Mathematische Texte

Inhaltsverzeichnis

Lizenz

Bitte lesen Sie diese Lizenz gründlich durch, bevor Sie Lazarus die Dokimentation oder Teile dieser Software verwenden wollen. Sollten Sie nicht mit den hier aufgeführten Bedingungen einverstanden sein, ist die Nutzung oder Verwendung des Quellcodes zu diesen Produkt (oder auch Teile davon) nicht gestattet.

Bei der Verwendung darf kein kommerzieller Zweck der Gewinnerzielung entstehen !

Freeware und Shareware-Programme zeichnen sich vor allem dadurch aus, dass sie kostenlos beziehungsweise gegen einen relativ geringen Preis dem Nutzer die Verwendung teilweise hochwertiger Computerprogramme ermöglichen, die sich durchaus mit kommerziellen Produkten messen lassen können. Daher sind diese Softwaretypen gerade in Zeiten begrenzter finanzieller Mittel - auch im schulischen Umfeld sehr beliebt und kommen in vielfältigen Gebieten zum Einsatz (zum Beispiel bei der Netzwerkadministration, bei der Einrichtung von Servern, bei der Installation von Internet-Software auf den Nutzer-PCs oder bei Office-Anwendungen).

Da es sich auch bei Free- und Sharewareprogrammen um Computerprogramme handelt, die gemäß § 69a Urheberrechtsgesetz (UrhG) urheberrechtlich geschützt sind, liegt es bei diesen Softwareprodukten weitgehend in den Händen des Rechteinhabers zum Beispiel des Programmierers - zu bestimmen, in welchem Umfang diese durch dritte Personen genutzt werden dürfen.

Dabei können Freeware- und Sharewareprogramme nach den Lizenzbedingungen in der Regel beliebig kopiert und weitergegeben werden, dagegen ist vor allem eine Veränderung der Programme üblicherweise nicht gestattet. Es ist deshalb eigentlich auch nicht korrekt, wenn im Zusammenhang mit Free- und Shareware immer wieder der Begriff der "Public-Domain-Software" verwendet wird, der frei übersetzt "Software, die im öffentlichen Eigentum steht" bedeutet. Denn dies impliziert, dass die Software von der Öffentlichkeit beliebig genutzt und damit auch verändert werden darf (also gemeinfrei ist); letzteres ist bei Free- und Shareware aber gerade nicht der Fall.

Shareware kann man in zahlreiche Unterkategorien einteilen. So kann Shareware beispielsweise werbegestützte Software oder “Adware” sein, welche dem User Werbung anzeigen soll. Dies hat den Zweck Einnahmen für den Eigentümer zu generieren. Eine weitere Art bezeichnet man als Demoware, wobei es sich hier nur um eine Demoversion der Software handelt.
Dabei ist es oft eine Testversion mit einer festgelegten Zeitspanne (“Trialware”). In anderen Fällen kann die gesamte Funktionalität der Anwendung deaktiviert sein, so dass man zwar alle Funktionen sehen kannst, aber dafür bezahlen muss (“Crippleware”).

Weiterhin gibt es auch “Freemium”-Shareware, bei welcher einige Funktionen in der kostenlosen Version verfügbar sind, Sie diese jedoch bezahlen müssen, um die volle Funktionalität freischalten zu können.
Obwohl Shareware eine gute Option für jeden ist, der Software testen möchte, bevor man sich zu einem Kauf verpflichtet, sollten Sie dennoch vorsichtig sein. Denn Cyberkriminelle sind nur allzu bereit, den Eifer der Menschen auszunutzen, etwas gratis zu bekommen.

Dieses Versprechen freier Software ist eine gängige Social-Engineering-Taktik, mit der sie Internetnutzer dazu bringen, bösartige Software herunterzuladen.
Bei Shareware gibt es ebenso wie bei anderen Softwareprodukten keine Garantie dafür, dass die Software virenfrei ist. Aus diesem Grund ist es wichtig, dass der Benutzer sein Antivirenprogramm auf dem aktuellen Stand hält, bevor er Shareware herunterlädt.

Prinzipiell kann jeder Shareware herunterladen, solange er einen Computer hat, der die Systemanforderungen des Programms erfüllt. Es gibt jedoch einige Seiten, die nur Benutzern in bestimmten Ländern das Herunterladen von Shareware erlauben.

Nicht alle Shareware-Programme werden überall zum Kauf angeboten. Der beste Weg, um zu sehen, wo man ein bestimmtes Programm kaufen kann, ist, die Website des Entwicklers zu besuchen und herauszufinden, auf welchen Plattformen das Programm erhältlich ist.

Freeware ist Software, die kostenlos heruntergeladen werden kann, aber keine Lizenzgebühren mehr verlangt. Im Gegensatz dazu erfordert Shareware normalerweise eine Lizenzgebühr für den vollen Zugang zu allen Funktionen. Beide Modelle ermöglichen es Benutzern, Software kostenlos auszuprobieren, aber

Shareware ist normalerweise etwas umfangreicher und bietet mehr Funktionen.

Dieses Software-Produkt besitzt eine Hybrid-Lizenz zwischen Free- und Share-Ware. Sie können es kostenlos einsetzen und an andere Personen weitergeben, solange Sie keine Teile der Software ändern oder kopieren. Es bedarf einer schriftlichen Einwilligung des Entwicklers (Jens Kallup), sollten Teile geändert oder übernommen werden.

Sie können dieses Produkt auf mehreren lokalen Computer gleichzeitig nutzen. Eine öffentlich zugängliche Nutzung ist nicht bestrebt mit der aktuellen Version.
Obwohl bei der Entwicklung dieser Software viel Sorgfalt gepflegt wurde, können Fehler nicht ausgeschlossen werden.

Es werden daher keine Garantien auf entstanden Schäden oder Kosten übernommen, die während der Verwendung dieser Software entstehen. Alles erfolgt auf Eigenes Risiko !!!

Für zukünftige Entwicklungen können Sie jedoch einen kleinen Obolus in Form von Spenden hinterlegen.

Überblick

GNU DOKUMENTATION

GNU Manifest
Free Software und OpenSource Software

Von Richard M. Stallman

Deutsche Übersetzung: Peter Gerwinski

Den offiziellen englischen Originaltext finden Sie unter http://www.gnu.org/gnu/manifesto.html.

