The **EMBOSS** Administrators Guide

David Martin
EMBnet Norway

This guide relates to **EMBOSS** 1.3.0

Contents

1 Introduction

1.1 About this document

This guide has been writen to assist system administrators and developers with the installation and configuration of **EMBOSS**. If you are reading this to find out how to do bioinformatics then you are wasting your time. You are referred instead to the Resources chapter below where there is a list of more relevant literature and web sites. Experienced users may find this document useful for configuring their own databases and customising their **EMBOSS** experience.

1.1.1 Credits

The principal author of this guide is David Martin[1] at the Norwegian EMBnet node.[2] It is however the result of a team effort. Thanks are due in particular to Alan Bleasby who answered a lot of silly questions, killed more than a few bugs, and provided clear explanations, and to the other **EMBOSS** developers at the 'Hinxton triangle'. I would also like to thank Johann Visagie for the FreeBSD information and Peter Rice for the SRS information.

1.1.2 Reproduction

1.2 What is EMBOSS?

EMBOSS is a freely available suite of bioinformatics applications and libraries. It can be downloaded via the internet, copied, customised, and passed on under the terms of the various General Public Licenses. **EMBOSS** has been developed in response to the need for a powerful, adaptable suite of software that can interface readily with many different situations and meet the need of professional bioinformaticists, particularly those needing high throughput and/or scriptable capabilities.

EMBOSS has primarily been developed by those responsible for the public extensions to the GCG package. Whilst **EMBOSS** duplicates much of EGCG it includes far better database interaction and has the benefit of freely accessible source code so novel applications can be developed rapidly and at minimal cost.

[1] damartin@hgmp.mrc.ac.uk
[2] http://www.no.embnet.org

EMBOSS is currently only available for Unix/Linux systems but it has been known to compile and run on Windows NT. This document will only consider the UNIX version and will assume the reader has some familiarity with UNIX system administration.

1.2.1 Where do I get it?

EMBOSS is available for download from the primary site at the UK EMBnet node by anonymous ftp.[3] This directory contains the **EMBOSS** package and several associated packages (collectively known as EMBASSY) that are distributed with **EMBOSS**. Download these to a suitable location. Documentation is available on the WWW at the **EMBOSS** web site.[4]

FreeBSD distributions from 4.2 onwards now include **EMBOSS** as an optional package maintained by Johann Visagie.[5] Please see section 2.5 for more information on installation on FreeBSD.

[3] `ftp.uk.embnet.org/pub/EMBOSS/`
[4] `http://www.uk.embnet.org/Software/EMBOSS`
[5] `johann@egenetics.com`

2 Installation

2.1 Retrieving EMBOSS by anonymous ftp

2.1.1 Interactive FTP

Change directory to the location in which you wish to download the **EMBOSS** source code. In this example we will download the source to /packages/EMBOSS. Then start your ftp client and poit it to ftp.uk.embnet.org.

```
% ftp ftp.uk.embnet.org
Connected to tantalum.hgmp.mrc.ac.uk.
220 tantalum FTP server ready.
Name (ftp.uk.embnet.org:admmast):
```

We are using anonymous FTP so type the username anonymous.

```
Name (ftp.uk.embnet.org:admmast): anonymous
331 Guest login ok, send your complete e-mail address as password.
Password:
```

Enter your email address here as the password for user *anonymous*.

```
Password:
230-######################################################################
230-
230-  Welcome to the UK HGMP Resource Centre anonymous ftp service
230-
230-       Please contact support@hgmp.mrc.ac.uk regarding
230-                any problems with this service
230-
230-######################################################################
230-
230-Please read the file README
230-  it was last modified on Wed Aug 13 15:40:25 1997 - 1093 days ago
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Move to the **EMBOSS** directory and list the files. The output has been truncated a little to save space.

```
ftp> cd /pub/EMBOSS
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 5264
...  1871128 Aug  9 22:25 EMBOSS-1.1.0.tar.gz
...   133557 Aug  9 22:31 MSE-0.0.4.tar.gz
...   571095 Aug  9 22:31 PHYLIP-3.573c.tar.gz
```

```
...    81586 Aug  9 22:31 TOPO-0.1.tar.gz
226 Transfer complete.
ftp>
```

Now download the source files

```
ftp> get EMBOSS-1.1.0.tar.gz
200 PORT command successful.
150 Opening BINARY mode data connection for EMBOSS-1.1.0.tar.gz
(1871128 bytes).
...
ftp>
```

And repeat for each file. Or use `mget *gz` to download all the files at once. Exit your ftp session with the command `bye`.

2.1.2 FTP using WGET

The program WGET can be used to download a remote directory noninteractively. More details on WGET can be obtained from the Free Software Foundation.[1] Assuming you have WGET installed, use the following command which generates a lot of output on the screen:

```
% wget -m 'ftp://ftp.uk.embnet.org/pub/EMBOSS'
--15:04:41--  ftp://ftp.uk.embnet.org:21/pub/EMBOSS
           => `ftp.uk.embnet.org/pub/.listing'
Connecting to ftp.uk.embnet.org:21... connected!
Logging in as anonymous ... Logged in!
==> TYPE I ... done.  ==> CWD pub ... done.
==> PORT ... done.    ==> LIST ... done.

...
many pages truncated
...

FINISHED --15:04:55--
Downloaded: 2,657,366 bytes in 4 files
```

A new directory *f*tp.uk.embnet.org has been created and EMBOSS can be found at *f*tp.uk.embnet.org/pub/EMBOSS. You may wish to create a symbolic link to this from your /packages directory for convenience.

2.2 Unpacking

You will have downloaded the **EMBOSS** and EMBASSY packages to a suitable directory. For this example we will assume you have downloaded them to /packages so you should now have the following files (or similar) and maybe more packages in EMBASSY.

```
% ls
EMBOSS-1.0.0.tar.gz
PHYLIP-3.573c.tar.gz
MSE-0.0.4.tar.gz
TOPO-0.1.tar.gz
```

First unpack the **EMBOSS** distribution

```
% gunzip EMBOSS-1.0.0.tar.gz
% tar xf EMBOSS-1.0.0.tar
```

[1] http://www.gnu.org

This will create a new directory, *E*MBOSS-1.0.0 or similar. You may wish to use `tar xpf` for unpacking **EMBOSS**.

Enter the **EMBOSS** directory

```
% cd EMBOSS-1.0.0
```

create a directory for the EMBASSY packages

```
% mkdir embassy
```

Now copy the EMBASSY packages to the EMBASSY directory

```
% cp ../MSE-0.0.4.tar.gz PHYLIP-3.573c.tar.gz \
  TOPO-0.1.tar.gz embassy
```

Go into the EMBASSY directory and unpack those packages.

