C&I
www.scl.org GPL

GPL – the Linking Debate

With GPL very much in the spotlight, **Andrew Katz** focuses on the linking mechanism used by programmers and its special implications.

One of the best-kept secrets of programmers-turned-lawyers is that there is a lot in common between drafting and programming[1] This in part, explains why GPL[2] version 2 is such a great piece of drafting, as a collaboration between a coder (Richard Stallman) and a lawyer (Eben Mogien)[3] I'm assuming more readers of this piece will have experience of drafting than of coding so I'll use a drafting analogy to explain what linking is

Imagine you're drafting a contract You want to insert a 'termination for insolvency' clause, and you need a definition of 'unable to pay its debts as they fall due' Being creatively lazy" like all the best lawyers (and programmers) you II want to use a precedent There are essentially three ways of achieving the aim

1. You can cut and paste the relevant words from your firm's precedent.
2 Make a note to your secretary to import the relevant words from your firm's precedent, checking the cross references. numbering and defined terms
3 You can make a reference to s 123 of the Insolvency Act 1986

Likewise, if you are a programmer writing a program, and you want a piece of code which terminates the program if the computer runs low on memory[5] you have three options:

1 Copy and paste a piece of code from somewhere else
2 You can 'include a reference to a static code library (which is, essentially, a set of precedents, sitting on your disk)
3 You can 'call' a subroutine in an external code library which carries out the same function

Copying and pasting is directly comparable When you include' a piece of code, in practice, what the compiler does (like a good secretary) is to import the text of the code from the precedent system somewhere else on the computer, check all the cross references and variables (essentially making sure the numbering scheme in your contract is consistent, and that the terminology is correct - fixing references to 'Vendor' in the precedent so that they read Seller' to match the rest of the contract, for example) The process of importing and correcting the cross-references, etc is called 'linking' and works in a virtually identical way whether you are drafting or coding If your imported code doesn't make sense (because something in the imported code, or text, needs to be defined) then a good compiler (and a good secretary) will point out the error and you can have another go

Method one and method two will both result in a stand-alone piece of code (or contract). When you send it out to your client, the client won't need to cross-refer to another document to understand it This variety of [inking is called 'static linking'

Method three is more interesting If your drafting says 'A company is insolvent if it is unable to pay its debts as they fall due as defined in section 123 of the Insolvency Act 1986' your client wont be able to understand exactly what you mean unless they have a copy of s 123 of the Insolvency Act 1986 to hand

Likewise when you are writing computer code you can make a reference to an external library function.

When you send the code to the client unless you know that they already have the library in place on their computer you will have to send them the library as well If you look in the \windows\system directory your Windows computer, you will see files ending ' dil': 'DLL stands for'dynamic link library'. These libraries perform tasks as diverse as providing mathematical functions decoding MP3s and video files interfacing with hardware and providing database functions. Many of these libraries are already supplied with Windows and will be updated from time to time through the Windows update function, to add functionality and correct errors[6] Subsequent versions are designed to be backwards compatible (so that programs written for earlier versions will continue to function if the DLL is upgraded to a later version) When you run a program which uses DLLs, the operating system first checks that the DLL is there (and generates an error - or may offer to download it) and then links to the functions in the DLL just as the program needs them The linking takes place each time the program is run (not just the one time when the program is compiled) - hence the term 'dynamic.

All major operating systems have a mechanism for calling dynamic link libraries: they have a number of advantages over static linking

1 The binaries (object code files) are smaller, as they don't have to include the code included in the DLL
2 If there is a bug in the DLL (or the core application) then only that component needs fixing
3 One single DLL (for example a video codec) may be used by a number of different programs, so each program doesn t need to incorporate code for that function thereby saving space
4 The DLL may be upgraded and maintained by a third party, and not the original coder
5 If you're a closed source software house, its much easier to keep the internal workings of your library and its source code, confidential if you release it as a binary DLL with a published interface
6 You can have different DLLs for different hardware environments but the underlying application remains the same
7. A DLL can be written in a different language from the application that calls it

It's perfectly possible to write a program using calls to DLLs by relying on their published specification, without ever calling the DLL at ali. Of course, you can't test it or run it unless you have the DLL in question (or a compatible one), but it is possible. This is an important point for reasons I'll come onto later[7]

So where does the GPL come in?

