Installation guide for *esys-Escript*

Release - 5.9
(r1643944147)

Escript development team

February 4, 2022

Centre for Geoscience Computing (GeoComp)
The University of Queensland
Brisbane, Australia
Email: `esys@esscc.uq.edu.au`

Guide to Documentation

Documentation for `esys.escript` comes in a number of parts. Here is a rough guide to what goes where.

install.pdf	"Installation guide for *esys-Escript*": Instructions for compiling *escript* for your system from its source code.
cookbook.pdf	"The *escript* COOKBOOK": An introduction to *escript* for new users from a geophysics perspective.
user.pdf	"*esys-Escript* User's Guide: Solving Partial Differential Equations with Escript and Finley": Covers main *escript* concepts.
inversion.pdf	"`esys.downunder`: Inversion with *escript*": Explanation of the inversion toolbox for *escript*.
sphinx_api directory	Documentation for *escript Python* libraries.
escript_examples(.tar.gz)/(.zip)	Full example scripts referred to by other parts of the documentation.
doxygen directory	Documentation for C++ libraries (mostly only of interest for developers).

Contents

CHAPTER
ONE

Introduction

This document describes how to install *esys-Escript*[1] on to your computer. To learn how to use Escript please see the Cookbook, User's guide or the API documentation.

Escript is primarily developed on Linux desktop, SGI ICE and MacOS X systems. It can be installed in several ways:

1. Binary packages – ready to run with no compilation required. These are available in Debian and Ubuntu repositories, so just use your normal package manager (so you don't need this guide). They are also available for Anaconda Python 3.

2. Using flatpak

3. From source – that is, it must be compiled for your machine. This is the topic of this guide.

See the site `https://answers.launchpad.net/escript-finley` for online help. Chapter 5 covers installing from source. Appendix A lists some c++ features which your compiler must support in order to compile escript. This version of escript has the option of using `Trilinos` in addition to our regular solvers. Appendix B covers features of `Trilinos` which escript needs.

[1] For the rest of the document we will drop the *esys-*

CHAPTER
TWO

Installing from Flatpak

To install `esys.escript` on any linux distribution using flatpak[1], type

```
flatpak install flathub au.edu.uq.esys.escript
```

This wil download and install `esys.escript` on your machine. The `esys.escript` build installed utilises both the Trilinos and PASO solver libraries, with openMP but without OPENMPI.

After flatpak has finished downloading and installing `esys.escript`, you can launch an `esys.escript` window from the menu or run `esys.escript` in a terminal with the command:

```
flatpak run au.edu.uq.esys.escript [other arguments to pass to escript]
```

Finally, to uninstall `esys.escript` from your machine, type

```
flatpak uninstall escript
```

[1] For most linux distributions this can be installed from the repository. Otherwise, flatpak is available at https://flathub.org/home

CHAPTER
THREE

Installing inside Anaconda

There are precompiled binaries of `esys.escript` available for Anaconda python (for Linux), available in the conda-forge channel on Anaconda cloud. They can be installed using the command

```
conda install esys-escript -c conda-forge
```

CHAPTER
FOUR

Installing from Docker

To install an `esys.escript` Docker container on your machine, first install Docker and then type:

```
docker pull esysescript/esys-escript
```

Once installed, you can launch an escript session using the command:

```
docker run -ti esysescript/esys-escript run-escript
```

If you would also like to mount a folder, you can use the command:

```
docker run -ti -v $(pwd):/app/ esysescript/esys-escript run-escript
```

Additionally, if you wish to run an escript named test_program.py, you can use the command:

```
docker run -ti -v $(pwd):/app/ esysescript/esys-escript \
                                        run-escript [path to test_program.py]
```

CHAPTER
FIVE

Installing from Source

This chapter describes installing `esys.escript` from source on unix/posix like systems (including MacOSX) and Windows 10.

5.1 Parallel Technologies

It is likely that the computer you run `esys.escript` on, will have more than one processor core. Escript can make use of multiple cores [in order to solve problems more quickly] if it is told to do so, but this functionality must be enabled at compile time. Section 5.1.1 gives some rough guidelines to help you determine what you need.

There are two technologies which `esys.escript` can employ here.

