GNU fortran

Paul Thomas,
Glyme Consultancy Limited

BCS Meeting 27th September 2018

- **Contributors, past and present:**

  Harald Anlauf, Janne Blomqvist, Steven Bosscher, *Paul Brook*, Tobias Burnus, François-Xavier Coudert, Bud Davis, Jerry DeLisle, Arnaud Desitter, Eric Edelsonn, Alessandro Fanfarillo, Daniel Franke, Katherine Holcomb, Dominique d'Humieres, Steven G. Kargl, Nicolas Koenig, Thomas Koenig, Daniel Kraft, Louis Krupp, Rainer Orth, Asher Langton, Mike Kumbera, Victor Leikehman, Dave Love, Toon Moene, Mikael Morin, Fritz Reese, Jeurgen Reuter, Damian Rouson, Tobias Schlüter, Lars Segerlund, Gerhard Steinmetz, Paul Thomas, *Andy Vaught*, Andre Vehrschild, Feng Wang, Janus Weil, Canqun Yang and Xiaoqiang Zhang.

- **GCC maintainers who have helped significantly:**

  Jakub Jelnik, Richard Biener, Jan Hubicka, Andrew Pinski, Bernhard Reutner-Fischer, Eric Botcazou, Qing Zhao, Martin Liska, Richard Sandiford, Cesar Philippidis, Thomas Schwinge, Nathan Sidwell, Martin Sebor and many others

- **...and all the contributors of bug reports on gcc Bugzilla**

Maintainers/Developers are mostly part-time volunteers

- Pros:
  - They are fortran users (mainly graduate students or post-docs)
  - *They are well committed to what they want or need to do.*
- Cons:
  - *They are well committed to what they want or need to do*… and not, by and large, much more. A steering committee would be faced with the “herding cats” problem.
  - There is a lack of consistency in design approach, which has led to some fuzziness between parsing, resolution and translation.
  - Some parts of the code have become rather unstructured by piling bug fix on bug fix.
  - Some of the more difficult bugs require a deep knowledge of TREE_SSA and back-end functioning to fix.

A Brief History:

- Since GCC version 4.0.0, released in April 2005, gfortran has replaced the older g77 compiler.
- The new Fortran front-end for GCC was rewritten from scratch, largely by Any Vaught and Paul Brook, after the principal author and maintainer of g77, Craig Burley, decided in 2001 to stop working on the g77 front end.
- gfortran forked off from g95 in January 2003, which itself started in early 2000. As far as I can tell, Andy Vaught stopped maintaining g95 sometime around 2010/11.
- Since 2010 the front-end, like the rest of the GCC project, was migrated to C++, while it was previously written in C.
- The initial aim was to provide a frontend with complete F95 compliance, reasonable performance and legacy g77 support. That was achieved sometime in the late “naughties”. Since then, F20xx and DEC/Cray legacy features have been added.

Structure of gfortran (i):

- The parser and resolution stages produce an intermediate representation of the fortran code.
- This can be exposed with the option **–fdump-parse-tree**

```
implicit none
   type :: mytype
      integer :: i
      integer, allocatable, dimension(:) :: j
   end type mytype
   type(mytype) :: x
   x%i = 42
   x%j = [1,2,3,4]
   print *, x%i
   print *, x%j
end
```