Anmerkung des Übersetzers: Dies ist eine inoffizielle Übersetzung des GNU-Manifests. Sie soll helfen, den Text zu verstehen und damit zur Verbreitung des GNU-Projekts beitragen. Der Übersetzer übernimmt jedoch keinerlei Garantie für die Richtigkeit der Übersetzung; wenn Sie in offiziellen Angelegenheiten bezug auf das GNU-Manifest nehmen wollen, halten Sie sich bitte an den englischsprachigen Originaltext.

This is an inofficial translation of the GNU Manifesto into German. The translation is intended to help people to understand the text and to help the GNU project to spread. The translator does not provide any warranty for correctness of the translation; if you intend to refer to to GNU Manifesto in official concerns, please refer to the authentic English version.

Das unten abgedruckte GNU-Manifest wurde von Richard Stallman zur Entstehungszeit des GNU-Projekts geschrieben, um zu Teilnahme und Unterstützung aufzurufen. Während der ersten Jahre wurde es geringfügig geändert, um neue Entwicklungen zu berücksichtigen, aber jetzt erscheint es uns am besten, es in der Form zu belassen, in der es die meisten Leute gesehen haben.

Seit dieser Zeit haben wir Einiges über häufige Mißverständnisse gelernt, die sich durch eine andere Wortwahl hätten vermeiden lassen. 1993 wurden Fußnoten ergänzt, um diese Punkte zu klären.

Aktuelle Informationen über verfügbare GNU-Software finden Sie auf den GNU-Webseiten*, insbesondere in der* Software-Liste*. Wie Sie selbst beitragen können, finden Sie unter* http://www.gnu.org/help*.*

Was ist GNU? GNU ist nicht gleich Unix!

GNU steht für „**G**NU ist **n**icht gleich **U**nix“ und ist der Name eines kompletten, Unix-kompatiblen Software-Systems, das ich gerade schreibe, um es dann an jeden, der es brauchen kann, frei weiterzugeben. (1) Einige andere Freiwillige unterstützen mich. Beiträge in Gestalt von Zeit, Geld, Programmen und Material werden dringend benötigt.

Bis jetzt haben wir einen Emacs-Texteditor mit Lisp, um Editorkommandos zu schreiben, einen Quelltextdebugger, einen yacc-kompatiblen Parsergenerator, einen Linker und etwa 35 Dienstprogramme. Ein Kommandointerpreter (*shell*) ist beinahe fertig. Ein neuer, portabler, optimierender C-Compiler hat sich selbst compiliert und könnte dieses Jahr herausgegeben werden. Anfänge zu einem *Kernel* existieren, aber es werden noch viele Funktionen benötigt, um Unix zu emulieren. Sobald der Kernel und der Compiler vollendet sind, wird es möglich sein, ein zur Programmentwicklung einsatzfähiges GNU-System herauszugeben. Wir werden TeX als Textsatzsystem einsetzen, aber auch an einem nroff wird gearbeitet. Auch werden wir das freie, portable X-Window-System verwenden. Danach werden wir ein portables *Common Lisp* hinzufügen, ein *Empire*-Spiel, ein Tabellenkalkulationsprogramm und hunderte anderer Dinge plus *on-line*-Dokumentation. Wir hoffen, schließlich alles Nützliche, was normalerweise zu einem Unix-System gehört, anbieten zu können – und mehr.

GNU wird in der Lage sein, Unix-Programme laufen zu lassen, aber es wird nicht mit Unix identisch sein. Auf der Grundlage unserer Erfahrungen mit anderen Betriebssystemen werden wir alle gebräuchlichen Verbesserungen vornehmen; insbesondere sind längere Dateinamen geplant, Datei-Versionsnummern, ein absturzsicheres Dateisystem, eventuell Dateinamen-Komplettierung, terminalunabhängige Ausgabe und schließlich ein auf Lisp basierendes Window-System, durch welches mehrere Lisp-Programme und gewöhnliche Unix-Programme ein- und denselben Bildschirm miteinander teilen können. Sowohl C als auch Lisp werden als Systemsprogrammiersprachen verfügbar sein. Für die Kommunikation beabsichtigen wir, UUCP, MIT-Chaosnet sowie die Internet-Protokolle zu unterstützen.

GNU zielt zunächst auf Maschinen der 68000/16000-Klasse mit virtuellem Speicher, weil es auf diesen am leichtesten lauffähig gemacht werden kann. Die Zusatzarbeit, es auf kleinere Maschinen zu portieren, überlassen wir jemandem, der es auf diesen verwenden will.

Um fürchterliche Verwechselungen zu vermeiden, sprechen Sie bitte auch im Englischen das „G“ in „GNU“ mit aus.

Warum ich GNU schreiben muß

Ich glaube, daß es das Gebot der Nächstenliebe verlangt, daß ich ein Programm, das mir gefällt, mit anderen teile, denen es ebenfalls gefällt. Software-Anbieter hingegen wollen die Anwender isolieren und beherrschen, wobei sie jeden Anwender dazu verpflichten, nicht mit anderen zu teilen. Ich weigere mich, die Solidarität mit anderen Anwendern in dieser Weise zu brechen. Ich kann nicht mit gutem Gewissen einen Nichtoffenbarungsvertrag oder einen Software-Lizenzvertrag unterzeichnen.

Damit ich ehrlich bleiben und trotzdem weiterhin Computer benutzen kann, habe ich mich entschlossen, eine genügend große Sammlung von freier Software zusammenzustellen, so daß ich in der Lage sein werde, ohne jegliche nicht-freie Software auszukommen. Ich habe meinen Beruf im *AI lab* aufgegeben, um dem *MIT* keinen rechtlichen Vorwand zu bieten, mich daran zu hindern, GNU weiterzugeben.

Warum GNU Unix-kompatibel sein wird

Unix ist nicht mein ideales Betriebssystem, aber es ist nicht übel. Die wesentlichen Eigenschaften von Unix scheinen gut zu sein, und ich denke, daß ich fehlendes ergänzen kann, ohne die guten Eigenschaften zu verderben. Außerdem wird ein Unix-kompatibles System für viele Menschen eher annehmbar sein.

