dem Sinn und Zweck der deklaratorischen »Garantieklausel« ist es aber nicht verboten, die Garantie auch ohne eine Gegenleistung (also kostenlos) anzubieten, zumal ohnehin jeder Nicht-Lizenznehmer eine kostenlose Garantie für das Programm anbieten darf.

Ziffer 2 GPL

Till Jaeger

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:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) 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.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works 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.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License. This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

Deutsche Übersetzung von Katja Lachmann und Peter Gerwinski

Sie dürfen Ihre Kopie(n) des Programms oder eines Teils davon verändern, wodurch ein auf dem Programm basierendes Datenwerk entsteht; Sie dürfen derartige Bearbeitungen unter den Bestimmungen von Paragraph 1 vervielfältigen und verbreiten, vorausgesetzt, daß zusätzlich alle im folgenden genannten Bedingungen erfüllt werden:

a) Sie müssen die veränderten Dateien mit einem auffälligen Vermerk versehen, der auf die von Ihnen vorgenommene Modifizierung und das Datum jeder Änderung hinweist.

b) Sie müssen dafür sorgen, daß jede von Ihnen verbreitete oder veröffentlichte Arbeit, die ganz oder teilweise von dem Programm oder Teilen davon abgeleitet ist, Dritten gegenüber als Ganzes unter den Bedingungen dieser Lizenz ohne Lizenzgebühren zur Verfügung gestellt wird.

c) Wenn das veränderte Programm normalerweise bei der Ausführung interaktiv Kommandos einliest, müssen Sie dafür sorgen, daß es, wenn es auf dem üblichsten Wege für solche interaktive Nutzung gestartet wird, eine Meldung ausgibt oder ausdruckt, die einen geeigneten Copyright-Vermerk enthält sowie einen Hinweis, daß es keine Gewährleistung gibt (oder anderenfalls, daß Sie Garantie leisten), und daß die Benutzer das Programm unter diesen Bedingungen weiter verbreiten dürfen. Auch muß der Benutzer darauf hingewiesen werden, wie er eine Kopie dieser Lizenz ansehen kann. (Ausnahme: Wenn das Programm selbst interaktiv arbeitet, aber normalerweise keine derartige Meldung ausgibt, muß Ihr auf dem Programm basierendes Datenwerk auch keine solche Meldung ausgeben).

Diese Anforderungen gelten für das bearbeitete Datenwerk als Ganzes. Wenn identifizierbare Teile des Datenwerkes nicht von dem Programm abgeleitet sind und vernünftigerweise als unabhängige und eigenständige Datenwerke für sich selbst zu betrachten sind, dann gelten diese Lizenz und ihre Bedingungen nicht für die betroffenen Teile, wenn Sie diese als eigenständige Datenwerke weitergeben. Wenn Sie jedoch dieselben Abschnitte als Teil eines Ganzen weitergeben, das ein auf dem Programm basierendes Datenwerk darstellt, dann muß die Weitergabe des Ganzen nach den Bedingungen dieser Lizenz erfolgen, deren Bedingungen für weitere Lizenznehmer somit auf das gesamte Ganze ausgedehnt werden – und somit auf jeden einzelnen Teil, unabhängig vom jeweiligen Autor.

Somit ist es nicht die Absicht dieses Abschnittes, Rechte für Datenwerke in Anspruch zu nehmen oder Ihnen die Rechte für Datenwerke streitig zu machen, die komplett von Ihnen geschrieben wurden; vielmehr ist es die Absicht, die Rechte zur Kontrolle der Verbreitung von Datenwerken, die auf dem Programm basieren oder unter seiner auszugsweisen Verwendung zusammengestellt worden sind, auszuüben.

Ferner bringt auch das einfache Zusammenlegen eines anderen Datenwerkes, das nicht auf dem Programm basiert, mit dem Programm oder einem auf dem Programm basierenden Datenwerk auf ein- und demselben Speicher- oder Vertriebsmedium dieses andere Datenwerk nicht in den Anwendungsbereich dieser Lizenz.

Literatur: Dankwardt, Kevin, Are non-GPL loadable Linux drivers really not a problem?, *http://www.linuxdevices.com/articles/AT5041108431.html*; Jaeger, Till/Metzger, Axel, Open Source Software, Rechtliche Rahmenbedigungen der Freien Software, München 2002, S. 37 ff.; Koch, Frank, Urheber- und kartellrechtliche Aspekte der Nutzung von Open Source Software (II), CR 2000, S. 333 ff.; Lejeune, Mathias, Rechtsprobleme bei der Lizenzierung von Open Source Software nach der GNU GPL, ITRB 2003, S. 10 ff.; Marly, Jochen, Softwareüberlassungsverträge, 3. Auflage, Rz. 432 ff.; Metzger, Axel/Jaeger, Till, Open Source Software und deutsches Urheberrecht, GRUR Int. 1999, S. 839 ff.; Spindler, Gerald, Rechtsfragen bei Open Source, Köln 2004; Ravicher, Dan, Software Derivative Work: A Circuit Dependent Determination, *http://www.pbwt.com/Attorney/files/ravicher_1.pdf*; Rosen, Lawrence, The Unreasonable Fear of Infection, *http://www.rosenlaw.com/html/GPL.PDF*; Torvalds, Linus, The Linux Edge, in: DiBona, Chris/Ockman, Sam/Stone, Mark, Open Sources – Voices from the Open Source Revolution, Cambridge u.a. 1999; Välimäki, Mikko, GNU General Public License and the Distribution of Derivative Works, *http://www.soberit.hut.fi/~msvalima/ gpl_derivative.pdf*; Wheeler, David, Make Your Open Source Software GPL-Compatible. Or Else., *http:// www.dwheeler.com/essays/gpl-compatible.html*; Wuermeling, Ulrich/Deike, Thies, Open Source Software: Eine juristische Risikoanalyse, CR 2003, S. 87 ff.

Übersicht

1 Ziffer 2 GPL räumt dem Lizenznehmer ein Bearbeitungsrecht an der Software ein und regelt, unter welchen Bedingungen veränderte Versionen der Software weiterverbreitet werden dürfen. Neben den allgemeinen Pflichten, die auch bei der Weitergabe von unveränderten Programmen einzuhalten sind (siehe oben Ziffer 1 GPL Rz. 29), müssen bei der Weitergabe der bearbeiteten Software auch einige besondere Pflichten beachtet

werden, insbesondere der »Copyleft-Effekt« (siehe unten Rz. 10). Ziffer 2 GPL betrifft die Nutzung des Programms im Source-Code, sofern die Weitergabe allein im Objekt-Code oder als Executable erfolgt, müssen zusätzlich die Pflichten der Ziffer 3 GPL erfüllt werden.

Man kann Ziffer 2 GPL mit gutem Recht als die Kernregelung, als das »Herz« der Lizenz bezeichnen. Wie jede andere freie Software-Lizenz erlaubt die GPL, modifizierte Versionen der Software zu erstellen. Was ihren besonderen Charakter ausmacht, ist das »Copyleft« – von Kritikern »viraler Effekt« genannt, von Anhängern »Impf-Effekt«. Damit ist die Pflicht gemeint, Weiterentwicklungen und Änderungen einer GPL-Software ebenfalls nur unter der GPL weitergeben zu dürfen. Mit diesem völlig neuartigen Modell wollte Richard Stallman sicherstellen, dass Freie Software auch frei bleibt. Anders als bei Programmen, die unter einer BSD-Lizenz stehen, sollte es nicht möglich sein, Weiterentwicklungen »proprietär« zu vertreiben und damit faktisch insgesamt zu einer proprietären Software zu machen. Von besonderer Bedeutung ist daher, dass das Landgericht München I in seinem Urteil zur Durchsetzbarkeit der GPL ausdrücklich darauf hingewiesen hat, dass es »keine Bedenken gegen die Zulässigkeit« der Bedingungen in den Ziffern 2 und 3 GPL hat (LG München I, Urteil v. 19. Mai 2004, S. 18, *http://www.jbb.de/urteil_lg_muenchen_gpl.pdf*, vergleiche auch Ziffer 4 GPL Rz. 14). 2