- OpenMP – the more efficient of the two [thread level parallelism].
- MPI – Uses multiple processes (less efficient), needs less help from the compiler (not supported on Windows).

Escript is primarily tested on recent versions of the GNU and Intel suites ("g++" / "icpc"), and Microsoft Visual C++ (MSVC). However, it also passes our tests when compiled using "clang++". Escript now requires compiler support for some features of the C++11 standard. See Appendix A for a list.

Our current test compilers include:

- g++ 8
- clang++ 7
- intel icpc v17
- MSVC 2017 or 2019

Note that:

- OpenMP will not function correctly for g++ $\leq$ 4.2.1 (and is not currently supported by clang).
- icpc v11 has a subtle bug involving OpenMP and C++ exception handling, so this combination should not be used.

5.1.1 What parallel technology do I need?

If you are using any version of Linux released in the past few years, then your system compiler will support OpenMP with no extra work (and give better performance); so you should use it. MSVC 2017 and 2019 also have OpenMP support on Windows (OpenMP 2.0). You will not need MPI unless your computer is some form of cluster. MPI is not recommended on Windows as it will interfer with Jupyter.

If you are using BSD or MacOSX and you are just experimenting with `esys.escript`, then performance is probably not a major issue for you at the moment so you don't need to use either OpenMP or MPI. This also applies if you write and polish your scripts on your computer and then send them to a cluster to execute. If in the future you find escript useful and your scripts take significant time to run, then you may want to recompile `esys.escript` with more options.

Note that even if your version of `esys.escript` has support for OpenMP or MPI, you will still need to tell the system to use it when you run your scripts. If you are using the `run-escript` launcher, then this is controlled by the `-t`, `-p`, and `-n` flags. If not, then consult the documentation for your MPI libraries (or the compiler documentation in the case of OpenMP [1]).

If you are using MacOSX, then see the next section, if not, then skip to Section 5.3.

5.2 MacOS

This release of `esys.escript` has only been tested on OSX 10.13. For this section we assume you are using either `homebrew` or `MacPorts` as a package manager[2]. You can of course install prerequisite software in other ways. For example, we have had *some* success changing the default compilers used by those systems. However this is more complicated and we do not provide a guide here.
Both of those systems require the XCode command line tools to be installed[3].

5.3 Building

Escript is built using *SCons*. To simplify the installation process, we have prepared *SCons _options.py* files for a number of common systems[4]. The options files are located in the *scons/templates* directory. We suggest that the file most relevant to your OS be copied from the templates directory to the scons directory and renamed to the form XXXX_options.py where XXXX should be replaced with your computer's (host-)name. If your particular system is not in the list below, or if you want a more customised build, see Section 5.3.11 for instructions.

- Debian - 5.3.1
- Ubuntu - 5.3.2
- Arch Linux - 5.3.3
- OpenSuse - 5.3.4
- Centos - 5.3.5
- Fedora - 5.3.6
- MacOS (macports) - 5.3.7
- MacOS (homebrew) - 5.3.8
- FreeBSD - 5.3.9
- Windows - 5.3.10

Once these are done proceed to Section 5.4 for cleanup steps.
All of these instructions assume that you have obtained the `esys.escript` source (and uncompressed it if necessary).

[1] It may be enough to set the `OMP_NUM_THREADS` environment variable.

[2] Note that package managers will make changes to your computer based on programs configured by other people from various places around the internet. It is important to satisfy yourself as to the security of those systems.

[3] As of OSX10.9, the command `xcode-select --install` will allow you to download and install the commandline tools.

[4] These are correct a time of writing but later versions of those systems may require tweaks. Also, these systems represent a cross section of possible platforms rather than meaning those systems get particular support.

