1. Compiling RTTOV9

1.1. Creating a Makefile

The creation of a Makefile for compiling RTTOV is automated. The Makefile is created by a script which analyses the dependencies between RTTOV source units. These dependencies are of two kinds:

- Module dependency: file `a.F90` contains a `use b` statement. Hence if module b is recompiled then unit `a.F90` has to be recompiled.
- Interface dependency: file `a.F90` includes `b.interface`. If the interface of `b` changes then it is necessary to recompile `a.F90`.

To create the Makefile, change to the subdirectory `src` of the RTTOV distribution and type:

```
$ ../build/Makefile.PL
```

Note that this script has to be run every time dependencies change. Adding a new subroutine or a new program in RTTOV `src/` directory implies running the script again, and so does adding a use statement or including an interface. Not updating the Makefile may lead to create spurious executable code.

1.2. Compiling for a specific architecture

Compiling RTTOV requires identifying the target machine; the directory `build/arch` contains a list of architectures which have been tested by the project team. Choose the one which appears the most appropriate for your machine.
Change to the `src` directory of your RTTOV distribution, and type:

```
$ make ARCH=myarch
```

This will build RTTOV for the `myarch` architecture; object files will be kept in the `obj/` subdirectory of your RTTOV distribution, modules files in `mod/`, executables in `bin/`, and interfaces in `include/`. Note that the creation of these interface files are part of the building process (see section « Interface files »).

It is possible to specify a target directory to install RTTOV; this is very useful when compiling RTTOV with different flags or a different compiler on the same machine:

```
$ make ARCH=myarch BLDDIR=../mydir
```

The command above, when issued from the `src/` directory will install the `obj/`, `bin/`, `include/` and `mod/` directories in the `mydir/` directory in your RTTOV distribution. Note that the following restriction currently holds: the `mydir` directory has to be located in the RTTOV distribution main directory; but once it is compiled and tested, you are free to move it where you like.

When you compile RTTOV for a specific architecture, a `tmp-myarch` is created in the RTTOV top directory; this makes possible to compile RTTOV in parallel (for different architectures) and to keep the listings issued by some compilers.

Some other targets exist in the Makefile:

- `clean` : removes all object files, libraries, executables, module files and interfaces created by the Makefile.
- `dist` : typing « `make ARCH=myarch dist` » will create a gzipped tarball of RTTOV source and test definition directories.

1.3. Interface files

Interface files are created automatically from the source code by the script `build/mkintf.pl`. Given a Fortran unit `a.F90` this script extracts the source code from `a.F90` up to the !INTF marker which shall appear in every Fortran unit which requires an interface (namely subroutines and functions). Hence a Fortran unit which needs an interface to be extracted shall be written as follows:

```
Subroutine a( x1, x2, x3, .... )
! use statements go here
Use m1
Use m2
! argument declarations go here
Real :: x1
Real :: x2
Real :: x3
...
!INTF
```

Note the `!INTF` mark at the end of arguments declaration.

The interface file is created when needed (that is, when the `make` executable request them); this implies that unreferenced interface files are never created. But you can create the interface of `a.F90` by typing:

```
$ ../build/mkintf.pl a.F90 a.interface
```

Note also that modifying `a.F90` does not imply that `a.interface` will be created again. It will actually be created only if it different from the one which already exists; this is to avoid unnecessary recompilation of the code.

1.4. Creating an architecture configuration file

If your architecture is not included in the `build/arch` directory bundled with RTTOV (or maybe you would like to customize the installation of RTTOV), it is possible to create your own configuration file.

This configuration file shall be installed in the `build/arch` directory and define the following macros:

- `FC` : the name of your Fortran 90 compiler.
- `FC77` : the name of your Fortran 77 compiler; this might be your Fortran 90 compiler with possibly some special options.
- `LDFLAGS_ARCH` : specific flags to pass to the linker.
- `FFLAGS_ARCH` : specific flags for your Fortran compiler.
- `AR` : the command to create a library from object files.

This configuration file may define the following macros:

- `FFLAG_MOD` : this is the flag used by your Fortran 90 compiler to locate module files; it defaults to `-I`, but you can override this setting.
- `CPP` : the name of your pre-processor; defaults to `cpp`.
- Specific flags for some RTTOV units; defining `FFLAGS_ARCH_a` will force the build system to compile unit `a.F90` with these specific flags.

We reproduce below the content of the configuration file for the NEC-SX F90 compiler with optimization:

```
FC=sxf90
FC77=sxf90
LDFLAGS_ARCH=

FFLAGS_HOPT= -Chopt
FFLAGS_SAFE= -Cvsafe
FFLAGS_NEC = -Wf,-pvctl loopcnt=200000 -Wf,-pvctl nomsg -Wf,-O nomove,-O nomsg -
DRTTOV_ARCH_VECTOR

FFLAGS_ARCH= $(FFLAGS_HOPT) $(FFLAGS_NEC)
FFLAGS_ARCH_rttov_alloc_prof      = $(FFLAGS_SAFE) $(FFLAGS_NEC)
FFLAGS_ARCH_rttov_alloc_predictor = $(FFLAGS_SAFE) $(FFLAGS_NEC)
FFLAGS_ARCH_rttov_tl              = $(FFLAGS_SAFE) $(FFLAGS_NEC)
FFLAGS_ARCH_rttov_ad              = $(FFLAGS_SAFE) $(FFLAGS_NEC)
AR=sxar rv
```

The previous configuration file shows that the Fortran 90 compiler on this platform is sxf90, the archive creation command is sxar rv, and that some files require that optimization be disabled (namely `rttov_alloc_prof.F90, rttov_alloc_predictor.F90, rttov_tl.F90, rttov_ad.F90`).