Wie GNU erhältlich sein wird

GNU ist nicht in der *public domain*. Zwar wird jedem gestattet sein, GNU zu modifizieren und weiterzugeben, aber keinem Distributor wird es erlaubt sein, die Weiterverbreitung von GNU einzuschränken; sprich: proprietäre Modifikationen werden nicht erlaubt sein. Ich möchte sicherstellen, daß alle Versionen von GNU frei bleiben.

Warum viele andere Programmierer mithelfen wollen

Ich habe viele andere Programmierer gefunden, die vom GNU-Projekt begeistert sind und ihre Hilfe anbieten.

Viele Programmierer sind mit der Kommerzialisierung von Systemsoftware unzufrieden. Es mag ihnen die Möglichkeit geben, mehr Geld zu machen, aber es zwingt sie gleichzeitig, andere Programmierer im allgemeinen als Gegner anstatt als Kameraden zu betrachten. Der fundamentale Akt der Freundschaft zwischen Programmierern ist das Teilen von Programmen; derzeitige Vermarktungspraktiken verbieten Programmierern im wesentlichen, sich gegenseitig als Freunde zu behandeln. Der Käufer von Software hat die Wahl zwischen Freundschaft und Gesetzestreue. Naturgemäß entscheiden viele, daß Freundschaft für sie wichtiger ist, aber diejenigen, welche an das Gesetz glauben, haben eine schwere Entscheidung. Sie werden zynisch und betrachten Programmierung nur noch als eine Möglichkeit, Geld zu machen.

Wenn wir an und mit GNU anstelle von proprietären Programmen arbeiten, können wir gleichzeitig zu jedem gastfreundlich sein und dem Gesetz genügen. Außerdem dient GNU uns als inspirierendes Beispiel und als Banner, andere zu sammeln, um sich uns beim Teilen anzuschließen. Dies kann uns ein Gefühl der Harmonie bringen, das beim Gebrauch nicht-freier Software unmöglich wäre. Für jeden zweiten Programmierer, mit dem ich gesprochen habe, ist dies ein wichtiges Glück, das durch Geld nicht ersetzt werden kann.

Wie Sie selbst beitragen können

Ich bitte Computerhersteller um Spenden in Gestalt von Maschinen und Geld. Ich bitte Einzelpersonen um Spenden in Gestalt von Programmen und Arbeit.

Wenn Sie uns eine Maschine zur Verfügung stellen, können Sie damit rechnen, daß GNU relativ früh darauf laufen wird. Die Maschinen sollten komplette, gebrauchsfertige Systeme sein, in einer Wohnung benutzt werden können und keine außergewöhnliche Kühlung oder Energieversorgung benötigen.

Ich habe sehr viele Programmierer gefunden, die bereit sind, einen Teil ihrer Arbeitszeit GNU zu widmen. Für die meisten Projekte würde derartig verteilte Teilzeitarbeit schwer zu koordinieren sein; die voneinander unabhängig geschriebenen Teile würden nicht zusammenarbeiten. Für die spezielle Aufgabe jedoch, Unix zu ersetzen, existiert dieses Problem nicht. Ein komplettes Unix-System enthält hunderte von Dienstprogrammen, von denen jedes separat dokumentiert ist. Die meisten Schnittstellen sind durch Unix-Kompatibilität festgelegt. Wenn jeder Beteiligte einen kompatiblen Ersatz für ein Unix-Dienstprogramm schreibt und dafür sorgt, daß es an der Stelle der Originalkomponente richtig arbeitet, dann werden diese Dienstprogramme auch richtig arbeiten, wenn man sie zusammensetzt. Selbst wenn wir Murphy erlauben, ein paar unerwartete Probleme zu produzieren, sollte das Zusammensetzen dieser Komponenten eine durchführbare Aufgabe sein. (Der Kernel wird eine engere Zusammenarbeit erfordern, daher wird eine kleine Gruppe daran arbeiten.)

Sollte ich Geldspenden erhalten, werden mich diese in die Lage versetzen, ein paar Leute in Voll- oder Teilzeitarbeit einzustellen. Die Gehälter werden für den Standard von Programmierern nicht hoch sein, aber ich suche Leute, für die das Bilden von Gemeinschaftsgeist wichtiger ist als Geld zu scheffeln. Ich sehe dies als einen Weg an, es ausgewählten Leuten zu ermöglichen, ihre gesamte Energie in ihre Arbeit an GNU zu investieren, indem ich sie von der Notwendigkeit freimache, ihren Lebensunterhalt auf andere Weise zu verdienen.

Warum dies allen Computerbenutzern nützen wird

Sobald GNU geschrieben sein wird, wird jeder in der Lage sein, gute Systemsoftware so frei wie Luft zu bekommen. (2)

Dies bedeutet mehr, als nur jedem den Preis für eine Unix-Lizenz zu erspraren. Es bedeutet, daß viele überflüssige Anstrengungen, Systemsoftware jedesmal neu zu programmieren, vermieden werden können. Dieselben Anstrengungen können stattdessen eingesetzt werden, uns auf dem Gebiet weiterzubringen.

Der komplette Quelltext des Systems wird für jeden verfügbar sein mit dem Ergebnis, daß ein Benutzer, der Veränderungen in dem System benötigt, immer die Freiheit haben wird, diese selbst vorzunehmen oder einen Programmierer oder eine Firma damit zu beauftragen. Die Benutzer werden nicht länger von der Gunst eines einzigen Programmierers oder einer einzigen Firma abhängig sein, welche den Quelltext besitzt und daher als einzige Veränderungen vornehmen kann.

Schulen können ein besseres pädagogisches Umfeld bieten, wenn sie die Schüler dazu anhalten, den Code des Betriebssystems zu studieren und zu verbessern. Das *Harvard's computer lab* verlangte, daß kein Programm im System installiert werden durfte, dessen Quelltext nicht öffentlich zugänglich war – und hielten dies aufrecht, indem bestimmte Programme tatsächlich nicht installiert wurden. Dies hat mich sehr inspiriert.

Und schließlich wird auch der Verwaltungsaufwand vermieden, zu überlegen, wem die Systemsoftware gehört und was man damit tun darf und was nicht.

