Mixing C and Fortran

Intrinsic module `iso_c_binding` defines the following variables

Type	Named constant	C type or types
`INTEGER`	`C_INT`	int, signed int
	`C_SHORT`	short int, signed short int
	`C_LONG`	long int, signed long int
	`C_LONG_LONG`	long long int, signed long long int
	`C_SIGNED_CHAR`	signed char, unsigned char
	`C_SIZE_T`	size_t
	`C_INT_LEAST8_T`	int_least8_t
	`C_INT_LEAST16_T`	int_least16_t
	`C_INT_LEAST32_T`	int_least32_t
	`C_INT_LEAST64_T`	int_least64_t
	`C_INT_FAST8_T`	int_fast8_t
	`C_INT_FAST16_T`	int_fast16_t
	`C_INT_FAST32_T`	int_fast32_t
	`C_INT_FAST64_T`	int_fast64_t
	`C_INTMAX_T`	c intmax_t
`REAL`	`C_FLOAT`	float, float _Imaginary
	`C_DOUBLE`	double, double _Imaginary
`COMPLEX`	`C_LONG_DOUBLE`	long double, long double _Imaginary
	`C_COMPLEX`	_Complex
	`C_DOUBLE_COMPLEX`	double _Complex
	`C_LONG_DOUBLE_COMPLEX`	long double _Complex
`LOGICAL`	`C_BOOL`	_Bool
`CHARACTER`	`C_CHAR`	char

Type `C_PTR` is interoperable with C pointer and `C_NULL_PTR` with `NULL` in C.

Keyword `BIND(C)` is ensuring interoperability with C, for example derived type may be passed to C as a structure

```
TYPE, BIND(C) :: MYTYPE
:
END TYPE MYTYPE
```

Example

```
typedef struct {
    int x;
    int y
    float val;
} element
```

is interoperable with

```
use iso_c_binding
type, bind(c) :: element
    integer(C_INT) :: x
    integer(C_INT) :: y
    real(C_FLOAT) :: val
end type element
```

Fortran can call C functions is an interface has been defined with proper binding, in general

```
function fortran_name(arg1, arg2, arg3, ...) bind(C, NAME='C_name')
```

for example

```
interface
    integer(C_INT) function function_name(a,b,c) bind(C,
name='C_function_name')
        use iso_c_binding
        implicit none
        ...
     end function function_name
end interface
```

An example program, C file

```c
int C_name(int a, int b, int c){
  // This is a simple C function
  return a+b+c;
}
```

and Fortran file

```fortran
program c_function_call

  use iso_c_binding

  implicit none

  integer(c_int) :: a,b,c
  integer(c_int) :: d

  interface
    integer(c_int) function triplet(a,b,c) bind(c,name='C_name')
      use iso_c_binding
      implicit none
      integer(c_int), value :: a,b,c
    end function triplet
  end interface

  a = 1
  b = 2
  c = 3

  d = triplet(a,b,c)

  print *,'d = ', d

end program c_function_call
```

a `Makefile` for this case will be

```
FC=gfortran
FCFLAGS=

CC=gcc
CFLAGS=

LIBS=
PROGRAM=example.x

F_SRC=example.o
C_SRC=function.o

all: $(F_SRC) $(C_SRC)
  $(FC) $(FCFLAGS) $(F_SRC) $(C_SRC) $(LIBS) -o $(PROGRAM)


%.o: %.f90
  $(FC) $(FCFLAGS) -c $<

%.o: %.c
  $(CC) $(CFLAGS) -c $<

clean:
  rm -r *.o *.mod $(PROGRAM)
```