[This document, a short excerpt from the first edition of *Software Portability with imake*, describes two instances of intertwining between the system and project sections of the X11R5 configuration files. Both instances were untangled as of R6.]

A separation of powers

For the most part, information in the system and project sections doesn't overlap. In theory, if you're using the X files, you could use them to configure a different project by plugging in a different *Project.tmpl* file. (This would also be true for any other set of files that splits parameters into system and project sections.)

In practice, it's not always easy to achieve a clean separation of configuration information into system and project components, so the breakdown is sometimes imperfect. For instance, the structure of the system's manual page hierarchy doesn't vary according to the project you're developing, so you might reasonably suppose that parameters relating to installation of manual pages into that hierarchy would go in the general system description section of the template.

That's not always true in the X11 files. For instance, two symbols related to manual page installation are `InstManFlags` (which specifies flags to use for the *install* command) and `ManDirectoryRoot` (which specifies the root of the manual page hierarchy). `InstManFlags` is in the system parameters section of template, as you'd expect. But `ManDirectoryRoot` is tied to an X-related symbol `XmanLocalSearchPath` (indentation added):

```
#ifndef ManDirectoryRoot
#  ifdef ProjectRoot
#    define ManDirectoryRoot Concat(ProjectRoot,/man)
#    ifndef XmanLocalSearchPath
#      define XmanLocalSearchPath ManDirectoryRoot
#    endif
#  else
#    if SystemV4
#      define ManDirectoryRoot /usr/share/man
#    else
#      define ManDirectoryRoot /usr/man
#    endif
#  endif
#endif
```

Due to this entanglement with an X-specific symbol, the definition of `ManDirectoryRoot` occurs in *Project.tmpl*.

A more convoluted example involves the specification of the *imake* command used to build a *Makefile*. In *Project.tmpl* the variable `IMAKE` is assigned a value to specify the name of the *imake* program. Where is `IMAKE` used? In `IMAKE_CMD`, which specifies the command line used to run *imake*. To find `IMAKE_CMD`, you go to *Imake.tmpl*. If you look at the definition of `IMAKE_CMD` there, you find it uses `IRULESRC` to indicate the directory in which the configuration files should be found. Where is `IRULESRC`? Back in *Project.tmpl*!

Initially, you might view these excursions back and forth between the various configuration files with some sense of adventure. But as you trace through their turnings and begin to wind your way further and further into their passageways, the true nature of the situation begins to dawn on you. What you thought was a quest to unlock the mysteries of the configuration files, is now revealed with frightful clarity as a trap for the unwary. The true secret of the files is this: they are the Labyrinth of Daedalus. Trapped inside, you can only await the approach of the menacing inhabitant concealed within: the Minotaur--in body half-man, half-bull; in spirit unbridled fury throughout. And now, because none escape the Labyrinth alive, only dread and despair lie before you--the fate of all who dare explore the X11 configuration files.

Well, actually, purple prose aside (or whatever my editors left of it), things aren't quite so bad as that. You're still here alive and reading, after all. But the sometimes maze-like interaction between the various

-- --

X11 configuration files does take its toll on ease of comprehension. For the case just described (the definition of IMAKE_CMD), the difficulty in cleanly partitioning the information in the files comes about because they not only use a version of *imake* distributed with and located within the X project tree, they provide a way to install *imake* on your system for public use as well. This is a nice convenience because then you can use *imake* for other projects. However, it makes it more difficult to decide whether *imake*-related information is properly classified as a property of the X project or of the system at large. When you're using the version of *imake* located within the source tree, its location is clearly a project parameter. But after *imake* is installed in a public directory, it isn't part of the X project per se, and the location can be considered a system parameter.