sends to stdout…..

```
Namespace: A-Z: (UNKNOWN 0)
procedure name = MAIN__
  symtree: 'MAIN__'        || symbol: 'MAIN__'
    type spec : (UNKNOWN 0)
    attributes: (PROGRAM PUBLIC  SUBROUTINE)
  symtree: 'Mytype'        || symbol: 'mytype'
    type spec : (UNKNOWN 0)
    attributes: (DERIVED )
    components:
    (i (INTEGER 4) ())
    (j (INTEGER 4) ALLOCATABLE DIMENSION (1 [0] AS_DEFERRED () () ))
    hash: 85374412
    Procedure bindings:
    Operator bindings:
  symtree: 'mytype'        || symbol: 'mytype'
    type spec : (UNKNOWN 0)
    attributes: (PROCEDURE  FUNCTION)
    Generic interfaces: mytype
  symtree: 'x'             || symbol: 'x'
    type spec : (DERIVED mytype)
    attributes: (VARIABLE )
    value: mytype(() , NULL())

  code:
  ASSIGN MAIN__:x % i 42
  ASSIGN MAIN__:x % j(FULL) (/ 1 , 2 , 3 , 4 /)
  WRITE UNIT=6 FMT=-1
  TRANSFER MAIN__:x % i
  DT_END
  WRITE UNIT=6 FMT=-1
  TRANSFER MAIN__:x % j(FULL)
  DT_END
```

specification part

code

Structure of gfortran (ii):

- The translation stage converts the intermediate representation into TREE_SSA for the middle- and back-ends.
- This can be exposed with the option **–fdump-tree-original**
- Output is written to:

<myfilename.f90>.<version code>.original

the line x%j = [1,2,3,4]

produces…..

Reallocation on assignment

```
{
  integer(kind=4)[0:] * restrict D.3794;
  integer(kind=8) D.3795;
  integer(kind=8) D.3796;
  integer(kind=8) D.3797;
  static integer(kind=4) A.1[4] = {1, 2, 3, 4};
  integer(kind=8) D.3800;

  D.3794 = (integer(kind=4)[0:] * restrict) x.j.data;
  D.3795 = x.j.offset;
  D.3796 = x.j.dim[0].lbound;
  D.3797 = x.j.dim[0].ubound;
  D.3800 = NON_LVALUE_EXPR <D.3796>;
  {
    integer(kind=8) S.2;

    {
      logical(kind=4) D.3804;
      integer(kind=8) D.3805;
      logical(kind=4) D.3806;

      D.3804 = (integer(kind=4)[0:] * restrict) x.j.data == 0B;
      if (D.3804) goto L.1;
      if (x.j.dim[0].lbound + 3 != x.j.dim[0].ubound) goto L.1;
      goto L.2;
      L.1:;
      if (D.3804)
        {
          D.3805 = 0;
        }
      else
        {
          D.3805 = MAX_EXPR <x.j.dim[0].ubound - x.j.dim[0].lbound, -1> + 1;
        }
      D.3806 = D.3805 != 4;
      x.j.dim[0].lbound = 1;
      x.j.dim[0].ubound = 4;
      x.j.dim[0].stride = 1;
      x.j.offset = -NON_LVALUE_EXPR <x.j.dim[0].lbound>;
      D.3795 = x.j.offset;
      D.3800 = NON_LVALUE_EXPR <x.j.dim[0].lbound>;
      if ((integer(kind=4)[0:] * restrict) x.j.data == 0B)
        {
          x.j.data = (void * restrict) __builtin_malloc (16);
          x.j.dtype = {.elem_len=4, .rank=1, .type=1};
        }
      else
        {
          if (D.3806)
            {
              x.j.data = (void * restrict) __builtin_realloc ((void *) x.j.data, 16);
            }
        }
      L.2:;
    }
    S.2 = 0;
    while (1)
      {
        if (S.2 > 3) goto L.3;
        (*D.3794)[(S.2 + D.3800) + D.3795] = A.1[S.2];
        S.2 = S.2 + 1;
      }
    L.3:;
  }
}
```

“scalarized” assignment

```
D.3794 = (integer(kind=4)[0:] * restrict) x.j.data;
  L.2:;
}
S.2 = 0;
while (1)
  {
    if (S.2 > 3) goto L.3;
    (*D.3794)[(S.2 + D.3800) + D.3795] = A.1[S.2];
    S.2 = S.2 + 1;
  }
L.3:;
}
}
```

Essential for debugging compiler

Useless for most users, except….