5.3.1 Debian

These instructions were prepared on Debian 10 *Buster*.
As a preliminary step, you should install the dependencies that Escript requires from the repository. If you intend to use Python 2.7, then you should install the following

```
sudo apt-get install python-dev python-numpy python-pyproj python-gdal
sudo apt-get install python-sympy python-matplotlib python-scipy
sudo apt-get install libboost-python-dev libboost-random-dev
sudo apt-get install libnetcdf-dev libnetcdf-cxx-legacy-dev libnetcdf-c++4-dev
sudo apt-get install scons lsb-release libsuitesparse-dev gmsh
```

If you intend to use Python 3.0+, then you should install the following

```
sudo apt-get install python3-dev python3-numpy python3-pyproj python3-gdal
sudo apt-get install python3-sympy python3-matplotlib python3-scipy
sudo apt-get install libboost-python-dev libboost-random-dev
sudo apt-get install libnetcdf-dev libnetcdf-cxx-legacy-dev libnetcdf-c++4-dev
sudo apt-get install libsuitesparse-dev scons lsb-release gmsh
```

In the source directory execute the following (substitute *buster_py2* or *buster_py3* for XXXX):

```
scons -j4 options_file=scons/templates/XXXX_options.py
```

If you wish to test your build, you can use the following:

```
scons -j4 py_tests options_file=scons/templates/XXXX_options.py
```

5.3.2 Ubuntu

These instructions were prepared on Ubuntu 20.04 LTS *Focal Fossa*.

As a preliminary step, you should install the dependencies that Escript requires from the repository.
For Python 3.0+, you should instead install the following packages:

```
sudo apt-get install python3-dev python3-numpy python3-pyproj python3-gdal
sudo apt-get install python3-sympy python3-matplotlib python3-scipy
sudo apt-get install libnetcdf-cxx-legacy-dev libnetcdf-c++4-dev libnetcdf-dev
sudo apt-get install libboost-random-dev libboost-python-dev libboost-iostreams-dev
sudo apt-get install scons lsb-release libsuitesparse-dev
```

Then navigate to the source directory and execute the following

```
scons -j4 options_file=scons/templates/focus_options.py
```

5.3.3 Ubuntu

These instructions were prepared on Arch Linux.

First, install the dependencies that escript uses:

```
pacman -Sy --noconfirm python python-numpy python-scipy python-pyproj
pacman -Sy --noconfirm community/netcdf community/netcdf-cxx
pacman -Sy --noconfirm extra/boost extra/boost-libs suitesparse
```

Now you can compile `esys.escript` using the command

```
scons options_file=scons/templates/arch_py3_options.py -j4 build_full
```

5.3.4 OpenSuse

These instructions were prepared using OpenSUSE Leap 15.2.

As a preliminary step, you should install the dependencies that Escript requires from the repository. If you intend to use Python 2.7, then you should install the following packages

```
sudo zypper in python-devel python2-numpy python-gdal
sudo zypper in python2-scipy python2-sympy python2-matplotlib
sudo zypper in gcc gcc-c++ scons netcdf-devel libnetcdf_c++-devel
sudo zypper in libboost_python-py2_7-1_66_0-devel libboost_numpy-py2_7-1_66_0-devel
sudo zypper in libboost_iostreams1_66_0-devel suitesparse-devel
```

If you intend to use Python 3.0, then you should instead install the following packages

```
sudo zypper in python3-devel python3-numpy python3-GDAL
sudo zypper in python3-scipy python3-sympy python3-matplotlib
sudo zypper in gcc gcc-c++ scons netcdf-devel libnetcdf_c++-devel
sudo zypper in libboost_python-py3-1_66_0-devel libboost_numpy-py3-1_66_0-devel
sudo zypper in libboost_random1_66_0-devel libboost_iostreams1_66_0-devel
sudo zypper in suitesparse-devel
```

Now to build escript itself. In the escript source directory execute the following (substitute *opensuse_py2* or *opensuse_py3* as appropriate for XXXX):

```
scons -j4 options_file=scons/templates/XXXX_options.py
```

If you wish to test your build, you can use the following:

```
scons -j4 py_tests options_file=scons/templates/XXXX_options.py
```

Now go to Section 5.4 for cleanup.

5.3.5 CentOS

It is possible to install `esys.escript` on both CentOS releases 7 and 8. We include separate instructions for each of these CentOS releases in this section.