Wenn Menschen für das Benutzen eines Programms einschließlich des Anfertigens von Kopien zahlen müssen, entstehen fürchterliche Kosten für die Gesellschaft durch den schwerfälligen Mechanismus, der notwendig ist, um herauszufinden für wie viel (d.h. für welche Programme) eine Person zahlen muß, und nur ein Polizeistaat kann jeden dazu zwingen, diese Verträge einzuhalten. Stellen Sie sich eine Raumstation vor, in der die Luft zu hohen Kosten hergestellt werden muß: Es mag fair sein, Atemluft pro verbrauchten Liter zu berechnen, aber das Tragen von Gasmasken mit Meßeinrichtungen – Tag und Nacht – ist intolerabel, selbst wenn der Luftpreis für jeden erschwinglich ist. Und allgegenwärtige Fernsehkameras, die überwachen, ob jemand die Maske abnimmt, sind abscheulich. Es ist besser, die Luftanlage durch eine Kopfsteuer zu finanzieren und die Masken wegzuwerfen.

Das ganze oder teilweise Kopieren eines Programms ist für einen Programmierer so natürlich wie Atmen – und so produktiv. Es sollte genauso frei sein.

Einige leicht zu entkräftende Einwände gegen die Ziele von GNU

„Niemand wird es benutzen, wenn es frei ist, weil dies bedeutet, daß Sie sich nicht auf Wartung verlassen können.“

„Sie müssen etwas für das Programm berechnen, um Service anbieten zu können.“

Wenn die Menschen lieber für GNU mit Service bezahlen, als GNU ohne Service frei zu erhalten, sollte eine Firma, die speziell diesen Service für Leute anbietet, die GNU frei erhalten haben, rentabel sein. (3)

Wir müssen zwischen Wartung in Gestalt von echter Programmierarbeit und Benutzerhilfe unterscheiden. Ersteres ist etwas, das Sie von einem Software-Händler nicht erwarten können: Wenn Ihr Problem nicht von genügend Leuten geteilt wird, wird der Händler Sie zum Teufel schicken.

Wenn Ihr Unternehmen darauf angewiesen ist, sich auf Wartung zu verlassen, ist der einzig gangbare Weg, alle nötigen Quelltexte und Werkzeuge vorliegen zu haben, denn dann können Sie jede verfügbare Person einstellen, Ihr Problem zu lösen, und sind nicht von der Gunst einer speziellen Person abhängig. Mit Unix ist dies infolge des hohen Preises der Quelltexte für die meisten Unternehmen unerschwinglich; mit GNU wird dies leicht sein. Es kann immer noch sein, daß keine kompetente Person zur Verfügung steht, aber dies liegt dann nicht an den Vertriebsbedingungen. GNU löst nicht alle Probleme der Welt, sondern nur bestimmte.

Gleichzeitig sind Anwender ohne Computerwissen auf Hilfe angewiesen: Dinge für sie erledigen, die sie leicht selbst tun könnten, aber nicht wissen, wie.

Derartige Dienste könnten von Firmen angeboten werden, die gerade solche Benutzerhilfen und Reparaturservice anbieten. Wenn es stimmt, daß Benutzer es vorziehen, für ein Produkt mit Service zu bezahlen, werden sie auch bereit sein, den Service zu bezahlen, während sie das Produkt frei erhalten haben. Die Serviceunternehmen werden in Qualität und Preis miteinander konkurrieren; die Benutzer werden nicht an ein spezielles gebunden sein. Gleichzeitig können diejenigen von uns, die den Service nicht benötigen, das Programm benutzen, ohne für den Service bezahlen zu müssen.

„Ohne Werbung können Sie nicht viele Leute erreichen, und Sie müssen etwas für das Programm berechnen, um dies zu ermöglichen.“

„Es ist nutzlos, für etwas zu werben, was man umsonst bekommen kann.“

Es gibt viele Formen kostenloser oder kostengünstiger Werbung, die dazu dienen kann, viele Computerbenutzer über so etwas wie GNU zu informieren. Es mag stimmen, daß man mehr Benutzer von Microcomputern durch Werbung erreichen kann; wenn dies so ist, sollte ein Unternehmen, das etwas für den Service des Kopierens und Verteilens von GNU berechnet, erfolgreich genug sein, um sich Werbung und mehr leisten zu können. Auf diese Weise zahlen auch nur diejenigen Benutzer für die Werbung, die von ihr profitieren.

Wenn andererseits viele Leute GNU von ihren Freunden erhalten und solche Unternehmen keinen Erfolg haben, zeigt dies, daß Werbung in Wirklichkeit gar nicht nötig war, um GNU zu verbreiten. Warum wollen die Vertreter der freien Marktwirtschaft nicht den freien Markt darüber entscheiden lassen? (4)

„Mein Unternehmen benötigt ein proprietäres Betriebssystem, um einen Wettbewerbsvorteil zu bekommen.“

GNU wird Betriebssystem-Software aus dem Bereich des Wettbewerbs entfernen. Sie werden keinen Vorteil auf diesem Gebiet erzielen können, aber umgekehrt wird auch Ihre Konkurrenz Sie nicht übervorteilen können. Sie werden auf anderen Gebieten in Wettbewerb treten, während Sie auf diesem Gebiet voneinander profitieren werden. Wenn Ihr Unternehmen vom Verkauf eines Betriebssystems lebt, werden Sie GNU nicht mögen, aber das ist Ihr Problem. Wenn Ihr Unternehmen anders ist, kann GNU Sie davor bewahren, in das teure Geschäft gedrängt zu werden, Betriebssysteme zu verkaufen.

Ich würde es gerne sehen, wenn viele Hersteller und Benutzer die Entwicklung von GNU durch Spenden unterstützen würden, um die Kosten für jeden einzelnen zu senken. (5)

„Verdienen nicht die Programmierer eine Belohnung für ihre Kreativität?“

Wenn irgendetwas eine Belohnung verdient, dann sind es soziale Beiträge. Kreativität kann ein sozialer Beitrag sein, aber nur, wenn die Gesellschaft die Freiheit hat, die Resultate zu nutzen. Wenn Programmierer eine Belohnung für das Schreiben innovativer Programme verdienen, müßten sie aus demselben Grunde bestraft werden, wenn sie die Nutzung dieser Programme einschränken.

„Sollte ein Programmierer nicht eine Belohnung für seine Kreativität verlangen dürfen?“

Es ist nichts Falsches darin, Bezahlung für Arbeit zu verlangen oder sein Einkommen maximieren zu wollen, solange man nicht destruktiv wird. Die zur Zeit auf diesem Gebiet gebräuchlichen Mittel basieren auf einer Form von Zerstörung.

