Forcheck

A Fortran source code analyzer

Tamás Fehér
High Level Support Team
Max-Planck-Institut für Plasmaphysik

tamas.bela.feher@ipp.mpg.de

Helios Webinar, June 20. 2013

Outline

- Introduction
  - Simple Forcheck example
- Forcheck Analysis
  - Input and output files
  - Preprocessor
  - Include directories
  - Forcheck library files
- More on Forcheck
  - False errors
  - Fortran standards and additional options
  - Forcheck in makefiles

Debugging and testing

- Static code analysis
  - analyze the code without execution
- Dynamic analysis
  - execute the code and check during runtime
    - Compile with run-time error checking

      ```
      ifort -O0 -g -check all source.f90
      ```
    - Valgrind package
- Test cases
  - check whether the implementation is correct
  - whether the resulting code fulfills its purpose

Static code analysis

- Compilers perform a certain level of static code analysis
  - `ifort -warn all helloworld.f90`

    turns on warning messages
  - `ifort -std f90 helloworld.f90`

    checks standard conformance
  - `ifort -diag-enable sc3 helloworld.f90`

    static security analysis, Intel Inspector is needed to view results
- Different compilers perform differently
- Forcheck: detects more anomalies than most compilers

Forcheck

- Fortran source code analyzer
- Performs a static analysis
- Supports all standards up to Fortran 2003, most Fortran 2008 features
- Understands language extensions by popular compilers
- Relatively easy to set up the analysis

```
$ forchk list_of_sources
```

- Helps to find and eliminate bugs
- Installed on Helios

Simple example

```
PROGRAM helloworld
IMPLICIT NONE
  CHARACTER(len=12) :: str1, str2

  str1 = 'Hello'
  str1 = 'World!'

  WRITE (*,*) str1, str2

END PROGRAM
```

```
$ module load intel
$ module load forcheck
$ forchk helloword.f90
```

Simple example

```
PROGRAM helloworld
IMPLICIT NONE
  CHARACTER(len=12) :: str1, str2

  str1 = 'Hello'
  str1 = 'World!'

  WRITE (*,*) str1, str2

END PROGRAM
```

```
$ module load intel
$ module load forcheck
$ forchk helloword.f90
```

```
                                …
     8  WRITE (*,*) str1, str2
(file: helloworld.f90, line: 8)
  STR2
**[312 E] no value assigned to this variable
(file: helloworld.f90, line: 3)
  STR2
**[307 E] variable not defined
                                …
 -- messages presented:

    1x[307 E] variable not defined
    1x[312 E] no value assigned to this
variable
    1x[315 I] redefined before referenced

number of error messages:                2
number of informative messages:          1
```

Simple example

```
PROGRAM helloworld
IMPLICIT NONE
  CHARACTER(len=12) :: str1, str2

  str1 = 'Hello'
  str1 = 'World!'

  WRITE (*,*) str1, str2

END PROGRAM
```

```
$ module load intel
$ module load forcheck
$ forchk helloword.f90
```

```
                                …
     8  WRITE (*,*) str1, str2
(file: helloworld.f90, line: 8)
  STR2
**[312 E] no value assigned to this variable
(file: helloworld.f90, line: 3)
  STR2
**[307 E] variable not defined
                                …
 -- messages presented:

    1x[307 E] variable not defined
    1x[312 E] no value assigned to this
variable
    1x[315 I] redefined before referenced

number of error messages:                2
number of informative messages:          1
```

Forcheck results

- Forcheck gives error, warning or informative messages
- Detects more than 800 different problems
- Some examples:
  - [145 I] implicit conversion of scalar to complex
  - [315 I] redefined before referenced
  - [307 E] variable not defined
  - [312 E] no value assigned to this variable
  - [318 E] not allocated
  - [571 E] argument type inconsistent with first occurrence

Outline

- Introduction
  - Simple Forcheck example
- Forcheck Analysis
  - Input and output files
  - Preprocessor
  - Include directories
  - Forcheck library files
- More on Forcheck
  - False errors
  - Fortran standards and additional options
  - Forcheck in makefiles

Forcheck output files

Forcheck output can be long, better to redirect into log files:

```
$ forchk -l forcheck.lst helloword.f90
```

- Creates a text file `forcheck.lst` that contains
  - source code annotated with Forcheck messages
  - message summary
  - additional analysis information

```
$ forchk -rep forcheck.rep helloword.f90
```

- More concise report in `forcheck.rep` text file

Multiple source files

We can simply give all the source files to Forcheck

```
forchk -l list.lst file1.F90 src/*.f* file2.f
```

- Default options based on file extension
  - `.F*` files to be preprocessed
  - `.f`, `.F` fixed format
  - `.f90`, `.F90`, ... free format

Format specification can be changed by global / local options

```
forchk -l list.lst -ff file1.F90 src/*.f* \
-nff file2.f
```

Global option (placed before sources): all files in free form

Local option (placed inside source list): the next file in the list is in fixed form

Preprocessor

- Use the same `-D` symbols as for the compiler
  - `ifort -Ddef1 -Ddef2 myprog.F90`
  - `forchk -define def1,def2 myprog.F90`
- cpp preprocessing is automatically invoked for .F, .FOR, .FTN, .FPP, .F90, .F95, .F03, or .F08
- Use -cpp -ncpp options to override defaults
- Alternatively one can preprocess the files, and call Forcheck for the preprocessed files

