General Controls Patch Technical Information

For Programmers Only

Here is the down and dirty info about what the General Controls panel does to cause a conflict, and what I propose to do about it.

Among other things, the INIT resource in General Controls patches _Pack3, the Standard File trap. Its purpose in doing so is to implement the control on the initial directory shown in directory dialogs, as shown in the lower right corner of the panel.

Most trap patches are of two types:

1. A standard “head patch” leaves the original arguments and return address on the stack, performs some action, and jumps to the previous address of the trap.

2. A standard “tail patch” calls the previous address of the trap as a subroutine, which involves copying the arguments, perhaps with modifications. This type is easier to write in a high-level language.

The patch in General Controls is an odd hybrid. It’s like a tail patch, in that control returns to the patch after the original trap routine has executed, but like a head patch, it does not make a copy of the arguments. Here is what it does: It saves the return address from the stack into a fixed location in its own code block, and then replaces the return address on the stack with an address in its code. Then it jumps to the previous trap address. When the old trap code is done, it returns to the General Controls patch, which does more work and returns to the address that was saved in its code block.

Now, do you see why the General Controls patch is not reentrant? Let’s say an application, call it WordMeister, calls StandardPutFile. The GC patch saves the return address, which points to somewhere in WordMeister’s code. Then, while that dialog is still displayed, you invoke an OtherMenu external, say Delete, that calls StandardGetFile. The GC patch starts up again, and saves the return address, which points to somewhere in Delete’s code. But notice that this has wiped out the return address that pointed into WordMeister’s code! Thus it has planted the seeds of destruction. When the Delete dialog is dismissed, it correctly returns to the Delete code, which finishes up and returns control to WordMeister’s save dialog. But when the save dialog is dismissed, the GC patch tries to return to Delete

again, which is no longer running.

To fix this problem without actually reprogramming General Controls, my idea was to patch _Pack3 after General Controls, in such a way that the return address seen by the GC patch is always the same. In this way, when the GC patch overwrites a return address with another, no harm is done. The return address for my patch is saved on the stack, above the arguments, rather than in a fixed location.

To ensure that my _Pack3 patch is installed immediately after GC's patch, I add my patch as an INIT resource in the General Controls file, and change the old INIT resource to type 'xNIT'. When my INIT runs, it loads and executes the 'xNIT' resource, then installs its patch.