Schutz der Privatsphäre

Oftmals wird übersehen, dass die Pflichten aus Ziffer 2 GPL nicht bereits bei der Bearbeitung des Programms entstehen, sondern erst dann, wenn das modifizierte Programm weiterverbreitet wird. Wer also für seinen privaten Gebrauch GPL-Programme verändert, muss diese Entwicklungen Dritten nicht zugänglich machen. Das ist schon deshalb sinnvoll, weil rein interne Programmierungen nicht kontrollierbar sind und es kaum einsehbar ist, warum jeder Test und unausgegorene Hack gleich publik gemacht werden sollte. Letztlich würde die GPL ansonsten auch nicht den Vorgaben der Free Software-Definition beziehungsweise der Open Source-Definition entsprechen, die ausdrücklich vorsehen, dass rein private Änderungen nicht veröffentlich werden müssen (vergleiche *www.fsf.org/philosophy/free-sw.html; www.opensource.org/docs/definition.html*). Das deutsche Urheberrechtsgesetz regelt hingegen für Software – abweichend von anderen Werkgattungen –, dass jegliche Bearbeitungen, auch die im privaten Bereich, zustimmungsbedürftig sind. Während für Software die spezielle Regelung des § 69c UrhG gilt (»Der Rechtsinhaber hat das ausschließliche Recht, folgende Handlungen vorzunehmen oder zu gestatten: 1. ...; 2. die Übersetzung, die Bearbeitung, das Arrangement und andere Umarbeitungen eines Computerprogramms sowie die Vervielfältigung der erzielten Ergebnisse. Die Rechte derjenigen, die das Programm bearbeiten, bleiben unberührt«), gilt für andere Werke wie Musik und Literatur § 23 UrhG (»Bearbeitungen oder andere Umgestaltungen des Werkes dürfen nur mit Einwilligung des Urhebers des bearbeiteten oder umgestalteten Werkes veröffentlicht oder verwertet werden.«). Damit gestattet die GPL die Veränderung eines Programms im privaten Bereich ohne Verpflichtungen. Erst wenn es außerhalb der privaten Sphäre genutzt werden soll, greift der »Copyleft«-Effekt ein. 3

4 Nicht ganz einfach ist die Frage zu beantworten, wann eine Handlung noch in die Privatsphäre fällt und wann sie schon als »Verbreitung« anzusehen ist. Die GPL enthält dazu keine weiteren Anhaltspunkte, so dass auf allgemeine Auslegungsgesichtspunkte zurückgegriffen werden muss. Nach den Regelungen zum Internationalen Privatrecht (siehe Anhang »Welches Recht ist anwendbar?«) kann sich hier je nach dem Wohnsitz der Vertragspartner ein unterschiedliches Verständnis davon ergeben, was als »Verbreitung« anzusehen ist. Im Anwendungsbereich des deutschen Urheberrechts kann auf die Definition des § 15 Absatz 3 UrhG zurückgegriffen werden (»Zur Öffentlichkeit gehört jeder, der nicht mit demjenigen, der das Werk verwertet, oder mit den anderen Personen, denen das Werk in unkörperlicher Form wahrnehmbar oder zugänglich gemacht wird, durch persönliche Beziehungen verbunden ist.«). Diese Vorschrift enthält die Abgrenzung von dem, was im Urheberrecht als »öffentlich« angesehen wird und damit nicht mehr »privat« ist. Danach ist also die Weitergabe an Familienmitglieder und Freunde noch keine »Verbreitung«, die die Pflichten von Ziffer 2 GPL auslösen würde. Hingegen ist ein Programm dann »öffentlich« gemacht worden, wenn es in dem Intranet eines Unternehmens für Personen zugänglich gemacht wird, die nicht eng an dem Programm zusammenarbeiten oder in sonstiger Form »durch persönliche Beziehungen verbunden sind« (siehe oben Ziffer 1 GPL Rz. 23). Für eine solche Verbundenheit ist es nicht erforderlich, dass ein unmittelbares Kennenlernen stattgefunden hat. Auch virtuell, etwa über das Internet, lassen sich persönliche Beziehungen entwickeln. Eine kleine, wenn auch weltweit verstreute Entwicklergruppe muss daher ein verändertes GPL-Programm nicht im Source-Code zugänglich machen, wenn der Austausch nur intern stattgefunden hat. Im Gegenteil: Erst wenn alle Miturheber sich einig sind, dass die Weiterentwicklung veröffentlicht werden soll, darf die Software auch außerhalb der Privatsphäre genutzt werden. Ansonsten kann ein Urheberrechtsverstoß wegen Verletzung des § 8 Absatz 2 Satz 1 UrhG vorliegen, wobei eine Veröffentlichung nicht »wider Treu und Glauben« verweigert werden darf (»Das Recht zur Veröffentlichung und zur Verwertung des Werkes steht den Miturhebern zur gesamten Hand zu; Änderungen des Werkes sind nur mit Einwilligung der Miturheber zulässig. Ein Miturheber darf jedoch seine Einwilligung zur Veröffentlichung, Verwertung oder Änderung nicht wider Treu und Glauben verweigern.«). So mag eine Software nicht ohne Einwilligung eines Beteiligten im Internet zum Download bereitgestellt werden, wenn noch Sicherheitsmängel befürchtet werden. Jedoch darf ein Programmierer nicht »blockieren«, nur weil er eine andere Softwarelösung besser findet.

Recht zur Bearbeitung

5 Ziffer 2 GPL erlaubt jedem, seine eigenen Programmkopien zu verändern. Ein Eingriff in eine fremde Programmkopie ist hingegen nicht gestattet – sofern der jeweilige Eigentümer nicht zustimmt (vergleiche zum Begriff der »Bearbeitung« Ziffer 0 GPL Rz. 17). Entsprechend dem Ziel Freier Software, dass auch veränderte Programmversionen für jedermann nutzbar sein sollen, enthält Ziffer 2 GPL neben dem Bearbeitungsrecht auch das Recht, die modifizierte Software zu vervielfältigen und zu verbreiten. Wie in Ziffer 1 GPL umfasst der Begriff »distribute« sowohl die Weitergabe von körperlichen Werkstü-

cken (zum Beispiel CD-ROMs, Festplatten, Embedded-Systeme) als auch die unkörperliche Weitergabe, etwa die Bereitstellung im Internet zum Download (siehe oben Ziffer 1 GPL Rz. 22).

Nach deutschem Urheberrecht ist eine solche Erlaubnis zur Vervielfältigung und Verbreitung von modifizierten Programmen notwendig, um die Modifikationen auch tatsächlich nutzen zu dürfen. Die oben genannten §§ 23, 69c UrhG (siehe oben Rz. 3) machen die »Verwertung«, also auch die Vervielfältigung und Verbreitung, von der Einwilligung des oder der Urheber der Ursprungsversion abhängig. Dies gilt jedenfalls dann, wenn die Veränderung des Programms nicht so unerheblich ist, dass kein »neues« Werk entsteht. Solche »kleinen« Veränderungen können bei manchen Bugfixes oder formalen Änderungen vorliegen, die keine individuelle Entwicklungsleistung erfordern. Auch wenn hier das Urheberrecht an sich keine Erlaubnis für die Veränderung erforderlich macht – in § 69d UrhG ist dies für Fehlerbehebungen sogar ausdrücklich geregelt (»Soweit keine besonderen vertraglichen Bestimmungen vorliegen, bedürfen die in § 69 c Nr. 1 und 2 genannten Handlungen nicht der Zustimmung des Rechtsinhabers, wenn sie für eine bestimmungsgemäße Benutzung des Computerprogramms einschließlich der Fehlerberichtigung durch jeden zur Verwendung eines Vervielfältigungsstücks des Programms Berechtigten notwendig sind.«) –, fallen solche »Fehlerbehebungen« insoweit in den Anwendungsbereich der GPL, als ihre gemeinsame Nutzung mit dem Programm nur unter Beachtung der in der GPL geregelten Pflichten erlaubt ist. Unabhängig von der urheberrechtlichen Lage verlangt die GPL, dass ihre Regelungen auf vertraglicher Ebene für das gesamte einheitliche Programm eingehalten werden. So muss zum Beispiel der gesamte Source-Code zugänglich gemacht werden – einschließlich solcher Fehlerbehebungen und »kleinen« Veränderungen. 6