```
% cd embassy

% gunzip MSE-0.0.4.tar.gz
% tar xf MSE-0.0.4.tar
```

and so on for each EMBASSY package.

Go back up one directory to the main **EMBOSS** package directory and prepare to start compilation.

2.3 Compilation

Building **EMBOSS** is easy. It follows the usual GNU style of `./configure`, `make`, `make install`. We'll take these steps one at a time.

2.3.1 Configure

To accept the default configuration, just type `./configure` and let **EMBOSS** get on with it. You may however want to make some changes to the configuration parameters according to your local policy. This section will not cover all the possibilities, just some of the more common. The configuration script will attempt to find the neccessary components in your system to determine how to successfully build **EMBOSS**. It typically expects the GNU C compiler (gcc) and several standard libraries that should already be part of your Unix/Linux system. **EMBOSS** should configure, compile and run on most modern Linux distributions straight out of the box.

Installation directory

You need to have write permission on the directory in which you eventually wish to install **EMBOSS**. You may also wish to put it somewhere else other than the standard location of /usr/local/emboss.

The installation directory is controlled by the `--prefix` argument. In my case I have all my applications owned by a non-priviledged user and installed in a package specific directory under /site/prog

```
% ./configure --prefix=/site/prog/emboss
```

will install **EMBOSS** under /site/prog/emboss. The binaries will be installed in /site/prog/emboss/bin with shared libraries installed in /site/prog/emboss/lib. System wide data are installed in /site/prog/emboss/data, and the configuration files (ACD files) for the applications will be installed in /site/prog/emboss/share in directories corresponding to the package name. Documentation is installed in /site/prog/emboss/doc.

The individual directories for installation can be modified with other configuration commands but this is usually not neccessary. Run `./configure --help` to get more information on the directories that can be changed and other configuration options.

Run `./configure` with the options you wish to use. This may take a short time as various messages scroll up the screen.

Depending on your system you may need to explicitly configure the graphics. Please see the section 'Configuring **EMBOSS** graphics' below.

All should be well with this and configure should exit with a message like this:

```
... much output skipped

creating ./config.status
creating plplot/Makefile
creating plplot/lib/Makefile
creating nucleus/Makefile
creating ajax/Makefile
creating emboss/Makefile
creating emboss/acd/Makefile
creating test/Makefile
creating test/data/Makefile
creating test/embl/Makefile
creating test/pir/Makefile
creating test/swiss/Makefile
creating test/swnew/Makefile
creating test/wormpep/Makefile
creating emboss/data/Makefile
creating emboss/data/CODONS/Makefile
creating emboss/data/REBASE/Makefile
creating emboss/data/PRINTS/Makefile
creating emboss/data/PROSITE/Makefile
creating Makefile
```

Configuration is now complete.

Configuring EMBOSS graphics

The PLPLOT library can produce output to many devices but requires certain libraries that are NOT distributed with **EMBOSS**

To get X-windows based output you must have X installed else PLplot will not build the required driver. You may need to specify the location of your X-windows library with the configuration options: `--x-includes=DIR` (X include files are in DIR) `--x-libraries=DIR` (X library files are in DIR)

To explicitly configure PLPLOT without X-windows, use `--without-x`.

To get PLPLOT to produce PNG images you will need to have the *z*, *p*ng and *g*d librarys installed. *g*d version >= 1.6.3 must be used as the older versions support GIF which is NOT supported in later versions. If for some reason you do not have the required libraries and your system support group will not update them for the system then install all three latest versions (*z*,*g*d,*p*ng) to a new directory and then add this new directory to your configure line for **EMBOSS** — `./configure --with-pngdriver=my_dir` where the *z*, *p*ng and *g*d libraries were each installed using `./configure --prefix=my_dir`

You can explicitly tell **EMBOSS** to not include PNG support with `--without-pngdriver` You can tell if `./configure` has found a suitable PNG library by watching for something like the following when running `./configure`:

```
checking if png driver is wanted... yes
checking for inflateEnd in -lz... (cached) yes
checking for png_destroy_read_struct in -lpng... (cached) yes
checking for gdImageCreateFromPng in -lgd... (cached) yes
```

This means that the configuration script has located the PNG libraries on your system. If you see a message indicating that `./configure` could not find the libraries or that the version of *gd* was too old then you should install the latest versions of the libraries yourself and rerun configure with the correct `--with-pngdriver` value.

2.3.2 Building EMBOSS

Building **EMBOSS** is a matter of typing '`make`' and going to find something else to do for the next ten minutes to half an hour depending on the speed of your system. **EMBOSS** will first build the shared libraries (*P*L_PLOT, *A*JAX, and *N*UCLEUS) and then build the applications.

You may see plenty of warnings (especially on SGI systems) complaining about libraries not being used to resolve any symbols. These can be safely ignored.

If all goes according to plan you should have built **EMBOSS** successfully. If not you will have to try to work out why the build failed. If you can't work it out yourself, send an email describing the problem to emboss-bug@embnet.org with a copy of the config.status and config.cache files from your **EMBOSS** directory. (These will tell the developers what state your system was in whaen compilation failed).

Assuming that compilation was successful, you can[2] now type '`make install`'. After a few minutes and many pagefuls of messages, **EMBOSS** should be installed where you specified in the `--prefix` option (or in the default location of /usr/local/emboss if `--prefix` was not specified).

2.3.3 Post compilation setup

You will now need to make a few adjustments to your enviromnent to ensure that **EMBOSS** runs smoothly. EMBOSS looks for certain environment variables to determine where the libraries and data are found. These instructions assumed you installed **EMBOSS** in /site/prog/emboss. Adjust these instructions to suit your installation. Insert the following lines at the end of /etc/cshrc (or ~/.cshrc for a personal installation)

```
setenv EMBOSS_DATA /site/prog/emboss/data
setenv PLPLOT_LIB /site/prog/emboss/lib
set path=( /site/prog/emboss/bin ${path} )
```

Or for bash/ksh/sh users, insert the following at the end of /etc/profile or ~/.bashrc

```
EMBOSS_DATA=/site/prog/emboss/data
PLPLOT_LIB=/site/prog/emboss/lib
PATH=/site/prog/emboss/bin:$PATH
export EMBOSS_DATA PLPLOT_LIB PATH
```

EMBOSS should now be ready for use.

2.3.4 Testing your EMBOSS installation

You can test your **EMBOSS** installation by trying the program '`wossname`'

```
% wossname -auto |more
```

This should give a long list of programs that are available. Press space to page down through the list. This is just the **EMBOSS** programs and doesn't include any of the EMBASSY programs.

[2] You don't have to do this. You can leave **EMBOSS** where it is and just add the path to the emboss directory to your `PATH`

2.4 Installing EMBASSY

As well as the base libraries and standard EMBOSS distribution, various extra packages (EMBASSY) are distributed with EMBOSS.

To install an EMBASSY package, go to the relevant directory. For example to install PHYLIP (which was unpacked into /packages/EMBOSS-1.0.0/embassy/PHYLIP-3.573c earlier) go to the relevant directory.

