Testing the emulation of ISO_C_BINDING in Fortran 95

Aleksandar Donev

June 25, 2002

1 Testing ISO_C_BINDING

Support on the C side:

"WEAVE.c" 1.1 ≡

```
@m ALLOCATE_IN_C 1

#include <stdlib.h>
#include <stdbool.h>

      // Give the prototype for the Fortran function we are calling:
   void *_BoundProcedure(AllocateTestArray_real_F)(_PassedByValue(size_t, n),
        _PassedByValue(_Bool, dp));

   void *_BoundProcedure(AllocateTestArray_real_C)(_PassedByValue(size_t, n),
          _PassedByValue(_Bool, dp))
      {    // Arguments passed by value:
      _AcceptValue(size_t, n)
      _AcceptValue(_Bool, dp)

        // Local variables
      int i;
      float *float_array;
      double *double_array;

      printf("Values passed: n=%d dp=%d\n", n, dp);

@#if (ALLOCATE_IN_C)
      if (dp)
        {
        printf("Allocating double array!\n");
        double_array = (double *)(malloc(n * sizeof(double)));
        if (float_array ≠ NULL)
          for (i = 0; i < n; i++)
            double_array[i] = (double) i;
```

```
      return ((void *) double_array);
      }
    else
      {
      printf("Allocating float array!\n");
      float_array = (float *)(malloc(n * sizeof(float)));
      if (float_array ≠ NULL)
        for (i = 0; i < n; i++)
          float_array[i] = (float) i;
      return ((void *) float_array);
      }
@#else
    return (_BoundProcedure(AllocateTestArray_real_F)(_PassByValue(n), _PassByValue(dp)));
@#endif
    }

  void _BoundProcedure(free_C)(_PassedByValue(void *, space))
    {
    _AcceptValue(void *, space)
    free(space);
    }

  struct point2D
    {
    int screen;
    float x, y;
    struct point2D *next;
    };

  void _BoundProcedure(DisplayPoint)(struct point2D *point)
    {
    struct point2D *next_point;

    next_point = point→next;
    printf("The point has: screen=%d, x=%f, y=%f \n", point→screen, point→x, point→y);
    printf("The next point has: screen=%d, x=%f, y=%f \n", next_point→screen,
        next_point→x, next_point→y);
    }

  typedef void FortranSubroutine(_PassedByValue(int, test));

  void _BoundProcedure(ExternalProcedures)(_PassedByValue(void *, f_address))
    {
    _AcceptValue(void *, f_address)
    FortranSubroutine *f_subroutine;
    int n = 113;

      /* This conversion should not be legal in C99 as object and function pointers are not interoperable.
        One can delcare the argument to this function as a correct procedure pointer. Note that there is
        no generic procedure pointer in C. */

    f_subroutine = (FortranSubroutine *) f_address;    // Convert from the void pointer

      // Call the Fortran subroutine
    printf("C calling procedure at integer address: %d \n", f_subroutine);
    f_subroutine(_PassByValue(n));
    }
```

§. `@I "C_Interop.hweb"` *Section(s) skipped...*

§. `@I "C.formatting.hweb"` *Section(s) skipped...*

§. `@I "Common.macros.hweb"` *Section(s) skipped...*

§. `@I "C.macros.hweb"` *Section(s) skipped...*