CentOS release 7

The core of escript works, however some functionality is not available because the default packages for some dependencies in CentOS are too old. At present, it is not possible to compile `esys.escript` using Python 3.0+ on CentOS 7 as Python 3.0+ versions of many of the dependencies are not currently available in any of the CentOS repositories, but this may change in the future. In this section we only outline how to install a version of `esys.escript` that uses Python 2.7.
First, add the `EPEL` repository.

```
yum install epel-release.noarch
```

Install packages:

```
yum install netcdf-devel netcdf-cxx-devel gdal-python
yum install python-devel numpy scipy sympy python2-scons
yum install python-matplotlib gcc gcc-c++ boost-devel
yum install boost-python gdal-devel suitesparse-devel pyproj
```

Now to build escript itself. In the escript source directory:

```
scons -j4 options_file=scons/templates/centos7_0_options.py
```

Now go to Section 5.4 for cleanup.

CentOS release 8

The core of escript works in CentOS 8, however some functionality is not available because the default packages for some dependencies in CentOS are too old. This install is for Python 3.

First, add the EPEL, PowerTools and Okay repositories:

```
yum update
yum install epel-release.noarch dnf-plugins-core
yum config-manager --set-enabled PowerTools
rpm -ivh http://repo.okay.com.mx/centos/8/x86_64/release/okay-release-1-3.el8.noarch.rpm
yum update
```

Now, install the packages:

```
yum install python3-devel python3-numpy python3-scipy python3-pyproj
yum install boost-devel boost-python3 boost-python3-devel
yum install gcc gcc-c++ scons
yum install suitesparse suitesparse-devel
```

Finally, you can compile `esys.escript` with the command

```
scons -j4 options_file=scons/templates/centos8_0_options.py
```

5.3.6 Fedora

These instructions were prepared using Fedora 31 Workstation.
To build the a version of `esys.escript` that uses Python 2.7, install the following packages:

```
yum install gcc-c++ scons suitesparse-devel
yum install python2-devel boost-python2-devel
yum install python2-scipy
yum install netcdf-devel netcdf-cxx-devel netcdf-cxx4-devel
```

To build the a version of `esys.escript` that uses Python 3.0+, install the following packages:

```
yum install gcc-c++ scons suitesparse-devel
yum install python3-devel boost-python3-devel
yum install python3-scipy python3-pyproj python3-matplotlib
yum install boost-python3 boost-numpy3 boost-iostreams boost-random
yum install netcdf-devel netcdf-cxx-devel netcdf-cxx4-devel
```

In the source directory execute the following (substitute *fedora_py2* or *fedora_py3* for XXXX):

```
scons -j4 options_file=scons/templates/XXXX_options.py
```

Now go to Section 5.4 for cleanup.

5.3.7 MacOS 10.10 and later (macports)

The following will install the capabilities needed for the `macports_10.10_options.py` file (later versions can use the same options file).

```
sudo port install scons
sudo port select --set python python27
sudo port install boost
sudo port install py27-numpy
sudo port install py27-sympy
sudo port select --set py-sympy py27-sympy
sudo port install py27-scipy
sudo port install py27-pyproj
sudo port install py27-gdal
sudo port install netcdf-cxx
sudo port install silo
```

```
scons -j4 options_file=scons/templates/macports_10.10options.py
```

5.3.8 MacOS 10.13 and later (homebrew)

The following will install the capabilities needed for the `homebrew_10.13_options.py` file.

```
brew install scons
brew install boost-python
brew install netcdf
```

There do not appear to be formulae for `sympy` or `pyproj` so if you wish to use those features, then you will need to install them separately.

```
scons -j4 options_file=scons/templates/homebrew_10.13_options.py
```

5.3.9 FreeBSD

At time of writing, `numpy` does not install correctly on FreeBSD. Since `numpy` is a critical dependency for `esys.escript`, we have been unable to test on FreeBSD.

5.3.10 Windows

These instructions were prepared for Microsoft Windows 10.
Start by installing `esys.escript` dependencies.

- Microsoft Visual Studio
  1. Download the Microsoft Visual Studio Community 2017 (or VS 2019 if preferred) installer from

     `https://visualstudio.microsoft.com`.
  2. Launch the Visual Studio installer, selecting Individual components:
     - VS 2017: **VC++ 2017 latest v141 tools**
       VS 2019: **MSVC v142 - VS 2019 C++ build tools**
     - **Windows 10 SDK**
     - **MSBuild**
     - **Visual C++ tools for CMake**
- Anaconda
  1. Download the Python 3.7 64-Bit Graphical Installer for Windows from

     `https://anaconda.org/`.
  2. Launch the Anaconda installer, selecting installation type: **Just Me** and destination folder: `C:\Users\%USERNAME%\Anaconda3`.

