

## Quickstart - Fortran Modernization with HYCOM and compile\_commands.json

1. This is a continuation of the Quickstart Guide to Codee. It works with the [performance-demos-fortran](#) Github repository, more specifically with [HYCOM/src](#).
2. In this case, the project comes with a Makefile, so we can directly use the *bear* tool (version 3 or later) to generate the `compile_commands.json` file required by Codee. Invoke *bear* as follows:

```
$ bear -- make ARCH=codee TYPE=demo
```

Basically, we must pass the invocation of *make* to the right of the '--' separator.

This command will produce the `compile_commands.json` file with all the compiler invocations needed to build the source files.

3. Produce the Screening Report of Codee using the compilation database (`--config`):

```
codee screening --check-id PWR001,PWR002,PWR003,PWR007,PWR008,PWR012,PWR063 --config compile_commands.json
```

```
50 total entries detected

Configuration file 'compile_commands.json' successfully parsed.
. . .
[ 1/50] /home/ulises/Appentra/repos/performance-demos-fortran/HYCOM/src/mod_dimensions.F90 ... Done
[ 2/50] /home/ulises/Appentra/repos/performance-demos-fortran/HYCOM/src/mod_xc.F90 ... Done
. . .
[50/50] /home/ulises/Appentra/repos/performance-demos-fortran/HYCOM/src/hycom.F90 ... Done

SCREENING REPORT

---Number of files---
Total | C C++ Fortran
-----|-----
50    | 1 0 49

Lines of code Analysis time # checks Profiling
-----|-----
44679      1 m 10 s      429      n/a

CHECKS PER CATEGORY AND PRIORITY LEVELS

| -----Checks per category----- | -Priority- |
| Scalar Control Memory Vector Multi Offload Quality | L1 L2 L3 |
| -----|-----|-----|-----|
| n/a n/a n/a n/a n/a n/a 429 | 221 20 188 |

Lines of code : total lines of code found in the target (computed the same way as the sloccount tool)
Analysis time : time required to analyze the target
# checks : total actionable items (opportunities, recommendations, defects and remarks) detected
Profiling : estimation of overall execution time required by this target

RANKING OF CHECKERS

Checker Level Priority # Title
-----|-----|-----|-----
PWR008 L1 P18 196 Declare the intent for each procedure parameter
PWR003 L1 P18 2 Explicitly declare pure functions
PWR063 L1 P12 23 Avoid using legacy Fortran constructs
PWR007 L2 P6 20 Disable implicit declaration of variables
PWR001 L3 P3 188 Declare global variables as function parameters

SUGGESTIONS
```

```
Focus the analysis on a specific file before proceeding with the Codee auto mode or the guided mode:
codee screening specific/file.c --check-id PWR001,PWR002,PWR003,PWR007,PWR008,PWR012,PWR063
--config compile_commands.json
```

```
Use --target-arch to focus on the checks most relevant to your hardware type [cpu | gpu | mcu],
e.g.:
codee screening --target-arch cpu --check-id PWR001,PWR002,PWR003,PWR007,PWR008,PWR012,PWR063
--config compile_commands.json
```

```
50 files, 251 functions, 2058 loops successfully analyzed and 0 non-analyzed files in 1 m 10 s
```

#### 4. Run the screening report in '--verbose' mode to get the subtotals per-file:

```
codee screening --check-id PWR001,PWR002,PWR003,PWR007,PWR008,PWR012,PWR063 --verbose --config
compile_commands.json
```

```
50 total entries detected
```

```
Configuration file 'compile_commands.json' successfully parsed.
```

```
Date: 2024-04-24 Codee version: 2024.2.2
```

```
[Fortran] target compiler: <none> (Compiler Agnostic Mode)
```

```
[C] target compiler: <none> (Compiler Agnostic Mode)
```

```
[ 1/50] mod_dimensions.F90 ... Done
```

```
[ 2/50] mod_xc.F90 ... Done
```

```
. . .
```

```
[50/50] hycom.F90 ... Done
```

```
. . .
```

Target	Lines of code	Analysis time	# checks	Profiling
forfun.F90	3809	2751 ms	55	n/a
s8gefs.F90	597	204 ms	39	n/a
mxkprf.F90	2520	3532 ms	23	n/a
diapfl.F90	921	1190 ms	20	n/a
. . .				
Total	44679	1 m 10 s	429	n/a

```
CHECKS PER CATEGORY AND PRIORITY LEVELS
```