```
% cd  /packages/EMBOSS-1.0.0/embassy/PHYLIP-3.573c
% ./configure --prefix=/site/prog/emboss
... output not shown
% make
... output not shown
% make install
... output not shown
```

Note. You MUST use the same arguments for configure that you used for the installation of the main **EMBOSS** package.

Repeat as neccessary for the other EMBASSY packages.

You should now find that running WOSSNAME as before lists the EMBASSY programs.

2.5 Installing EMBOSS in package format

EMBOSS can be installed on almost all Unix/Linux operating systems using the instructions above, but the package format can be far more convenient. A package is a precompiled set of binaries with installation instructions that can be set up on your system with a minimum of work. In some cases the package will check for the correct libraries and install those as neccessary.

Brief instructions are given here for the packages of which I am aware. These are maintained separately from the main source tree and may also install some files in operating system standard locations instead of the locations used by the 'raw' **EMBOSS** distribution. Please read the more detailed instructions that accompany each package.

2.5.1 Installing EMBOSS on FreeBSD

A FreeBSD **EMBOSS** package has been created by Johann Visagie[3] of Electric Genetics. This will be distributed on the installation CD's and through the normal distribution channels from FreeBSD version 4.2 onwards.

For the FreeBSD user with an up-to-date ports tree[4], installing **EMBOSS** reduces to two simple commands (as root):

```
# cd /usr/ports/biology/emboss
# make install
```

The FreeBSD specific parts of the port are that emboss.default is included with the other configuration files under /usr/local/etc as emboss.default.sample, and the **EMBOSS** documentation is installed in /usr/local/share/doc/EMBOSS instead of the default location. For further information on installation under FreeBSD you are referred to the Resources chapter.

[3] johann@egenetics.com

[4] FreeBSD users can update their ports tree through a variety of mechanisms. Please see the FreeBSD specific guide produced by Johann for more information

3 Configuration

EMBOSS can be readily configured to match your requirements. In a standard installation of **EMBOSS** the configuration directives are looked for in the following locations and in the following search order:

1. A file emboss.default in the share/EMBOSS subdirectory of your **EMBOSS** installation.[1][2]
2. A file .embossrc in the directory specified by the EMBOSSRC environment variable.
3. A file .embossrc in the users home directory.
4. A file .embossrc in the current directory.

emboss.default and .embossrc are plain text files that can readily be edited to suit.[3] Redefinitions of configuration parameters will override those previously defined. In the descriptions that follow only .embossrc will be mentioned but all directives can be placed in emboss.default for site wide configuration.

Several aspects of **EMBOSS** can be defined. These are:

- **EMBOSS** environment variables
- **EMBOSS** databases
- Default behaviour of **EMBOSS** programs

Databases are by far the most complex of these.

EMBOSS will ignore blank lines in the emboss.default and .embossrc files. It will also ignore any lines beginning with # or ! allowing comments to illuminate the declarations in the file.

3.1 EMBOSS environment variables

EMBOSS environment variables are set with an 'env' or a 'set' declaration. 'env' and 'set' are interchangeable. The most important environment variable is the location of the .acd files that describe each program.

```
set emboss_acdroot /site/prog/emboss/share/EMBOSS/acd
```

Environment variables are useful for simplifying maintenance of your .embossrc. For example you may want to specify the location of your databases as an environment variable. Then if you move the databases you only have to update one line in the configuration file.

```
set emboss_database_dir /data/databases/flatfiles
```

This would then be referred to later in .embossrcas

```
$emboss_database_dir/embl
```

for the directory /data/databases/flatfiles/embl

[1] This location may have been redefined in installations of **EMBOSS** that have been packaged for specific operating systems. See section 2.5 for further information on OS specific package installations.

[2] **EMBOSS** will also look in the emboss directory under the **EMBOSS** source distribution for emboss.default if it is not found under the installation directory

[3] A sample emboss.default is located in emboss/acd under the source distribution.

3.1.1 Configuring EMBOSS differently for different groups of users

It may be the case that you have users who need to share a specific setup. Maybe to have access to different sets of databases or need to use a different data directory.

It can be time consuming and error prone to maintain a series of individual .embossrc files or to cause users to have to work in the same directory or to copy an .embossrc to each directory they wish to work in. The environment variable `EMBOSSRC` can be set to point to an arbitrary directory containing an .embossrc which can then be used to give workgroup specific configuration. Each user then only needs to set `EMBOSSRC` in their .cshrc (CSH) or .profile (BASH) to get the workgroup specific setup.

In my case I have several groups of researchers for whom I maintain biological sequence databases. These databases have been made available under restrictive licenses so that I cannot allow researchers outside the groups to access the databases. Using `$EMBOSSRC` I can set up a common configuration for the members of each group by defining the databases in the *$*EMBOSSRC/.embossrc file.

3.2 Databases

3.2.1 Database access modes

EMBOSS offers three modes for accessing databases:

Single: EMBOSS retrieves a single sequence indexed by ID or accession number.

Query: EMBOSS retrieves a set of sequences corresponding to a wildcard query.

All: EMBOSS returns all the sequences in the database in no particular order

Each database definition can configure one or many of these modes for database access.

Typically **EMBOSS** uses variations on the EMBLCD system of database indexing to provide rapid access in single and query modes to flat file databases. The EMBLCD method is implemented in a variety of ways depending on the original format of your database. The EMBLCD method assumes that you have one or both of ID and accession number in each record and that they are unique for the whole database index. **EMBOSS** also provides methods for retrieving sequences via the WWW and a specific method for interaction with SRS[4]. For other non flatfile databases or flat file databases in formats not currently supported by **EMBOSS** you will have to configure an external application to retrieve sequences.

3.2.2 General database configuration.

Each database is configured using a DB declaration.

The generalised form is