Next, open Windows Command Prompt (`cmd.exe`) and set-up the `esys.escript` dependencies.

- Conda environment
  1. Create and activate a new environment

     ```
     C:\Users\%USERNAME%\Anaconda3\Scripts\activate
     conda create --name escript python=3.7
     conda deactivate
     C:\Users\%USERNAME%\Anaconda3\Scripts\activate escript
     ```
  2. Install required conda modules

     ```
     conda install numpy==1.15.4 matplotlib==2.2.2 sympy==1.1.1
         boost gdal git pyproj scons scipy m2-patch mumps gmsh
         -c defaults -c conda-forge
     ```
- Vcpkg

1. Build vcpkg package manager

   ```
   cd C:\Users\%USERNAME%
   git clone https://github.com/Microsoft/vcpkg.git
   cd vcpkg
   bootstrap-vcpkg
   ```

2. Install the CppUnit vcpkg package

   ```
   vcpkg install cppunit:x64-windows
   ```

Once the dependencies are installed and set-up, you can download and build `esys.escript` from source.

1. Activate the conda environment (if not active).

   ```
   C:\Users\%USERNAME%\Anaconda3\Scripts\activate escript
   ```

2. Set-up the Command Prompt for the 64-bit MSVC command line build environment.

   ```
   "C:\Program Files (x86)\Microsoft Visual Studio\2017\
       Community\VC\Auxiliary\Build\vcvars64.bat"
   ```

3. Add CppUnit to the Windows System Path.

   ```
   set PATH=%PATH%;C:\Users\%USERNAME%\vcpkg\packages\cppunit_x64-windows\bin
   ```

4. Download the `esys.escript` source code tarball from

   `https://launchpad.net/escript-finley`

   Extract the tarball to `C:\Users\%USERNAME%\escript`

5. Build and install the netCDF-4 C++ library before starting the `esys.escript` build. Download the netCDF-4 C++ v4.3.1 source code tarball from

   `https://github.com/Unidata/netcdf-cxx4/archive/v4.3.1.tar.gz`

   Extract the tarball to `C:\Users\%USERNAME%\escript`

6. Apply the provided patch.

   ```
   cd C:\Users\%USERNAME%\escript\netcdf-cxx4-4.3.1
   patch -p1 < ..\src\tools\anaconda\Anaconda3\netcdf-cxx4.patch
   ```

7. Configure, build, and install netcdf-cxx4.

   ```
   mkdir build
   cd build
   cmake -G "Visual Studio 15 2017 Win64" -DBUILD_SHARED_LIBS=OFF
       -DCMAKE_INSTALL_PREFIX="%CONDA_PREFIX%\Library"
       -DCMAKE_LIBRARY_PATH="%CONDA_PREFIX%\Library\lib"
       -DCMAKE_PREFIX_PATH="%CONDA_PREFIX%\Library"
       -DNETCDF_LIB_NAME="netcdf" -DHDF5_LIB_NAME="hdf5" ..
   cmake --build . --config Release
   cmake --build . --config Release --target install
   ```

8. Kick-off the `esys.escript` build when the netCDF-4 C++ install is complete.

   ```
   cd C:\Users\%USERNAME%\escript\src
   scons -j4 build_full options_file=scons/templates/windows_msvc141_options.py
   ```

9. Once the build completes successfully, you can validate `esys.escript` using the provided test script.

   ```
   python utest.py C:\Users\%USERNAME%\escript\src\build -t8
   ```

5.3.11 Other Systems / Custom Builds

`esys.escript` has support for a number of optional packages. Some, like `netcdf` need to be enabled at compile time, while others, such as `sympy` and the projection packages used in `esys.downunder` are checked at run time. For the second type, you can install them at any time (ensuring that python can find them) and they should work. For the first type, you need to modify the options file and recompile with scons. The rest of this section deals with this.