….can be used to convert F20xx into C

Reallocation on assignment

```
if ((integer(kind=4)[0:] * restrict) x.j.data == 0B)
        {
          x.j.data = (void * restrict) __builtin_malloc (16);
          x.j.dtype = {.elem_len=4, .rank=1, .type=1};
        }
      else
        {
          if (D.3806)
            {
              x.j.data = (void * restrict) __builtin_realloc ((void *) x.j.data, 16);
            }
        }
```

"scalarized" assignment

```
94 = (integer(kind=4)[0:] * restrict) x.j.data;
 L.2:;
}
S.2 = 0;
while (1)
 {
  if (S.2 > 3) goto L.3;
  (*D.3794)[(S.2 + D.3800) + D.3795] = A.1[S.2];
  S.2 = S.2 + 1;
 }
L.3:;
}
```

```
{
 integer(kind=8) S.2;

 {
  logical(kind=4) D.3804;
  integer(kind=8) D.3805;
  logical(kind=4) D.3806;

  D.3804 = (integer(kind=4)[0:] * restrict) x.j.data == 0B;
  if (D.3804) goto L.1;
  if (x.j.dim[0].lbound + 3 != x.j.dim[0].ubound) goto L.1;
  goto L.2;
  L.1:;
```

```
D.3795 = x.j.offset;
D.3800 = NON_LVALUE_EXPR <x.j.dim[0].lbound>;
if ((integer(kind=4)[0:] * restrict) x.j.data == 0B)
 {
  x.j.data = (void * restr
  x.j.dtype = {.elem_len
 }
else
 {
  if (D.3806)
   {
    x.j.data = (void * re
   }
 }
```

```
S.2 = 0;
      while (1)
       {
        if (S.2 > 3) goto L.3;
        (*D.3794)[(S.2 + D.3800) + D.3795] = A.1[S.2];
        S.2 = S.2 + 1;
       }
      L.3:;
    }
```

Descriptors

- The array gfortran descriptor was designed at a time when memory usage was still an issue. This limited max dimensions to 7 and no allowance was made for pointers to components of arrays of derived types; *array_ptr => myderived(:)%component*

- The present array descriptors, although recently partially upgraded to fix above and for eventual F2018 CFI compliance, are still defective in one important respect: The 'dim' triplet should be {lbound,sm,extent} and not {lbound,stride,ubound}

- It appears that F2003 Parameterized Derived Types need a descriptor, similar to that of polymorphic entities: (Bug 82649 - (PDT) Invalid error for assumed parameters in ALLOCATE typespec
  - Allocate (matrix(rk, *, *) :: o_matrix) fails to pick up the declared LEN parameter
  - Both KIND and LEN parameters are fields in array elements

F20xx compliance

- gfortran was originally targeted to be a fully compliant F95 compiler with legacy g77 support. By ~2006/7 or so, this had been achieved. (primarily Paul Brook & Andy Vaught)

- F2003 compliance has more or less been achieved. The most significant defects are that the implementation of Parameterized Derived Types is only partially compliant, as described previously, and finalization still does not occur in all mandated cases.

- F2008 compliance is close. There are still some bugs in polymorphic assignment, especially where allocatable components or deferred string length are involved. A lot of the missing features in http://www.fortran.bcs.org/2017/fortran_2003_2008_compiler_support.pdf are now implemented.

- F2018 is primarily awaiting implementation of C/Fortran Interoperability features.

Ian D Chivers & Jane Sleightholme – November ’17
F2003 FEATURES

PDTs and Finalization need work

F2003	Absoft	Cray	Fujitsu	gfortran	IBM	Intel	NAG	Oracle	PGI
	14	8.4.0	2.0.3	7.2	15.1.5	18.0	6.1	8.8	16.4
Y	37	55	56	57	57	53	57	53	56
Y with notes	0	1	1	1	0	4	0	2	1
N	16	0	0	0	0	0	1	3	0
N with notes	0	0	0	0	0	0	0	0	0
P	1	0	0	0	1	0	0	0	0
P with notes	1	1	1	0	0	1	0	0	0
No information	3	1	0	0	0	0	0	0	1
	58	58	58	58	58	58	58	58	58
	Absoft	Cray	Fujitsu	gfortran	IBM	Intel	NAG	Oracle	PGI