```
DB databasename [

Configuration options

]
```

The configuration options are tag/value pairs and must contain at least a description of the access method (using `method:` or one or more of `methodsingle:`, `methodquery:` and `methodall:`) and a description of the original format of the sequences (using `format:`). In addition to these tags there will be other tags that are needed for particular methods and other tags that are optional.

[4] `http://www.lionbio.co.uk`

Database access methods

The scope of each method is:

Single mode - s Supports retrieval of a single sequence.

Query mode - q Supports retrieval of a subset of the sequences in the database specified using a wild card query in the USA[5]

All mode - a Supports retrieval of all sequences in the database as a stream of data.

An example entry for each access method is shown.

DIRECT Modes: a
Direct accesses the flatfile directly. It returns all the database entries, one after the other. It assumes no indexing.

```
DB mydb [
#required parameters
   method: direct
   format: fasta
   dir: $emboss_db_dir/mydb
   file: *.dat
#optional parameters
   type: N
   release: 63.0
   comment: "My own database with no indices"
   exclude: "est*.dat"
]
```

SRS Modes: a q s
SRS returns entries from a local installation of SRS using the -e switch to getz to return entries in the original format.

```
DB mydb [
#required parameters
   method: srs
   format: embl
   app: getz
#optional parameters
   dbalias: embl
   type: N
   comment: 'My srs indexed database'
   release: '63.0'
]
```

SRSFASTA Modes: a q s
As SRS but returns the sequences in FASTA format.

[5]Please see the **EMBOSS** documentation for description of Uniform Sequence Address format

URL Modes: `s`
URL uses a defined web server to retreive a specific entry. EMBOSS may fail if the HTML causes complications with parsing of the entry.

```
DB mydb [
# required parameters
    method: url
    format: genbank
    url: "http://www.infobiogen.fr/srs5bin/cgi-bin/wgetz?-e+[genbank-id:%s]"
#optional parameters
    type: N
    comment: "Genbank by ID from InfoBiogen"
]
```

The %s in the URL string indicates where **EMBOSS** will insert the identifier portion of the USA.

EMBLCD Modes: `a q s`
EMBLCD uses EMBLCD indices created with DBIFLAT or DBIFASTA to access flatfile databases in the original format.

```
DB mydb [
   method: emblcd
   format: embl
   dir: $emboss_db_dir/embl
   file: *.dat
#optional parameters
   type: N
   release: 63.0
   comment: "my comment"
   exclude: est*.dat
   indexdir: $emboss_db_dir/indices
]
```

This method can require careful setup. Please read the more specific descriptions below.

GCG Modes: `a q s`
GCG uses EMBLCD indices created with DBIGCG to access databases in GCG format. This method uses the .seq and .header files created by the GCG suite of programs.

```
DB mygcgdb [
   method: gcg
   format: embl
   dir: $emboss_db_dir/gcgembl
   file: *.seq
#optional parameters
   type: N
   release: 63.0
   comment: "my comment"
   exclude: est*
   indexdir: $emboss_db_dir/indices
]
```

BLAST Modes: `a q s`
BLAST uses EMBLCD indices created with DBIBLAST to access databases in BLAST format.

EXTERNAL Modes: `a q s`
EXTERNAL uses an external application to retrieve sequences. The ID is passed as an argument to the application, either replacing %s in the command string (if present) or as an additional arguement (if there is no %s). EXTERNAL expects the application to return the sequence on STDOUT.

```
DB mydb [
#required parameters
    method: app
    format: fasta
    app: "getfromdb mydb"
#optional parameters
    type: P
    comment: "my own protein database with a custom retrieval program"
]
```

APP Modes: `a q s`
APP is the same as EXTERNAL.

For any given `method:` declaration, **EMBOSS** will use that method for those access modes supported by the method.

If you wish to specify which access mode (all, query or single) should be handled by which database retrieval method then the `methodsingle:`, `methodquery:` and `methodall:` declarations should be used instead of `method:`

```
DB mydb [
methodsingle: app
format: fasta
app: "customapp myproteindb"
methodall: direct
dir: $emboss_db_dir/myproteindb
file: myproteindb.dat
type: P
comment: "single and all access for myproteindb"
]
```

3.2.3 Indexing and configuring flatfile databases

Flatfile databases are plain text files in a defined format such as those released by EMBL, Swissprot and so on. The **EMBOSS** program DBIFLAT is used to generate EMBLCD indices that can be used for all types of database access. DBIFLAT can process databases in EMBL, SWISSPROT and GENBANK format. Pseudo EMBL format databases which do not have unique ID and AC entries may cause DBIFLAT to do mysterious things and should be avoided.

DBIFLAT (and the EMBLCD access method) requires the databases to be uncompressed. The examples given here will not probe the deeper secrets of DBIFLAT (for which the reader is referred to the documentation, or failing that the source code) but will show a typical installation for a common database.

We assume that **EMBOSS** has been installed and works. This can be tested with the command `wossname -auto` which should list all the programs available.

In this example we will index and configure the EMBL database for use with **EMBOSS**.

First download and unpack the EMBL database. This will require a considerable amount of disk space. If you do not have sufficient space available then just download a subset of the database.

Use `cd` to move the directory in which you have unpacked EMBL. This should look something like this when you run `ls`:

```
% ls
est_fun.dat
```

```
est_hum1.dat
est_hum10.dat
.
Output truncated
.
syn.dat
unc.dat
vrl.dat
vrt.dat
```

Run DBIFLAT to create the EMBLCD indices.

