Introduction to OpenIFS

Author: Glenn Carver
Date: 09-Jun-2014 10:50
URL: https://software.ecmwf.int/wiki/display/OIFS/Introduction+to+OpenIFS

Table of Contents

1 Introduction

OpenIFS is an ECMWF led project which provides an easy-to-use, exportable version of the IFS system in use at ECMWF for operational weather forecasting. The project aims to develop and promote research, teaching and training on numerical weather prediction (NWP) and NWP-related topics with academic and research institutions.

In this tutorial, OpenIFS has been **pre-installed** for you. If you want to learn more about how OpenIFS can be downloaded, configured and installed, please see the OpenIFS website at: http://software.ecmwf.int/oifs/.

1.1 In this tutorial..

1.1.1 What you will do:

- Explore installed OpenIFS files.
- Run T21 model forecast.
- Verify model works in serial (1 process) and parallel (2 tasks & 2 threads).
- Carry out acceptance test.
- Use the model namelist to change settings
- Explore model output data (GRIB)

1.1.2 At the end of this tutorial, you will:

- learnt about how OpenIFS is installed and organised.
- run it at T21 resolution and know how to run it serially (1 process) and in parallel with both MPI & OpenMP.
- know how to use grib tools to look at model output.

2 OpenIFS directories

In this section we:

- Set the OpenIFS environment
- Examine the OpenIFS installation

Tasks - Set OpenIFS environment

1. Start virtual machine (see separate handout)
2. Open a terminal window by clicking on the 'terminal' icon in the toolbar.

3. Type the command: cd openifs.
4. Type the command: source ./oifs.setup

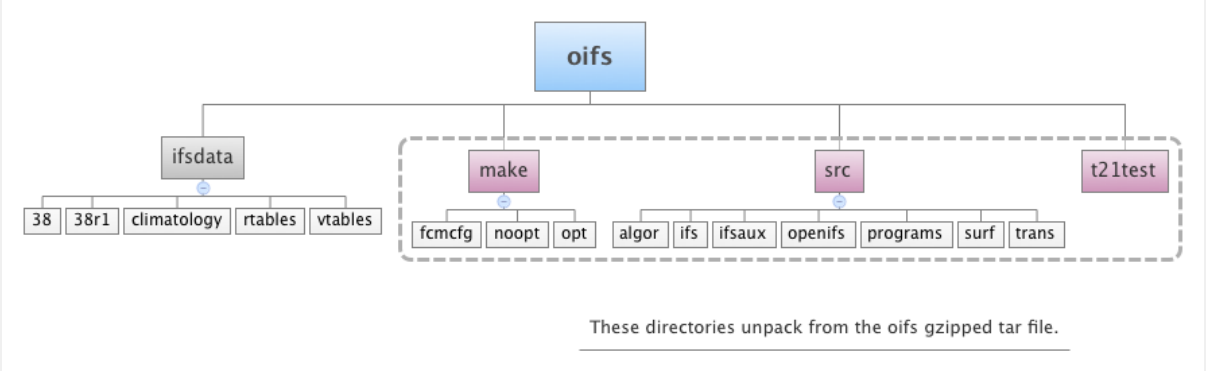
The 'oifs.setup' file sets a number of Unix shell environment variables which define the type of OpenIFS compiled installation and location of files.

Make sure you type the *source* command every time you start the VM otherwise OpenIFS will not run correctly.

For more information on how to configure OpenIFS, please see the OpenIFS website: http://software.ecmwf.int/oifs/.

Tasks - Examine directory layout

1. Type: cd openifs
2. Type: ls



Directories:

- src - contains all the source code for the model and supporting programs.
- make - contains the build configuration files for the FCM compile command. Object files and executables will be in this directory organised according to the type of build (OIFS_BUILD environment variable).
- t21test - self contained T21 model run for verifying the installation is correct.
- ifsdata - contains additional input files for the model e.g. climatologies. Available as separate tarfiles.

Note:

- OpenIFS has been precompiled on the virtual machine.
- All source code has been removed due to licensing restrictions.
- OpenIFS builds 'out-of-source'; object (.o) and executables are not mixed with source code.

3 OpenIFS T21 test forecasts

In this section of the tutorial, we'll run the pre-compiled OpenIFS model on a simple T21 forecast.

You will:

- learn about OpenIFS input and output files.
- learn some switches to control OpenIFS.
- learn how to run the model in parallel.
- learn how to run the acceptability test.

