Fortran and X [4]

Although you cannot call Xlib functions directly from Fortran, you can mix Fortran functions and C functions in one binary file. Using the techniques described in this section, you can use Fortran for computation, and write Fortran-callable C functions that use Xlib to handle the graphics. Similar interfaces can be used for other languages, such as Pascal and Ada.

The following differences between Fortran and C must be considered:

- A Fortran restriction requires that the subroutine or function name be in uppercase letters when Fortran subroutines are called from C, and vice versa.
- The Fortran calling sequence is call-by-address, and the C calling sequence is call-by-value. Therefore, any parameters that are passed from C to Fortran or from Fortran to C must be pointers.
- Fortran and C handle character strings differently.

Figure 2, page 20, shows the architecture of a client that uses both Fortran and C. It consists of a main program that interfaces with functions and subroutines.

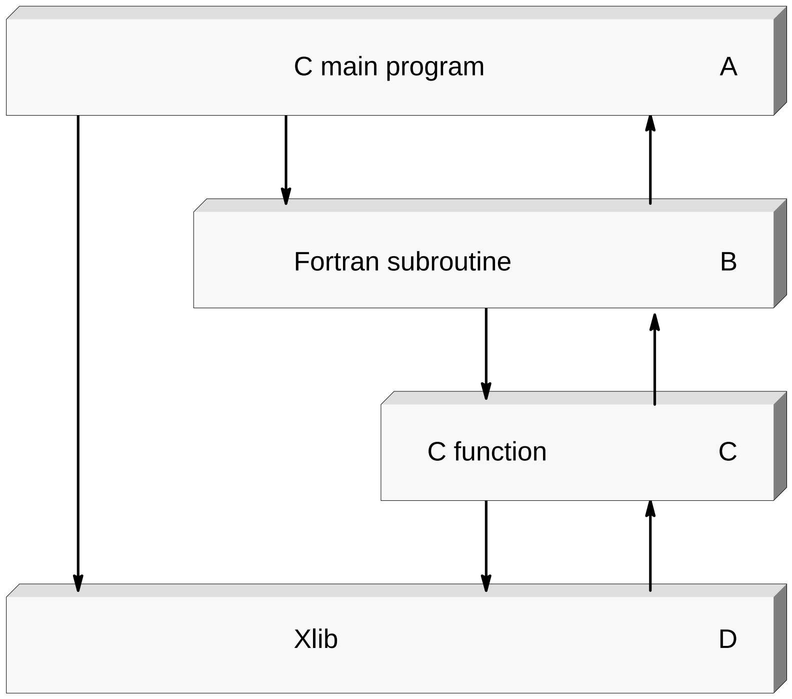

Figure 2. Architecture of C and Fortran client

The main program, represented by box A, performs tasks such as initialization, opening the display, creating windows, creating command buttons, and so on. If the main program is written using only Xlib, it contains its own event-handling loop; if it uses the Intrinsics toolkit, it contains a call to `XtMainLoop`. You can call the Fortran subroutine represented by box B from the main program's event loop if Xlib is being used; it can be called by a *callback procedure* (a routine called when a button is pressed) if the Intrinsics toolkit is being used. To do graphic output, the Fortran subroutine calls a C function (box C) to generate the appropriate Xlib calls. Xlib (box D) generates the protocol requests.

To illustrate these techniques, the remainder of this section contains examples of code. You can find complete listings of the main program (`mandelf`) in "Complete `xmandelf` Source Code," page 27.

The following example shows C calling Fortran and Fortran calling C. The main program uses the Intrinsics toolkit (Xt), and declares a *callback table* (a set of callback procedures).

```
void mandel ();
XtCallbackRec mandel_callbacks[] = {
    { mandel, NULL },
    { NULL, NULL },
},
```

The main program also provides packaging for the rest of the code by performing toolkit initialization, creating widgets such as command buttons, and invoking Xt's main loop, as follows:

```
main(argc,argv)
{
        Widget toplevel, mandelbutton;
        Arg argies[10];
        toplevel = XtInitialize(”Anyname”, ”Anyname”, NULL, 0, &argc, argv);
         -
         -
         -
        XtSetArg( argies[i], XtNcallback, (XtArgVal) mandel_callbacks); i++;
        mandelbutton = XtCreateManagedWidget(”mandel”, commandWidgetClass,
                 form, argies, i);
         -
         -
         -
        XtRealizeWidget(toplevel);
        XtMainLoop();
}
```

The callback table referred to in the previous example is a set of C language routines that will be called when a certain command button named mandel is pressed. The following routine, also called mandel, handles the callback and sets up to call the Fortran subroutine GENERATE, which does the actual calculations.

This mandel routine might be in the same file as the main routine.

Note

Because Fortran is a call-by-address process, all of the parameters on the GENERATE subroutine call are addresses.

```
mandel(w, closure, call_data)
    Widget w;
    caddr_t closure;
    caddr_t call_data;
{
        -
        -
        -

    GENERATE(&wheight,&y0,&incry,&wwidth,&x0,&incrx,&iter);
}
```

The following example is an excerpt from the Fortran subroutine named GENERATE, which the mandel function called. When it completes its calculations, it calls a C function (called FPUTI) to build the Xlib graphics requests.

```
subroutine GENERATE(wheight,y0,incry,wwidth,x0,incrx,iter)
       -
       -
       -
    call FPUTI(ixlo,iylo,index)
```

The following code is the C function `FPUTI`, which must convert parameters from the form that Fortran produces to the form that the call to Xlib requires (in this case, the `XPutImage` call):

```
FPUTI(pix,piy,index)
int *pix,*piy;
int index[64*64];
{
       -
       -
       -
       XPutImage(draw_d,draw_win,draw_gc,xip,0,0,*pix,*piy,VSIZE,VSIZE);
}
```