The Code itself is the Contract... Or is it?

The Architecture of Intent within Linux and its Legal Effect on GPL Compliance

DevLegal

Sabir Ibrahim

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

About Me

- IP Attorney
- OSS Licensing and Compliance
- ex-OSS counsel at Amazon, Roku
- Founder of Dev Legal
- Co-Founder of Chinstrap Community, a resource center for COSS entrepreneurship

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

“Linux is a cancer that attaches itself in an intellectual property sense to everything it touches.”

–Microsoft CEO Steve Ballmer (June 1, 2001)

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

"Today we're excited to announce that we're launching Microsoft's open source Linux distribution, a version of Linux supported by Microsoft, available on Azure, for anybody to use."

-Microsoft VP of Azure Brendan Burns (May 18, 2026)

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

What are we talking about?

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

What are we talking about technically?

What are we talking about legally?

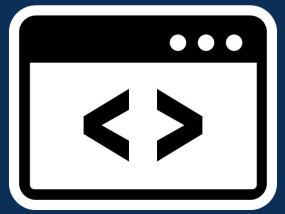
01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

What are we talking about technically?

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

What are we talking about technically?

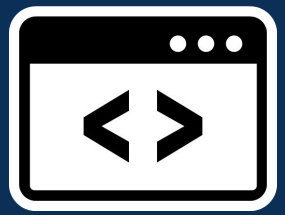
Non-GPL Software



01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

What are we talking about technically?

Non-GPL Software

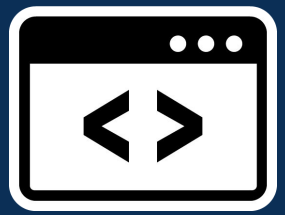


Linux Kernel (GPLv2)

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

What are we talking about technically?

Non-GPL Software



Linux Kernel (GPLv2)

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

What are we talking about legally?

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

What are we talking about legally?

- 1964: The U.S. Copyright Office began registering computer programs
- 1978: The Copyright Act was amended to designate software as a copyrightable “literary work”
- 1980: The Computer Software Copyright Act created a legal definition of “computer program” and established rules for making copies.

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

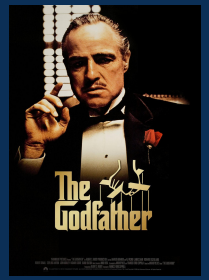
What are we talking about legally?

A derivative work is "a work based upon one or more preexisting works [.]" A work is "based on" a preexisting work when it is "recast, transformed, or adapted" from the preexisting work.

(17 USC § 101)

What are we talking about legally?

The Godfather
by Francis Ford Coppola

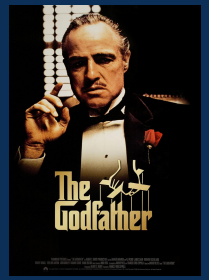


Dev Legal

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

What are we talking about legally?

The Godfather **by Francis Ford Coppola**



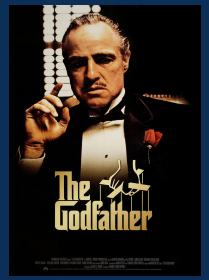
The Godfather **by Mario Puzo**

Dev Legal

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

What are we talking about legally?

The Godfather by Francis Ford Coppola



This...

...is a derivative work of...

...this.

The Godfather by Mario Puzo

Dev Legal

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

What are we talking about legally?

Copyleft is “a general method for making a program (or other work) free (in the sense of freedom, not ‘zero and requiring all modified and extended ons of the program to be fr ”

(Free Software Foundation, “What is Copyleft?”)

What are we talking about legally?

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and 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. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

(General Public License version 2, Section 0)

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

What are we talking about legally?

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

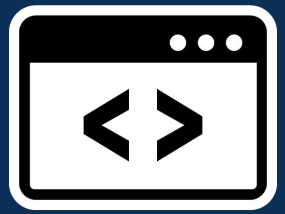
a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

(General Public License version 2, Section 3)

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

What are we talking about legally?

Non-GPL Software

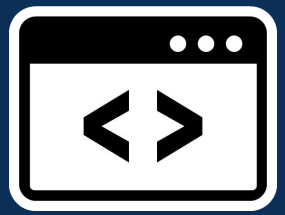


Linux Kernel (GPLv2)

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

What are we talking about legally?

Non-GPL Software



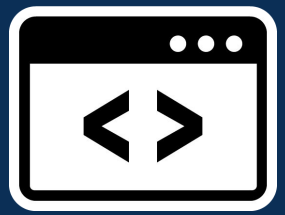
Is this...

