Building the Offline Surface Model

On this page...

Prerequisites

You should have completed: the install GRIB library and downloaded the OSM source code.

Set your environment

The OSM uses software called 'FCM' to handle the compilation of the code. FCM is distributed in the tarball downloaded from the ftp site.

Before trying to compile, make sure the FCM directory is added to your PATH environment variable so the `fcm` command can be found (if you already have fcm available on your system, we still recommend using the distributed version):

e.g. assuming osm is unpacked in $HOME/ecmwf/osm/

```
export PATH=$PATH:$HOME/ecmwf/osm/fcm/bin
```

Build configuration

At ECMWF...

The script `compile_fcm.ksh` is provided that correctly sets the compilation environment by using module load command and then compiling the model code using the command fcm. You may need to edit this script for your own account name and directories.

In the 'make' directory, change the link 'cfg' so it points to 'cfg.ecmwf' rather than 'cfg.external'.

You can ignore the rest of these instructions though they are a useful reference.

Before compiling the model, the build configuration must first be set by several environment variables:

OSM_COMP - This sets the choice of compiler. The default is 'gnu' which means the gfortran/gcc compilers will be used.

OSM_BUILD - This sets the type of build.

These environment variables directly correspond to the names of the FCM configuration files in the `make/cfg/` directory in the source. Please see this directory for the choices provided. Typically configurations are available for: GNU, Intel, Cray and PGI compilers.

Build types: OSM_BUILD sets the type of build and there will be one FCM configuration file for each type. Build types provided are:

- *opt* - Recommended optimized compile settings for OpenIFS for this compiler. Tested to provide best performance for this platform & compiler.
- *noopt* - Debugging options. No optimization set for all of the code. Suitable for use with a debugger. OpenMP is disabled.

e.g.

```
export OSM_COMP=gnu
export OSM_BUILD=opt
```

means using the gfortran compiler and the model will be compiled with full optimization compiler settings. So the FCM build system will expect to find a file:

```
make/cfg/gnu-opt.cfg
```

Setting:

```
export OSM_COMP=intel
export OSM_BUILD=noopt
```

means FCM will expect to find a file:

```
make/cfg/intel-noopt.cfg
```

suitable for building using the Intel compilers using non-optimized settings.

Building for other systems/environments

As the environment variables refer to the corresponding filename in the `make/cfg` directory it is straightforward to create a build configuration for other environments. Take a copy of an existing file and modify it as necessary. It's also possible to alter compile options by the use of additional environment variables (see below). Which approach you use depends on your personal preference. Be aware however that the optimization compiler flags in the provided configuration files are the recommended ones. It is impossible to test all combinations of compiler flags so the model may become unstable or performance may reduce if higher optimizations are tried.

Setting the location of libraries

Some environment variables need to be set for the GRIB and netCDF libraries.

For the **GRIB** library, environment variables need to be defined to set the library and the include paths in as compiler options:

- **GRIB_LIB** - Defines the GRIB library link options
  e.g. `GRIB_LIB="-L$HOME/ecmwf/eccodes/lib -leccodes-f90 -leccodes"`
- **GRIB_INCLUDE** - Defines the GRIB include path compile option
  e.g. `GRIB_INCLUDE="-I$HOME/ecmwf/eccodes/include"`

And, similarly for the **netCDF** library and include files:

- **NETCDF_LIB** - Defines the netCDF library link options:
  e.g. `NETCDF_LIB="-L/usr/local/lib -lnetcdff -lnetcdf"`
- **NETCDF_INCLUDE** - Defines the netCDF include path compile option:
  e.g. `NETCDF_INCLUDE="-I/usr/local/include"`

Or if you prefer, you can edit the values in the appropriate FCM configuration file in the make/cfg directory or make a copy of the supplied configuration file (.cfg) and use that.

Using a script

It can be useful to put the definition of the environment variables into a small script which is run before the main compilation:

Example of simple script to setup compilation environment

```
% cat setup.sh

export OSM_COMP=gnu
export OSM_BUILD=opt

export GRIB_LIB="-L/usr/lib64 -leccodes_f90 -leccodes"
export GRIB_INCLUDE="-I/usr/include"

export NETCDF_LIB="-L/usr/local/lib -lnetcdff -lnetcdf"
export NETCDF_INCLUDE="-I/usr/local/include"
```

Compiling OSM

Once the environment variables are set, OpenIFS can be compiled (make sure you have edited your PATH environment variable to add the FCM installation `/bin` directory). Assuming you are in the directory where the OpenIFS source code was unpacked:

```
cd make
fcm make -f osm.cfg
```

The command '`fcm make`' starts the compilation. The -f option specifies the location of the master configuration file for the surface model For more information about fcm command options, try: `fcm --help`.

This generates output similar to:

```
[init] make                # 2016-11-21T19:50:13Z
[init] make config-parse   # 2016-11-21T19:50:13Z
[info] config-file=........./make/osm.cfg
[info] config-file= - ......./make/cfg/gnu-opt.cfg
[done] make config-parse   # 0.0s
[init] make dest-init      # 2016-11-21T19:50:13Z
[info] dest=...../bin/gnu-opt
[info] mode=new
[done] make dest-init      # 0.0s
[init] make build          # 2016-11-21T19:50:13Z
[info] sources: total=299, analysed=299, elapsed-time=2.7s, total-time=2.6s
[info] target-tree-analysis: elapsed-time=0.3s
[info] compile   targets: modified=203, unchanged=0, failed=0, total-
time=44.2s
[info] compile+  targets: modified=128, unchanged=0, failed=0, total-
time=0.0s
[info] ext-iface targets: modified=59, unchanged=0, failed=0, total-time=0.
4s
[info] install   targets: modified=26, unchanged=0, failed=0, total-time=0.
0s
[info] link      targets: modified=8, unchanged=0, failed=0, total-time=0.
4s
[info] TOTAL     targets: modified=424, unchanged=0, failed=0, elapsed-
time=45.7s
[done] make build          # 48.4s
[done] make                # 48.4s
```

Verbose compilation

The default output of FCM is quite terse. To get more verbose output use the '-v' option. Also if you have a multicore machine you can use the -j option to specify additional processes to build in parallel.

e.g.

```
fcm make -v -j 2 -f osm.cfg
```

Successful compilation

After a successful compilation, the executables can be found in the '`bin/gnu-opt/osm/bin`' subdirectory:

```
% ls bin/gnu-opt/build/bin
adjust_forc.exe  caldtdz.exe  conv_forcing.exe  convNetcdf2Grib.exe
create_grid_info.exe  create_init_clim.exe  find_points.exe  master1s.exe
```

where **master1s.exe** is the OSM master executable. The other executables are utilities.

The 'gnu-opt' subdirectory is generated by FCM using the values of the OSM_COMP & OSM_BUILD environment variables. The `include` directory contains all the files referenced in the code by a '`#include`' statement. This includes the Fortran interface block files which are auto-generated by FCM and end in `.intfb.h`. The o directory contains all the compiled object files. The build does not place any additional files in the `src` directory (known as building '*out of source*').

Compilation problems

If you get the message:

```
fcm make -f osm.cfg
/bin/bash: fcm: not found [No such file or directory]
```

you need to add the 'bin' directory of the FCM installation to your PATH environment variable. FCM is supplied with OpenIFS and is in the directory 'fcm'.