```
% dbiflat

Index a flat file database
      EMBL : EMBL
     SWISS : Swiss-Prot, SpTrEMBL, TrEMBLnew
        GB : Genbank, DDBJ
Entry format [SWISS]: EMBL
Database name: embl
Database directory [.]:
Wildcard database filename [*.dat]:
Release number [0.0]: 63.0
Index date [00/00/00]: 31/07/00
```

DBIFLAT should happily chug away for some considerable time (up to a few hours depending on the speed of your machine) and will generate (eventually) the following index files:

```
% ls
acnum.hit
acnum.trg
division.lkp
entrynam.idx
```

Now we create an entry in the **EMBOSS** configuration files to acces sthe database. It is probably a good idea to try new database definitions in your local configuration file first.

Put the following entry in your .embossrc

```
DB embl [
   type: N
   method: emblcd
   format: embl
   dir: $emboss_db_dir/embl
   file: "*.dat"
   release: "63.0"
   comment: "EMBL release 63.0"
]
```

you will have needed to predefine `$emboss_db_dir` using a directive such as

```
set emboss_db_dir /path_to_databases
```

somewhere in your emboss.default or .embossrc.

Save .embossrc and try SHOWDB. You should see a line that looks like:

```
% showdb
.. output deleted
embl          N    OK  OK  OK  EMBL release 63.0
.. output deleted
```

3.2.4 Fine tuning the installation:

It is probably a good idea to set up subsections of the database so that end users can search just the regions they wish to search. This section applies to all access methods that use EMBLCD style indexes and probably to others as well.

Files can be included with the declaration `file:` or excluded with the declaration `exclude:`

In order to just take the EST files in our EMBL database try the following:

```
DB emblest [
   type: N
   method: emblcd
   format: embl
   dir: $emboss_db_dir/embl
   file: "est*.dat"
   release: "63.0"
   comment: "EMBL release 63.0"
]
```

Files can also be given as a space separated list enclosed in quotes. For example to set up a database of all mamallian sequences (except genomes) try the following:

```
DB emblallmam [
   type: N
   method: emblcd
   format: embl
   dir: $emboss_db_dir/embl
   file: "rod*.dat hum*.dat mam*.dat"
   release: "63.0"
   comment: "EMBL release 63.0"
]
```

As you can see from these two examples, the `file:` tag takes a space delimited list of filenames enclosed in quotes that can contain normal wildcard (`?*`) characters.

It can be quite tedious to set up a long list of sequences to search. In many cases you can use the `exclude:` tag to make things easier.

```
DB emblnoest [
   type: N
   method: emblcd
   format: embl
   dir: $emboss_db_dir/embl
   file: "*.dat"
   exclude: "est*.dat"
   release: "63.0"
   comment: "EMBL release 63.0"
]
```

This configures the emblnoest database to contain all of EMBL except the EST's.

3.2.5 Indexing and configuring GCG format databases

EMBOSS can access GCG formatted databases, thus avoiding having multiple copies of the same databases in different formats for those who still use GCG alongside the flatfiles. **EMBOSS** creates EMBLCD like indices for the GCG format databases using the program DBIGCG. This runs in much the same way as DBIFLAT. You will need the GCG format .seq and .header files in order to create an EMBLCD indexed database.

Move to the GCG database directory containing your data and run DBIGCG

```
Index a GCG formatted database
      EMBL : EMBL
     SWISS : Swiss-Prot, SpTrEMBL, TrEMBLnew
        GB : Genbank, DDBJ
       PIR : NBRF
Entry format [EMBL]:
Database name: embl
Database directory [.]:
Wildcard database filename [*.seq]:
Release number [0.0]: 63.0
Index date [00/00/00]: 31/07/00
```

The program will chug along for a while and will then generate the EMBLCD index files for the GCG format database.

When DBIGCG prompts for the entry format (`Entry format [EMBL]:`) you should enter the original database format before you ran EMBLTOGCG or similar to generate the GCG databases.

The following entry should be put in your .embossrc

```
DB gcgembl [
   type: N
   method: gcg
   format: embl
   dir: $emboss_db_dir/embl
   file: "*.dat"
   release: "63.0"
   comment: "EMBL release 63.0"
]
```

SHOWDB should show your newly configured database.

You can configure subsets of the databases in the same way as for the original format databases, described in section 3.2.4 above.

3.2.6 Indexing and configuring BLAST databases

BLAST format databases are generated for efficient homology searching using the BLAST programs. It can be convenient to avoid redundant copies of databases so **EMBOSS** provides a mechanism for accessing these databases.

BLAST format databases are those generated using the tools distributed with NCBI-BLAST or with WU-BLAST. For indexing of one BLAST database, move to the directory containing your BLAST format databases and run DBIBLAST

```
Index a BLAST database
Database name: blastsw
Database directory [.]:
database base filename [blastsw]:
Release number [0.0]:
Index date [00/00/00]:
         N : nucleic
         P : protein
         ? : unknown
Sequence type [unknown]: p
         1 : wublast and setdb/pressdb
         2 : formatdb
         0 : unknown
Blast index version [unknown]: 2
```

The program will chug along for a while and will then generate the EMBLCD index files for the BLAST format database.

The following entry (or one like it that is more appropriate to your particular installation) should be put in your .embossrc

```
DB blastsw [
   type: P
   method: blast
   format: ncbi
   dir: $emboss_db_dir/blastsw
   file: "blastsw"
   release: "38.9"
   comment: "BLAST format Swissprot"
]
```

SHOWDB should show your newly configured database.

Because of the way BLAST works, many sites may group their BLAST databases in the same directory. You can index these *in situ* with DBIBLAST but this may require some extra steps if your databases are not of the same type as generation of subsequent index files will overwrite those that already exist. To avoid overwriting of index files you can index many databases with one set of index files, or you can use the `indexdir` options to place the indices in a different directory.

There are two requirements for indexing several databases together in one index. The first is that the databases are the same type (protein/nucleic acid) and generated with the same tool (pressdb or formatdb); the second is that all the ID and accession numbers in the combined databases are unique.

Run DBIBLAST as before but specify all the databases you wish to be included when prompted for the database filename.

```
Index a BLAST database
Database name: alldbs
Database directory [.]:
database base filename [alldbs]: dbone dbtwo dbthree dbfour
Release number [0.0]:
Index date [00/00/00]:
         N : nucleic
         P : protein
         ? : unknown
Sequence type [unknown]: p
         1 : wublast and setdb/pressdb
         2 : formatdb
         0 : unknown
Blast index version [unknown]: 2
```

These can then be configured as described in section 3.2.4 above by using the '`file:`' and '`exclude:`' tags as appropriate.[6]

When you have databases of different types, generated with different programs or where the ID/accession numbers are duplicated between databases the preferred strategy is probably to keep the source data for the individual databases in separate directories and index them there.[7]

[6] There is one difference to the standard EMBLCD access method in that the database indexes will not allow the generation of exclusive subsections of the combined database. If an ID or accession number is specified that is present in the index then the sequence will be returned irrespective of which database it is in.

[7] Keeping one directory with symbolic links for your BLAST installation will ensure that BLAST continues to function correctly if you set BLASTDB to point to the directory containing the symbolic links. The EMBOSS indices can be placed wherever you wish as long as you remember to run DBIBLAST with the appropriate options and put an appropriate `indexdir` tag in the DB configuration in your /.embossrc

Alternatively you can place the index files in a separate directory. This requires that you run DBIBLAST with the `-indexdirectory` option and set the `indexdir:` tag in the database configuration to point to the correct database. The example below illustrates database configuration using the `indexdir` options.