Pflichten bei der Weitergabe – Änderungsvermerk

In den Ziffern 2a) bis 2c) GPL sind die Bedingungen aufgelistet, die bei der Weiterverbreitung von bearbeiteter Software eingehalten werden müssen. Nach Ziffer 2a) GPL muss jede veränderte Programmdatei einen Hinweis darauf enthalten, wer die Änderungen vorgenommen hat und wann dies geschehen ist. In der Praxis geschieht das durch Header wie den folgenden: 7

```
/* -*- linux-c -*-
 * APM BIOS driver for Linux
 * Copyright 1994, 1995, 1996 Stephen Rothwell
 *                            (Stephen.Rothwell@canb.auug.org.au)
 *
 * This program is free software; you can redistribute it and/or modify it
 * under the terms of the GNU General Public License as published by the
 * Free Software Foundation; either version 2, or (at your option) any
 * later version.
 *
 * This program is distributed in the hope that it will be useful, but
 * WITHOUT ANY WARRANTY; without even the implied warranty of
```

```
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * General Public License for more details.
 *
 * $Id: apm_bios.c,v 0.22 1995/03/09 14:12:02 sfr Exp $
 *
 * October 1995, Rik Faith (faith@cs.unc.edu):
 *    Minor enhancements and updates (to the patch set) for 1.3.x
 *    Documentation
 * January 1996, Rik Faith (faith@cs.unc.edu):
 *    Make /proc/apm easy to format (bump driver version)
 * March 1996, Rik Faith (faith@cs.unc.edu):
 *    Prohibit APM BIOS calls unless apm_enabled.
 *    (Thanks to Ulrich Windl <Ulrich.Windl@rz.uni-regensburg.de>)
 * April 1996, Stephen Rothwell (Stephen.Rothwell@canb.auug.org.au)
 *    Version 1.0 and 1.1
 * May 1996, Version 1.2
 *
 * History:
 *    0.6b: first version in official kernel, Linux 1.3.46
 *    0.7: changed /proc/apm format, Linux 1.3.58
 *    0.8: fixed gcc 2.7.[12] compilation problems, Linux 1.3.59
 *    0.9: only call bios if bios is present, Linux 1.3.72
 *    1.0: use fixed device number, consolidate /proc/apm into this file,
 *         Linux 1.3.85
 *    1.1: support user-space standby and suspend, power off after system
 *         halted, Linux 1.3.98
 *    1.2: When resetting RTC after resume, take care so that the the time
 *         is only incorrect by 30-60mS (vs. 1S previously) (Gabor J. Toth
 *         <jtoth@princeton.edu>); improve interaction between
 *         screen-blanking and gpm (Stephen Rothwell); Linux 1.99.4
 *
```

8 Es ist dabei nicht erforderlich, seinen wirklichen Namen zu nennen. Auch ein Pseudonym, etwa ein Nickname, oder eine anonyme Änderung sind zulässig:

```
/* linux/mm/vmalloc.c
* Copyright (C) 1993 Linus Torvalds
* Support of BIGMEM added by anonymous, July 2001
*/
```

Selten enthalten die Header genaue Datumsangaben, sondern zumeist nur einen Urhebervermerk mit Jahreszahl, was aber wohl ausreichen dürfte. Bei der Verwendung eines Versionkontrollsystems, zum Beispiel dem gebräuchlichen CVS, kann hingegen der exakte Zeitpunkt einer Änderung (jedenfalls innerhalb des Versionkontrollsystems) ermittelt werden. Die Angaben in einem Versionkontrollsystem reichen jedoch alleine nicht aus, um den Anforderungen der Ziffer 2a) GPL zu genügen. Denn bei der Weitergabe des Source-Codes sind für den Empfänger die Angaben im Versionkontrollsystem nicht einsehbar, jedenfalls nicht ohne entsprechenden Hinweis auf den Ort des Versionkontrollsystems; der Ort hilft zudem nicht weiter, wenn die Plattform mit dem Versionkontrollsystem nicht weiterbetrieben wird. Dennoch kann ein Versionskontrollsystem bei einem etwaigen Streitfall, wer Urheber einer Software ist und wann ein Programm

entwickelt oder verändert wurde, von großer praktischer Bedeutung sein. Es ist daher zu empfehlen, ein Versionkontrollsystem sorgfältig zu betreiben und jede Änderung der Software zu dokumentieren.

Es genügt, wenn der Änderungsvermerk im Source-Code enthalten ist, da jeder, der die 9
Software als Executable oder sonst im Objekt-Code weitergibt, zumindest verpflichtet ist, den Source-Code anzubieten, wenn er ihn nicht schon direkt mitliefert (dazu ausführlich unten Ziffer 3 GPL Rz. 11). So ist sichergestellt, dass der Empfänger die Möglichkeit hat, von dem Änderungsvermerk Kenntnis zu nehmen. Dennoch macht es Sinn, auch im Objekt-Code einen Urhebervermerk anzubringen, da er die Durchsetzung der GPL wesentlich erleichtern kann. Besteht der Verdacht, dass jemand ein GPL-Programm als Executable verbreitet, ohne die Lizenzbedingungen der GPL einzuhalten, ist zunächst ein – möglicherweise aufwändiges – Re-Engineering oder eine erlaubnispflichtige Dekompilierung erforderlich, wenn nachgewiesen werden soll, dass man Urheber der betroffenen Software ist. Urhebervermerke in Textform können hingegen einfach durch einen Editor ermittelt werden. Selbstverständlich ist die Löschung eines solchen Vermerks möglich, allerdings wird damit auch offenbar, dass eine vorsätzliche GPL-Verletzung vorliegt.

Pflichten bei der Weitergabe – das »Copyleft«

Wer die Software oder einen Teil davon verändert und das so veränderte Programm wei- 10
tergibt oder veröffentlicht, muss die Software gemäß Ziffer 2b) GPL insgesamt unter der GPL lizenzieren. Dabei müssen jedem (»to all third parties«) lizenzgebührenfrei (»at no charge«) dieselben Nutzungsrechte eingeräumt werden, die auch der Bearbeiter von dem oder den ursprünglichen Urhebern unter der GPL erhalten hat. Das Prinzip ist so einfach wie klar: Zusätzliche oder andere Lizenzbedingungen sind nicht zulässig, die Weitergabe darf nur unter der GPL erfolgen. Lizenzänderungen sind damit ausgeschlossen. Dies gilt für alle Fälle, in denen das Ursprungsprogramm oder Teile davon so verwendet werden, dass ein neues, abgeleitetes (»derivative«) Werk entsteht. Wann dies der Fall ist, gehört wiederum zu den schwierigsten und umstrittensten Fragen der GPL-Auslegung, die unten ausführlich behandelt wird (siehe unten Rz. 14 ff.).

Pflichten bei der Weitergabe – Anzeige bei interaktiven Kommandos

Etwas umständlich formuliert, fordert Ziffer 2c) GPL, dass besondere Hinweispflichten 11
bestehen, wenn ein Programm bei der Ausführung »interaktiv Kommandos einliest«. Damit sind alle Programme gemeint, die nicht nur im Hintergrund ablaufen, wie zum Beispiel Systemprogramme, sondern mit Hilfe von Eingaben gesteuert werden. Dies betrifft sowohl Programme, die mittels Kommandozeile bedient werden (wie dies früher nahezu ausschließlich bei Freier Software üblich war), als auch die im Massenmarkt inzwischen weiter verbreiteten Anwendungsprogramme, die graphisch bedient werden – also über Fenster und mit anzuklickenden Schaltflächen.

Die nach Ziffer 2c) GPL erforderlichen Angaben sind: Urhebervermerke, der Hinweis auf 12
den Gewährleistungsausschluss, sofern nicht Gewährleistung explizit eingeräumt wird

(siehe oben Ziffer 1 GPL Rz. 58), der Hinweis darauf, dass das Programm unter den Bedingungen der GPL weiterverbreitet werden darf, sowie den Ort, an dem sich der Text der GPL findet, also etwa der Dateiname (zum Beispiel »COPYING«). Während bei Programmen, die ausschließlich über die Kommandozeile bedient werden, die gesonderte Einblendung auf dem Bildschirm üblich und auch GPL-konform ist, sind solche Hinweise bei graphisch bedienten Programmen im Menü unter Rubriken wie »About« oder »Info« unterzubringen. Der Nutzer erwartet dort Informationen über Lizenzbestimmungen und den Anbieter.

13 Diese Hinweispflichten bestehen allerdings nicht, wenn das bearbeitete Programm schon interaktiv war und solche Hinweise selbst nicht enthielt. Daher müssen solche Angaben nur gemacht werden, wenn sie schon vorhanden waren oder eine interaktive Funktion erst eingeführt wird.

»Derivative Work« – viraler Effekt?

14 Der Copyleft-Effekt, also die Pflicht, Modifikationen von GPL-Programmen bei der Weitergabe ebenfalls nur unter der GPL zu lizenzieren, scheint auf den ersten Blick ein einfaches Modell zu sein. In der Praxis ergeben sich allerdings zahlreiche Auslegungs- und Wertungsprobleme, die hier genauer betrachtet werden sollen. Vorab soll aber der Mythos vom »viralen Effekt« widerlegt werden: Software gerät niemals »automatisch« unter die GPL, sondern stets nur auf Grund einer entsprechenden Handlung des Rechtsinhabers. Daher ist die Kombination von GPL-Software und eigenen Programmmodulen an sich noch nicht ausreichend, damit das eigene Programm unter der GPL lizenziert wird. Nur wenn der Rechtsinhaber ausdrücklich oder durch schlüssiges Handeln seine Software der GPL unterstellt, greift die Lizenz auch ein. Das ist etwa der Fall, wenn bei der Weitergabe eine »Copying«-Datei mit dem Lizenztext beigefügt wird. Wird aber ein GPL-Programm zusammen mit einer eigenen Software, die anderen Lizenzbestimmungen unterliegt, vertrieben, so kann die GPL dieses eigene Programm nicht »anstecken« oder automatisch einbeziehen. Ist das mit der GPL-Software gemeinsam vertriebene Programm aber ein »derivative work«, kann eine Urheberrechtsverletzung vorliegen, wenn es nicht ebenfalls unter der GPL zugänglich gemacht wird. Dem Verletzer bleiben dann aber zwei Möglichkeiten: Zum einen kann er sein Programm ebenfalls der GPL unterstellen und die GPL-Verletzung damit »heilen«, zum anderen kann er auf den (weiteren) gemeinsamen Vertrieb verzichten und seine Software gar nicht oder ohne GPL-Bestandteile vertreiben. Die Lizenzierung geschieht nicht automatisch, sondern es bestehen lizenzrechtlich abgesicherte Pflichten, die bei der Weitergabe beachtet werden müssen.