Include files

- Call Forcheck with the same `-I` options as the compiler
  - `forchk -I list_of_directories`
  - comma separated list
- example.F90 have statements:

  INCLUDE 'mpif.h'

  INCLUDE 'fftw3.f'
- Command to compile:
  - `ifort -c -I ${FFTW_DIR}/include \`

    `-I /opt/mpi/bullxmpi/1.1.16.5/include example.F90`
- Forcheck command:
  - `forchk -l forcheck.lst -I ${FFTW_DIR}/include,\`

    `/opt/mpi/bullxmpi/1.1.16.5/include example.F90`

Checking MPI calls

- If MPI is interfaced by `INCLUDE 'mpif.h'` statement, then there can be several messages like

```
71x[571 E] argument type inconsistent with first occurrence
```

- Better to have `use MPI` statements in the code
  - Forcheck (and also the compilers) can check the argument list properly

- Forcheck comes with `MPI.FLB` file that stores MPI interface information

```
forchk -l forcheck.lst mpi_example.F90 \
${FCKDIR}/share/forcheck/MPI.FLB
```

Creating Forcheck library files

- Forcheck runs even without information about external libraries (but produces extra error messages)
- It is better to provide the library sources:
  - `forchk -l mylist.lst my_prog/*.f90 library/*.F90`
- Alternative: create an FLB file for the library
  - Stores the global information of one or more program units

    ```
    forchk -l list_lib.lst library/*.f90 -create library.flb
    ```
  - Later it can be used for program analysis

    ```
    forchk -l mylist.lst my_prog/*.f90 library.flb
    [library2.flb ...]
    ```
  - It is the standard way of interfacing external libraries

Forcheck analysis summary

- Same -I and -D options as for compiler (but comma separated list)
- External libraries
  - Include files will be found if `-I` is specified
  - If module interface is used:

    Create .FLB files for libraries **or** add the library sources to the analysis
- Run Forcheck:

```
forchk -I include,directories \
       -define def1,def2 -l mylist.lst \
        path/to/src/*.f90 [ lib_src/*.f* ] \
        ${FCKDIR}/share/forcheck/MPI.FLB
```

- Summary of errors at the end of mylist.lst
  - Search for message labels: [xxx E], [xxx W], [xxx I]

Outline

- Introduction
  - Simple Forcheck example
- Forcheck Analysis
  - Input and output files
  - Preprocessor
  - Include directories
  - Forcheck library files
- More on Forcheck
  - False errors
  - Fortran standards and additional options
  - Forcheck in makefiles

Files missing from the analysis

- If some include files/modules are not found:
  - The Forcheck analysis still continues
  - Can results in a high number of false error messages
- To fix the problem, check for the `FCK--` messages:

```
(file: src/part.F90, line: 3)
BACKGROUND
FCK-- module information not
found
```

- Check the list of source files

```
(file: src/mpi_mod.F90, line: 8)
mpif.h
FCK-- open error on include file
```

- Specify directories for include files

False errors

- Forcheck performs a complex analysis
- Sometimes it gives false error messages
- Such errors can be reported to the developer of Forcheck
- Bugs are fixed usually quickly

Fortran standards

- FORCHECK can handle many Fortran levels, dialects and language extensions
- Determined by the FCKCNF environment variable
- Default on Helios: Intel compiler emulation with Fortran 95 syntax
- Alternative compiler configuration files:

```
ls ${FCKDIR}/share/forcheck/*.cnf

export FCKCNF=${FCKDIR}/share/forcheck/f03.cnf
```

- `-standard:` checks conformance to standard selected by FCKCNF

Other options

- If you compile with double precision reals:

```
ifort -r8 a.f90
ifort -real-precision 64 a.f90
```

  - Then use the -dp option for Forcheck

```
forchk -dp -l list.lst a.f90
```

- Check all columns for fixed format (beyond 72)

```
forchk -allc ...
```

- Suppress certain messages: `-ninf -nwarn`
- List reference structure (call tree): `-shref`
- Integrated development environment: ForcheckIDE

-ancmpl option

More detailed analysis between separate compilation units. Reports:

- Unreferenced procedures and modules
- Unreferenced and undefined common blocks and public module variables
- Unsaved common blocks and module data
- Can lead to false positives
- Recommended to use in the second step

Forcheck in the Makefile

Assume we have a list of sources in SRCLIST and all compiler options in FFLAGS:

```
COMMA =,

SPACE :=

SPACE +=


INC_TMP = $(filter -I%,$(FFLAGS))

DEF_TMP = $(filter -D%,$(FFLAGS))

FCK_INCS = $(strip $(subst -I,,$(subst $(SPACE)-I,$(COMMA),$
(INC_TMP))))

FCK_DEFS = $(strip $(subst -D,,$(subst $(SPACE)-D,$(COMMA),$
(strip $(DEF_TMP)))))


forcheck :
        forchk -allc -ancmpl -l mycode.lst -ff \
        -define $(FCK_DEFS) -I $(FCK_INCS) $(SRCLIST) \
        $(FCKDIR)/share/forcheck/MPI.FLB
```

Summary

- Forcheck can be used for static code analysis
- Relatively easy to set up
- Helps the development process
- Can locate bugs that would be very difficult to find otherwise
- Forcheck helps to write more portable code