Fortran on Linux — Free Compilers

- f2c
  - Written at AT&T.
  - Compiles f77 into C.
  - Comes with Fortran math and I/O libraries.
  - Accepts most, but not all, of f77 standard. Has a few extensions traditionally found in Unix Fortran compilers, but no DEC Fortran extensions.
- g77
  - The GNU f77 compiler. In beta release.
  - Uses the gcc backend. $\rightarrow$ Should be able to generate code for all machines on which gcc runs.
  - Uses f2c libraries.
  - Accepts most, but not all, of f77 standard. Supports more extensions than f2c, including `%val`, mil-std bitwise functions, etc.

Fortran on Linux — Commercial Products

- NAG f90.
  - Commercial product.
  - Full f90 compiler.
  - Compiles into ANSI C.
  - Not available for ELF?
- Microway?

g77

- Current version is 0.5.17. Work in 0.5.18 is in progress.

- Compiled CERNLIB and most of DØ offline software. ($\sim$ 1M lines.)
  - Made a extensions to the compiler to reduce changes needed in sources.
  - Only a handful of changes needed in CERNLIB.
  - In DØ code, have had to make minor changes in $\sim$ 160 modules (out of $\sim$ 5000).

- No problems yet with bad code being generated.
  - But not tested very thoroughly.
  - A few failures in genlib test suite remain to be investigated.

- Programs working:
  - PAW.
  - Basic DØ framework for reading, unpacking, and writing event data in Zebra format.

Some features being added in g77 0.5.18

- `-fno-second-underscore` option. External name styles:
  - Default (f2c):
    - `name` → `name_`
    - `two_words` → `two_words__`
  - `-fno-underscore`:
    - `name` → `name`
    - `two_words` → `two_words`
  - `-fno-second-underscore` (SGI):
    - `name` → `name_`
    - `two_words` → `two_words_`

Some features being added in g77 0.5.18

- Allow `SAVE` to follow `DATA`.

```
integer ivar
data ivar/0/
save ivar
```

- `INTEGER*1`, `INTEGER*2`, `LOGICAL*1` types.
  - Still don't work with intrinsics. (Maybe in final 0.5.18 release?)

- `BYTE` and `WORD` keywords.

- Passing typeless constants to bitwise intrinsics.

```
val = iand (val, 'ffff'x)
```

Constructs not accepted by g77 (1)

- String concatenation for an actual argument:

```
      character*(*) foo
      call bar (foo // 'asdf')
```

- ENTRY and FORMAT statements before IMPLICIT. (To be fixed in 0.5.18?)

```
      subroutine foo
      entry bar
      implicit none
```

- Usage of intrinsics in PARAMETER statements. (On long-term projects list.)

```
      character tab
      parameter (tab = char(8))
```

- Variable formats. (On long-term projects list.)

```
      integer i, width
      write (*, 10) i
 10   format (i<width>)
```

Constructs not accepted by g77 (2)

- Declaring a type for a subroutine.

```
integer bar
call bar
```

- `READONLY`, `DISPOSE`, and `CARRIAGECONTROL` keywords in an `OPEN` statement.

- Using an integer variable in a logical context.

```
integer ivar
logical lvar
if (ivar .and. lvar) call bar
```

- `TAND`, `COSD`, `SIND`, `ATAND`, `ACOSD`, and `ASIND` intrinsics.

- `RAN` intrinsic.

- `TYPE` I/O statement.

```
type *, 'hey dude'
```

Constructs not accepted by g77 (3)

- Omitting argument list in a `FUNCTION` declaration, when `IMPLICIT NONE` is used.

```
      integer function foo
      implicit none
      foo = 1
      end
```

- Omitting count on `X` format specifier.

```
      write (*, 10)
   10 format (x)
```

- Omitting field width for format specifications.

```
      integer ivar
      real    fvar
      write (*, 10)
 10   format (i, f)
```

- Omitting commas in format specifications (gives warning only).

```
      integer ivar
      write (*, 10) ivar
 10   format ('ivar is ' ivar)
```

Constructs not accepted by g77 (4)

- Commons with the same name as subroutines. (Allowed by DEC Fortran.)

```
subroutine foo
integer bar
common /foo/ bar
```

- Specifying a variable twice in `DATA` statements.

```
integer bar
data bar/0/
data bar/0/
```

- Using character strings in a numeric context.

```
integer ivar
if (ivar .eq. 'IVAR') call bar
```

- `EQUIVALENCE` of overlapping variables in `COMMON`.

```
real ary(2)
real  var1, var2
equivalence (var1, ary(1))
equivalence (var2, ary(2))
common /com/ var1, var2
```

Constructs not accepted by g77 (5)

- `INTEGER*2` as index of implied-`DO` loop.
- `INTEGER*2` as argument of intrinsic functions.
- `REAL*16` type.
- DEC Fortran direct I/O.

```
read (lun'rec, '(i4)') var
```

- DEC Fortran `PARAMETER` syntax.

```
parameter par = 5
```

- `STRUCTURE` definitions.