Target	Checks per category							-Priority-		
	Scalar	Control	Memory	Vector	Multi	Offload	Quality	L1	L2	L3
forfun.F90	n/a	n/a	n/a	n/a	n/a	n/a	55	24	0	31
s8gefs.F90	n/a	n/a	n/a	n/a	n/a	n/a	39	20	19	0
mxkprf.F90	n/a	n/a	n/a	n/a	n/a	n/a	23	12	0	11
diapfl.F90	n/a	n/a	n/a	n/a	n/a	n/a	20	10	0	10
. . .										
Total	n/a	n/a	n/a	n/a	n/a	n/a	429	221	20	188

```
. . .
```

```
50 files, 251 functions, 2058 loops successfully analyzed and 0 non-analyzed files in 1 m 10 s
```

#### 5. Choose one of the files that has the highest amount of high priority checkers (L1). Then run the 'checks report' with it. For example:

```
codee checks --check-id PWR001,PWR002,PWR003,PWR007,PWR008,PWR012,PWR063 --config compile_commands.json
s8gefs.F90
```

```
CHECKS REPORT
```

```
s8gefs.F90:955:7 [PWR003] (level: L1): Explicitly declare pure functions
s8gefs.F90:4:7 [PWR008] (level: L1): Declare the intent for each procedure parameter
s8gefs.F90:149:7 [PWR008] (level: L1): Declare the intent for each procedure parameter
s8gefs.F90:355:7 [PWR008] (level: L1): Declare the intent for each procedure parameter
s8gefs.F90:470:7 [PWR008] (level: L1): Declare the intent for each procedure parameter
s8gefs.F90:600:7 [PWR008] (level: L1): Declare the intent for each procedure parameter
s8gefs.F90:651:7 [PWR008] (level: L1): Declare the intent for each procedure parameter
s8gefs.F90:804:7 [PWR008] (level: L1): Declare the intent for each procedure parameter
```

```

s8gefs.F90:909:7 [PWR008] (level: L1): Declare the intent for each procedure parameter
s8gefs.F90:955:7 [PWR008] (level: L1): Declare the intent for each procedure parameter
s8gefs.F90:1750:7 [PWR008] (level: L1): Declare the intent for each procedure parameter
s8gefs.F90:1819:7 [PWR008] (level: L1): Declare the intent for each procedure parameter
s8gefs.F90:1887:7 [PWR008] (level: L1): Declare the intent for each procedure parameter
s8gefs.F90:1985:7 [PWR008] (level: L1): Declare the intent for each procedure parameter
s8gefs.F90:2081:7 [PWR008] (level: L1): Declare the intent for each procedure parameter
s8gefs.F90:2176:7 [PWR008] (level: L1): Declare the intent for each procedure parameter
s8gefs.F90:2227:7 [PWR008] (level: L1): Declare the intent for each procedure parameter
s8gefs.F90:2258:7 [PWR008] (level: L1): Declare the intent for each procedure parameter
s8gefs.F90:2305:7 [PWR008] (level: L1): Declare the intent for each procedure parameter
s8gefs.F90 [PWR063] (level: L1): Avoid using legacy Fortran constructs
s8gefs.F90:4:7 [PWR007] (level: L2): Disable implicit declaration of variables
s8gefs.F90:149:7 [PWR007] (level: L2): Disable implicit declaration of variables
s8gefs.F90:355:7 [PWR007] (level: L2): Disable implicit declaration of variables
s8gefs.F90:470:7 [PWR007] (level: L2): Disable implicit declaration of variables
s8gefs.F90:600:7 [PWR007] (level: L2): Disable implicit declaration of variables
s8gefs.F90:651:7 [PWR007] (level: L2): Disable implicit declaration of variables
s8gefs.F90:804:7 [PWR007] (level: L2): Disable implicit declaration of variables
s8gefs.F90:909:7 [PWR007] (level: L2): Disable implicit declaration of variables
s8gefs.F90:955:7 [PWR007] (level: L2): Disable implicit declaration of variables
s8gefs.F90:1750:7 [PWR007] (level: L2): Disable implicit declaration of variables
s8gefs.F90:1819:7 [PWR007] (level: L2): Disable implicit declaration of variables
s8gefs.F90:1887:7 [PWR007] (level: L2): Disable implicit declaration of variables
s8gefs.F90:1985:7 [PWR007] (level: L2): Disable implicit declaration of variables
s8gefs.F90:2081:7 [PWR007] (level: L2): Disable implicit declaration of variables
s8gefs.F90:2152:7 [PWR007] (level: L2): Disable implicit declaration of variables
s8gefs.F90:2176:7 [PWR007] (level: L2): Disable implicit declaration of variables
s8gefs.F90:2227:7 [PWR007] (level: L2): Disable implicit declaration of variables
s8gefs.F90:2258:7 [PWR007] (level: L2): Disable implicit declaration of variables
s8gefs.F90:2305:7 [PWR007] (level: L2): Disable implicit declaration of variables

