Proprietary plug-ins for GPL programs: what about interpreted languages? [closed]

Asked 16 years, 3 months ago Modified 6 years, 6 months ago

Viewed 3k times

9

Closed. This question does not meet Stack Overflow guidelines. It is not currently accepting answers.

This question does not appear to be about programming within the scope defined in the help center.

Closed 9 years ago.

Improve this question

I am developing a GPL-licensed application in Python and need to know if the GPL allows my program to use proprietary plug-ins. This is what the FSF has to say on the issue:

> **If a program released under the GPL uses plug-ins, what are the requirements for the licenses of a plug-in?**

> It depends on how the program invokes its plug-ins. If the program uses fork and exec to invoke plug-ins, then the plug-ins are separate programs, so the license for the main program makes no requirements for them.
>
> If the program dynamically links plug-ins, and they make function calls to each other and share data structures, we believe they form a single program, which must be treated as an extension of both the main program and the plug-ins. This means the plug-ins must be released under the GPL or a GPL-compatible free software license, and that the terms of the GPL must be followed when those plug-ins are distributed.
>
> If the program dynamically links plug-ins, but the communication between them is limited to invoking the ‘main’ function of the plug-in with some options and waiting for it to return, that is a borderline case.

The distinction between fork/exec and dynamic linking, besides being kind of artificial, doesn't carry over to interpreted languages: what about a Python/Perl/Ruby plugin, which gets loaded via `import` or `execfile`?

(edit: I understand why the distinction between fork/exec and dynamic linking, but it seems like someone who wanted to comply with the GPL but go against the "spirit"

--I don't-- could just use fork/exec and interprocess communication to do pretty much anything).

The best solution would be to add an exception to my license to explicitly allow the use of proprietary plugins, but I am unable to do so since I'm using Qt/PyQt which is GPL.

python plugins open-source licensing interpreted-language

Share

Improve this question

Follow

edited Jun 20, 2020 at 9:12

Community Bot
1 ● 1

asked Aug 28, 2008 at 0:26

dF.
75.7k ● 31 ● 135 ● 137

4 I'm voting to close this question as off-topic because it is about licensing and legal issues, not programming or software development. See here for details, and the help center for more. – Pang Jun 12, 2015 at 1:49

3 Answers

Sorted by: Highest score (default)

> he distinction between fork/exec and dynamic linking, besides being kind of artificial,

7

I don't think its artificial at all. Basically they are just making the division based upon the level of integration. If the program has "plugins" which are essentially fire and forget with no API level integration, then the resulting work is unlikely to be considered a derived work. Generally speaking a plugin which is merely forked/exec'ed would fit this criteria, though there may be cases where it does not. This case especially applies if the "plugin" code would work independently of your code as well.

If, on the other hand, the code is deeply dependent upon the GPL'ed work, such as extensively calling APIs, or tight data structure integration, then things are more likely to be considered a derived work. Ie, the "plugin" cannot exist on its own without the GPL product, and a product with this plugin installed is essentially a derived work of the GPLed product.

So to make it a little more clear, the same principles could apply to your interpreted code. If the interpreted code relies heavily upon your APIs (or vice-versa) then it would be considered a derived work. If it is just a script that executes on its own with extremely little integration, then it may not.

Does that make more sense?

Share Improve this answer Follow

answered Aug 28, 2008 at 0:33

jsight

28.4k ● 25 ● 109 ● 140

1

@Daniel

> The distinction between fork/exec and dynamic linking, besides being kind of artificial, doesn't carry over to interpreted languages: what about a Python/Perl/Ruby plugin, which gets loaded via import or execfile?

I'm not sure that the distinction **is** artificial. After a dynamic load the plugin code shares an execution context with the GPLed code. After a fork/exec it does not.

In anycase I would guess that `import`ing causes the new code to run in the same execution context as the GPLed bit, and you should treat it like the dynamic link case. No?

Share Improve this answer Follow

answered Aug 28, 2008 at 0:32

dmckee --- ex-moderator kitten

101k ● 25 ● 146 ● 235

0

How much info are you sharing between the Plugins and the main program? If you are doing anything more than just executing them and waiting for the results (sharing no data between the program and the plugin in the process) then you could most likely get away with them being proprietary, otherwise they would probably need to be GPL'd.

Share Improve this answer

Follow

answered Aug 28, 2008 at 0:33

Alasdair

778 ● 7 ● 10