(Incidentally. I'm concentrating on GPL2 here The overwhelming majority of GPL software is released under GPL2 Although there are some major projects which have announced transition to GPL3 (such as Samba), the reality is that GPL2 will be the overwhelmingly popular GPL licence (and arguably open source/free software licence) for the next few years at least if not indefinitely Aiex Newson in the accompanying article

points out many of the issues with GPL3) I'm also assuming a general familiarity with the terms of GPL2

The Free Software Foundation is Richard Staffman's baby[8] and the guardian of the GPL, in all its forms The FSF FAQ[9] states very simply that any software linking to a library, either statically or dynamically, which is covered by GPL2[10] must itself be released under GPL2

Luckily, the FSF is not a judicial body, and the GPL will, ultimately be subject to interpretation in the courts of the relevant jurisdiction The FSF FAQ is not a legally binding document "

The relevant questions are:

1 Does linking an application dynamically to a GPL library render that application subject to the GPL?
2 Does linking an application statically to a GPL library render that application subject to the GPL?
3 Does cutting and pasting GPL code into an application render that application subject to the GPL?

Does linking an application dynamically to a GPL library render that application subject to the GPL?

No. !n spite of the FSF's comments in their FAQ, linking a piece of software to GPL code dynamically does not render that application subject to the GPL There are several reasons for this:

1 It is the end-user at run-time, and not the programmer who causes the application to interface with the DLL It follows that the programmer cannot be in breach of copyright as he has not undertaken the act which would be an alleged breach of the GPL[12]
2 The alleged act of infringement could take place after the creation of the application has been completed.
3 The applications calls to DLL may be equally applicable to a number of different DLLs each with consistent APIs all of which are under separate copyright ownership and which may be subject to different licensing terms, and it is the user's choice which DLL they wish to use.[13]

Note that none of this reasoning even requires looking at the wording of the GPL itself: it is a consequence of the fact that the GPL, as a software licence only purports to deal with acts which would otherwise be a breach of copyright. It is the end-user and not the programmer, who is undertaking those acts, and therefore the GPL has no impact on the programmer in these circumstances For completeness and from the end-user's perspective the end-user is merely running the program an act which is not restricted in any way by the GPL Of course, if the programmer modifies the GPL library and redistributes it (or indeed if the end-user does) then the GPL obligations to license the modified library to all third parties applies as do various other obligations in the GPL: but the question focuses on whether linking an application to a GPL library requires the *application* (as opposed to the library) to be released under GPL code

Does linking an application statically to a GPL library render that application subject to the GPL?

This is a somewhat more complex question, and does require an analysis of the text of the GPL

The most relevant clause is 2(b):

'You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program and copy and distribute such modifications or work under the terms of section 1 above, provided that you also meet all of these conditions . you must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof to be licensed as a whole at no charge to all third parties under the terms of this License'

This is qualified by the following:

'These requirements apply to the modified work as a whole If identifiable sections of that work are not derived from the Program and can reasonably be considered independent and separate woiks in themselves, then this License and its terms, do not apply to those sections when you distribute them as separate works But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it'

It is also important to bear in mind the definition of 'work based on the Program':

'a work based on the Program means either the Program or any derivative work under copyright law, that is to say a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language'

The GPL therefore only attaches to a 'work based on the Program'. Assuming the library is a 'Program', is the application calling it a 'work based on the Program? Can, in the legal sense, the application be considered a derivative work of the library?

Just to be clear: statically linking to the library requires a licence It's an act of copying, and as such, will be[15] a breach of the Copyright Act in the absence of a licence The question is, whether the licence in question GPL2, permits that copying if the resultant application (or at least the parts not incorporating the library) is to be released under a licence other than GPL2.

My view is that that the answer to the question 'Does linking an application statically to a GPL library render that application subject to the GPL? is a somewhat more hesitant 'no': the application is not a derivative work of the library and the application as a whole does not have to be licensed under the GPL. If you distribute the resultant application it's clear that you will be distributing GPL code: namely those parts of the application derived from the library, and you'll have to comply with the GPL to the extent that it attaches 1o those parts of the code. However, I'd argue that what you are distributing is a collective work,[16] and not a derivative work and all the relevant definitions in the GPL refer to a 'work based on the Program' which is defined (in the GPL) as a derivative work as determined by applicable copyright law[17] We are on slightly shakier ground here I m confident that the GPL *could* have been drafted to ensure that static linking to a GPL library brought the whole

application within the ambit of the GPL, and indeed, it seems clear to me from other wording in the licence that the intention of Stallman and Mogien was that it should