Linux Kernel (GPLv2)

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

What are we talking about legally?

Non-GPL Software



Is this...

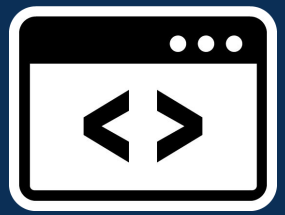
...a derivative work of...

Linux Kernel (GPLv2)

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

What are we talking about legally?

Non-GPL Software



Is this...

...a derivative work of...

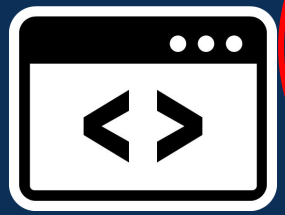
...this?

Linux Kernel (GPLv2)

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

What are we talking about legally?

Non-GPL Software



Linux Kernel (GPLv2)

The answer depends on the details of this.

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

What do the Linux kernel maintainers have to say about it?

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

The Kernel's "Architecture of Intent"

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

The Kernel's "Architecture of Intent"

- **A Clear Exception for Userspace: the Syscall Note**

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

The Kernel's "Architecture of Intent"

- **A Clear Exception for Userspace: the Syscall Note**
- **An Internal Boundary for Kernel Modules:** `MODULE_LICENSE` **and** `EXPORT_SYMBOL_GPL`

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

The Kernel's "Architecture of Intent"

- **A Clear Exception for Userspace: the Syscall Note**
- **An Internal Boundary for Kernel Modules: `MODULE_LICENSE` and `EXPORT_SYMBOL_GPL`**
- **A Tracking System for Code Under a GPL-Incompatible License: the 'Tainted Kernel' Flag (`P`)**

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

A Userspace Exception: The Syscall Note

```
13    NOTE! This copyright does *not* cover user programs that use kernel
14  services by normal system calls - this is merely considered normal use
15  of the kernel, and does *not* fall under the heading of "derived work".
16  Also note that the GPL below is copyrighted by the Free Software
17  Foundation, but the instance of code that it refers to (the Linux
18  kernel) is copyrighted by me and others who actually wrote it.
19
20  Also note that the only valid version of the GPL as far as the kernel
21  is concerned is _this_ particular version of the license (ie v2, not
22  v2.2 or v3.x or whatever), unless explicitly otherwise stated.
23
24                          Linus Torvalds
25
```

(/LICENSES/exceptions/Linux-syscall-note)

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

A Userspace Exception: The Syscall Note

Spotify
(Non-GPL)

Linux Kernel
(GPLv2)

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

A Userspace Exception: The Syscall Note

Spotify
(Non-GPL)

Linux Kernel
(GPLv2)

👍 OK because of the Userspace Exception (set forth in the Syscall Note).

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

Can licensors grant exemptions?

"In legal terms, releasing a work under the GNU General Public License... is a statement by the copyright holders authorizing users to do certain things. The same copyrig legally free to additional permission for use of t "

(Free Software Foundation, "Interpreting, enforcing and changing the GNU GPL, as applied to combining Linux and ZFS")

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

The Internal Boundary: `MODULE_LICENSE` and `EXPORT_SYMBOL_GPL`

“Loadable kernel modules also require a `MODULE_LICENSE()` tag… The sole purpose of this tag is to provide sufficient information whether the module is free software or proprietary for the kernel module loader and for user space tools.”

(Linux Kernel Organization, “Linux kernel licensing rules”)

The Internal Boundary: `MODULE_LICENSE` and `EXPORT_SYMBOL_GPL`

“[T]he symbols exported by `EXPORT_SYMBOL_GPL()` can only be seen by modules with a `MODULE_LICENSE()` that specifies a GPL compatible license. It implies that the function is considered internal issue, and not really an interface.”

(Linux Kernel Organization, “Chapter 9, Symbols”)

A Social Deterrent: The ‘Tainted’ Flag (P)

“Modules tagged [as ‘Proprietary’] are tainting the kernel with the ‘P’ flag.”

(Linux Kernel Organization, “Tainted kernels”)

A Social Deterrent: The ‘Tainted’ Flag (P)

“[T]he kernel will print the tainted state when it notices an internal problem (a ‘kernel bug’), a recoverable error (‘kernel oops’) or a non-recoverable error (‘kernel panic’).”

