Installation guide for *esys-Escript*

Release - 3.4
(r4488)

Escript development team

June 28, 2013

Earth Systems Science Computational Centre (ESSCC)
The University of Queensland
Brisbane, Australia
Email: esys@esscc.uq.edu.au

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. 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. It can be installed in two ways:

1. Binary packages – ready to run with no compilation required. Bundles are available for:
   - .deb files for Debian and Ubuntu Linux

   Please see Appendix A for Windows instructions. The rest of this guide assumes you are using a posix like system.
2. From source – that is, it must be compiled for your machine. This will be required if there is no binary package for your machine or if extra functionality is required such as MPI parallelisation.

The major change from the point of view of installation is that we no longer provide binary packages compiled against the "support bundles". This means, there are no longer binary packages for MacOS or generic Linux. One can still build from source and we have endevoured to make this process as straight forward as possible.

See the site `https://answers.launchpad.net/escript-finley` for online help. Chapter 2 describes how to install binary packages on Debian/Ubuntu systems. Chapter 3 covers installing from source. Appendix A gives brief instructions for Windows.

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

CHAPTER
TWO

Debian/Ubuntu Binary Installation

We provide `.deb` files for the following distributions:

Debian (i386 or amd64):

- 6 — *Squeeze*
- 7 — *Wheezy*

Ubuntu (i386 or amd64):

- 12.04 — *Precise* Pangolin (LTS)
- 12.10 — *Quantal* Queztal
- 13.04 — *Raring* Ringtail

Two packages make up the `esys.escript` system: The main package which contains all system itself and the (optional) documentation package. The main package will be named `python-escript-X-D_A.deb` where `X` is the version, `D` is the distribution codename (eg "`wheezy`" or "`precise`") and `A` is the architecture. For example, `python-escript-3.4-1-precise_amd64.deb` would be the file for Ubuntu 12.04 for 64bit processors. There is a common documentation for all distributions called `python-escript-doc-X_all.deb`. To install Escript, download the appropriate `.deb` file(s) and execute the following commands as root (you need to be in the directory containing the file):

(For Ubuntu users)
You will need to either install `aptitude`[1] or substitute `apt-get` where this guide uses `aptitude`.

```
sudo apt-get install aptitude
```

```
dpkg --unpack python-escript*.deb
aptitude install python-escript python-escript-doc
```

Installing escript should not remove any (non-escript) packages from your system. If aptitude suggests removing `python-escript` then choose 'N'. If it wants to remove `escript-noalias` or `escript`, then choose 'Y'. It should then suggest installing some dependencies choose 'Y' here.

If you use sudo (for example on Ubuntu) enter the following instead:

```
sudo dpkg --unpack python-escript*.deb
sudo aptitude install python-escript python-escript-doc
```

There are a number of optional dependencies which you should also install unless you are sure you don't need them:

```
aptitude install python-sympy python-matplotlib python-scipy
aptitude install python-pyproj python-gdal
```

This should install Escript and its dependencies on your system. Please notify the development team if something goes wrong.

[1] Unless you are short on disk space `aptitude` is recommended

CHAPTER
THREE

Installing from Source

This chapter assumes you are using a unix/posix like system.

3.1 Intro

Some initial questions:

1. Are you using MacOS? (If not, then skip to Section 3.1.2).
2. Which parallel technologies do you wish to use?
3. Which packaging system are you using?

3.1.1 MacOS and compilers

This information is as accurate as far as we can tell at time of writing but things may change.

In order to install `esys.escript` from source, you need a compiler. This is a bit harder on MacOS than on other systems which either provide a compiler as part of the base install (BSD) or allow people to download one as part of the install process (Linux). In earlier releases of its operating system ("Snow Leopard" and earlier), Apple included an optional "developer tools" package (`XCode`) on the install media. For more recent releases, `XCode` is available for "free" from Apple's application store. Why "free"? The only way to access the application store is with an AppleID which requires either:

- purchasing an iTunes gift card.
- giving Apple access to your credit card.
- signing up as an Apple developer[1] and giving up personal information.

If you install `XCode`, you will need to download the "command line tools" optional package [see `XCode` documentation for details].

There are also a number of projects on the net which aim to deliver compilers for MacOS. Use at your own risk. For example:

- `http://hpc.sourceforge.net`
- `http://kennethreitz.org/experiments/xcode-gcc-and-homebrew`

[1] If you do this you can download a "command line tools" package which installs the relevant compilers without needing to install all of `XCode`.

3.1.2 Parallel Technologies

It is extremely 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.

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

- OpenMP – 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 or Intel suites ("gcc, g++" / "icc, icpc"). However, it also passes our tests when compiled using "clang, clang++". The table below shows what methods are available with which compilers.

	Serial	OpenMP	MPI
$\leq$ gcc-4.2.1	✓	[2]	✓
gcc (recent $\geq$ 4.3.2)	✓	✓	✓
icc(10)	✓	✓	✓
icc(11)	✓	[3]	✓
icc(12)	✓	✓	✓
clang	✓		✓

Where both OpenMP and MPI are marked, `esys.escript` can be compiled with either or both. A ✓mark means that combination passes our tests.