15 Entscheidend ist in jedem Falle die Frage, wann ein »derivative work«, also ein »abgeleitetes Werk«, vorliegt. Die Antwort auf diese Frage entscheidet, ob der Rechtsinhaber sein Programm der GPL unterstellen muss, oder ob er frei in der Lizenzwahl ist, also eine andere Open Source-Lizenz oder eine proprietäre Lizenz verwendet werden darf. Die folgenden Ausführungen zeigen, dass es keine scharfe Trennlinie gibt, sondern eine Reihe von mehr oder weniger klaren Fallgruppen, bei denen der Copyleft-Effekt eingreift oder

nicht, sowie einen Graubereich, in dem Wertungen getroffen werden müssen, die von Einzelfall zu Einzelfall unterschiedlich sein können. Hierzu werden Kriterien vorgeschlagen, die die Einschätzung erleichtern sollen, wann ein abgeleitetes Werk vorliegt und wann nicht.

Kein »derivative work« – eindeutige Fallgruppen

Zunächst stellt die GPL in Ziffer 2, 4. Hauptabsatz, klar, dass GPL-Programme und Software, die unter anderen Bedingungen lizenziert werden, grundsätzlich problemlos auf einem Datenträger verbreitet werden können: »In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.« Die Tatsache an sich, dass sich die Programme auf einem Datenträger – zum Beispiel einer Festplatte oder einer DVD – befinden, führt also noch nicht dazu, dass sie als voneinander abgeleitet angesehen werden müssen. 16

Eindeutig ist die Lage auch für Anwendungsprogramme, die zusammen mit Linux ausgeliefert werden und allein mittels der gewöhnlichen Systemaufrufe auf den Kernel zugreifen. Zwischen die Applikation und den Linux-Kernel ist regelmäßig die Programmbibliothek »glibc« geschaltet, die unter der Lesser General Public License beziehungsweise Library General Public License (LGPL) lizenziert wird, siehe Abbildung 2-2. Daher muss die Applikation nur den Anforderungen der LGPL genügen, die – anders als die GPL – grundsätzlich auch für solche (Anwendungs-)Programme, die mit dem von der Bibliothek (glibc) ein einheitliches Werk bilden, die Verwendung einer anderen Lizenz als der LGPL zulässt. 17

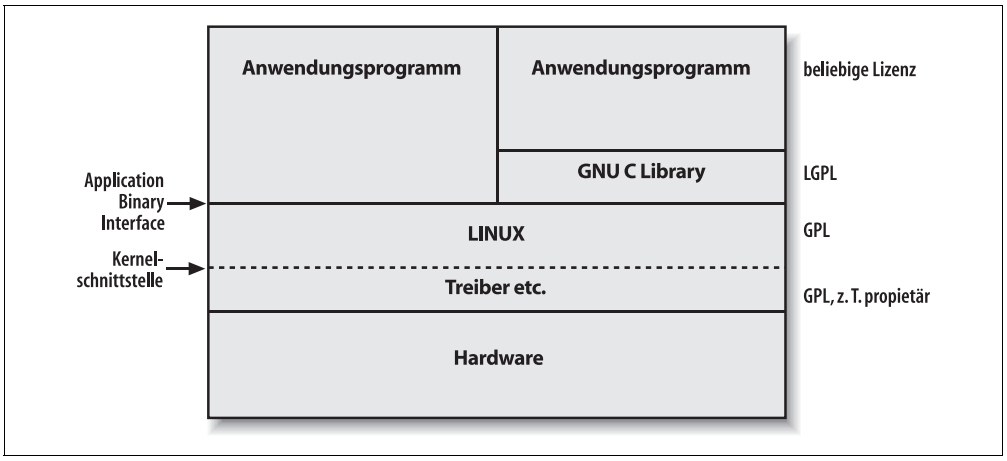

Abbildung 2-2: Aufbau des Linux-Kernels und seine Schnittstellen

Aber auch unabhängig davon, ob die »glibc« verwendet wird, führen bloße Systemaufrufe nicht zu einem abgeleiteten Werk. Linus Torvalds, der Urheber der ersten Linux-Version, hat die GPL für Linux mit einer weiteren Klausel versehen, die folgenden Wortlaut hat:

> NOTE! This copyright does *not* cover user programs that use kernel services by normal system calls – this is merely considered normal use of the kernel and does *not* fall under the heading of »derived work«. Also note that the GPL below is copyrighted by the Free Software Foundation, but the instance of code that it refers to (the Linux kernel) is copyrighted by me and others who actually wrote it.
>
> Also note that the only valid version of the GPL as far as the kernel is concerned is _this_ license (ie v2), unless explicitly otherwise stated.
>
> *Linus Torvalds*

Dieser Zusatz erlaubt es ausdrücklich, den Kernel zu benutzen – etwa direkt über das vom Kernel zur Verfügung gestellte ABI (Application Binary Interface) –, ohne deswegen die Applikation unter die GPL stellen zu müssen. Dies würde auch ohne diese explizite Erklärung gelten, da Anwendungsprogramme, die nur Systemaufrufe starten, so eigenständig sind, dass sie keine »derivative works« darstellen. Dies wurde von Linus Torvalds zur Vermeidung von Auslegungszweifeln nochmals betont. Der Hinweis zeigt aber auch, dass er sich durchaus bewusst darüber war, dass die Beurteilung dessen, was unter den Begriff »derivative work« fällt, problematisch ist.

18 Unabhängig von der Frage, wann ein »derivative work« im Einzelfall vorliegt, ist der Vertrieb von eigener Software alleine immer dann unter einer beliebigen Lizenz zulässig, wenn sie keinen GPL-Code enthält. Selbst wenn die »Verbindung« der eigenen Software mit dem GPL-Programm ein »abgeleitetes Werk« ergeben würde, wäre der alleinige Vertrieb gestattet. Das liegt daran, dass in diesem Fall nicht das bearbeitete Programm vertrieben wird, sondern nur eigener Code. Dass damit der Copyleft-Effekt durch entsprechende Vertriebskonstruktionen umgangen werden kann, lässt sich wohl nicht vermeiden. Wo die Grenzen solcher Umgehungsmöglichkeiten liegen, muss letztlich die Rechtsprechung zeigen.

»Derived« – eindeutige Fallgruppen

19 Umgekehrt muss eigener Code eindeutig der GPL unterstellt werden, wenn das vorbestehende GPL-Programm in seiner bestehenden Form geändert wird. Erweiterungen, Kürzungen und Abänderungen des Codes führen stets dazu, dass das so geänderte Programm der GPL unterstellt werden muss – immer vorausgesetzt, es soll überhaupt vertrieben und nicht nur privat eingesetzt werden. Dies trifft in der Regel für Bugfixes und Patches zu. Dabei kommt es nicht darauf an, ob ein Patch an sich schon urheberrechtlich schutzfähig ist, da das gepatchte Programm ein »abgeleitetes Werk« darstellt und der Umfang der Änderung dafür irrelevant ist (siehe oben Rz. 6). Eine Bedeutung kommt dem nur für die Frage zu, ob der Autor eines Patchs dann selbst auch Bearbeiterurheber und Lizenzgeber wird. Dies kann zum Beispiel Bedeutung für die Frage haben, ob dieser Autor selbst gegen eine GPL-Verletzung gerichtlich vorgehen kann.

Auch die klassische Weiterentwicklung durch Codeergänzungen führt stets zu einem abgeleiteten Werk. Dies gilt zunächst unabhängig davon, ob solche Änderungen in einer neuen Datei oder in bestehenden Dateien implementiert werden. Damit dürfte die Mehrzahl von Bearbeitungen eines GPL-Programms durch den Copyleft-Effekt betroffen sein. 20

Problematisch sind dagegen die Fälle, in denen nicht nur eine einfache Weiterentwicklung stattfindet, sondern ein GPL-Programm mit anderen Programmen oder Programmbestandteilen kombiniert wird. Für die Beurteilung enthält die GPL einige Hinweise, die allerdings alles andere als eindeutig sind. 21

Allgemeine Grundlagen – Wann ist ein Softwaremodul ein »derived work«?