```
% dbiblast -indexdir=/databases/indices/mydb
Index a BLAST database
Database name: mydb
Database directory [.]:
database base filename [mydb]:
Release number [0.0]:
Index date [00/00/00]:
         N : nucleic
         P : protein
         ? : unknown
Sequence type [unknown]: p
         1 : wublast and setdb/pressdb
         2 : formatdb
         0 : unknown
Blast index version [unknown]: 2
```

The corresponding entry in ~/.embossrc (or emboss.default) would look like:

```
DB mydb [
   type: P
   method: blast
   format: ncbi
   dir: $emboss_db_dir/blastsw
   indexdir: /databases/indices/mydb
   file: mydb
   release: "1.0"
   comment: "My BLAST DB with an index in a different directory"
]
```

Again, multiple indices cannot coexist in the same directory so care should be taken when using the `indexdir` options that an existing dataabse index is not overwritten.

3.2.7 Indexing and configuring FASTA databases

The FASTA specifications just define the sequence file as a header line that begins with > and subsequent lines containing the sequence. The header line can be present in an almost infinite number of formats, several of which can be processed by **EMBOSS**. **EMBOSS** attempts to determine the accession number and/or ID for each sequence. For indexing purposes there is no semantic difference between an accession number and an ID. In the real world, acession numbers are immutable, ie. they do not change with subsequent releases of the dataabse, but ID's may change. In any case IDs and accession numbers are unique, and that is all that matters for database indexing **EMBOSS**.

The program used to process FASTA format databases is DBIFASTA. It can recognise the following header line formats:

simple	`>id ...`
idacc	`>id accno ...`
dbid	`>db id ...`[8]
gcgid	`>db:id ...`[8]
gbidacc	`>db:id acc ...`[8]
gbid	`>db:id ...`[8]
ncbi	`>...[|accno]|id ...`[9]

Other header formats will not be recognised by DBIFASTA and will cause indexing and/or database lookup to fail. If you have a different header format that DBIFASTA cannot yet handle you have two options:

1. (The preferred option) Get a C programmer to modify the source code for DBIFASTA and recompile. If you are a community spirited person you will also contribute these changes to the main **EMBOSS** source tree. (email emboss-dev@embnet.org for more information on contributing changes to the **EMBOSS** source code and/or read the **EMBOSS** developers documentation)

2. (The quick hack) Write a custom script (using e.g. BioPerl[10]) to access your database and use `method: external` to configure it. This is less desirable as you may be limited in the access modes you can use.

To index a FASTA format database, run DBIFASTA.

```
% dbifasta
Index a fasta database
    simple : >ID
     idacc : >ID ACC
     gcgid : >db:ID
  gcgidacc : >db:ID ACC
      ncbi : >blah|...[|ACC]|ID
ID line format [idacc]:
Database name: mydb
Database directory [.]:
Wildcard database filename [*.dat]: mydb.fasta
Release number [0.0]:
Index date [00/00/00]:
```

DBIFASTA will chug along for a little while and will produce the index files. You can use the same `indexdir` options as for DBIFLAT,DBIGCG and DBIBLAST to place the indices in a different directory.

Place the following entry in your .embossrc

```
DB mydb [
        type: P
        method: emblcd
        format: fasta
        dir: $emboss_db_dir/mydb
file: mydb.fasta
        comment: "My database"
]
```

`format:` should be `fasta`, `ncbi` or `dbid`, possibly not the format you specified when running DBIFASTA. The same `file:` and `include:` tags can be used as for the other database indexing programs.

[8] *db* is one word

[9] The ID is always taken to be the characters after the last bar (|). The previous field is also indexed but ONLY if it looks like an accession number (e.g. AC00001).

[10] `http://www.bioperl.org`

3.2.8 Configuring EMBOSS to use SRS for database lookup.

`method: srs` is really a special case of `method: external` with some additional features.

SRS is a powerful database querying system that can cross reference between different databases, launch applications and so on. SRS can be run either through a web interface (see the description of the URL method above for an example) or via the command line program GETZ. Indexing and configuring databases for SRS is outside the scope of this document which will describe how to connect to preconfigured and indexed SRS databases.[11] If GETZ is already in your `PATH` environment variable then insert the following (or similar) in your .embossrc:

```
DB emblgetz [
    type: N
    method: srs
    release: "63"
    format: embl
    comment: 'EMBL using getz'
    dbalias: embl
    app: getz
]
```

This will provide access to the SRS database 'embl' as `emblgetz:acc`. If the SRS database has a different name to the **EMBOSS** database (as is the case here) then the `dbalias:` tag should be used to access the correct SRS database.

This configuration can be extremely slow for the all access mode. It is probably a better idea to set up the database as follows:

```
DB emblgetz [
    type: N
    methodquery: srs
    release: "63"
    format: embl
    comment: 'EMBL using getz'
    dbalias: embl
    app: getz
    methodall: direct
    file: "*.dat"
    dir: $emboss_db_dir/embl
]
```

which will use `method: srs` for the `query` access mode but will use `method: direct` for the `all` access mode, thus speeding up reading of the whole database.

The SRSFASTA access method is identical to the normal SRS method except that it returns the sequence in FASTA format and so does not need a `format:` tag.

3.2.9 Indexing and configuring other databases

Many institutions may have local databases set up in their own Laboratory Information Management System. **EMBOSS** provides a simple mechanism for interfacing with such systems.

As long as a program is available that can be called noninteractively and returns the specified sequence on standard output, **EMBOSS** can interface with it. Use method: app or external (the two are equivalent) and app: "program command". The ID given in the USA will be appended to the command used to run the program. It is probably best to specify the methods available using the method subsets, methodall:, methodquery: and methodsingle: rather than using the generic method: tag.

[11] For information on configuring and indexing SRS databases please look at the SRS administrators guide www/doc/srsadmin.pdf in your SRS 6 installation

3.3 Other data

EMBOSS can be integrated with some common biological databases. These are described in this section.

3.3.1 REBASE

Rebase is the restriction enzyme database maintained by New England Biolabs. It is needed for programs such as remap and restrict.

The latest version of Rebase can be obtained by anonymous FTP.[12] **EMBOSS** needs the *withrefm* file. The data is extracted for **EMBOSS** with the program REBASEEXTRACT.

If you installed **EMBOSS** with the –prefix option you may need to create the REBASE directory under the **EMBOSS** data directory (/site/prog/emboss/data in this example) This directory only needs creating once.

