Installation guide for esys-Escript

Release - 4.2
(r5872)

Escript development team

15 January 2016

Earth Systems Science Computational Centre (ESSCC)
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. Also briefly covers installing .deb packages for Debian and Ubuntu.

- **cookbook.pdf**: "The escript COOKBOOK": An introduction to escript for new users from a geophysics perspective.


- **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

1 Introduction  

2 Installing from Source  
   2.1 Parallel Technologies  
   2.1.1 What parallel technology do I need?  
   2.2 MacOS  
   2.3 Building  
      2.3.1 Debian  
      2.3.2 Ubuntu  
      2.3.3 OpenSuse  
      2.3.4 Centos  
      2.3.5 Fedora  
      2.3.6 MacOS 10.10/10.11 (macports)  
      2.3.7 10.10/10.11 (homebrew)  
      2.3.8 FreeBSD  
      2.3.9 Other Systems / Custom Builds  
   2.4 Cleaning up  
   2.5 Optional Extras  

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>5</td>
</tr>
<tr>
<td>2 Installing from Source</td>
<td>7</td>
</tr>
<tr>
<td>2.1 Parallel Technologies</td>
<td>7</td>
</tr>
<tr>
<td>2.1.1 What parallel technology do I need?</td>
<td>7</td>
</tr>
<tr>
<td>2.2 MacOS</td>
<td>8</td>
</tr>
<tr>
<td>2.3 Building</td>
<td>8</td>
</tr>
<tr>
<td>2.3.1 Debian</td>
<td>8</td>
</tr>
<tr>
<td>2.3.2 Ubuntu</td>
<td>9</td>
</tr>
<tr>
<td>2.3.3 OpenSuse</td>
<td>9</td>
</tr>
<tr>
<td>2.3.4 Centos</td>
<td>10</td>
</tr>
<tr>
<td>2.3.5 Fedora</td>
<td>10</td>
</tr>
<tr>
<td>2.3.6 MacOS 10.10/10.11 (macports)</td>
<td>10</td>
</tr>
<tr>
<td>2.3.7 10.10/10.11 (homebrew)</td>
<td>11</td>
</tr>
<tr>
<td>2.3.8 FreeBSD</td>
<td>11</td>
</tr>
<tr>
<td>2.3.9 Other Systems / Custom Builds</td>
<td>11</td>
</tr>
<tr>
<td>2.4 Cleaning up</td>
<td>12</td>
</tr>
<tr>
<td>2.5 Optional Extras</td>
<td>12</td>
</tr>
</tbody>
</table>
This document describes how to install *esys-Escript* on to your computer. To learn how to use Escript please see the Cookbook, User’s guide or the API documentation. If you use the Debian or Ubuntu and you have installed the `python-escript-doc` package then the documentation will be available in the directory `/usr/share/doc/python-escript-doc`, otherwise (if you haven’t done so already) you can download the documentation bundle from launchpad.

Escript is primarily developed on Linux desktop, SGI ICE and MacOS X systems. This guide covers installing the packages from source.

See the site https://answers.launchpad.net/escript-finley for online help. Chapter 2 covers installing from source.

---

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

---

Chapter 1. Introduction 5
CHAPTER
TWO

Installing from Source

This chapter assumes you are using a unix/posix like system (including MacOSX).

2.1 Parallel Technologies

It is likely that the computer you run esys.escript on, will have more than one processor core. esys.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 2.1.1 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.

Escript is primarily tested on recent versions of the GNU and Intel suites (“g++” / “icpc”). However, it also passes our tests when compiled using ”clang++”.

Our current test compilers include:

• g++ 4.7.2, 4.9.2
• clang++ (OSX 10.10 default)
• intel icpc v15

Note that:

• OpenMP will not function correctly for g++ ≤ 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.

2.1.1 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. You will not need MPI unless your computer is some form of cluster.

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 \(-t\) and \(-p\) options. 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 2.3.

2.2 MacOS

This release of \texttt{esys.escript} has only been tested on OSX 10.11. For this section we assume you are using
either \texttt{homebrew} or \texttt{MacPorts} as a package manager\(^2\). You can of course install prerequisite software in other
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\(^3\).

2.3 Building

To simplify things for people, we have prepared \texttt{__options.py} files for a number of systems\(^4\). The \texttt{__options.py}
files are located in the \texttt{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 \texttt{XXXX_options.py} where \texttt{XXXX}
should be replaced with your computer’s name. If your particular system is not in the list below, or if you want a
more customised build, see Section 2.3.9 for instructions.

- Debian - 2.3.1
- Ubuntu - 2.3.2
- OpenSuse - 2.3.3
- Centos - 2.3.4
- Fedora - 2.3.5
- MacOS (macports) - 2.3.6
- MacOS (homebrew) - 2.3.7
- FreeBSD - 2.3.8

Once these are done proceed to Section 2.4 for cleanup steps.

All of these instructions assume that you have obtained the \texttt{esys.escript} source (and uncompressed it if
necessary).

