71.3.5 G_NOARGS

Whenever a Perl subroutine is called using one of the *call_** functions, it is assumed by default that parameters are to be passed to the subroutine. If you are not passing any parameters to the Perl subroutine, you can save a bit of time by setting this flag. It has the effect of not creating the `@_` array for the Perl subroutine.

Although the functionality provided by this flag may seem straightforward, it should be used only if there is a good reason to do so. The reason for being cautious is that even if you have specified the G_NOARGS flag, it is still possible for the Perl subroutine that has been called to think that you have passed it parameters.

In fact, what can happen is that the Perl subroutine you have called can access the `@_` array from a previous Perl subroutine. This will occur when the code that is executing the *call_** function has itself been called from another Perl subroutine. The code below illustrates this

```
sub fred
  { print "@_\n"  }

sub joe
  { &fred }

&joe(1,2,3) ;
```

This will print

```
1 2 3
```

What has happened is that `fred` accesses the `@_` array which belongs to `joe`.

71.3.6 G_EVAL

It is possible for the Perl subroutine you are calling to terminate abnormally, e.g., by calling *die* explicitly or by not actually existing. By default, when either of these events occurs, the process will terminate immediately. If you want to trap this type of event, specify the G_EVAL flag. It will put an *eval { }* around the subroutine call.

Whenever control returns from the *call_** function you need to check the `$@` variable as you would in a normal Perl script.

The value returned from the *call_** function is dependent on what other flags have been specified and whether an error has occurred. Here are all the different cases that can occur:

- If the *call_** function returns normally, then the value returned is as specified in the previous sections.
- If G_DISCARD is specified, the return value will always be 0.
- If G_ARRAY is specified *and* an error has occurred, the return value will always be 0.
- If G_SCALAR is specified *and* an error has occurred, the return value will be 1 and the value on the top of the stack will be *undef.* This means that if you have already detected the error by checking `$@` and you want the program to continue, you must remember to pop the *undef* from the stack.

See *Using G_EVAL* for details on using G_EVAL.