However the crux of my argument relies on the definition of 'work based on the Program' and my contention that a program and its libraries are a collection of works of separate copyright, and therefore in no sense is the program as a whole a derivative work' of any of its components, but a collective work[18] If you look at the source code of any reasonably-sized application, in all probability the preamble will contain a list of headers linking to external libraries The external libraries will not be modified by the linker portion of the compiler but will simply be imported almost verbatim [19]

Lawrence Rosen in the seminal work 'Open Source Licensing'[20] makes the argument that, unless the relevant GPL components are modified in some way, merely combining or linking them with other works does not render the whole work subject to the GPL His book is based on US copyright law but for the purposes of this argument, I contend that there is no material difference between English law and the US law However, this article is limited to English law.

Note also that where an ambiguity exists, the *contra proferentem* rule would traditionally require the court to resolve the ambiguity against the licensor and in favour of the licensee

There is also the interesting argument, using s 50C of the Copyright, Designs and Patents Act 1988 (as amended), that the lawful user of a piece of software may copy or adapt it as long as this is not prohibited by contract The GPL clearly does not prohibit copying or adaptation," and therefore the lawful user automatically has the right to copy and adapt Of course, this then invites the counter argument 'but if you copy and adapt (and distribute) and do not follow the other requirements of the GPL (including making the source available) then you are no longer a lawful user'

Does cutting and pasting GPL code into an application render that application subject to the GPL?

The arguments that apply to static linking apply to cutting and pasting: a programmer cutting and pasting code and amending the cross-references, is, essentially doing the job of a linker, but by hand. However, where the code is inline, there is a much greater temptation to modify and intermingle the code and therefore it may become increasingly difficult to discern which portions of the code are separate, at which point it becomes more likely that the whole is a derivative work

in practice

There are versions of GPL2 which are explicitly intended to deal with linking: these are the LGPL and GPL with classpath exception. I won't go into those in detail in this article, which is aimed squarely at GPL2 However, were I a licensor

Endnotes

1 There's also a lot in common between knitting patterns and programming, but maybe that's an article for Knitter's World.

2 When I talk about the GPL in this article, I'm specifically referring to version 2. However, occasionally I specifically refer to GPL2 to make it absolutely clear that those comments are referable only to that version

3 It doesn't explain why GPL3 is, to put it charitably, a less good piece of drafting, but I'll leave that point to Alex Newsom in the accompanying article. It may have something to do with the fact that two authors were responsible for GPL 2, as opposed to something over 1000 for GPL 3

4 There should be a synonym with less negative connotations

5 I am hoping that the carefully chosen analogy is appreciated.

6 And, occasionally, to remove functionality, if it turns out that a particular library breaches the IPRs of a third party, provides access to functions, like decryption of DVD copy protection, or has been cracked by rogue coders

7 I wrote this article without looking at s 123 of the Insolvency Act 1986 and indeed I'm sure commercial lawyers frequently make the reference without bothering to check it. In my case, it's the only statutory reference I've committed to memory and in fact I'd be quite happy if it were wrong, if only to prove my point that while it is possible to code to a DLL without testing, it's not very wise

8 Actually, GNU is Richard Stallman's baby: the FSF is the Zoo, of which he is head keeper, in which it is kept and nurtured

9 You'll have to trust me on this: the release of GPL3 has meant that the FAQ section of the FSF web site has been updated to cover GPL3, but the original GPL2 FAQ has gone walkabout (a link will be posted on the SCL site if and when it re-emerges) GPL3 has some other linking issues, which I will address in a subsequent article, if anyone reading this far asks me to.

10 This does not apply to libraries covered by the LGPL, or the GPL + classpath exception, which are topics for another day.