2.3.1 Debian

\begin{verbatim}
sudo aptitude install python-dev python-numpy libboost-python-dev libnetcdf-dev
sudo aptitude install scons lab-release libboost-random-dev
sudo aptitude install python-sympy python-matplotlib python-scipy
sudo aptitude install python-pyproj python-gdal
\end{verbatim}

If you are running Jessie, (or if \texttt{wheezy-backports} is in your \texttt{apt} sources) you can use:

\begin{verbatim}
sudo aptitude install gmsh
\end{verbatim}

to add extra meshing functionality.

\textit{Optional step}

If for some reason, you wish to rebuild the documentation, you would also need the following:

\(^{1}\)It may be enough to set the \texttt{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 \texttt{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.
In the source directory execute the following (substitute wheezy for XXXX):

```bash
scons -j1 options_file=scons/templates/XXXX_options.py
```

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

```bash
scons -j1 py_tests options_file=scons/templates/XXXX_options.py
```

### 2.3.2 Ubuntu

If you have not installed `aptitude`, then substitute `apt-get` in the following.

```bash
sudo aptitude install python-dev python-numpy libboost-python-dev
sudo aptitude install libnetcdf-dev libboost-random-dev
sudo aptitude install scons lsb-release
sudo aptitude install python-sympy python-matplotlib python-scipy
sudo aptitude install python-pyproj python-gdal gmsh
```

**Optional step**

If for some reason, you wish to rebuild the documentation, you would also need the following:

```bash
sudo aptitude install python-sphinx doxygen python-docutils texlive
sudo aptitude install zip texlive-latex-extra latex-xcolor
```

In the source directory execute the following (substitute precise, quantal or raring as appropriate for XXXX):

```bash
scons -j1 options_file=scons/templates/XXXX_options.py
```

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

```bash
scons -j1 py_tests options_file=scons/templates/XXXX_options.py
```

### 2.3.3 OpenSuse

These instructions were prepared using release 13.2.

Install packages from the main distribution:

```bash
sudo zypper install libboost_python1_54_0 libboost_random1_54_0
sudo zypper install python-devel python-numpy libnetcdf_c++-devel
sudo zypper install python-scipy python-sympy python-matplotlib
sudo zypper install gcc gcc-c++ scons boost-devel netcdf-devel
```

These will allow you to use most features except some parts of the `esys.downunder` inversion library. If you wish to use those, you will need some additional packages [python-pyproj, python-gdal]. This can be done now or after Escript installation.

```bash
sudo zypper addrepo \
```

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

```bash
scons -j1 options_file=scons/templates/opensuse13.1_options.py
```

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

```bash
scons -j1 py_tests options_file=scons/templates/opensuse13.1_options.py
```

Now go to Section 2.4 for cleanup.
2.3.4 Centos

These instructions were prepared using centos release 7.0. The core of escript works, however some functionality is not available because the default packages for some dependencies in Centos are too old.

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 scons boost-devel
yum install python-matplotlib gcc gcc-c++
yum install boost-python
```

The above packages will allow you to use most features except the esys.downunder inversion library. If you wish to use those it, you will need to install some additional packages.

For some coordinate transformations, esys.downunder can also make use of the python interface to a tool called proj. There does not seem to be an obvious centos repository for this though. If it turns out to be necessary for your particular application, the source can be downloaded.

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

```
scons -j1 options_file=scons/templates/centos7_0_options.py
```

Now go to Section 2.4 for cleanup.

2.3.5 Fedora

These instructions were prepared using release 21.5.

Install packages:

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

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

```
scons -j1 options_file=scons/templates/fedora21_5_options.py
```

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

```
scons -j1 py_tests options_file=scons/templates/fedora21_5_options.py
```

Now go to Section 2.4 for cleanup.

2.3.6 MacOS 10.10/10.11 (macports)

The following will install the capabilities needed for the macports_10.10_options.py 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 instal silo
```

```
scons -j1 options_file=scons/templates/macports_10.10options.py
```

10 2.3. Building
2.3.7 10.10/10.11 (homebrew)

The following will install the capabilities needed for the homebrew_10.10_options.py file. For OS 10.11 use homebrew_11_options.py instead.

```
brew install scons
brew install boost-python
brew install homebrew/science/netcdf --with-cxx-compat
```

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 -j1 options_file=scons/templates/homebrew_10.10_options.py
```

2.3.8 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.

2.3.9 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`. 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'`.

**Python3**

`esys.escript` works with `python3` but until recently, many distributions have not distributed python3 versions of their packages. You can try it out though by modifying or adding the following variables in your options file:

```
pythoncmd='python3'
usepython3=True
pythonlibname='whateveryourpython3libraryiscalled'
```

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

2.4 Cleaning up

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

- bin
- esys
- lib
- doc
- CREDITS.TXT
- README_LICENSE

The last two 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.

2.5 Optional Extras

Some other packages which might be useful include:

- support for silo 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 files.
- gmsh — meshing software used by our pycad library.
- Mayavi2 — another visualisation tool.