Geld von Benutzern zu kassieren, indem man den Gebrauch eines Programms einschränkt, ist destruktiv, weil die Einschränkungen die Häufigkeit und die verschiedenen Weisen begrenzen, in denen das Programm benutzt werden könnte. Dies begrenzt den Reichtum, der aus dem Programm für die Menschheit entsteht. Die schädlichen Auswirkungen einer bewußten Beschränkung sind eine bewußte Form von Zerstörung.

Der Grund, weshalb ein guter Bürger derart destruktive Mittel nicht anwendet, um reich zu werden, ist, daß, wenn dies jeder täte, wir alle durch die wechselseitige Zerstörung ärmer würden. Dies ist Kantsche Ethik – oder das Gebot der Nächstenliebe. Die Konsequenzen, die daraus entstünden, daß ein jeder Information horten würde, gefallen mir nicht, daher sehe ich mich gezwungen, es für falsch zu befinden, wenn sich einer so verhält. Insbesondere begründet der Wunsch, für Kreativität belohnt zu werden, es nicht, die gesamte Welt all dieser Kreativität oder von Teilen davon zu berauben.

„Werden die Programmierer nicht verhungern?“

Ich könnte antworten, daß niemand gezwungen ist, ein Programmierer zu sein. Die meisten von uns könnten nicht davon leben, auf der Straße zu stehen und Possen zu reißen, aber wir sind deswegen noch lange nicht dazu verdammt, auf der Straße zu stehen, Possen zu reißen und zu verhungern – wir machen etwas anderes.

Aber dies ist die falsche Antwort, weil sie die implizite Annahme des Fragestellers akzeptiert, daß nämlich Programmierer keinen Pfennig erhalten würden, wenn es keinen Besitz von Software gibt. Es wird angenommen, daß es um „alles oder nichts“ geht.

Der wirkliche Grund, weshalb Programmierer nicht verhungern werden, ist, daß es für sie immer noch möglich sein wird, Geld für Programmierung zu erhalten, nur eben nicht so viel, wie dies im Moment der Fall ist.

Eingeschränktes Kopieren ist nicht die einzige geschäftliche Basis in Sachen Software. Es ist die üblichste Basis, weil sie am meisten Geld einbringt. Wäre diese Basis verboten oder würde sie von den Kunden abgelehnt, würde sich das Software-Geschäft auf andere organisatorische Grundlagen begeben, die zur Zeit weniger häufig verwendet werden. Es gibt immer viele Wege, Geschäfte zu organisieren.

Sicherlich wird das Programmieren auf dieser neuen Basis nicht so lukrativ sein, wie es jetzt ist, aber dies ist kein Argument gegen den Wechsel. Man betrachtet es im allgemeinen nicht als ungerecht, daß Verkäufer die Gehälter bekommen, die sie bekommen. Würden Programmierer die gleichen Gehälter beziehen, wäre dies ebenfalls nicht ungerecht. (In der Praxis werden sie auch weiterhin deutlich mehr beziehen.)

„Haben Menschen nicht das Recht, zu kontrollieren, wie ihre Ideen verwendet werden?“

„Kontrolle über den Gebrauch von Ideen“ konstituiert in Wirklichkeit Kontrolle über das Leben anderer Menschen, und sie wird normalerweise eingesetzt, um den Menschen das Leben schwerer zu machen.

Leute, die das Thema geistigen Eigentums sorgfältig studiert haben (z.B. Anwälte) sagen, daß es kein intrinsisches Recht auf intellektuelles Eigentum gibt. Die von der Regierung anerkannten Arten angenommenen intellektuellen Eigentums wurden durch spezielle Gesetze für spezielle Zwecke geschaffen.

Beispielsweise wurde das Patentsystem etabliert, um Erfinder zu ermutigen, die Details ihrer Erfindungen zu offenbaren. Der Sinn war eher der Gesellschaft zu helfen als dem Erfinder. Zu jener Zeit war die Lebensdauer von 17 Jahren für ein Patent klein verglichen mit der Geschwindigkeit des Fortschritts. Weil Patente nur für Hersteller ein Thema sind, für welche die Kosten und Mühen einer Lizenz klein verglichen mit den Produktionskosten sind, schaden Patente meistens nicht viel. Die meisten Einzelpersonen, die patentierte Produkte benutzen, werden dadurch nicht behindert.

Die Idee des Urheberrechts existierte früher nicht, als Autoren häufig andere Autoren in nicht-fiktionalen Werken kopierten. Diese Praxis war nützlich, und nur auf diesem Wege sind die Werke vieler Autoren

wenigstens teilweise erhalten geblieben. Das System des Urheberrechts wurde ausdrücklich dafür entwickelt, Autorenschaft zu ermutigen. Auf dem Gebiet, für das es erfunden wurde – Bücher, die nur auf einer Druckerpresse auf ökonomische Weise vervielfältigt werden konnten – schadete es wenig, und die meisten Leser wurden dadurch nicht behindert.

Alle intellektuellen Eigentumsrechte wurden von der Gesellschaft lizensiert, weil man – richtig oder falsch – glaubte, daß die gesamte Gesellschaft von der Einführung dieser Rechte profitieren würde. In einer speziellen Situation jedoch müssen wir uns fragen, ob wir wirklich besser damit fahren, solche Rechte zu lizensieren. Welche Art von Handlungen erlauben wir den Personen?

Der Fall von Computerprogrammen heute ist sehr verschieden von dem von Büchern vor hundert Jahren. Die Tatsache, daß der einfachste Weg, ein Programm zu kopieren, von Nachbar zu Nachbar geht, die Tatsache, daß zu einem Programm sowohl Quelltext als auch Objectcode gehören, die verschieden sind, und die Tatsache, daß ein Programm benutzt und nicht gelesen und genossen wird, schaffen zusammen eine Situtation, in der eine Person, die ein Urheberrecht einfordert, der gesamten Gesellschaft sowohl materiellen als auch spirituellen Schaden zufügt – eine Situtation, in der eine Person kein Urheberrecht einfordern sollte, unabhängig davon, ob die Gesetze es ihr erlauben.