Tasks - Examine t21test directory

1. Make sure you are in the directory: openifs//t21test.
2. List the files in this directory.

The directory `t21test` contains a number of files:

```
% ls t21test
ICMGGepc8      ICMGGepc8INIUA  ICMSHepc8INIT  ifsdata  namelists
ICMGGepc8INIT  ICMSHepc8       README         job      ref_021_0144
```

Files beginning with 'ICM'.
These are the input files for this T21 experiment. They are in GRIB format. Do not move them from this directory. OpenIFS expects to find it's input files in the same directory as the main executable.

epc8 - this is the Experiment ID. Experiments IDs are used at ECMWF and initial conditions provided by ECMWF will always have an expt id.
ICMGGepc8 - 'GG' indicates these contain gridpoint fields.
ICMSHepc8 - 'SH' indicates these contains spherical harmonic fields.

job
Simple shell script to run the model. Described in more detail below.

ifsdata
Climate data fields used for T21 test integration. You should not move or rename this directory as the model will expect to find the climate files it needs in a directory of this name.

namelists
This file contains all of the input model fortran NAMELISTS. Not all of the namelists have their variables listed, only the variables commonly changed are listed here. Users should copy this file and modify it for the tests described below.

ref_021_0144
This file is reference output for the model tests. The model can be run in 'reference' mode where it checks it is working correctly by comparing some mathematical norms against these files. Reference runs are described in more detail under 'Acceptance testing' below.

Tasks - Examine grib files

Use the *grib_ls* and *grib_dump* commands to examine the contents of the ICM files.

Tasks - Run model

Run the model:

% ./job

What happens?

The model fails. Look in the NODE_001.01 file and find the subroutine traceback. Near the top of the traceback you will find the error messages.

Whenever the model fails, it will produce this traceback (controlled by DR_HOOK=1 in the *job* file).

3.1 Single process test

Tasks - Run the model as a single process

Copy the file *namelists* and run the model with a single task and single thread by executing the job script:

```
% cp namelists fort.4
% ./job
```

The model will expect to find a file called `fort.4` in the same directory as the executable. This script copies the executable from `make/opt/bin`.

If the run works you will see output like:

```
...
signal_drhook(SIGSYS=31): New handler installed at 0x4d06cf; old preserved at 0x0
MPL_BUFFER_METHOD:  2           0
   16:03:46 STEP    0 H=   0:00 +CPU=  3.598
   16:03:46 STEP    1 H=   0:10 +CPU=  0.535
   16:03:47 STEP    2 H=   0:20 +CPU=  0.537
   16:03:48 STEP    3 H=   0:30 +CPU=  0.537
   16:03:48 STEP    4 H=   0:40 +CPU=  0.527
   16:03:49 STEP    5 H=   0:50 +CPU=  0.526
   16:03:49 STEP    6 H=   1:00 +CPU=  0.530
```

This test runs only 6 timesteps.

3.1.1 Model output

The model writes its output to a several files.

NODE_001.01 contains the text output (WRITE/PRINT statements). The numbers refer to task number and thread number. Only output from the master task & thread is normally output but this can be changed for debugging purposes.

*ICM*epc8+0000* is the model output in GRIB format split into 2 files; one for the gridpoint, the other for spectral fields. These contain only a few output variables in this test. This file is a mix of GRIB1 and GRIB2 messages. See the Documentation for how to process this output.

ifs.stat is a small file that prints the model steps, time taken for each step and a 'norm' measure. This file can be usually ignored but is useful for debugging.

Tasks - Examine model output

Look at the output from the model in the NODE_001.01 file from this successful run. Note the output of the model namelists and the statistics printed at the end. IFS has very comprehensive logging output which is useful for debugging and understanding the model's performance.

Look at the grib output files using the grib commands.

3.2 Parallel runs

These next short tests verify the model works correctly with either OpenMP parallel threading, MPI tasks and both and follow on from the serial tests above.

Tasks - Enable OpenMP

Edit the file 'job' and change the line: `export OMP_NUM_THREADS=1` to `export OMP_NUM_THREADS=2`

Run ./job as above.

Do the reported CPU times change?

Use the grib_ls command to look at grib output files - what do you notice?

OpenMP threads is only enabled for optimized 'opt' builds

If this works, look in the NODE_001.01 output file for the line:

```
NUMBER OF THREADS                2
```

to verify the model ran with 2 OpenMP threads.

Tasks - Enable MPI

Edit the file 'job' and change `OMP_NUM_THREADS` back to 1.

Change the line: `NPROC=1` to `NPROC=2`.

Also, edit the fort.4 file and change `NPROC` to 2.

Rerun the job:
./job

Do the reported CPU times change?