To avoid having to specify the options file each time you run scons, copy an existing `_options.py` file from the `scons/` or `scons/templates/` directories. Put the file in the `scons` directory and name it *your-machinename*`_options.py`.[5]. For example: on a machine named toybox, the file would be `scons/toybox_options.py`.

Individual lines can be enabled/disabled, by removing or adding # (the python comment character) to the beginning of the line. For example, to enable OpenMP, change the line

```
#openmp = True
```

to

```
openmp = True
```

If you are using libraries which are not installed in the standard places (or have different names) you will need to change the relevant lines. A common need for this would be using a more recent version of the boost::python library. You can also change the compiler or the options passed to it by modifying the relevant lines.

MPI

If you wish to enable or disable MPI, or if you wish to use a different implementation of MPI, you can use the `mpi` configuration variable. You will also need to ensure that the `mpi_prefix` and `mpi_libs` variables are uncommented and set correctly. To disable MPI use, `mpi = 'none'`.

Testing

As indicated earlier, you can test your build using `scons py_tests`. Note however, that some features like `netCDF` are optional for using `esys.escript`, the tests will report a failure if they are missing.

5.4 Cleaning up

Once the build (and optional testing) is complete, you can remove everything except:

- bin
- esys
- lib
- doc
- CREDITS
- LICENSE
- README

The last three aren't strictly required for operation. The `doc` directory is not required either but does contain examples of escript scripts.

You can run escript using *`path_to_escript_files`*`/bin/run-escript`.
Where *`path_to_escript_files`* is replaced with the real path.

> *Optional step*
> You can add the escript `bin` directory to your `PATH` variable. The launcher will then take care of the rest of the environment.

[5]If the name has - or other non-alpha characters, they must be replaced with underscores in the filename

5.5 Optional Extras

Some other packages which might be useful include:

- Lapack and UMFPACK — direct solvers (install the relevant libraries and enable them in the options file).
- support for the Silo file format (install the relevant libraries and enable them in the options file).
- VisIt — visualisation package. Can be used independently but our `weipa` library can make a Visit plug-in to allow direct visualisation of escript simulations.
- gmsh — meshing software used by our `pycad` library.
- Mayavi2 — another visualisation tool.

5.5.1 Trilinos

`esys.escript` now has some support for Trilinos[6] solvers and preconditioners. The most significant limitation is that the current Trilinos release does not support block matrices so `esys.escript` can only use Trilinos solvers for single PDEs (i.e. no PDE systems).

If your distribution does not provide Trilinos packages you can build a working version from source. (See Appendix B)

5.6 Testing `esys.escript`

`esys.escript` has extensive testing that can be used to confirm that the program is working correctly. To run the unit testing, compile `esys.escript` with the flag `build_full`. This will build `esys.escript` normally and then create a shell script named `utest.sh`. Once this file has been created, you can run unit testing using the command

```
sh utest.sh path_to_build_folder '-tT -nN -pP'
```

where `T`, `N` and `P` represent the number of threads, nodes and processes to run the testing on. Some of these terms can be omitted. For example, to run the testing in serial, you would run

```
sh utest.sh path_to_build_folder '-t1'
```

Note that a careless selection of these parameters may cause the testing program to skip many of the tests. For example, if you compile `esys.escript` with OpenMP enabled but then instruct the testing program to run on a single thread, many of the OpenMP tests will not be run.

[6] https://trilinos.org/

APPENDIX
A

Required compiler features

Building escript from source requires that your c++ compiler supports at least the following features:

- `std::complex<>`
- Variables declared with type `auto`
- Variables declared with type `decltype(T)`
- `extern template class` to prevent instantiation of templates.
- `template class` *`classname<type>`*`;` to force instantiation of templates
- `isnan()` is defined in the `std::` namespace

The above is not exhaustive and only lists language features which are more recent that our previous baseline of c++99 (or which we have recently begun to rely on). Note that we test on up to date versions of `g++, icpc & clang++` so they should be fine.

Note that in future we may use c++14 features as well.

APPENDIX
B

Trilinos

In order to solve PDEs with complex coefficients, escript needs to be compiled with `Trilinos` support. This requires that your version of Trilinos has certain features enabled. Since some precompiled distributions of `Trilinos` are not built with these features, you may need to compile `Trilinos` yourself as well.