3.1.3 Packaging System

The packaging system (also known as the package manager) is the tool you use to search for and install new open source software. For Linux, there will be one set up by default: the apt tools on Debian and Ubuntu, yast on Suse, yum on the RedHat family. On BSD systems this will be a combination of `pkg_add` and the `ports` tree.

For MacOS, this is a bit more tricky. There are a number of possible systems including `macports` and `homebrew`[4], but they do not come pre-installed so if you want to make use of one you will need you will need to install it.

Packaging systems 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.

If you are using Linux or have decided that you don't want to use OpenMP skip to Section 3.2.

Changing Compilers on MacOS and BSD

Not for the faint-hearted.

In order to use OpenMP, your compiler must support it. The compilers provided in Apple's package and BSD do not support OpenMP. `Clang` has no support for it at all, while `GCC4.2.1` has some, but it does not work very well[5]. You can use the packaging system to install a more up to date version of `gcc`. However, you will need to install a number of other dependencies in order for escript to operate. If these dependencies end up fighting over which versions of libraries to use, there will be problems. [6] To avoid this, if you are intend to change the compiler or python versions, you should install the new compiler first, then the new version of python. After this, you can install the other dependencies. *Please note that none of the Mac based package managers nor BSD recommend changing the default C compiler so do so at your own risk. If you want OpenMP though, there does not seem to be a choice.*

The following steps seemed to work when we tested them but we cannot make guarantees.

[2] The OpenMP support in gcc-4.2.1 is buggy/non-functional.
[3] There is a subtle bug in icc-11 when OpenMP and c++ exception handling are combined.
[4] There is also `fink`, but we have not experimented with that.
[5] Rejects valid code, crashes etc
[6] For example: Trying to use two versions of `Python` in the same program or two versions of the `c++` standard library.

Changing compiler on FreeBSD(9.1)

The following sequence passes our unit tests under OpenMP.

- Install `gcc46` from ports.
- Modify `/etc/make.conf` to set the default compiler to be `gcc46`
- Install the remaining dependencies.
- Configure `esys.escript` to build with OpenMP.

We chose version 4.6 rather than a later one because one of the optional dependencies (scipy) will try to install it anyway.

Changing compiler on MacPorts

The following sequence passes our unit tests under OpenMP.

- `port install gcc47`
- Set the default compiler for the command line with: `port select --set gcc mp-gcc47`
- Set the default compiler for macports by adding the following line to `/opt/local/etc/macports/macports.conf`:

```
#Added by user to coerce macports into using a newer compiler
default_compiler        macports-gcc-4.7
```

- Now install the remainder of the dependencies using `port -s install X`. This will build them from source rather than downloading precompiled versions. (This will unfortunately mean larger downloads.). In some cases, some dependencies will not build properly from source. In those cases(where XYZ is the dependency that fails to build), `port clean XYZ` then `port install XYZ`. Then *try to install the rest from source*.

  Installing from source via macports is necessary to convince macports to link against the libraries provided by your new compiler rather than the default one.

Changing compiler on Homebrew

There is no configuration file that needs to be changed in order to use a different compiler. You do need to do things in the correct order and make sure that you have made any required changes to your environment variables before moving on to the next step. In particular, the compiler must be installed before anything else, followed by Python.

3.2 Building

To simplify things for people, we have prepared `_options.py` files for a number of systems[7]. If your particular system is not in the above list, or if you want a more customised build[8], see Section 3.2.9 for instructions.

- Debian - 3.2.1
- Ubuntu - 3.2.2
- OpenSuse - 3.2.3
- Centos - 3.2.4
- Fedora - 3.2.5

[7] 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.

[8] for example, you want MPI functionality or you wish to use a different compiler

- MacOS (macports) - 3.2.6
- MacOS (homebrew) - 3.2.7
- FreeBSD - 3.2.8

Once these are done proceed to Section 3.3 for cleanup steps.
All of these instructions assume that you have obtained the source (uncompressed it if necessary).

3.2.1 Debian

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

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

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

In the source directory execute the following (substitute squeeze or wheezy as appropriate for XXXX):

```
scons -j1 options_file=scons/os/XXXX_options.py
```

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

```
scons -j1 py_tests options_file=scons/os/XXXX_options.py
```

3.2.2 Ubuntu

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

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

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

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

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

```
scons -j1 options_file=scons/os/XXXX_options.py
```

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

```
scons -j1 py_tests options_file=scons/os/XXXX_options.py
```

3.2.3 OpenSuse

These instructions were prepared using release 12.3.
Install packages from the main distribution:

```
sudo yast2 --install libboost_python1_49_0 python-devel python-numpy
sudo yast2 --install python-scipy python-sympy python-matplotlib libnetcdf_c++-devel
sudo yast2 --install gcc-c++ scons boost-devel netcdf-devel
```

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

Optional step
Add http://ftp.suse.de/pub/opensuse/repositories/Application:/Geo/openSUSE_12.3/ to your repositories in YaST.

```
sudo yast2 --install python-pyproj, python-gdal
```

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

```
scons -j1 options_file=scons/os/opensuse12.3_options.py
```

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

```
scons -j1 py_tests options_file=scons/os/opensuse12.3_options.py
```

Now go to Section 3.3 for cleanup.