(Linux Kernel Organization, “Tainted kernels”)

A Social Deterrent: The ‘Tainted’ Flag (P)

“[B]ug reports from tainted kernels will often be ignored by developers.”

(Linux Kernel Organization, “Tainted kernels”)

Why does the intent of the Kernel maintainers matter?

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

A Software License, Even an OSS License, is a Binding Contract

"Defendant contends that Plaintiff's reliance on the GPL fails to plausibly demonstrate… the existence of a contract. Not so. The GPL provides that the Ghostscript user agrees to its terms if the user does not obtain a commercial license… These allegations sufficiently plead the existence of a contract."

(*Artifex Software, Inc. v. Hancom, Inc.* , No. 16-cv-06982-JSC, 2017 WL 1477373 (N.D. Cal. Apr. 25, 2017))

The Parties' Intent is the Most Important Factor in Interpreting a Contract

"A contract must be so interpreted as to give effect to the mutual intention of the parties as it existed at the time of contracting, so far as the same is ascertainable and lawful."

(CAL. CIV. CODE § 1636)

Do the maintainers have an Achilles' Heel? Yes they do.

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

GPL Licensors Can Grant Additional Permissions, but they Can't Impose Additional Restrictions

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

(General Public License version 2, Section 6)

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

What are the Formal Criteria for Determining Which Functions are Placed Within `EXPORT_SYMBOL_GPL`?

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

What are the Formal Criteria for Determining Which Functions are Placed Within `EXPORT_SYMBOL_GPL`?

THERE ARE NONE.

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

The NVIDIA HMM Controversy

The Heterogeneous Memory Management (HMM) subsystem, which allows developers to access some low-level memory-management functionality, was initially made available via `EXPORT_SYMBOL`.

(Linux Kernel Organization, "The proper use of `EXPORT_SYMBOL_GPL()`")

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

The NVIDIA HMM Controversy

NVIDIA used HMM to develop proprietary drivers, prompting calls by some contributors for it to be moved to `EXPORT_SYMBOL_GPL()`.

(Linux Kernel Organization, "The proper use of `EXPORT_SYMBOL_GPL()`")

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

The NVIDIA HMM Controversy

After a heated debate at the 2018 Kernel Maintainers Summit, a consensus was reached to change the HMM symbols to `EXPORT_SYMBOL_GPL`.

(Linux Kernel Organization, "The proper use of `EXPORT_SYMBOL_GPL()`")

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

The NVIDIA HMM Controversy

Some contributors criticized the move, arguing that it was politically-driven and made for no reason other than to target a specific vendor.

(Linux Kernel Organization, "The proper use of `EXPORT_SYMBOL_GPL()`")

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

Does `EXPORT_SYMBOL_GPL` amount to a "further restriction"?

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

(General Public License version 2, Section 6)

Does `EXPORT_SYMBOL_GPL` amount to a "further restriction"?

"In general, the kernel community has long worked to maintain a vague and scary ambiguity around the legal status of proprietary modules while being unwilling to attempt to ban such modules outright."

(Jonathan Corbet, Questioning `EXPORT_SYMBOL_GPL()`, LWN.net (June 23, 2014))

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

If a proprietary vendor were to ever challenge the maintainers on legal grounds, that's what it would argue.

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

What if the vendor won?

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

What if the vendor won?

The court would have to delve into the details and decide for itself, substituting its own analysis of the "structure, sequence, and organization" of the Linux kernel for that of the maintainers.

(*Computer Assocs. Int'l, Inc. v. Altai, Inc.* , 982 F.2d 693, 702 (2d Cir. 1992))

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

Let's imagine a world where `EXPORT_SYMBOL_GPL` has no legal relevance.

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

Disclaimer: THIS IS PURE SPECULATION AND SHOULD NOT BE TAKEN AS LEGAL ADVICE.

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

Scenario 1 - Data Exchange Between Firmware and Kernel Space

A surgical device manufacturer bypasses `EXPORT_SYMBOL_GPL` and directly accesses `struct dma_buf` and related structures to perform zero-copy transfers. It does this in order to meet mission-critical latency targets, without which the device cannot be used safely.

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

Scenario 1 – Data Exchange Between Firmware and Kernel Space

By directly accessing and manipulating the fields within these structs, the proprietary firmware is is intimately weaving its own logic into the very fabric of the kernel's internal workings.

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

Scenario 1 - Data Exchange Between Firmware and Kernel Space

It's directly accessing and relying upon the specific "structure, sequence, and organization" of the Linux Kernel.