While we can't provide support for building `Trilinos`, we provide here two configuration files which seem to work for Debian 10 “buster'. One of these cmake script builds `Trilinos` with MPI support and one builds `Trilinos` without MPI support.

B.1 Debian “buster” configuration

B.1.1 Required packages

The following packages should be installed to attempt this build:

cmake

g++

libsuitesparse-dev

libmumps-dev

libboost-dev

libscotchparmetis-dev

libmetis-dev

libcppunit-dev

B.1.2 Example configuration file (without MPI)

```
#!/bin/bash

# Set this to the root of your Trilinos source directory.
TRILINOS_PATH=../trilinos_source

rm -f CMakeCache.txt

EXTRA_ARGS=$@

cmake \
      -D CMAKE_C_COMPILER=`which gcc` \
      -D CMAKE_CXX_COMPILER=`which g++` \
      -D CMAKE_Fortran_COMPILER=`which gfortran` \
      -D PYTHON_EXECUTABLE=/usr/bin/python3 \
      -D CMAKE_INSTALL_PREFIX=/usr/local/ \
      -D Trilinos_ENABLE_CXX11=ON \
      -D CMAKE_C_FLAGS=' -w -fopenmp' \
      -D CMAKE_CXX_FLAGS=' -w -fopenmp'\
      -D Trilinos_ENABLE_Fortran=ON \
      -D BUILD_SHARED_LIBS=ON \
      -D TPL_ENABLE_BLAS=ON \
      -D TPL_ENABLE_Boost=ON \
      -D TPL_ENABLE_Cholmod=ON \
      -D TPL_ENABLE_LAPACK=ON \
      -D TPL_ENABLE_METIS=ON \
      -D TPL_ENABLE_SuperLU=OFF \
      -D TPL_ENABLE_UMFPACK=ON \
      -D TPL_BLAS_INCLUDE_DIRS=/usr/include/suitesparse \
      -D TPL_Cholmod_INCLUDE_DIRS=/usr/include/suitesparse \
      -D TPL_Cholmod_LIBRARIES='libcholmod.so;libamd.so;libcolamd.so' \
      -D TPL_UMFPACK_INCLUDE_DIRS=/usr/include/suitesparse \
      -D TPL_Boost_INCLUDE_DIRS=/usr/local/boost/include \
      -D TPL_Boost_LIBRARY_DIRS=lib \
      -D Trilinos_ENABLE_Amesos=ON \
      -D Trilinos_ENABLE_Amesos2=ON \
      -D Trilinos_ENABLE_AztecOO=ON \
      -D Trilinos_ENABLE_Belos=ON \
      -D Trilinos_ENABLE_Ifpack=ON \
      -D Trilinos_ENABLE_Ifpack2=ON \
      -D Trilinos_ENABLE_Kokkos=ON \
      -D Trilinos_ENABLE_Komplex=ON \
      -D Trilinos_ENABLE_ML=ON \
      -D Trilinos_ENABLE_MueLu=ON \
      -D Trilinos_ENABLE_Teuchos=ON \
      -D Trilinos_ENABLE_Tpetra=ON \
      -D Trilinos_ENABLE_ALL_OPTIONAL_PACKAGES=ON \
      -D Kokkos_ENABLE_AGGRESSIVE_VECTORIZATION=ON \
      -D Tpetra_INST_COMPLEX_DOUBLE=ON \
      -D Trilinos_ENABLE_COMPLEX_DOUBLE=ON \
      -D Teuchos_ENABLE_COMPLEX=ON \
      -D Tpetra_INST_INT_INT=ON \
      -D Tpetra_ENABLE_DEPRECATED_CODE=ON \
      -D Trilinos_ENABLE_OpenMP=ON \
      -D Trilinos_ENABLE_MPI=OFF \
      -D Trilinos_ENABLE_EXPLICIT_INSTANTIATION=ON \
      -D Kokkos_ENABLE_COMPILER_WARNINGS=ON \
      -D Amesos2_ENABLE_Basker=ON \
      -D Tpetra_INST_SERIAL:BOOL=ON \
      -D Trilinos_ENABLE_TESTS=OFF \
$EXTRA_ARGS \
$TRILINOS_PATH
```