„Im Wettbewerb werden Dinge besser ausgeführt.“

Das Musterbeispiel eines Wettbewerbs ist ein Wettrennen: Indem wir den Gewinner belohnen, ermutigen wir jeden Teilnehmer, schneller zu laufen. Wenn der Kapitalismus tatsächlich auf diese Weise funktioniert, ist es gut, aber seine Befürworter haben Unrecht mit der Annahme, daß es immer so funktioniert. Wenn die Läufer vergessen, weshalb der Preis ausgesetzt wurde und unbedingt gewinnen wollen – egal wie –, entdecken sie vielleicht andere Strategien, zum Beispiel, andere Läufer anzugreifen. Wenn die Läufer in einen Faustkampf geraten, werden alle erst spät durchs Ziel gehen.

Proprietäre und geheime Software sind das moralische Äquivalent von Rennläufern in einem Faustkampf. Es ist traurig, daß sich der einzige vorhandene Schiedsrichter nicht um Faustkämpfe sorgt und sie lediglich reguliert („Pro zehn gelaufene Meter dürft ihr einen Schuß abfeuern“). Er sollte sie stattdessen auseinanderbringen und Läufer bereits für den Versuch eines Angriffs bestrafen.

„Wird ohne finanziellen Ansporn nicht jeder aufhören zu programmieren?“

Tatsächlich werden viele Menschen absolut ohne jeden finanziellen Ansporn programmieren. Das Programmieren übt eine unwiderstehliche Faszination auf manche Leute aus – normalerweise diejenigen Leute, die darin am besten sind. Es gibt keinen Mangel an Berufsmusikern, die bei ihrem Beruf bleiben, obwohl sie keinerlei Hoffnung haben, damit ihren Lebensunterhalt bestreiten zu können.

Aber in Wirklichkeit ist diese Frage, obwohl sie häufig gestellt wird, der Situation nicht angemessen. Die Bezahlung für Programmierer wird nicht verschwinden, sondern nur weniger werden. Die richtige Frage ist daher: Wird irgendjemand bei reduziertem finanziellen Ansporn programmieren? Meine Erfahrung zeigt, daß jemand es tun wird.

Vor über zehn Jahren haben einige der besten Programmierer der Welt im *Artificial Intelligence Lab* für weit weniger Geld gearbeitet, als sie anderswo hätten verdienen können. Sie erhielten nicht-finanzielle Belohnungen in vielerlei Art, zum Beispiel Berühmtheit und Dank. Außerdem ist die Kreativität selbst eine Freude, eine Belohnung für sich.

Dann geschah es, daß die meisten dieser Programmierer verschwanden, als sie eine Chance bekamen, dieselbe interessante Arbeit für viel Geld zu tun.

Die Tatsachen zeigen, daß Menschen aus anderen Gründen als Bereicherung programmieren werden, aber wenn man ihnen eine Möglichkeit gibt, außerdem noch viel Geld zu scheffeln, geschieht es, daß sie dieses Geld erwarten und verlangen. Niedriglohn-Organisationen machen sich schlecht neben Höchstlohn-Organisationen; dies entfällt, wenn die Höchstlohn-Organisationen verboten sind.

„Wir sind auf die Programmierer angewiesen. Wenn die verlangen, daß wir aufhören, unseren Nachbarn zu helfen, müssen wir nachgeben.“

Sie können niemals so verzweifelt sein, daß Sie derartigen Forderungen nachgeben müssen. Vergessen Sie

nicht: Millionen für die Verteidigung – keinen Pfennig für Tribut!

„Programmierer müssen von irgendetwas leben.“

Kurzfristig existiert dieses Problem. Es gibt aber viele Wege, wie Programmierer ihren Lebensunterhalt bestreiten können, ohne das Recht zu verkaufen, eine Programm zu benutzen. Der derzeitige Weg ist üblich, weil er Programmierern und Geschäftsleuten das meiste Geld einbringt und nicht, weil es der einzige Weg ist, seinen Lebensunterhalt zu verdienen. Es ist leicht, andere Wege zu finden, wenn man sie sucht; einige Beispiele folgen.

Ein Computerhersteller, der einen neuen Computer einführt, zahlt für die Portierung des Betriebssystems auf die neue Hardware.

Programmierer können in der Schulung, Benutzerhilfe und Wartung unterkommen.

Leute mit neuen Ideen können Programme als FreeWare verteilen und zufriedene Benutzer um Spenden bitten oder Benutzerhilfen anbieten. Ich bin einigen Leuten begegnet, die bereits erfolgreich in dieser Weise arbeiten.

Benutzer mit ähnlichen Bedürfnissen können Interessensgemeinschaften bilden und Beiträge zahlen. Die Gruppe würde dann Programmierfirmen damit beauftragen, Programme zu schreiben, die die Mitglieder gerne benutzen würden.

Alle Arten von Weiterentwicklung könnten durch eine Software-Steuer finanziert werden:

Stellen wir uns vor, jeder, der einen Computer kauft, muß x Prozent des Preises an Software-Steuer entrichten. Die Regierung gibt dieses Geld einer Agentur wie der NSF, um es für Software-Entwicklung einzusetzen.

Wenn allerdings der Computerkäufer selbst für Software-Entwicklung spendet, wird die Spende mit der Software-Steuer verrechnet. Er kann für das Projekt seiner Wahl spenden – häufig in der Hoffnung ausgewählt, das Ergebnis benutzen zu können. Er kann beliebig hohe Spenden anrechnen lassen bis hin zum Gesamtvolumen der zu zahlenden Steuer.

Der Steuersatz könnte durch die Steuerzahler selbst durch Wahl festgelegt werden, gewichtet gemäß der Menge der zu zahlenden Steuer.

Die Konsequenzen:

- *Die Gemeinschaft der Computerbenutzer unterstützt die Software-Entwicklung.*
- *Die Gemeinschaft entscheidet, wieviel Unterstützung benötigt wird.*
- *Benutzer, denen es darauf ankommt, an welchen Projekten sie sich beteiligen, können dies selbst entscheiden.*

Auf lange Sicht ist das Freigeben von Programmen ein Schritt in Richtung einer Welt ohne Mangel, in der niemand hart arbeiten muß, um sein Leben zu bestreiten. Die Menschen werden frei sein, sich Aktivitäten zu widmen, die Freude machen, zum Beispiel Programmieren, nachdem sie zehn Stunden pro Woche mit notwendigen Aufgaben wie Verwaltung, Familienberatung, Reparatur von Robotern und der Beobachtung von Asteroiden verbracht haben. Es wird keine Notwendigkeit geben, von Programmierung zu leben.