Ausgangspunkt für die Auslegung der GPL ist Ziffer 2, 2. Hauptabsatz, mit dem Kriterium, dass selbständige Softwaremodule nicht unter der GPL lizenziert werden müssen, wenn sie als eigenständige Werke weitergegeben werden: 22

> If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works.

Danach genügt es also nicht, dass die Softwarebestandteile inhaltlich selbständig sind, auch deren Verbreitung muss »als eigenständige Werke« erfolgen. Bemerkenswert ist, dass die GPL an dieser Stelle danach unterscheidet, in welcher Form der Vertrieb der Softwarebestandteile erfolgt. Eine Verbreitungsform, die als eigenständig anzusehen ist, liegt dann nicht vor, wenn ein selbständiges Softwaremodul mit den GPL-Bestandteilen als »Teil eines Ganzen« verbreitet wird

> 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.

Die GPL ist an dieser Stelle allerdings widersprüchlich: Zunächst scheint der Grundsatz, wonach selbstständige, nicht abgeleitete Softwarebestandteile nicht unter die GPL gestellt werden müssen, durch den Halbsatz »when you distribute them as separate works« eingeschränkt. In dem folgenden, oben zitierten Satz in Ziffer 2 GPL wird dies dahingehend erläutert, dass eigenständige Bestandteile, die als »Teil eines Ganzen« (»as part of a whole«) verbreitet werden, dennoch der GPL unterstellt werden müssten. Der Ausdruck »part of a whole« wird dahingehend konkretisiert, dass es ein »work based on the program« sein müsse. Der Begriff »work based on the program« wird in Ziffer 0 GPL wiederum als »abgeleitetes Werk« (»derivative work«) definiert, das das ursprüngliche Programm oder Teile davon enthält. Somit wird nicht eindeutig klar, wann ein Softwarebestandteil einerseits ein eigenständiges Werk, andererseits zugleich Teil eines Ganzen sein kann, da beide Begriffe (»separate works«, »part of a whole«) letztlich über den Begriff »derivative work« definiert werden. 23

Trotz dieses »Zirkelschlusses« lässt sich der GPL unter Berücksichtigung von Ziffer 2, 3. Hauptabsatz GPL entnehmen, dass unabhängige Softwarebestandteile unter anderen Lizenzen als der GPL (auch proprietären Lizenzen) verbreitet werden dürfen:

> Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

Hier wird deutlich, dass dabei auf die Kontrolle der »Verbreitung« der Software abgestellt wird. Ziffer 2 GPL, letzter Absatz, stellt klar, dass die Bewertung dessen, ob zwei Softwaremodule ein einheitliches Ganzes bilden oder ein abgeleitetes Werk (»derivative work«) besteht, nicht davon abhängt, ob sie zusammen auf einem Datenträger oder auf einer Hardware vertrieben werden. Andere formale Aspekte müssen daher für die Abgrenzung entscheidend sein:

24 Vor diesem Hintergrund ergibt die Unterscheidung in Ziffer 2, Absatz 2 der GPL, ob ein einheitliches »Ganzes« verbreitet wird oder »as separate works«, dann Sinn, wenn sie folgendermaßen ausgelegt wird: Es soll vermieden werden, dass GPL-Software mit selbständigen Softwaremodulen in einer Form verbunden wird, die es dem Nutzer unmöglich macht, die GPL-Bestandteile ohne weiteres als eigenständige Teile zu erkennen und zu nutzen. Daher muss die Ziffer 2 GPL wohl so verstanden werden, dass sowohl inhaltliche als auch formale Voraussetzungen erfüllt sein müssen, damit ein Softwaremodul nicht als »derivative work« anzusehen ist, obwohl es mit dem GPL-Programm in irgendeiner Form technisch verbunden wurde.

25 Bei der Bewertung dieser inhaltlichen und formalen Kriterien ist die technische Form der Verknüpfung von zwei Softwarebestandteilen nur von indizieller Bedeutung. Zwar werden bei inhaltlich selbständigen Programmen in der Regel übliche Kommunikations- und Verknüpfungstechniken verwendet, aber es ist einem Programmierer auch möglich, diese Techniken künstlich auf inhaltlich einheitliche Programme anzuwenden. Damit kann aber rechtlich noch keine »Selbstständigkeit« herbeigeführt werden, da ansonsten eine einfache Umgehung des »Copyleft« möglich wäre. Andererseits sprechen Techniken, wie die gemeinsame Ausführung von Softwaremodulen in einem Adressraum, regelmäßig für eine enge inhaltliche Einheit. Dies muss aber im Einzelfall nicht so sein, sondern kann auch bloß technische Gründe haben.

Daneben ist die technische Gestaltung aber auch bedeutsam bei der Beurteilung der formalen Trennung. So werden Pipes, Sockets und Kommandozeilenargumente als übliche Kommunikationsmittel zwischen selbständigen Softwarebestandteilen angesehen, während bei Änderungen des GPL-Source-Codes oder dem Vertrieb in einem Executable stets ein »derivative work« vorliegt und demzufolge dann die gesamte Software nur unter der GPL weitergegeben werden darf.

26 Es ist daher notwendig, inhaltlich *und* funktional zu bewerten, ob zwei Softwarebestandteile eine Einheit bilden oder ob ihnen selbstständige und unabhängige Funktionen zukommen. Im Einzelfall kann dies zu schwierigen Auslegungsfragen führen, die sich

nicht immer zweifelsfrei beantworten lassen. Ein starkes Indiz für eine inhaltliche Abhängigkeit ist sicherlich der Umstand, dass ein Softwarebestandteil speziell für das GPL-Programm entwickelt wurde und nur mit diesem ablauffähig ist. Ist hingegen ein Programm nicht nur mit dem GPL-Programm ablauffähig, sondern auch mit anderer Software, etwa mit einem anderen unixartigen Betriebssystemkernel als Linux, so ist dies ein überzeugendes Argument für eine inhaltliche Selbstständigkeit, auch wenn für die jeweilige Portierung kleine Anpassungen (zum Beispiel für Schnittstellen) erforderlich sind. Das heißt aber noch nicht, dass ein Programm, das nur für eine Plattform (zum Beispiel Linux) entwickelt wurde, deswegen nie eigenständig sein könnte. Hier ist im Einzelfall eine Wertung nach inhaltlichen und formalen Kriterien erforderlich.

27 Wie auch immer die technische Verbindung zwischen zwei Softwarebestandteilen gestaltet ist – man wird verlangen müssen, dass die Softwaredateien in deutlich abgegrenzter Form (etwa in voneinander getrennten Dateien) verbreitet werden, so dass auch technisch-formal eigenständige Softwarebestandteile vorliegen, sei es im Source-Code, sei es im Objekt-Code. Wird hingegen nur ein Softwaremodul mit dem GPL-Programm verbunden und die beiden Teile nicht in einer Form verbreitet, in der sie deutlich voneinander abgegrenzt sind, liegt faktisch ein einheitliches Werk vor: Der Nutzer kann die Bestandteile nicht mehr ohne weiteres trennen und damit die mit dem GPL-Bestandteil verbundenen Freiheiten nicht nutzen. Es kann also schon einen Unterschied machen, ob die Softwarebestandteile in einer Objekt-Code-Datei zusammengeworfen wurden oder ob sie sorgfältig getrennt wurden. Umgekehrt heißt das aber nicht, dass bloße Weiterentwicklungen nur deshalb als eigenständige Programme angesehen werden dürfen, nur weil sie in eine eigene Datei »ausgegliedert« wurden (siehe oben Rz. 25). Hier mangelt es an der inhaltlichen Eigenständigkeit, so dass stets ein abgeleitetes Werk vorliegt. Denn sowohl die inhaltliche als auch die formale Trennung müssen für die Eigenständigkeit vorliegen.

28 Vor dem Hintergrund der hier vertretenen Abgrenzungskriterien (Gerichtsentscheidungen existieren dazu noch nicht) ergibt sich folgende Abstufung für Software, die zusammen mit einem GPL-Programm verbreitet werden soll:

1. **Programme oder Softwarebestandteile, die (inhaltlich) nicht voneinander abgeleitet sind, können unter unterschiedlichen Lizenzen verbreitet werden, wenn sie auch formal getrennt vorliegen;**
2. **Programme oder Softwarebestandteile, die (inhaltlich) nicht voneinander abgeleitet sind, müssen aber dann insgesamt unter der GPL verbreitet werden, wenn sie ein »Ganzes« bilden, weil keine formale Trennung besteht;**
3. **Programme oder Softwarebestandteile, die (inhaltlich) voneinander abgeleitet sind, dürfen stets nur unter der GPL gemeinsam verbreitet werden.**

Im Folgenden werden anhand von Beispielen einige typische Problemfälle vorgestellt.