B.1.3 Example configuration file (with MPI)

```
#!/bin/bash

# Set this to the root of your Trilinos source directory.
```

```
TRILINOS_PATH=../trilinos_source

#
# You can invoke this shell script with additional command-line
# arguments.  They will be passed directly to CMake.
#
EXTRA_ARGS=$@

rm -f CMakeCache.txt

cmake \
      -D MPI_C_COMPILER=`which mpicc` \
      -D MPI_CXX_COMPILER=`which mpic++` \
      -D MPI_Fortran_COMPILER=`which mpif90` \
      -D PYTHON_EXECUTABLE=/usr/bin/python3 \
      -D CMAKE_INSTALL_PREFIX=/usr/local/ \
      -D Trilinos_ENABLE_CXX11=ON \
      -D CMAKE_C_FLAGS=' -w -fopenmp' \
      -D CMAKE_CXX_FLAGS=' -w -fopenmp'\
      -D Trilinos_ENABLE_Fortran=ON \
      -D BUILD_SHARED_LIBS=ON \
      -D TPL_ENABLE_BLAS=ON \
      -D TPL_ENABLE_Boost=ON \
      -D TPL_ENABLE_Cholmod=ON \
      -D TPL_ENABLE_LAPACK=ON \
      -D TPL_ENABLE_METIS=ON \
      -D TPL_ENABLE_SuperLU=OFF \
      -D TPL_ENABLE_UMFPACK=ON \
      -D TPL_ENABLE_PARMETIS=ON \
      -D TPL_ENABLE_SCALAPACK=ON \
      -D TPL_ENABLE_MUMPS=OFF \
      -D TPL_BLAS_INCLUDE_DIRS=/usr/include/suitesparse \
      -D TPL_Cholmod_INCLUDE_DIRS=/usr/include/suitesparse \
      -D TPL_Cholmod_LIBRARIES='libcholmod.so;libamd.so;libcolamd.so' \
      -D TPL_UMFPACK_INCLUDE_DIRS=/usr/include/suitesparse \
      -D TPL_Boost_INCLUDE_DIRS=/usr/local/boost/include \
      -D TPL_Boost_LIBRARY_DIRS=lib \
      -D TPL_MUMPS_INCLUDE_DIRS='/usr/include/mumps-seq-shared/' \
      -D Trilinos_ENABLE_Amesos=ON \
      -D Trilinos_ENABLE_Amesos2=ON \
      -D Trilinos_ENABLE_AztecOO=ON \
      -D Trilinos_ENABLE_Belos=ON \
      -D Trilinos_ENABLE_Ifpack=ON \
      -D Trilinos_ENABLE_Ifpack2=ON \
      -D Trilinos_ENABLE_Kokkos=ON \
      -D Trilinos_ENABLE_Komplex=ON \
      -D Trilinos_ENABLE_ML=ON \
      -D Trilinos_ENABLE_MueLu=ON \
      -D Trilinos_ENABLE_Teuchos=ON \
      -D Trilinos_ENABLE_Tpetra=ON \
      -D Trilinos_ENABLE_ALL_OPTIONAL_PACKAGES=ON \
      -D KOKKOS_ENABLE_AGGRESSIVE_VECTORIZATION=ON \
      -D Tpetra_INST_COMPLEX_DOUBLE=ON \
      -D Teuchos_ENABLE_COMPLEX=ON \
      -D Tpetra_INST_INT_INT=ON \
      -D Tpetra_ENABLE_DEPRECATED_CODE=ON \
      -D Trilinos_ENABLE_OpenMP=ON \
      -D Trilinos_ENABLE_MPI=ON \
      -D Trilinos_ENABLE_EXPLICIT_INSTANTIATION=ON \
      -D KOKKOS_ENABLE_COMPILER_WARNINGS=ON \
      -D Amesos2_ENABLE_Basker=ON \
      -D Tpetra_INST_SERIAL:BOOL=ON \
      -D Trilinos_ENABLE_TESTS=OFF \
$EXTRA_ARGS \
$TRILINOS_PATH
```