Wir haben bereits die Menge an Arbeit, welche die Gesellschaft für ihre Produktivität aufbringen muß, gewaltig reduzieren können, aber nur ein kleiner Teil davon übertrug sich in mehr Freizeit für Arbeiter, weil jede produktive Aktivität zwangsläufig von viel unproduktiver Aktivität begleitet wird. Die Hauptursache hierfür ist Bürokratie und gegenseitiges Bekämpfen, ohne dabei voran zu kommen, anstelle von Wettbewerb. Freie Software wird diese Auswüchse auf dem Gebiet der Software-Entwicklung in großartiger Weise reduzieren. Wir müssen so handeln, um technische Fortschritte in Sachen Produktivität zu erzielen, die sich in weniger Arbeit für uns alle äußern werden.

Fußnoten:

(1)

Diese Wortwahl war ein wenig sorglos. Die Absicht war, daß niemand für die *Erlaubnis* zahlen muß, das GNU-System zu benutzen. Diees geht jedoch nicht aus der Formulierung hervor, und man interpretiert dies häufig so, daß Kopien von GNU stets gegen kein oder höchstens geringes Entgelt verteilt werden sollen. Dies war niemals die Absicht; weiter unten erwähnt das Manifest die Möglichkeit für Firmen, GNU gegen Bezahlung zu verteilen. Später habe ich es gelernt, sorgfältig zwischen „frei“ im Sinne von „Freiheit“ und „frei“ im Sinne von „Preis“ zu unterscheiden. Freie Software ist Software, deren Benutzer die Freiheit haben, sie weiterzugeben und zu verändern. Manche werden ihre Kopien gratis erhalten, während andere dafür bezahlen werden – und wenn diese Geldsumme dazu beiträgt, die Software weiter zu verbessern, um so besser. Wichtig ist, daß jeder, der eine Kopie hat, auch die Freiheit hat, beim Gebrauch dieser Kopie mit anderen zu kooperieren.

(2)

Dies ist eine weitere Stelle, an der ich versämut habe, sorgfältig zwischen den beiden verschiedenen Bedeutungen von „frei“ zu unterscheiden. So, wie der Satz geschrieben ist, ist er nicht falsch – Sie können Kopien von GNU-Software kostenlos von Ihren Freunden oder über das Netz erhalten –, aber er suggeriert die falsche Idee.

(3)

Inzwischen gibt es mehrere solche Firmen.

(4)

Die *Free Software Foundation* bezieht den größten Teil ihres Kapitals von einem solchen Verteilungsservice, obwohl sie eher eine caritative Einrichtung als eine Firma ist. Wenn sich *niemand* dafür entscheidet, seine Kopien von der FSF zu beziehen, wird diese ihre Arbeit nicht machen können. Dies bedeutet nicht, daß proprietäre Einschränkungen gemacht werden sollten, um jeden Benutzer zum Bezahlen zu zwingen; wenn ein kleiner Bruchteil aller Benutzer seine Kopien von der FSF bezieht, genügt dies, die FSF flüssig zu halten. Daher bitten wir Benutzer in dieser Weise um ihre Unterstützung. Haben Sie Ihren Teil bereits geleistet?

(5)

Eine Gruppe von Computerunternehmen hat vor kurzem Kapital angesammelt, um die Wartung des GNU-C-Compilers zu unterstützen.

* * *

Weshalb „freie Software“ besser ist als „Open Source“

Von Richard M. Stallman

Deutsche Übersetzung: Peter Gerwinski, Dezember 2006

Den offiziellen englischen Originaltext finden Sie unter http://www.gnu.org/philosophy/free-software-for-freedom.html. Dieser Artikel wurde außerdem in dem Buch Free Software, Free Society: The Selected Essays of Richard M. Stallman veröffentlicht.

Freie Software unter irgendeinem anderen Namen würde Ihnen dieselbe Freiheit geben. Trotzdem ist es ein großer Unterschied, welchen Namen Sie verwenden: Verschiedene Worte *transportieren verschiedene Konzepte*.

1998 begannen einige Leute in der Gemeinschaft der freien Software, den Begriff „Open-Source-Software“ anstelle von „freie Software“ zu verwenden, um ihre Tätigkeiten zu beschreiben. Schon bald wurde der Begriff „Open Source“ mit einem anderen Ansatz, einer anderen Philosophie und anderen Werten assoziiert und sogar mit unterschiedlichen Kriterien dafür, welche Software-Lizenzen akzeptabel seien. Heutzutage handelt es sich bei der Freie-Software- und der Open-Source-Bewegung um separate Bewegungen mit unterschiedlichen Ansichten und Zielen. Nichtsdestoweniger können wir in der Praxis an einigen Projekten zusammenarbeiten – und tun dies auch.

Der grundsätzliche Unterschied zwischen beiden Bewegungen liegt in ihren Wertesystemen, ihrer unterschiedlichen Art, die Welt zu betrachten. Für die Open-Source-Bewegung ist die Frage, ob Software Open-Source-Software sein soll, eine Frage der Zweckmäßigkeit und nicht der Ethik. Jemand brachte das einmal folgendermaßen auf den Punkt: „Open-Source ist eine Methode zur Software-Entwicklung; freie Software ist eine soziale Bewegung.“ Für die Open-Source-Bewegung ist nicht-freie Software eine suboptimale Lösung. Für die Freie-Software-Bewegung ist nicht-freie Software ein soziales Problem, und freie Software ist die Lösung.

Das Verhältnis zwischen der Freie-Software- und der Open-Source-Bewegung

Die Freie-Software-Bewegung und die Open-Source-Bewegung sind wie zwei politische Parteien innerhalb der Gemeinschaft der freien Software.

Die radikalen Gruppen in den 1960er Jahren waren für ihren Parteigeist berüchtigt: Organisationen fielen wegen Uneinigkeit über Details in der Vorgehensweise auseinander und betrachteten sich von da an als Gegner. Dies ist zumindest das Bild, das man von ihnen hat, ob nun wahr oder falsch.