Kernelmodule

29 In der Praxis ist stark umstritten, ob ein Kernelmodul als »derivative work« betrachtet werden muss. Die Auseinandersetzungen um Binär-Treiber für Linux werden mit Heftigkeit geführt. Man wird wohl nicht für sämtliche Kernelmodule eine einheitliche Antwort finden können: Wann ein Kernelmodul von Linux »abgeleitet« ist, hängt stark von der technischen Umsetzung ab und richtet sich nach den oben dargelegten Kriterien. Sofern die Kernelfunktionen nur insoweit genutzt werden, dass die eigenen Softwaremodule mit dem Kernel über eine Schnittstelle »kommunizieren«, kann ein unabhängiges Softwaremodul vorliegen. Entscheidend ist im Regelfall, ob das Kernelmodul funktional eigenständig ist. Dafür spielen viele Aspekte eine Rolle. Wird ein Gerätetreiber eigens für Linux entwickelt – soll er also dessen integraler Bestandteil werden –, sprechen gute Gründe dafür, von einem »derivative work« auszugehen. Entsprechendes gilt, wenn das Kernelmodul technisch weitergehend in den Kernel eingebunden ist (etwa mittels »Hooks«) oder dessen »Infrastruktur« mehr als nur oberflächlich nutzt, so dass auf diese Weise offenbar wird, dass das Kernelmodul nicht unabhängig ist.

30 Es existieren allerdings auch Kernelmodule, die älter sind als Linux, etwa das Dateisystem AFS. Dort liegt es auf der Hand, dass sie als funktional eigenständig anzusehen sind, da sie gar nicht »für Linux« geschrieben sein können. Dies dürfte jedenfalls dann gelten, wenn für die Lauffähigkeit unter Linux nicht mehr als eine Anpassung für die entsprechenden Linux-Schnittstellen erforderlich war. Entsprechendes gilt für Kernelmodule, die zwar nicht älter sind als Linux, aber auch mit anderen Unix-Versionen ablauffähig sind und auf diese Weise ihre Eigenständigkeit zeigen. Es ist zulässig, solche Kernelmodule unter einer proprietären Lizenz zu verbreiten, wenn sie auch formal klar vom Linux-Kernel getrennt werden. Anders mag die Lage sein, wenn für die Portierung des Kernelmoduls auf Linux tiefergreifende Änderungen erforderlich sind als nur die Anpassung für die Linux-Schnittstelle. Dies muss jeweils für den Einzelfall bewertet werden.

31 In der Praxis wird es von den Kernelentwicklern vielfach geduldet, dass reine Binär-Kernelmodule in Linux eingefügt werden, um dem Betriebssystem damit weitere Hardware zu erschließen. Dies heißt keinesfalls, dass dieses Vorgehen auch rechtlich zulässig ist oder eine Selbstbindung nach sich zieht oder dass trotz einer möglichen GPL-Verletzung niemand mehr rechtlich dagegen vorgehen dürfte. Denn diejenigen Programmierer, die Beiträge zu Linux geleistet haben, ohne sich mit solchen Treibern einverstanden zu erklären, werden durch die Duldung anderer nicht verpflichtet (vergleiche etwa Erik Anderson *http://www.uwsg.iu.edu/hypermail/linux/kernel/0309.1/2152.html*).

32 Um die weitere Verbreitung von Binär-Kernelmodulen zu verhindern, wurde im Jahr 2001 bei der Entwicklung des Kernels 2.4 für die Einbindung von Kernelmodulen die Angabe der Lizenz des Moduls im »include/linux/module.h«-Headerfile technisch vorgeschrieben (*http://kerneltrap.org/sup/2991/module.h*, vergleiche auch *http://linuxdevices.com/articles/AT7432187078.html*):

```
/*
 * The following license idents are currently accepted as indicating free
 * software modules
 *
 *       "GPL"             [GNU Public License v2 or later]
 *       "GPL v2"          [GNU Public License v2]
 *       "GPL and additional rights"[GNU Public License v2 rights and more]
 *       "Dual BSD/GPL"    [GNU Public License v2
 *                          or BSD license choice]
 *       "Dual MPL/GPL"    [GNU Public License v2
 *                          or Mozilla license choice]
 *
 * The following other idents are available
 *
 *       "Proprietary"     [Non free products]
 *
 * There are dual licensed components, but when running with Linux it is the
 * GPL that is relevant so this is a non issue. Similarly LGPL linked with GPL
 * is a GPL combined work.
 *
 * This exists for several reasons
 * 1.    So modinfo can show license info for users wanting to vet their setup
 *       is free
 * 2.    So the community can ignore bug reports including proprietary modules
 * 3.    So vendors can do likewise based on their own policies
 */
#define MODULE_LICENSE(_license) MODULE_INFO(license, _license)
```

Damit muss sich jedes Kernelmodul zunächst »anmelden«. Steht das Modul nicht unter der GPL, wird der Kernel als »tainted« markiert (/proc/sys/kernel/tainted) und die Meldung »modulname: module license ›xxx‹ taints kernel.« ausgeworfen. Erklärtes Ziel dieser Kontrolle, die sich als Digital Rights Management-System bezeichnen lässt, ist es, technische Anfragen (Bug reports) für einen »Tainted-Kernel« im Vornherein abzulehnen und auf den Hersteller des Binär-Kernelmodules zu verweisen (vergleiche *http://www. tux.org/lkml/#s1-18; http://lwn.net/2001/0906/a/ac-license.php3*). Dahinter steht das praktische Problem, dass Kernelmodule mit neuen Kernelversionen kompiliert werden müssen und so Fehler auftreten können, die nur mit Kenntnis des Source-Codes des Kernelmoduls behebbar sind. Zudem können solche Fehler an unerwarteten Stellen auftreten, so dass sich erst mit erheblichem Aufwand die »Herkunft« des Problems ermitteln lässt. Dies war zum Beispiel bei Nividia-Treibern der Fall.

Ähnliche Wirkung haben die »GPL-only«-Kernelsymbole. Mit dem Makro »EXPORT_ 33
SYMBOL_GPL«, das Kernelfunktionen für Module zugänglich macht, können manche Symbole nicht gegen einen Code gelinkt werden, der nicht unter der GPL lizenziert ist. Damit wird nicht nur ein Warnhinweis erzeugt, sondern die betroffenen Symbole können überhaupt nicht von Kernelmodulen genutzt werden, die einer proprietären Lizenz unterstellt sind. In den entsprechenden FAQ der Kernelliste wird darauf hingewiesen, dass bereits bestehende Symbole nicht in »GPL-only« geändert werden sollen (*http://www.tux.org/lkml/#s1-19*).

34 In diesem Zusammenhang sei noch die Schnittstelle für »Linux Security Modules« (LSM) erwähnt, an die – wie der Name schon sagt – sicherheitsrelevante Module angebunden werden können. Hier enthält das »include«-Headerfile seit dem Kernel 2.4.10 den Hinweis:

> This file is GPL. See the Linux Kernel's COPYING file for details. There is controversy over whether this permits you to write a module that #includes this file without placing your module under the GPL. Consult your lawyer for advice.

Damit soll bei der Erstellung von LSMs zum einen auf das Problem hingewiesen werden, dass schon durch die Einbeziehung eines Headerfiles unter der GPL das gesamte Kernelmodul als »derivative work« angesehen werden könnte (siehe unten Rz. 37); zum anderen sollte gerade bei sicherheitsrelevanten Bestandteilen der Source-Code offen liegen, um eine adäquate Sicherheitskontrolle zu ermöglichen und um einfach nachprüfen zu können, ob die Gefahr besteht, mit US-Exportkontrollgesetzen für Kryptographie in Konflikt zu geraten (siehe unten Ziffer 7 GPL Rz. 4).