```
% mkdir /site/prog/emboss/data/REBASE
% rebaseextract
Extract data from REBASE
Full pathname of WITHREFM: /data/rebase/withrefm.008
```

Rebase is now installed and ready to use.

3.3.2 TRANSFAC

Transfac is the transcription factor binding site database. It is available by anonymous FTP.[13] Unpacking the distribution reveals a file called site.dat. This is the one **EMBOSS** needs.

Run TFEXTRACT to extract the data from TRANSFAC.

```
% tfextract
Extract data from TRANSFAC
Full pathname of transfac SITE.DAT: /databases/transfac/site.dat
```

TFSCAN can now access the TRANSFAC database.

3.3.3 PROSITE

Prosite is a database of regular expressions that match potentially diagnostic regions for structural/functional classification of proteins. **EMBOSS** needs this database for the patmatmotifs program.

PROSITE can be obtained via anonymous FTP.[14]

You may need to create a PROSITE subdirectory under data in the **EMBOSS** installation directory.

Then run PROSEXTRACT to build the **EMBOSS** Prosite database.

```
% prosextract
Builds the PROSITE motif database for patmatmotifs to search
Enter name of prosite directory: /data/prosite
```

PROSITE is now integrated into your EMBOSS installation.

[12] ftp://ftp.ebi.ac.uk/pub/databases/rebase
[13] ftp://transfac.gbf.de/pub/transfac/ascii/
[14] ftp://ftp.ebi.ac.uk/pub/databases/prosite

3.3.4 PRINTS

Prints is a database of diagnostic patterns of blocks of sequence homology in protein families. The PRINTS database can be searched using the **EMBOSS** program PSCAN.

PRINTS can be obtained via anonymous FTP.[15] The database is made available as compressed files which should be uncompressed using GZIP before integrating them into **EMBOSS**

PRINTS is integrated with **EMBOSS** using the program PRINTSEXTRACT

```
% printsextract
Extract data from PRINTS
Input file: /data/prints/prints27_0.dat
```

The PRINTS database is now integrated with **EMBOSS**.

3.3.5 Miscellaneous data files

Other data files should be kept in the data directory under the main **EMBOSS** installation. Individual users personal data files can be kept in the current working directory, a subdirectory .embossdata of the current directory, their home directory or a subdirectory .embossdata of their home directory. **EMBOSS** will search these locations in this order and will stop as soon as it finds a matching file. If the personal directories do not contain the desired file, **EMBOSS** will search the system wide data directory, /site/prog/emboss/data in this example.

Apparently inexplicable errors when running **EMBOSS** programs may be caused by the system not using the data files one expects. The search path can be displayed in search order using the command EMBOSSDATA.

3.4 Default program settings

As with many other areas, the default behaviour of programs can be controlled by setting appropriate values in .embossrc.

All general qualifiers[16] can be specified as

```
set emboss_QUALIFIER 1
```

where `QUALIFIER` is one of the general qualifiers and the value can be `1` or `0` for true or false respectively. Setting the qualifier value to true has the effect of running every program with that qualifier set.[17] multiple qualifiers can be set and will work in the same way as if you set them when running the program. For example you can `set emboss_verbose 1` and the program will run normally, but when the program is run with the `-help` qualifier, the output will be in verbose form.

Qualifiers that can be set:

HELP Print help text.

VERBOSE Causes `-help` to print verbose text.

ACDTABLE Causes `-help` to print HTML formatted table of options.

STDOUT Causes all output to go to STDOUT as default.

ACDPRETTY Rewrites the ACD file in a nicely formatted way.

ACDLOG Enables ACD file processing to be logged. Useful for hunting bugs.

DEBUG Writes debugging output to a file. Also useful for finding bugs.

[15] ftp://ftp.ebi.ac.uk/pub/databases/prints

[16] See the **EMBOSS** Quick Guide or the web documentation (or use `wossname -help -verbose`) for an overview of general qualifiers.

[17] You can specifically unset it by using the `-noQUALIFIER` command line option

OPTIONS Enable prompting for optional parameters.

FILTER Take input from STDIN and send it to STDOUT.

AUTO Do not prompt for any options but accept the defaults if no values are given.

These qualifiers are typically used by advanced users (`-options`, `-verbose`) or by developers (`-debug -acdlog`).

Other program options that can be set are `emboss_format`, `emboss_acdroot`, and `emboss_data`. The value of `emboss_format` determines which default sequence format to use for output. for example, if you are running **EMBOSS** alongside GCG you may wish to have the following entry in your .EMBOSSRC

```
set emboss_FORMAT gcg
```

which has the effect of using GCG format by default.[18]

`emboss_acdroot /path/to/acd` can be set if you wish to use a different directory for the ACD files, and `emboss_data /path/to/data` if you wish to use a separate data directory.

3.5 Logging

Many system administrators may wish to make use of the logging facilities of **EMBOSS**. Setting the variable `emboss_logfile` in emboss.default or .embossrc allows the system to keep a log of which programs are used when and by whom.

```
set emboss_logfile /site/log/emboss.log
```

The log file structure is very simple. Three tab separated fields are stored, program name, user name, and the date and time.

```
prettyplot      joeuser         Wed Aug 02 14:29:13 2000
```

The file set in emboss_logfile should be world writable. The following command ensures logging can occur.

```
chmod +w /site/log/emboss.log
```

All settings can be overridden in a users .embossrc files by redefining the relevant variables. eg. to prevent my system usage being logged I can redefine emboss_logfile by putting the following entry in my .embossrc file.

