

Re: Ah've got them Function Pointer blues

Source: http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2008-08/msg02809.html

- *From:* Ben Bacarisse <ben.usenet@xxxxxxxx>
 - *Date:* Fri, 22 Aug 2008 15:15:12 +0100
-

"MikeC" <Mike.Best@xxxxxxxxxxxxxxxx> writes:

"August Karlstrom" <fusionfile@xxxxxxxx> wrote in message

<snip>

How come the functions have different signature and what do you want to achieve? Please, tell us a bit more about the underlying problem.

<snip>

However, as you ask ...

I haven't finished specifying the program yet – I keep having bigger and better ideas – but I want to write a text macro engine. It will interpret commands from a (text) command file, executing them on an input (read only) file, and producing an output file. The commands will be, for example

```
copy off // don't copy anything from the input file to the output file
find_forward "a text string"
move_left 6 // characters
copy on // this causes any character scanned by the cursor to be copied to
the output file
loop 6 // times
{ <more commands>
}
etc....
```

... you get the idea.

I wanted to run through the command (program) file and compile it into a forth-like stack (yes, I'm a dinosaur), with each stack element being a structure, which contains, among other things, a pointer to the function that will execute the command. The commands do different things, so they have different signatures, hence my question.

This does not follow automatically from what you have said. I have done similar things and there is no reason why the functions /need/ to

Re: Ah've got them Function Pointer blues

have different types. One way to look at it is that the functions all modify the state of a "text copying virtual machine". I.e. each one takes a pointer to a structure that describes that program's state: the file positions, variable bindings, stack and so on.

While there is a cost here (all the state get piled into one place) the payoff is that you don't need a big switch effectively doing run-time type checking. Remember I said that at the time of the call the function pointer must be cast to the correct type? You end up with a messy if-then-else chain (or a switch) resolving the correct way to call each of the various kinds. It is well worth trying to get them all to be the same.

--

Ben.

.