35 Die genannten »DRM-Systeme« zur Durchsetzung der GPL für Kernelmodule werfen zahlreiche Fragen auf. Dies betrifft sowohl die Zulässigkeit solcher Maßnahmen als auch die rechtliche Möglichkeit, sie zu umgehen. Erlaubt die GPL überhaupt die Verwendung von »GPL-only«-Makros? Die Beantwortung dieser Frage hängt von zwei Aspekten ab: Kommt diesem Makro eine lizenzrechtliche oder nur technische Bedeutung zu und wie ist der Lizenzstatus von Kernelmodulen an sich? Wenn man wie hier davon ausgeht, dass nicht alle Kernelmodule mit Linux ein »derivative work« bilden, kommt es letztlich auf den ersten Aspekt an: Wird nur eine technische Hürde aufgebaut oder möchte der Autor von Symbolen mit dem Makro »EXPORT_SYMBOL_GPL« auch rechtlich durchsetzen, dass ausschließlich die unter der GPL lizenzierten Kernelmodule verwendet werden dürfen? Dies würde in der Tat gegen Ziffer 6 der GPL (»You may not impose any further restrictions ...«, siehe unten Ziffer 6 GPL Rz. 6) verstoßen, da dem Lizenznehmer aufgezwungen würde, nicht nur abgeleitete Werke unter der GPL zu lizenzieren, sondern jedes Kernelmodul, wenn er die Software zusammen verbreiten will. Linus Torvalds schreibt dem Makro nur eine Warnfunktion zu, die lizenzrechtlich irrelevant sei (vergleiche *http://www.uwsg.iu.edu/hypermail/linux/kernel/0210.2/0617.html*). Folgt man dieser Auffassung, die sicherlich die besseren Gründe für sich hat – »EXPORT_SYMBOL_GPL« sei nur eine technische Anweisung und enthalte keine nach außen hinreichend erkennbare rechtliche Forderung (vergleiche dazu Alan Cox, *http://www.uwsg.iu.edu/hypermail/linux/kernel/0309.1/1929.html*) – so darf auch jedermann Linux so verändern, dass Binär-Kernelmodule alle Kernelfunktionen nutzen können, die durch das Makro »EXPORT_SYMBOL_GPL« ausgeschlossen sein sollen. Entsprechendes gilt für die Anmeldung von Kernelmodulen über MODULE_LICENSE. Ein Beispiel dafür ist die Umgehung durch das Unternehmen LinuxAnt (vergleiche *http://kerneltrap.org/node/ view/2991*).

36 Alan Cox hat darauf hingewiesen, dass die Entfernung oder Beeinträchtigung der Lizenzabfrage einen Verstoß gegen die DRM-Vorschriften des US-Rechts darstellen könnten (*http://lwn.net/2001/0906/a/ac-tainted.php3*). Wäre dies der Fall, würde eine Weiterver-

breitung von Linux mit dem »Tainted-Kernel«-Hinweissystem entsprechend Ziffer 7 der GPL wegen Verstoßes gegen die GPL verboten sein, weil dem Nutzer entgegen Ziffer 6 der GPL zusätzliche Bedingungen aufgezwungen werden, die nicht mit der GPL vereinbar sind (immer vorausgesetzt, dass nicht alle Kernelmodule als »derivative work« anzusehen sind). Würden also die US-Gesetze zum DRM die Bearbeitung von Linux in einer bestimmten Form verbieten, nämlich die Umgehung der »MODULE_LICENSE«-Lizenzabfrage, wäre die Weitergabe mit diesem Abfragesystem lizenzrechtlich verboten. Nach deutschem Urheberrecht stellt sich dieses Problem nicht, da eine entsprechende Veränderung des Linux-Kernels nicht gegen die entsprechende DRM-Regelung des § 95c) UrhG verstoßen würde, da diese Vorschrift nach § 69a) Absatz 5 UrhG (»Die Vorschriften der §§ 95a bis 95d finden auf Computerprogramme keine Anwendung.«) gar nicht auf Software anwendbar ist. Daher ist in Deutschland sowohl der Vertrieb der Linux-Versionen mit »EXPORT_SYMBOL_GPL« zulässig als auch die Umgehung dieser Hinweissysteme – unabhängig davon, ob ein bestimmtes Kernelmodul ein »derivative work« darstellt oder nicht.

Headerfiles

Ein Headerfile enthält eine Funktionsdeklaration des Programms, zu dem es gehört. Oft- 37
mals ist dies eine Programmbibliothek. Darin sind Parameter und Strukturinformationen enthalten, die erst die Kommunikation zwischen zwei Programmmodulen erlauben. Damit kommen Headerfiles an zwei Stellen zum Einsatz: Zum einen gehören sie zu dem Programm, dessen Deklaration sie enthalten, zum anderen werden sie beim Kompilieren von anderen Programmen (oder auch dem eigenen Programm) durch den Befehl #include <name.h> in den Quelltext eingefügt (in der Programmiersprache C).

Regelmäßig gehören Headerfiles inhaltlich nur zu einem Programm (etwa einer Bibliothek) 38
und ihre Einbeziehung durch ein anderes Programm beinträchtig dessen Selbständigkeit per se noch nicht. Allerdings wird man nicht umhin können, ein Executable, das den Objekt-Code eines kompilierten, unter der GPL lizenzierten Headerfile enthält, als abgeleitetes Werk anzusehen, da dann die für eine andere Lizenzierung erforderliche formale Trennung nicht vorliegt. Dies gilt jedenfalls dann, wenn ein Headerfile alleine die für einen urheberechtlichen Schutz erforderliche Schöpfungshöhe erreicht (dieses Erfordernis ergibt sich aus § 2 Absatz 2 UrhG. (»Werke im Sinne dieses Gesetzes sind nur persönliche geistige Schöpfungen.«). Sofern ein Headerfile nicht ausdrücklich von der Anwendung der GPL ausgenommen ist, bleibt so nur die Möglichkeit eines Vertriebs, bei dem kein einheitliches »Ganzes« entsteht. Dafür kommt als Möglichkeit in Betracht, die Dateien übersetzt im Objekt-Code, aber noch unverlinkt zu verbreiten oder gleich unkompiliert im Source-Code. Im Übrigen gehen die Autoren der LGPL in Ziffer 5 ebenfalls davon aus, dass es für die Frage, ob ein »derivative work« entsteht, wesentlich auch auf die Frage ankommt, ob das Headerfile im Objekt-Code oder Source-Code vorliegt:

> When a »work that uses the Library« uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not.

Zwar ist die LGPL für die Auslegung der GPL nicht maßgeblich – ihr Text liegt dem Lizenznehmer in der Regel nicht vor –, aber sie gibt einen interessanten Hinweis auf das Verständnis der Autoren der Lizenz.

39 Die oben beschriebene Vorgehensweise erscheint zulässig, solange auf diese Weise nur die formale Trennung herbeigeführt werden soll. Die inhaltliche Selbständigkeit von zwei Programmen ist davon unabhängig und gesondert zu beurteilen. Wenn sie nicht vorliegt, genügt auch eine formale Trennung nicht, um die Lizenzierung unter der GPL zu vermeiden. Wird zum Beispiel in ein Headerfile ein Inline-Code aus dem GPL-Programm ausgelagert (siehe unten Rz. 40), das für ein anderes Programm verwendet werden soll, indem das Headerfile eingefügt wird, so ist das andere Programm inhaltlich von dem GPL-Code abhängig. Es darf als abgeleitetes Werk zusammen mit dem Headerfile nur unter der GPL vertrieben werden, unabhängig davon, ob dies im Source-Code, Objekt-Code oder als Executable geschieht.

Inlinefunktionen

40 Inlinefunktionen (oder »inline code«) werden im Regelfall noch nicht im Source-Code von einem Programmmodul in ein anderes Programmmodul übernommen, sondern erst während des Kompiliervorgangs beim Linken. Insbesondere Headerfiles (siehe oben Rz. 38) können solche Inlinefunktionen enthalten, um die Kommunikation mit einem anderen Programm oder die Vereinfachung bestimmter Abläufe leistungsfähiger zu gestalten.

```
inline int max (int a, int b)
{
 if (a > b) return a;
 else return b;

}
```

41 Zumeist handelt es sich bei Inlinefunktionen um eher kleine Codebestandteile. Sofern diese so wenig Eigenart besitzen, dass die erforderliche Schöpfungshöhe nicht erreicht wird, um als »Werk« urheberrechtlich geschützt zu sein, führt ihre Verwendung auch nicht zu einem »derivative work«. Umfangreicherer Inline-Code kann aber durchaus schutzfähig sein und ein Executable, das solche Inlinefunktionen enthält, die der GPL unterstellt sind, darf ebenfalls nur unter der GPL weiterverbreitet werden.

42 Erfolgt der Vertrieb in einer Form, die noch zu keiner formalen Verbindung der Inlinefunktion mit einem anderen Programm führt, kann entsprechend den für Headerdateien beschriebenen Voraussetzungen (siehe oben Rz. 38 ff.) ein Vertrieb unter verschiedenen Lizenzen zulässig sein.