```

#### SUGGESTIONS

Use `--verbose` to get more details, e.g:  
`codee checks --verbose --check-id PWR001,PWR002,PWR003,PWR007,PWR008,PWR012,PWR063 --config compile_commands.json s8gefs.F90`

Use `--level` to filter checks with a specific level of priority, e.g:  
`codee checks --level L1 --config compile_commands.json s8gefs.F90`

More details on the defects, recommendations and more in the Open Catalog of Best Practices for Performance:  
<https://github.com/codee-com/open-catalog/>

1 file, 19 functions, 40 loops successfully analyzed and 0 non-analyzed files in 191 ms

- Re-run the checks report with `--verbose` to get more details for each check. Note that the output can be large, so it is useful to combine the verbose mode with `--level L1` to list only the highest priority checkers. You can also directly restrict the list of checkers to a specific one. For example, here is the output of the verbose mode for the PWR063 checker only:

```
codee checks --check-id PWR063 --verbose --config compile_commands.json s8gefs.F90
```

#### CHECKS REPORT

```
s8gefs.F90 [PWR063] (level: L1): Avoid using legacy Fortran constructs
```

```

Assumed size array:
149:    SUBROUTINE S8GECO(A,LDA,N,IPVT,RCOND,Z)
223:    INTEGER LDA,N,IPVT(*)
224:    REAL A(LDA,*),Z(*)
355:    SUBROUTINE S8GEFA(A,LDA,N,IPVT,INFO)
413:    INTEGER LDA,N,IPVT(*),INFO
414:    REAL A(LDA,*)
470:    SUBROUTINE S8GESL(A,LDA,N,IPVT,B,JOB)
540:    INTEGER LDA,N,IPVT(*),JOB
541:    REAL A(LDA,*),B(*)
1750:   INTEGER FUNCTION IS8AMAX(N,SX,INCX)
1786:   REAL SX(*),SMAX,XMAG
1819:   REAL FUNCTION S8ASUM(N,SX,INCX)
1855:   REAL SX(*)
1887:   SUBROUTINE S8AXPY(N,SA,SX,INCX,SY,INCY)
1928:   REAL SX(*),SY(*),SA

```

```

1985:      REAL FUNCTION S8DOT(N,SX,INCX,SY,INCY)
2025:      REAL SX(*),SY(*)
2081:      SUBROUTINE S8SCAL(N,SA,SX,INCX)
2118:      REAL SA,SX(*)
DATA:
842:      DATA KOUNT(1),KOUNT(2),KOUNT(3),KOUNT(4),KOUNT(5), &
845:      DATA KOUNTX/0/
1052:      DATA IMACH( 1) /    5 /
1053:      DATA IMACH( 2) /    6 /
1054:      DATA IMACH( 3) /    7 /
1055:      DATA IMACH( 4) /    6 /
1056:      DATA IMACH( 5) /   32 /
1057:      DATA IMACH( 6) /    4 /
1058:      DATA IMACH( 7) /    2 /
1059:      DATA IMACH( 8) /   31 /
1060:      DATA IMACH( 9) / 2147483647 /
1061:      DATA IMACH(10) /    2 /
1067:      DATA IMACH(11) /   53 /
1068:      DATA IMACH(12) / -1021 /
1069:      DATA IMACH(13) /  1024 /
1070:      DATA IMACH(14) /   53 /
1071:      DATA IMACH(15) / -1021 /
1072:      DATA IMACH(16) /  1024 /
2219:      DATA IPARAM(1),IPARAM(2),IPARAM(3),IPARAM(4)/0,2,0,10/
2220:      DATA IPARAM(5)/1/
2221:      DATA IPARAM(6),IPARAM(7),IPARAM(8),IPARAM(9)/0,0,0,0/
Equivalence:
1032:      EQUIVALENCE (IMACH(4),OUTPUT)
Suggestion: Remove the legacy fortran constructs and refactor the code to comply with modern Fortran
standards.
Documentation: https://github.com/codee-com/open-catalog/tree/main/Checks/PWR063

SUGGESTIONS

Use --level to filter checks with a specific level of priority, e.g:
codee checks --level L1 --verbose --config compile_commands.json s8gefs.F90

More details on the defects, recommendations and more in the Open Catalog of Best Practices for
Performance:
https://github.com/codee-com/open-catalog/

1 file, 19 functions, 40 loops successfully analyzed and 0 non-analyzed files in 205 ms

```