```
set emboss_logfile /dev/null
```

This behaviour may change in the future to prevent users redefining some system settings.

[18] This can of course be overridden using the `-osformat` associated qualifier. See the **EMBOSS** Quick Guide for more information.

4 Resources

4.0.1 Programs

EMBOSS source code ftp://ftp.uk.embnet.org/pub/EMBOSS

EMBOSS Documentation http://www.uk.embnet.org/Software/EMBOSS

BLAST tools Tools for generating BLAST format databases are contained in the NCBI toolkit which can be obtained from NCBI at:

http://www.ncbi.nlm.nih.gov/

SRS software The SRS software can be obtained from Lion Bioscience.[1] This is a commercial package but at the time of writing is available free of charge to academic institutions.

WGET Various useful utilities including the WGET program are available from teh Free Software Foundation.[2]

4.0.2 Databases

Most of the databases mentioned in the text along with many others can be obtained via anonymous ftp from the European Bioinformatics Institute (EBI) at:

ftp://ftp.ebi.ac.uk/pub/databases

Please use a mirror site where possible to avoid overloading of the EBI's resources.
Other databases can be obtained from NCBI (Genbank,UniGene etc.)

4.0.3 Other Documentation

Please review the **EMBOSS** documentation available on the WWW at the URL above.

The EMBOSS Quick guide A pocket reference guide to using **EMBOSS**[3].

The EMBOSS Tutorial A tutorial to give an introduction to using **EMBOSS** for bioinformatics users.[4]

The updated ABC guide This is a series of bioinformatics practicals based predominantly on **EMBOSS**.[5]

EMBOSS-FreeBSD-HOWTO Detailed documentation on installation of **EMBOSS** on FreeBSD.[6]

4.1 Maintainance of your EMBOSS installation

EMBOSS is a rapidly evolving software packages. It is constantly being improved, new features added and 'issues' resolved. In addition there are new applications added and you probably want to make use of these.

[1] `http://www.lionbio.co.uk`
[2] `http://www.gnu.org`
[3] `ftp://ftp.no.embnet.org/pub/EMBOSS-extra/emboss-qg.ps`
[4] `http://www.hgmp.mrc.ac.uk/Registered/Option/emboss.html`
[5] `ftp://ftp.no.embnet.org/pub/ABC`
[6] `ftp://ftp.no.embnet.org/pub/EMBOSS-extra/EMBOSS-FreeBSD-HOWTO`

4.1.1 Automated installation of EMBOSS and EMBASSY

Once you have installed **EMBOSS** and got it to work you have solved the hardest part of the struggle. Updating **EMBOSS** as new releases appear[7] can be quite tedious. UNIX is designed for the lazy, so here is my lazy mans guide to always having an up to the minute **EMBOSS** installation.

The following script can be run manually (it should probably be 'sourced' rather than executed directly) or can be fired off with cron (in the early hours of the morning is a good time). It assumes you are installing **EMBOSS** outside the source directory and have write permissions to do so.

EMBOSS will update **EMBOSS** distributed files but will not alter or overwrite your own datafiles[8] or your emboss.default.

```
# This script should be sourced, not run.
# EMBOSS UPDATE.
# it assumes $packages_dir/EMBOSS is a symbolic link to
# $mirror_dir/ftp.uk.embnet.org/pub/EMBOSS
#

#site specific variables: season according to taste..

set mirror_dir=('/ftp/mirrors')
set packages_dir=('/site/newprog')
set emboss_config_options=\
('--prefix=/site/prog/emboss --with-pngdriver=/site/lib')

# Now the script proper

set oldpwd=`pwd`

cd $mirror_dir
echo 'updating EMBOSS'
if ( `wget -m 'ftp://ftp.uk.embnet.org/pub/EMBOSS' |& \
  tail -1 | awk '/^Downloaded:/{print $5}'` != "0" ) then

    cd ${packages_dir}/EMBOSS
    echo 'new EMBOSS programs found .. installing'
    set latest_emboss=`ls -t EMBOSS*|tail -1`

    cd $packages_dir
    rm -Rf EMBOSS-*
    tar zxf EMBOSS/$latest_emboss
    set emboss_dir=`ls -dt EMBOSS-*[^z]|tail -1`

#the next line is necessary on my system but may not be for yours.
    setenv LD_LIBRARYN32_PATH /site/lib

    cd $emboss_dir

# If you have any site specific changes to the source code
# that you want to include, copy them in here

    ./configure $emboss_config_options &&\
```

[7] **EMBOSS** is rebuilt nightly from CVS, tested, and, assuming it passes the compilation tests, the latest version is posted to the **EMBOSS** FTP server.

[8] Assuming of course that you haven't overwritten **EMBOSS** datafiles with your own to begin with.

```
    make && \
    make install

#Now unpack and build EMBASSY

    mkdir embassy
    cd embassy

#Unpack and build each package one at a time

    foreach embassadir ( `ls ../../EMBOSS/*gz |grep -v E
MBOSS-` )

tar zxf $embassadir
set embassadir_arch=$embassadir:t
set embassadir_root=$embassadir_arch:r

cd $embassadir_root:r
./configure  $emboss_config_options &&\
make && \
make install

cd ..
    end
else
    echo 'No new version of EMBOSS available'
endif

cd $oldpwd
```

4.1.2 Automated database updating

In the same way, scripts can be written to automatically update the biological databases. An example is given here for REBASE. As all the parameters for **EMBOSS** programs can be specified on the command line it is a trivial matter to include index generation in your nightly update scripts. The management of a bioinformatic resource is beyond the scope of this document, though **EMBOSS** goes a long way towards easing the burden of management.

Automated update of REBASE

This script will look for a new version of REBASE and install it in **EMBOSS** using REBASEEXTRACT.

```
# This script should be sourced, not run.
# REBASE UPDATE. Should be run just after the beginning of the month.
set mirrors_dir=('/ftp/mirrors')
set oldpwd=`pwd`

cd $mirrors_dir

if ( ` wget -m 'ftp://ftp.ebi.ac.uk/pub/databases/rebase/*' |& \
  tail -1 | awk '/^Downloaded:/{print $5}'` != "0" ) then
cd ftp.ebi.ac.uk/pub/databases/rebase
cp `ls -t withrefm.*.Z|head -1` withrefm.Z
uncompress withrefm.Z
rebaseextract \
```

```
  ${mirrors_dir}/ftp.ebi.ac.uk/pub/databases/rebase/withrefm
rm withrefm
endif

cd $oldpwd
```

I make no guarantees that these scripts will work correctly on your system. If it deletes all your files, spams your associates, scratches your CD's and initiates a nuclear strike on a small unpopulated pacific island it is NOT MY FAULT. It just happens to work for me.

5 Acknowledgements

The acknowledgements and credits are found at the front of this guide because no one ever reads them if they are at the back.