Note that increasing the number of tasks requires changing the number of tasks in *two* places.

Look in the NODE_001.01 output file for the line: "`NUMBER OF TASKS 2`" to verify that two MPI tasks was used.

3.2.1 Mixed mode: OpenMP/MPI

The model can also be set to use NPROC=2 and OMP_NUM_THREADS=2 to use a total of 4 processes. However, this would require a computer with at least 4 cores to see any benefit and is not beneficial on the virtual machine.

4 Acceptance testing

The final step is to check the model is producing the numerical answers within acceptable limits, even if it runs the short tests above without failing. OpenIFS includes code that will compute internal statistical norms and compare against numbers supplied by ECMWF. The file: `ref_021_0144` in the `t21test` directory contains statistical norms computed by the model run at ECMWF.

Task - run acceptance test

To do the acceptance test, edit the namelists in `fort.4` and look for the NAMCT0 namelist:

```
&NAMCT0
 LREFOUT=false,
 NSTOP=6,
```

change the number of timesteps to 144 to run the model for 1 day (assuming you have not changed the default timestep of 10mins at T21) and set the LREFOUT to TRUE:

```
&NAMCT0
 LREFOUT=true,
 NSTOP=144,
```

With LREFOUT=true, at the last timestep OpenIFS will read the ref_021_0144 file and produce a new file: res_021_0144 (note the similar filenames!). The contents of the file should be similar to:

```
% cat res_021_0144

               Results of ERROR calculation

 The error calculated from the results shows
 that the calculations are correct

 The maximum error is =         0.11345 %
```

As long as the model reports 'calculations are correct' and the error is less than a few percent then the model is behaving satisfactorily in your compilation and run environment.

5 How to control model output

In this section, the main NAMELIST variables that control the output of the model are presented.

If you have time, try changing the variables, run the model and using the grib_ls and grib_dump commands to view the output grib file contents.

5.1 How to control output frequency

The namelist variables that determine the output from the model as it runs are:

5.1.1 Namelist : NAMCT0

`LFPOS` - this should be set TRUE in order to turn on model output and diagnostics.

`NFRHIS` - this is the output frequency of the 'history' files, that is, the model's prognostic variables on the *model* levels. `NFRPOS` - this is the output frequency of the prognostic variables on *pressure* levels.

It's recommended these are set the same.

If `NFRHIS/NFRPOS` are positive, the units are in model timesteps. If a negative value is used, the units are in hours.

NPOSTS & NHISTS - these are integer arrays that control the write times of the history files. They can be used for non-regular output intervals.

5.1.2 Examples

Regular output at fixed timesteps

```
NFRHIS=4,NFRPOS=4,NPOSTS=0,NHISTS=0,
```

This simple example will cause the model to produce history file output every 4 timesteps.

For this to work correctly, `NFRHIS * timestep` must equal an integer number of hours. The GRIB output will not work correctly if this isn't the case.

Non-regular output

```
NFRHIS=1,
NFRPOS=1,
NHISTS(0)=3,
NHISTS(1:3)=0,-3,-9,
NPOSTS(0)=-3,
NPOSTS(1:3)=0,-3,-9,
```

The minus sign indicates the units are in hours rather than timesteps. `NFRHIS/NFRPOS` in this case must be set to 1. The 0th element of `NHISTS/NPOSTS` determines how many outputs are produced in total by the model, the first to nth elements determine the actual output times (hours in this case because of the negative values used).

In this example, the model will write 3 separate output files at the first timestep (0hrs), 3hrs and 9hrs and then no more regardless of how long the model runs for.

6 How to change the output variables and post-processing

The namelist NAMFPC is the main namelist for the post-processing. Variables in this list can be sensitive to changes as many combinations are possible but not all work.

6.1 Model level output

To control model level output the following namelist variables (in NAMFPC) are used:

NRFP3S - list of the model levels on which post-processed output is required.
e.g. for a 60 level model run where output on all levels was required set:
NRFP3S=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,

For a 91 level model, this would give output on the first 60 levels (starting from the top).

NFP3DFS - number of 3D fields to be output on model levels. Must equal number of entries in MFP3DFS.
MFP3DFS - list of grib codes of 3D variables to be output on model levels. See How to control OpenIFS output on the OpenIFS website for valid codes.

e.g
NFP3DFS = 5,
MFP3DFS = 130, 135, 138, 155,
would output the temperature (130), vert. vel. (135), relative vorticity (138), divergence (155) on model levels.

Tasks - Change model output

Using the information above, try:

1. Adjusting the output frequency of the model.
2. Changing the list of model levels used for output and the output variables.