Language Features (i)

PDTs and Finalization need work

Language Feature	Absoft	g95	gfortran	Intel	Lahey	NAG	Pathscale	PGI	Oracle
Fortran 95	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
TR15581 (Allocatable dummy arguments, derived type components etc.)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Fortran 2003 Support[2]	Partial	Partial	Partial	Yes	No	Yes	No	Yes	Yes
Fortran 2008 Support[2]	No	No	Partial	Partial	No	Partial	No	No	No
OpenMP	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Tabbed source form	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Symbolic names with $	Yes	Option	Option	Yes	Yes	Yes	Yes	Yes	Yes
Hollerith data	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
DOUBLE COMPLEX	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Varying length for named COMMON	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Mix numeric and character in COMMON and EQUIVALENCE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
INTEGER*n	1,2,4,8	1,2,4,8	1,2,4,8,16	1,2,4,8	1,2,4,8	1,2,4,8	1,2,4,8	1,2,4,8	1,2,4,8
LOGICAL*n	1,2,4,8	1,2,4,8	1,2,4,8,16	1,2,4,8	1,2,4,8	1,2,4,8	1,2,4,8	1,2,4,8	1,2,4,8
REAL*n	4,8,16	4,8,10	4,8,10,16	4,8,16	4,8,16	4,8,16	4,8	4,8	4,8,16

[2] Ian D Chivers & Jane Sleightholme – November ’17 (gfortran 7.2 and Intel 18.0)

Language Features (ii)

Language Feature	Absoft	g95	gfortran	Intel	Lahey	NAG	Pathscale	PGI	Oracle
VAX style debug (D) lines	Yes	Yes	Yes	Yes	No[1]	No	Yes	Yes	Yes
C style string constants (e.g. 'hello \n world')	Option	Option	Option	Option	No	No	No	Option	Option
VAX style STRUCTURE, RECORD, UNION etc.	Yes	No	Yes	Yes	Yes	No	No	Yes	Yes
Initialization in TYPE statements	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes
ENCODE and DECODE	Yes	No	No	Yes	Yes	No	Yes	Yes	Yes
variable format expressions using <>	Yes	No	No	Yes	Yes	No	No	No	Yes
\ edit descriptor	Yes	Yes	No	Yes	Yes	No	Yes	No	No
Q edit descriptor	Yes	No	No	Yes	Yes	No	Yes	Yes	Yes
$ edit descriptor	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
in line assembler	No	No	No	No	No	No	No	No	Partial
CRAY Pointers	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes
OPEN for "Transparent" or stream I/O (e.g. FORM='BINARY')	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Language Features (iii)

Language Feature	Absoft	g95	gfortran	Intel	Lahey	NAG	Pathscale	PGI	Oracle
Get Command Line	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Get Environment Variable	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Invoke External command	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Get Files in Directory	No	No	No	Yes	No	Yes	Yes	No	No
Get File Size, Date, Attributes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
VAX style system intrinsics (SECNDS etc.)	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Unix style system library (getenv, etime etc.)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Posix style library (pxfputc, pxfopen etc.)	No	No	No	Yes	No	No	Yes	No	No

Diagnostic Comparisons

Source	Run-time Error	Absoft	g95	gfortran	Intel	Lahey	NAG	Pathscale	PGI	Oracle
	Percentage Passes [1]	**34%**	**45%**	**53%**	**53%**	**92%**	**91%**	**38%**	**28%**	**42%**
	TFFT execution time with diagnostic switches (seconds)[2]	10	16	6	12	446	60		19	9
	Can mix checked and unchecked code	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes

Polyhedron Benchmarks

- gfortran benefits from the optimization provided by gcc back-end.
- Front-end has some optimization passes; e.g. to avoid library calls.

