67.3.18 Documenting your Extension

There is absolutely no excuse for not documenting your extension. Documentation belongs in the .pm file. This file will be fed to pod2man, and the embedded documentation will be converted to the manpage format, then placed in the blib directory. It will be copied to Perl's manpage directory when the extension is installed.

You may intersperse documentation and Perl code within the .pm file. In fact, if you want to use method autoloading, you must do this, as the comment inside the .pm file explains.

See *perlpod* for more information about the pod format.

67.3.19 Installing your Extension

Once your extension is complete and passes all its tests, installing it is quite simple: you simply run "make install". You will either need to have write permission into the directories where Perl is installed, or ask your system administrator to run the make for you.

Alternately, you can specify the exact directory to place the extension's files by placing a "PREFIX=/destination/directory" after the make install. (or in between the make and install if you have a brain-dead version of make). This can be very useful if you are building an extension that will eventually be distributed to multiple systems. You can then just archive the files in the destination directory and distribute them to your destination systems.

67.3.20 EXAMPLE 5

In this example, we'll do some more work with the argument stack. The previous examples have all returned only a single value. We'll now create an extension that returns an array.

This extension is very Unix-oriented (struct statfs and the statfs system call). If you are not running on a Unix system, you can substitute for statfs any other function that returns multiple values, you can hard-code values to be returned to the caller (although this will be a bit harder to test the error case), or you can simply not do this example. If you change the XSUB, be sure to fix the test cases to match the changes.

Return to the Mytest directory and add the following code to the end of Mytest.xs:

```
void
statfs(path)
        char *  path
    INIT:
        int i;
        struct statfs buf;

    PPCODE:
        i = statfs(path, &buf);
        if (i == 0) {
                XPUSHs(sv_2mortal(newSVnv(buf.f_bavail)));
                XPUSHs(sv_2mortal(newSVnv(buf.f_bfree)));
                XPUSHs(sv_2mortal(newSVnv(buf.f_blocks)));
                XPUSHs(sv_2mortal(newSVnv(buf.f_bsize)));
                XPUSHs(sv_2mortal(newSVnv(buf.f_ffree)));
                XPUSHs(sv_2mortal(newSVnv(buf.f_files)));
                XPUSHs(sv_2mortal(newSVnv(buf.f_type)));
                XPUSHs(sv_2mortal(newSVnv(buf.f_fsid[0])));
                XPUSHs(sv_2mortal(newSVnv(buf.f_fsid[1])));
        } else {
                XPUSHs(sv_2mortal(newSVnv(errno)));
        }
```

You'll also need to add the following code to the top of the .xs file, just after the include of "XSUB.h":