Programmbibliotheken

Programmbibliotheken spielen bei der Softwareentwicklung eine bedeutsame Rolle: Mit 43
ihrer Hilfe können wiederkehrende Programmroutinen in einer Datei zusammengefasst und für den Zugriff von verschiedenen Programmen oder Programmbestandteilen bereitgestellt werden. Technisch ist danach zu unterscheiden, ob eine Bibliothek mit dem zugreifenden Programm statisch oder dynamisch verlinkt wird. Bei der statischen Verlinkung wird die Bibliothek mit einem Linker während des Kompilierens eingebunden, so dass in der Regel ein einheitliches Executable entsteht. Für den Austausch der statisch gelinkten Bibliothek ist daher die Erstellung eines neuen Executables notwendig. Dem gegenüber werden dynamische Bibliotheken gesondert kompiliert und während der Laufzeit zu dem zugreifenden Programm geladen. Sie werden dabei nur einmal in den Arbeitsspeicher geladen, auch wenn sie von mehreren Programmen gleichzeitig verwendet werden. Das ist besonders bei Standardbibliotheken von Bedeutung. Dynamische Bibliotheken sind separat austauschbar, etwa durch entsprechende Updates.

Entscheidend für die Frage, ob eine Programmbibliothek und ein auf sie zugreifendes 44
GPL-Programm abhängige Werke im Sinne der Ziffer 2 b) der GPL sind, ist der Zeitpunkt der Beurteilung. Entsprechendes gilt für eine Bibliothek unter der GPL und das darauf zugreifende Programm. Stellt man auf die Laufzeit ab, bilden sowohl statische als auch dynamische Bibliotheken mit dem zugreifenden Programm ein einheitliches »Ganzes«. Geht man hingegen von dem Zeitpunkt des Vertriebs aus, ließe sich vertreten, dass statische Bibliotheken »derivative works« sind, weil sie mit dem zugreifenden Programm ein einheitliches Executable bilden, während dynamische Bibliotheken auch in einer getrennten Datei selbständig vorliegen können. Da die GPL aber selbst auf den Vertriebsvorgang abstellt (»... when you distribute them as separate works«), erscheint es überzeugender, nicht auf die Laufzeit, sondern auf den Zeitpunkt des Vertriebs abzustellen.

Nun ist es sicherlich richtig, dass die Frage, ob eine Bibliothek statisch oder dynamisch 45
verlinkt ist, im Wesentlichen eine Frage der Programmiertechnik ist. Dennoch kann von dieser Technik abhängen, ob ein »derivative work« vorliegt oder nicht, da sie für die formale Trennung relevant ist (siehe oben Rz. 24 f.). Das heißt aber nicht, dass deswegen jede dynamisch verlinkte Bibliothek selbstständig von dem zugreifenden Programm ist. Hierfür kommt es wiederum auf den inhaltlichen Aspekt an; bei einer GPL-Applikation zum Beispiel darauf, ob die betroffene Bibliothek auch für andere Programme verwendbar ist oder gerade für ein konkretes GPL-Programm entwickelt wurde (siehe oben Rz. 26). Für den Fall, dass eine GPL-Bibliothek betroffen ist, kann die inhaltliche Selbständigkeit davon abhängen, ob das zugreifende Programm auch mit anderen Bibliotheken oder ohne die GPL-Bibliothek sinnvoll ablauffähig ist. Unabhängig davon kann ein Programm auch ohne Bibliothek unter einer eigenen Lizenz vertrieben werden, wenn kein GPL-Code enthalten ist (siehe oben Rz. 18).

Es lässt sich also keine pauschale Aussage zu der lizenzrechtlichen Einordnung von Bib- 46
liotheken als abgeleitetes Werk treffen. Wie oben ausgeführt, hängt dies von der Form des Vertriebs und von inhaltlichen Aspekten ab. Während Executables mit statisch ver-

linkten Bibliotheken stets insgesamt nur unter der GPL weitergegeben werden dürfen, kommt es bei dynamisch verlinkten Bibliotheken auf das Verhältnis zwischen Bibliothek und zugreifendem Programm an – besonders darauf, ob eine austauschbare Standardbibliothek vorliegt oder eine speziell auf ein Programm zugeschnittene Bibliothek. Es lässt sich durchaus vertreten, dass auch bei statischem Linken eine formale Trennung herbeigeführt werden kann, etwa indem Applikation und Bibliothek unkompiliert oder ungelinkt mit ausgeliefert werden.

47 Um freie Bibliotheken möglichst als Standards durchzusetzen, hat die FSF wichtige Programmbibliotheken wie die »glibc« einer eigenen Lizenz unterstellt, der LGPL (Library General Public License, seit der Version 2.1 Lesser General Public License genannt). Diese Lizenz erlaubt die Verlinkung mit proprietären Programmen, wenn eine Reihe von Bedingungen erfüllt sind, die die faktische Modifizierbarkeit der Bibliothek sichern sollen. Da die LGPL in ihrer Ziffer 3 zugleich die Nutzung unter der GPL gestattet – also eine Form des »Dual Licensing«, der Doppellizenzierung –, können GPL-Programme problemlos mit LGPL-Bibliotheken verbunden werden, und sind damit insgesamt unter der GPL lizenziert. Im Übrigen stellt die LGPL für die Bewertung, was als »derivative work« anzusehen ist, in ihrer Ziffer 5 ebenfalls auf die Form des Vertriebs ab, besonders darauf, ob ein Executable vorliegt oder das zugreifende Programm »in isolation«. Auch wenn die LGPL nicht für die Auslegung der GPL herangezogen werden kann, zeigt sich hier doch das Verständnis der Autoren der Lizenz.

Softwaretools

48 Grundsätzlich führt die Verwendung von Kompilern, Editoren und anderen Softwarewerkzeugen nicht dazu, dass die damit erstellte, kompilierte oder bearbeitete Software lizenzrechtlich beeinflusst wird. Mit dem GNU C Compiler kompilierte Programme müssen daher nicht unter die GPL gestellt werden. Etwas anderes gilt nur, wenn der Code des Softwarewerkzeugs in den Code des bearbeiteten Programms eingefügt wird und dieser eingefügte Code die notwendige Schöpfungshöhe besitzt (siehe oben Rz. 38). Dies kann zum Beispiel bei dem Parser Bison der Fall sein, der eigene Codebestandteile weiterverwendet. Um Bison eine weite Verwendbarkeit auch mit proprietären Programmen zu ermöglichen, wurde der GPL eine zusätzliche Ausnahme hinzugefügt:

> /* As a special exception, when this file is copied by Bison into a Bison output file, you may use that output file without restriction. This special exception was added by the Free Software Foundation in version 1.24 of Bison. */

49 Besonderheiten gelten für Softwaretools, die zur Installation oder Kompilierung des GPL-Programms erforderlich sind. Hier kann es nach Ziffer 3, 2. Hauptabsatz der GPL erforderlich sein, solche Tools mit dem Objekt-Code auszuliefern, selbst wenn sie nicht unter der GPL lizenziert werden müssen (siehe unten Ziffer 3 GPL Rz. 26).

Plugins

Für Plugins, die zumeist als »shared library« zu einem Programm hinzugebunden wer- 50
den, gelten die oben gemachten Erläuterungen für Programmbibliotheken und Kernelmodule entsprechend (siehe oben Rz. 29 ff., 43 ff.).

»GPL-kompatible« Lizenzen

Bei der Kombination eines GPL-Programms mit anderer Open Source Software kann eben- 51
falls die Situation entstehen, dass ein abgeleitetes Werk entsteht und die gesamte Software nur unter den Bedingungen der GPL vertrieben werden darf. In diesen Fällen stellt sich die Frage, ob die Lizenzen der betroffenen Programme »GPL-kompatibel« sind, d.h. keine Verpflichtungen enthalten, die die GPL nicht vorsieht, zugleich aber die Etablierung zusätzlicher Pflichten gestattet und damit einen »Wechsel« der gesamten Software zur GPL erlaubt. Dies ist insbesondere bei der modifizierten BSD-Lizenz ohne Werbeklausel der Fall (vergleiche dazu *http://www.gnu.org/philosophy/bsd.html*). Die BSD-Lizenzen erlauben ohne weiteres, Modifikationen der ihr unterstellten Software unter anderen Lizenzbedingungen zu nutzen. Das können nicht nur proprietäre Lizenzen sein, sondern eben auch die GPL. Im Ergebnis heißt dies, dass Software-Code unter GPL-kompatiblen Lizenzen in GPL-Programmen verwendet werden darf.

Die FSF gibt in ihrer Lizenz-Liste Hinweise darauf, welche Lizenzen sie als GPL-kompatibel 52
betrachtet und welche nicht (siehe *http://www.fsf.org/licenses/license-list.html#GNUGPL*). Ihre Auffassung ist zwar nicht rechtsverbindlich, gibt aber eine gute erste Übersicht. Zu den GPL-kompatiblen Lizenzen gehören danach insbesondere:

- Lesser General Public License V. 2.0
- BSD-Lizenz (ohne Werbeklausel)
- X11-License
- Artistic License v. 2.0
- Open LDAP License v. 2.7
- eCos License v. 2.0
- Public Domain

Ziffer 3 GPL

Olaf Koglin

> 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,