Linux/64 on Intel Processor

	Absoft 17.0	Absoft(AP) 17.0	gfortran 5.2.0	Intel 17.0	Intel(AP) 17.0	NAG 6.1	Oracle 12.5	PGI 16.9	open64 4.5.2.1
AC	5.01	4.96	6.36	4.47	4.44	7.32	18.91	6.84	5.07
AERMOD	11.35	11.51	15.60	11.52	12.26	18.64	11.13	11.98	15.79
AIR	3.50	2.18	3.16	2.60	1.98	4.94	2.78	3.32	3.46
CAPACITA	20.26	17.75	19.21	16.83	17.32	22.17	21.49	15.36	19.33
CHANNEL2	73.53	28.58	83.00	84.76	28.95	105.87	84.26	81.57	103.80
DODUC	19.28	19.34	18.70	15.12	14.97	24.20	15.96	18.11	18.72
FATIGUE2	63.09	66.65	67.08	55.62	55.75	117.37	82.94	89.12	77.66
GAS_DYN2	73.96	49.13	86.23	62.25	38.42	177.37	74.91	111.19	79.02
INDUCT2	83.68	76.03	80.99	71.82	50.96	132.20	138.92	127.38	144.11
LINPK	5.23	5.49	4.93	4.37	4.47	6.22	4.70	5.96	5.73
MDBX	9.68	7.98	8.07	6.47	4.85	8.68	8.54	9.08	9.35
MP_PROP_DESIGN	120.85	13.13	157.61	62.78	10.97	254.30	196.16	88.35	127.22
NF	8.13	8.21	7.30	7.58	7.54	9.00	8.98	8.47	7.96
PROTEIN	21.58	21.16	21.13	23.68	25.02	20.31	22.09	23.22	21.95
RNFLOW	15.55	15.23	13.66	12.52	9.65	16.66	17.34	17.00	21.54
TEST_FPU2	61.24	43.00	50.15	43.44	39.64	82.37	64.90	48.28	57.10
TFFT2	58.66	61.10	46.74	58.95	62.43	60.41	58.91	56.78	58.55
Geometric Mean	22.93	17.39	22.95	19.33	14.96	30.95	26.44	24.23	25.45

gfortran	gfortran -ffast-math -funroll-loops –param max-unroll-times=2 -Ofast -march=native
Intel	ifort -O3 -fast -ipo -nostandard-realloc-lhs

- 2 x Xeon E5-2643 3.30GHz quad core + 64-bit Linux Mint 17.1
- gfortran doing well among those without Automatic Parallelization turned on.
- Equals Intel when linked to Intel library
- *"Autoparallelization settings are not used on any other compilers because we found that they produced no significant performance benefits on this benchmark set. "*

Coarray support

- Native support currently only supports a single image.
  fcoarry=single

- Multiple image coarrays currently supported via MPI. Requires OpenCoarray library and runtime wrapper to be installed.
  fcoarray=lib

- An effort was made to include OpenMPI and OpenCoarrays in the gcc build but it has been recently decided to drop this because of maintenance and licensing issues.

- Work is underway to provide alternative implementation, which will provide native support: e.g.
  fcoarray=pthread images=10

