply and run:

- Open https://www.giss.nasa.gov/tools/panoply/download/
- Download and extract the zipped file
- `cd PanoplyJ-{version}`
- `cd PanoplyJ`
- `./panoply.sh # if bash: permission denied follow below commands`

- `chmod u+r+x panoply.sh`
- `./panoply.sh`
- `sudo apt install default-jre # if java is not installed, install using this command`

Once it is opened, we can load .h5 files directly and get the plots.

Merge Tool

The **h5mgm** merge tool is a post-processor that takes the multi-grid HDF output from LUMA and converts this data into a time-series of VTU mesh files readable into Paraview. A guide to building the merge tool is given beow:

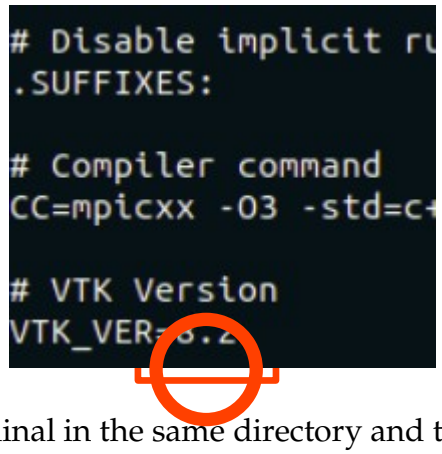
Open terminal from the LUMA source and type:

- **cd tools/post_processors/h5mgm**
- Open the merge tool makefile using **vi** by typing
- **vi makefile**

Change the version of VTK in the makefile with the version buid and installed in your system. Use vi editor commands.

Go to **/usr/include** search for **vtk-{version}** folder, and use the version exactly as it is in the folder name.

Refer to the image of the folder and the makefile:



Once the make file is edited, save it and open terminal in the same directory and type:

- **make**

If no error is shown, the merge tool is made properly. Once the **h5mgm** executable is made in **h5mgm** folder, copy that file to the **output_{date}_{time}** folder created which contains the **.h5** data file. Open terminal there and type:

- **./h5mgm**

The output files are organised into a directory called **postprocessedoutput.** In this **postprocessedoutput** folder, a series of **.vtu** files are created. These files can be viewed through **paraview**. For that make sure you have **paraview** installed in your system.

###

The merge tool must be called from **within the directory containing the *.h5 file to be merged**. Parameters are passed as space separated command line argument as follows:

- **h5mgm <arg1> <arg2> ...**

Valid options for the argument are:

cut	Excludes solid sites from the reconstructed mesh.
legacy	Writes VTKs instead of VTUs.
loud	Prints out information to the screen as well as to the log file.
quiet	Will not write the log file.
sorter	Writes out cell velocities in a z, y, x, sorted list based on the HDF5 files.
version	Prints the version number of the tool compiled on your system.
XXX	Appends XXX to the output filename. Appends 000 if not specified. e.g. h5mgm 123 will produce files named something like "luma.123.<t>.<ext>"

The output files are organised into a directory called **postprocessedoutput.**
###

References:

- https://github.com/cfdemons/LUMA/wiki/Installing-LUMA
- https://www.stackoverflow.com/