GNU gfortran compatible version: Cosmos8.00

Sep. 2016

1 Why gfortran?

The structure construct having been used in Cosmos is not formal one. The formal structure construct was first introduced in $\sim$ 1990 (Fortran90 ?). However, even long before that, informal one was born with so-called workstations. Since Cosmos was born far before the Fortran90 era, Cosmos has been using such informal one. (Very similar to C style).

Many compilers understand both styles (e.g Intel compiler, ifort) but some do not (e.g, GNU gfortran). In 2015, the Intel compiler on Linux became not usable free of charge. Many wondered, "Is there any compiler that is free and can compile Cosmos ?" The answer is probably "No".

At that time, a new team (see the Web page) was formed to develop Cosmos further, and as their first target, they selected a task to convert Cosmos's informal features (mainly structure construct) to the formal one. As a target compiler, GNU gfortran was selected since it is free and complies with the recent formal Fortran grammar (gfortran made much advances in recent several years). The team also payed attention to keep Cosmos compatible with Intel ifort.

2 Practical aspect

2.1 What is to be modified

Example:

Old	New
structure /fmom/ real(8):: p(4) ... end structure	type fmom real(8):: p(4) ... end type fmom
record /fmom/ p, q	type(fmom):: p, q Before "::", save, parameter, intent statement can be insterted.
structure /ptcl/ record /fmom/ fm real(8):: mass integer(2):: code, subcode, charge end structure	type ptcl type(fmom):: fm real(8):: mass integer(2):: code, subcode, charge end type ptcl
record /ptcl/ a, b b.fm.p(:) = a.fm.p(:)	type(ptcl):: a, b b%fm%p(:) = a%fm%p(:)

Therefore, for example,

```
Move.Track.p.fm.p(:)
```

must be rewritten as[1]

```
Move%Track%p%fm%p(:)
```

Oh my God, poor readability (and unpleasant key position) ! Who claimed necessity of changing "." to "%" ?

2.2 A script to manage "old to new" conversion

The conversion of old to new style for Cosmos has been done. The user need not worry about it. If the user has own application programs in the old style, the user must convert them. For that purpose, a script command "dot2perc.sh" (in Comsos/Scrpt/) is prepared. As it is normal to assume that in the user's command search path, "Cosmos/Scrpt" is included when using Cosmos, we assume so here (see sec.3).

The basic usage is

dot2perc.sh file

where "file" is a file containing Fortran programs; the extension may normally be ".f" or ".f90" but can be any (say, .h) or without extension. Then, if the command did some change, a new file **"file"-bkup** will be created and "file" will become a new converted file. If no change, no "-bkup" file will be created and "file" will remain as it is.

If the user has a lot of files,

```
for f in *.f *.f90; do
   dot2perc.sh $f
done
```

or a similar simple script will work.

Note: Besides conversions shown in the table, all comment lines are modified to start with "!". This is to prepare for future format conversion from "fixed" to "free". No text is touched in the comment lines.

Important note: The script is **not perfect**. To make it perfect is a difficult task like making a compiler. Rather than wasting time for perfect one, we may compile converted files and detect errors quickly. Where to modify is normally easily detected.

For example, the current conversion script fails in the following cases (although for human eyes, obvious):

```
if(  a.le.  b ) then  --> converted to  if(a%le% b) then
if(  a .le. b ) then  --> This is OK

call copenf(21, "$COSMOS/Data/a.d", icon)
    --> a.d is conveted to a%d
       Unfortunately, this type of error cannot be detected
       until the user executes the program.
```

[1] Intel Fortran still understands the old dot style notation even everything has been modified for the new scheme, but the old dot style should not be used anymore.

2.3 Intel Fortran or gfortran

If you can use Intel ifort, it's probably better to use it since execution speed seems higher than gfortran (for many applications, the ratio is ∼3/2).

The choice is fixed by Cosmos/site.config. If you use Linux running on non-Mac PC, to use Intel Fortran (64 bit mode), copy (see however, next section),

site.configPCLinuxIFC64

to site.config in the Cosmos. (This is same as in older versions).
To use gfortran, to be copied is

site.configLinuxGfort.

Mac users may use

site.configMacIFC (for Intel ifort)

and

site.configMacGfort (for GNU gfortran).

Note: Reliability of both of ifort and gfortran on Mac are unfortunately less than Linux versions. In some case, they fail to compile a program which can be easily compiled by the Linux version.

2.3.1 Managing site.config

Although the description above says,

copy site.configXXXX site.config

(XXXX is, say, LinuxGfort), the same effect can be obtained by using "link" function; do in Cosmos,

1. `rm␣site.config`
2. `ln␣-s␣site.configXXXX␣site.config`

The merit of this method is that you can know the current assigned compiler by issuing

ls -l site.config.

In the case of "copy", we need to look into site.config like

head site.config.

etc.

3 Be prepared for version change

This applies to all Cosmos versions (i.e, not specific to the gfortran compatible new version). We see sometimes that a user does rather complex procedure when the version of Cosmos is changed (for using a new version or for using an older version).

One recommended way which requires simple minimum changes in such a case would be as follows. Let's take Cosmos as example.

- Preparation once for all.
  - Put next in your ∼/.bashrc.

```
export COSMOSTOP=~/Cosmos
export COSMOSINC=$COSMOSTOP/cosmos
export PATH=..:$PATH:$COSMOSTOP/Scrpt:...
```

The `$PATH` position may be elsewhere. For other shells, do similarly.

– Don't make "Cosmos" folder in your home. Store various versions of Cosmos in other folder(s) (let's say XXXX which may consist of more than one directory hierarchy like ∼/abc/xyz/).

- If you want to use a new version of Cosmos, put that version in XXXX.
- If "Cosmos" already exists in your home,

  rm Cosmos

- `ln␣-s␣XXXX/CosmosVVV␣Cosmos`

  where VVV means your desired version number.

- Fix "site.config" for your desired compiler, as described previously.

  That's it for the change.

- Now, you can think Cosmos is always in your home.

4 Other updates

- "fixIntmodel.sh" and "showIntModel.sh" are unified to "intModel.sh". If old one is used, the user is requested to use this new command. The "intModel.sh" command shows the current interaction models and also enables the user to change the models.
- For the dpmjet interaction model, a subroutine call like

  integer:: inttype

  call cqDpmIntType(inttype)

  is now usable to know what type of interaction (defraction, elastic etc) has occurred. (See Cosmos/Import/DPM/Interface/cdpmjet.f). It's not recommended to use this for judging if a collision is elastic or inelastic; for practical purpose better to see energy release.

- Others are not directly related to Cosmos execution but are for utility commands. In older versions, they must be issued after going to Cosmos/Util, but now can be used from anywhere, provided that the command search path is properly set (sec. 3)

  Trace display commands for Cosmos are renamed or newly introduced. Those use Gnuplot are not recommended (slow and not beautiful).

```
dispcosTraceByGeomv   (rename from disptracebygeomv)
dispcosTraceByGnupl    (new: use Gnuplot)
```

Note:
With the recent combination of Linux and Mac, Geomview window invoked on Linux seems not transmitted to Mac (i.e, when remote login is made via "ssh -Y" from Mac to Linux and Geomview command is issued). Direct workaround is not yet found.

However, "vnc" connection or "remote-desktop" connection (in some environment "vpn" connection may be required in advance) will resolve this problem.