Hosts and Targets

- aarch64*-*-*
- alpha*-*-*
- amd64-*-solaris2.10
- arm-*-eabi
- avr
- Blackfin
- DOS
- *-*-freebsd*
- h8300-hms
- hppa*-hp-hpux*
- hppa*-hp-hpux10
- hppa*-hp-hpux11
- *-*-linux-gnu
- i?86-*-linux*
- i?86-*-solaris2.10
- ia64-*-linux
- ia64-*-hpux*
- *-ibm-aix*
- iq2000-*-elf
- lm32-*-elf
- lm32-*-uclinux
- m32c-*-elf
- m32r-*-elf
- m68k-*-*
- m68k-uclinux
- microblaze-*-elf
- mips-*-*
- nds32le-*-elf
- nds32be-*-elf
- nvptx-*-none
- powerpc*-*-*
- powerpc-*-darwin*
- powerpc-*-elf
- powerpc*-*-linux-gnu*
- powerpc-*-netbsd*
- powerpc-*-eabisim
- powerpc-*-eabi
- powerpcle-*-elf
- powerpcle-*-eabisim
- powerpcle-*-eabi
- riscv32-*-elf
- riscv32-*-linux
- riscv64-*-elf
- riscv64-*-linux
- s390-*-linux*
- s390x-*-linux*
- s390x-ibm-tpf*
- *-*-solaris2*
- sparc*-*-*
- sparc-sun-solaris2*
- sparc-sun-solaris2.10
- sparc-*-linux*
- sparc64-*-solaris2*
- sparcv9-*-solaris2*
- c6x-*-*
- tilegx-*-linux*
- tilegxbe-*-linux*
- tilepro-*-linux*
- visium-*-elf
- *-*-vxworks*
- x86_64-*-*, amd64-*-*
- x86_64-*-solaris2.1[0-9]*
- xtensa*-*-elf
- xtensa*-*-linux*
- Microsoft Windows
- *-*-cygwin
- *-*-mingw32
- OS/2
- all ELF targets
  (SVR4, Solaris 2, etc.)

See the GFortran wiki for advice on binaries: https://gcc.gnu.org/wiki/GFortran

Note that the Windows bash shell supports gfortran-5.4.0

Professional support for gfortran

- Since Paul Brook disengaged himself from supporting gfortran, it has had no full-time professional involvement.
- The maintainers are primarily volunteers, most of whom are motivated initially by the need for some missing feature or other. *Most often, the maintainers are chemists, physicists or engineers who lack some (a lot?) of the required expertise.*
- The lack of a development plan shows. Frankly, there are quite significant chunks, especially in trans-*.c, that have grown like Topsy and are consequently a bit of a mess.
- We are very grateful for the help received from gcc maintainers but some dedicated, full-time (, professional) support is needed.
- *Ideas on how to obtain funding for this support?*

Benefits of continuing Fortran standardisation survey: interim results

Anton Shterenlikht Standards Officer, BCS Fortran Specialist Group

On balance, the responses are positive for gfortran. However, it is clear that the versions being used in the field are lagging far behind the development version 9.0.0.

Some quite considered remarks point out that bugginess in new features is shared between nearly all brands but the fact that they don't have the same bugs makes life difficult for those wanting to use them! (see third paragraph on next slide)

Of bugs in features that were mentioned explicitly, deferred character length comes out on top of the list. Bizarrely, though, more people claim to use PDTs than deferred character length and yet there are no complaints about the poor implementation in gfortran (mea culpa).

Maintainers can probably draw some development priorities from the document. ***It is clear, though, that bug fixing is the highest priority, as is obvious from the number of gfortran Problem Reports.***

Benefits of continuing Fortran standardisation survey: interim results

Some comments that stood out:

"As of today the only two things I can think of that will save Fortran is for Intel to follow NVIDIA/Portland Groups example and release a community edition (Ie free) version of their compiler and for the NVIDIA backed flang project to supplant gfortran as the ”open source” compiler. Just my 2 cents"

"There is no standard for the Fortran modules file format. This is terrible. …snip… The Fortran standards committee should have adressed this horribly lax policy a long time ago. Long overdue." (I can only agree. However, the array descriptor ABIs are not compatible either!)

"Gfortran 5.4.0, Nagfor 6207, ifort 17.0.5 None of these compilers supports the full (up to 2008) standard, and it is frustrating using trial and error to find the subset of useful features that they do all support. All three claim to support parts of the standard which they actually don’t.”

"gfortran - yes (although it would be nice if coarrays were better integrated, i.e. not having to use the opencoarray library explicitly).“

"Fortran is a dead language and its use should be banned by an act of Parliament."
(Well, I suppose that it would make a change from Brexit ☺)