3.2.4 Centos

These instructions were prepared using release 6.4.
Install packages from the main distribution:

```
yum install python-devel numpy scipy scons boost-devel
yum install python-matplotlib gcc-g++
yum install boost-python
```

The above packages will allow you to use most features except saving and loading files in netCDF format and the esys inversion library. If you wish to use those features, you will need to install some additional packages. NetCDF needs to be installed when you compile if you wish to use it.

Optional step
Add the EPEL repository.

```
rpm -U http://download.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
```

```
yum install netcdf-devel sympy gdal-python
```

For some coordinate transformations, esys 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/os/centos6.4_options.py
```

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

```
scons -j1 py_tests options_file=scons/os/centos6.4_options.py
```

Now go to Section 3.3 for cleanup.

3.2.5 Fedora

These instructions were prepared using release 18.
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/os/fedora18_options.py
```

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

```
scons -j1 py_tests options_file=scons/os/fedora18_options.py
```

Now go to Section 3.3 for cleanup.

3.2.6 MacOS (macports)

```
port install python27
port select --set python python27
port install scons
port install openmpi
port install py27-numpy
port install boost
port install py27-sympy
port select --set py-sympy py27-sympy
install py27-scipy
install py27-pyproj
install py27-gdal
install py27-netcdf4
install netcdf-cxx
```

```
scons -j1 options_file=scons/os/macports_options.py
```

3.2.7 MacOS (homebrew)

Note that these steps add "non-official" packages. You will also want to make sure that the homebrew Python is executed in preference to the system Python[9].

```
brew install scons
brew install python
brew install boost
brew tap samueljohn/python
brew tap homebrew/science
pip install nose
brew install gfortran
brew install samueljohn/python/numpy
brew install scipy
brew install gdal
brew install openmpi
brew install matplotlib
brew install netcdf --enable-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/os/homebrew_options.py
```

3.2.8 FreeBSD

The following was tested on the 9.1 release of FreeBSD. It passes the majority of tests but there is an issue related to some features in the inversion library. The following set of installations "works" but is not guaranteed to be minimal[10].

Install the following packages:

- subversion
- scons
- boost-python-libs
- bash

Now install the following ports:

- science/py-scipy

[9] Putting `/usr/local/bin` at the front of your PATH is one way to do this.

[10] Depending on your needs you might be able to get by with a smaller set of packages.

- science/py-netCDF4
- math/sympy
- graphics/py-pyproj
- graphics/py-gdal
- net/openmpi

You will need to add `/usr/local/mpi/openmpi/bin` to your path if you wish to build with MPI.
Next choose (or create) your options file. In this case we have three prepared in the `scons/os` directory:

- `freebsd91_options.py`
- `freebsd91_mpi_options.py` If you would like to use MPI.
- `freebsd91_gcc46_options.py` Use this if you have managed to change compilers to gcc4.6 (and would like to use OpenMP).

In the escript source directory (where ZZZ is your options file):

```
scons -j1 options_file=ZZZ
```

3.2.9 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` 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/os/` directories. Put the file in the `scons` directory and name it *yourmachinename*`_options.py`.[11]. 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 boostpython 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. To disable MPI use, `mpi = 'none'`. You will also need to ensure that the `mpi_prefix` and `mpi_libs` variables are uncommented and set correctly.

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 the following variables:

```
pythoncmd='python3'
```

```
usepython3=True
```

```
pythonlibname='whateveryourpython3libraryiscalled'
```

[11] If 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 textttnetCDF are optional for using `esys.escript`, the tests will report a failure if they are missing.

3.3 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.

3.4 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.
- mayavi — another visualisation tool.

APPENDIX
A

Windows binary installation

There is no automated install/uninstall procedure for Escript on Windows at this time. Where builds are available, they will be built for Windows XP and has not been tested on newer versions. If you want to use MPI, make sure to download the MPI version [You will also need MPICH2 1.0.8 (http://www.mcs.anl.gov/research/projects/mpich2/)]. Other dependencies are:

- pythonxy (http://www.pythonxy.com) or
  - Python 2.7 (http://python.org)
  - Numpy 1.3.0 (http://sourceforge.net/projects/numpy/files/NumPy)
  - SymPy 0.7.1 (http://sympy.org)
- Optional:
  - gmsh 2.4.0 (required to use pycad, must be in your PATH) - http://www.geuz.org/gmsh
  - matplotlib 0.99 - http://matplotlib.sourceforge.net

Unpack the escript zip file, then:

- copy the `esys` directory to your Python 2.7 site-packages folder[1] (usually `C:\Python27\Lib\site-packages`).
- copy the `.dll` files from `esys_dlls` to a directory on your PATH. For example copy the directory to `C:\Python27\libs\esys_dlls` and add `C:\Python27\libs\esys_dlls` to your PATH.[2]

[1] Substitute the relevant Python version

[2] Failing to do so my result in the error message: "This application has failed to start because the boost_python-vc71-mt-1_33_1.dll was not found."