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

Scenario 1 – Data Exchange Between Firmware and Kernel Space

This is analogous to copying a chapter from a book rather than merely citing it.

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

Scenario 1 – Data Exchange Between Firmware and Kernel Space

Probable verdict: DERIVATIVE WORK

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

Scenario 2 – Data Exchange Between Kernel Space and Application Layer

An automotive OEM is developing technology to enable Level 5 driving automation. It does this by directly reading sensor data from DMA buffers and writing control commands to hardware registers via mapped memory, bypassing the syscall interface and minimizing latency.

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

Scenario 2 - Data Exchange Between Kernel Space and Application Layer

Is this a more complex case, because it involves a user-space application? Not quite.

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

Scenario 2 – Data Exchange Between Kernel Space and Application Layer

The automotive OEM in this scenario has made a conscious decision to engineer a solution that circumvents this boundary entirely. It isn't using the standard, arm's-length communication channel; instead, the OEM is tunneling through it to directly access the kernel's private memory.

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

Scenario 2 – Data Exchange Between Kernel Space and Application Layer

Unlike the controversy around `EXPORT_SYMBOL_GPL`, there's no argument that the syscall note is arbitrary, since the distinction between kernel space and the application is a clear and unambiguous technical boundary.

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

Scenario 2 – Data Exchange Between Kernel Space and Application Layer

Probable verdict: DERIVATIVE WORK

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

Scenario 3 - A Cross-Platform Storage Driver with a Shim-Based Architecture

An established enterprise storage vendor manufactures a high-performance NVMe-based hardware RAID controller. The controller's competitive advantage comes from a proprietary, closed-source library that implements trade-secret algorithms.

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

Scenario 3 - A Cross-Platform Storage Driver with a Shim-Based Architecture

This core library was developed over many years and is highly portable, with existing drivers for many OSes. The vendor now wishes to offer support for Linux. The goal is to create a Linux kernel driver that keeps the library proprietary.

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

Scenario 3 - A Cross-Platform Storage Driver with a Shim-Based Architecture

The vendor does this by creating a "shim" that is only invoked by the library at initialization, solely to access certain kernel functions that are exported with `EXPORT_SYMBOL_GPL()`. The vendor will release the shim under GPL, but not the library.

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

Scenario 3 – A Cross-Platform Storage Driver with a Shim-Based Architecture

The use of shims is, at best, frowned upon in the Linux community. They are derisively referred to as "GPL condoms."

(Michael Larabel, *Kernel Developers Work To Block NVIDIA "GPL Condom" Effort Around New NetGPU Code*, Phoronix (Aug. 3, 2020), https://www.phoronix.com/news/Linux-Kernel-Blocking-NV-NetGPU)

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

Scenario 3 - A Cross-Platform Storage Driver with a Shim-Based Architecture

Would the use of a shim in this scenario draw the ire of a court because the Linux community regards it as a willful circumvention of the GPL? Not so fast.

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

Scenario 3 - A Cross-Platform Storage Driver with a Shim-Based Architecture

The combination of its pre-existing, cross-platform nature, its functional independence, and the use of a thin, GPL-licensed shim for superficial, initialization-only integration creates a strong case for legal separability.

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

Scenario 3 - A Cross-Platform Storage Driver with a Shim-Based Architecture

This scenario illustrates a critical boundary case: where GPL-only symbols are used not to access the kernel's unique creative logic, but as a restricted entry point to a standardized, public subsystem.

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

Scenario 3 - A Cross-Platform Storage Driver with a Shim-Based Architecture

Probable verdict: NOT A DERIVATIVE WORK

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

Thank you!

sabir@devlegal.io
https://devlegal.io

DevLegal

Sabir Ibrahim

01010111 01101001 01110100 01101000 00100000 01100101 01101110 01101111 01110101 01100111 01101000 00100000 01100101 01111001 01100101 01110011 00101100 00100000 01100001
01101100 01101100 00100000 01100010 01110101 01100111 01110011 00100000 01100001 01110010 01100101 00100000 01110011 01101000 01100001 01101100 01101100 01101111 01110111

Chinstrap Community

Chinstrap Community is a resource center for developers, founders, investors, and other tech professionals who are interested in commercial open source software (COSS).

- **chinstrap.community**

INTRODUCING

Cossmology

The definitive directory of commercial open source software (COSS) companies — 1,000+ organizations, their key people, OSS projects, funding, and the latest news, all in one place.

- **cossmology.com**