11 It is also worth pointing out that although the FSF is the copyright owner of a lot of GPL software, anyone can release software under the GPL and retain the copyright themselves, and have indeed done so

12 And neither, by using the library, is the end-user in breach of copyright, so the programmer cannot be liable for procuring any breach

13 A particularly pertinent point is Windows Media Player. Windows media player can play back a number of different formats of video and audio, through third party codecs. These codecs are essentially DLLs Several codecs (especially those for DivX and Ogg Vorbis) are available under the GPL It is rather bizarre to suggest, as FSF does, that by making it possible for Windows Media Player to interface with these DLLs, Microsoft is in breach of the GPL

pursuing a licensee for copyright infringement of my GPL code by using my libraries, I would certainly try to point out to any judge that if I had wanted software linked to my libraries to be free of the GPL, I would have chosen one of the other licences

My advice to clients who wish to release software under a licence of their choice (which may be another free or other source licence, and not just the GPL) is as follows

1 Try to find a library released under the LGPLor GPL with classpath exception (or another more permissive licence, like MIT or BSD, or even a proprietary licence with royalty-free distribution rights which are compatible with your out-licence)

2 If you need to use a GPL library try to persuade the copyright owner of the library to release it under another licence, or think hard about why you don't want to release the whole app under the GPL.

3 If you have no luck with that then link dynamically to the library, and if possible try to arrange for the end-users to download the library separately ^ If you can use a non-GPL (or even a commercially available) library during the testing phase, even better[25]

4 Failing that link statically, avoid amending the library, remember to comply with the GPL for the library when you distribute the object code containing it, and buy a rabbit's foot

5 Failing that, cut and paste the code you want, call it only by using the published interfaces, resist the temptation to modify the pasted code, remember to comply with the GPL for the library when you distribute it and wait for a cafl from gpl-violatians org

Remember to get local advice in any jurisdiction where your code might end up

Its also worth bearing in mind that the closer you sail to the wind the more likely it is that you will attract the ire of the vociferous GPL community If you make some useful and worthwhile amendments to a GPL library which you then release back to the community, then you re less likely to be hounded for GPL violations than if you try to conceal all the juicy bits in the closed part of your code ®

Andrew Katz is a partner at Moorcrofts LLP, where he specialises in technoiogy law with a bias towards open source software, in a former Eife he was a software developer and has released software under the GPL,. He can be reached at Andrew,. katz@moorcrofts .com,

Endnotes *(Continued)*

14 I'd argue that the part of section 1 quoted which follows 'that is to say is technically both irrelevant to the GPL's interpretation, and legally an incorrect summary of the law

15 Subject to fair use exceptions or unless copyright has expired or the code is otherwise in the public domain

16 As a reminder, a collective work is a set of works, each individually attracting copyright (or not, as the case may be) like an anthology of poems A derivative work is a work based on a previous work, such as a translation, or a recasting of a previous work (like *Return to the Forbidden Planet)* Both attract copyright in their own right, but will also require a licence from the previous copyright owner(s) for exploitation.

17 This is my view in English law - but the answer may be different in different jurisdictions,

18 One thing that makes me feel uncomfortable is that although this argument makes sense as far as the source is concerned, the object code generated from the source is undeniably a derivative work of the source - even if it is a collective work, and it would be much more difficult (although probably not impossible) to distinguish the each portion of the object code which correlate to the GPL and non-GPL portions of the source Still, if the source is not a 'work based on the Program', then it's difficult to see how a derivative work can be a 'work based on the Program' Note that, to comply with the GPL (if the object code is distributed), the source code of the GPL portions (and any amendments to those portions) needs to be made available, and this is easy to achieve, especially if the GPL parts of the code remain in separate libraries

19 The linker will make trivial changes to the cross-references and variables in the imported sections, but this does not affect the argument.

20 Prentice Hall 2005 - but watch out: if you have the same edition I have, some crucial clauses in the GPL are omitted from the copy in the book's appendix.

21 And is probably not a contract (another topic for another day)

22 If the library is downloaded separately, it's clearer that it must be the end-user who is doing the linking, and not the programmer. However, there is an explicit clause towards the end of section 2 of the GPL which makes it clear that 'mere aggregation' of files on the same storage medium does not render the non-GPL files subject to the GPL This is really a clarification and goes without saying

23 The logic here being that, a-la-Microsoft media player, if your app has never even touched a GPL library, even during testing and development, then it really is impossible to see how the app can be subject to the GPL if the user then chooses to use a GPL library, For example, if you are developing a video app, you might want to use an Nvidia MPEG2 codec for testing purposes (which is available for $20 or so from the Nvidia web site, but for singie user use only) but then release your app without a codec, allowing the customer to use the GPL codec from Sourceforge, or buy a commercial one from Nvidia (or another vendor)