Das Verhältnis zwischen der Freie-Software- und der Open-Source-Bewegung ist gerade das Gegenteil dieses Bildes. In den Grundprinzipien sind wir uns uneinig, aber hinsichtlich der praktischen Empfehlungen stimmen wir weitgehend überein. Daher können wir an vielen spezifischen Projekten gemeinsam arbeiten und tun dies auch tatsächlich. Wir sehen die Open-Source-Bewegung nicht als Gegner. Der Gegner ist die proprietäre Software.

Wir sind nicht gegen die Open-Source-Bewegung, aber wir wollen nicht mit ihr in einen Topf geworfen werden. Wir erkennen ihre Leistungen für unsere Gemeinschaft an, aber wir waren es, die diese Gemeinschaft aufgebaut haben, und wir möchten, daß die Leute das wissen. Wir wollen, daß man das, was wir erreicht haben, mit unseren Werten und unserer Philosophie in Verbindung bringt, nicht mit den ihrigen. Wir möchten gehört werden und nicht hinter einer Gruppe mit ganz anderen Ansichten versteckt werden. Um zu vermeiden, daß man uns für einen Teil der Open-Source-Bewegung hält, vermeiden wir mühsam das Wort “open” (offen), wenn wir von freier Software sprechen, und das Wort “closed” (geschlossen), wenn wir von nicht-freier Software sprechen.

Bitte erwähnen Sie daher die Freie-Software-Bewegung, wenn Sie über unsere Arbeit sprechen und über die Software, die wir entwickelt haben – wie zum Beispiel das Betriebssystem GNU/Linux.

Vergleich beider Begriffe

Der Rest dieses Artikels vergleicht die beiden Begriffe „freie Software“ und „Open Source“ miteinander und zeigt, weshalb der Begriff „Open Source“ keinerlei Probleme löst, sondern im Gegenteil Probleme aufwirft.

Mehrdeutigkeit

Der Begriff „freie Software“ hat ein Mehrdeutigkeitsproblem: Die unbeabsichtigte Bedeutung „Software, die man kostenlos bekommen kann“ paßt auf den Begriff genausogut wie die beabsichtigte Bedeutung, „Software, die einem gewisse Freiheiten gibt“. Wir gehen dieses Problem dadurch an, daß wir eine präzisere Definition von freier Software bereitstellen, aber diese Lösung ist nicht perfekt und kann das Problem nicht gänzlich beseitigen. Ein eindeutig korrekter Begriff wäre besser, wenn er keine anderen Probleme aufwerfen würde.

Leider sind alle Alternativen im Englischen auf ihre eigene Weise problematisch. Wir haben uns viele Alternativen angesehen, die uns vorgeschlagen wurden, aber keine ist derart eindeutig „richtig“ daß es eine gute Idee wäre, zu ihr zu wechseln. Jeder vorgeschlagene Ersatz für „freie Software“ wirft ähnlich geartete semantische Probleme auf – oder schlimmere. Dies gilt auch für „Open-Source-Software“.

Die offizielle Definition von „Open-Source-Software“ der Open-Source-Initiative ist unserer Definition von freier Software sehr ähnlich. Nichtsdestoweniger ist sie in mancher Hinsicht eine Spur lockerer, und es wurden ein paar Software-Lizenzen akzeptiert, die wir als für den Benutzer unakzeptabel restriktiv ansehen.

Wie dem auch sei, lautet die offensichtliche Bedeutung des Begriffs „Open-Source“ (quelloffen): „Sie haben Einblick in den Quelltext“. Dies ist ein wesentlich schwächeres Kriterium als freie Software, das zwar freie Software mit einschließt, aber auch semi-freie Computerprogramme wie z.B. Xv und sogar proprietäre Software, darunter Qt unter seiner ursprünglichen Lizenz (vor der QPL).

Diese offensichtliche Bedeutung von „Open Source“ ist nicht die von ihren Befürwortern beabsichtigte. Das Ergebnis ist, daß die meisten Leute mißverstehen, was hier befürwortet wird. Der Autor Neal Stephenson beispielsweise definiert „Open Source“ folgendermaßen:

Linux ist „Open-Source-Software“, was schlicht und einfach bedeutet, daß jedermann seinen Quelltext erhalten kann.

Ich glaube nicht, daß es in seiner Absicht lag, die „offizielle“ Definition zu diskutieren oder in Frage zu stellen. Ich glaube vielmehr, daß er einfach die Regeln der englischen Sprache angewendet hat, um die Bedeutung des Begriffs zu erschließen. Der US-Bundesstaat Kansas publizierte eine ähnliche Definition:

Verwenden Sie Open-Source-Software (OSS). OSS ist Software, deren Quelltext frei und öffentlich zugänglich ist, wenn auch die spezifischen Software-Lizenzen variieren, die besagen, was genau man mit dem Quelltext tun darf.

Natürlich haben die Open-Source-Leute versucht, diesem Problem durch eine präzise Definition des Begriffs zu begegnen, genauso wie wir es für den Begriff „freie Software“ getan haben.

Aber die Erklärungen für „freie Software“ sind einfach: Jemand, der das Konzept „freie Rede, nicht Freibier“ begriffen hat, wird es nicht wieder falsch verstehen. Es gibt keinen derart prägnanten Weg, die offizielle Bedeutung von „Open Source“ zu erklären und klar aufzuzeigen, weshalb die [sprachlich-]natürliche Definition falsch ist.

Angst vor der Freiheit

Das Hauptargument für den Begriff „Open-Source-Software“ ist, daß „freie Software“ manche Leute verängstigt. Es stimmt: Wer über die Freiheit, über ethische Fragen, über Luxus und Verantwortung spricht, fordert die Menschen auf, über Dinge nachzudenken, die sie lieber ignorieren würden. Dies kann Unbehagen auslösen, und manche Leute werden das Konzept deswegen ablehnen. Daraus folgt aber nicht, daß es der Gesellschaft besser erginge, wenn wir über diese Themen nicht mehr reden würden.

Vor Jahren haben die Entwickler freier Software diese Reaktion des Unbehagens erkannt, und manche haben einen Ansatz erkundet, dies zu vermeiden. Sie fanden heraus, daß sie, wenn sie zu Ethik und Freiheit schwiegen und nur über die unmittelbaren praktischen Vorteile bestimmter freier Software sprachen, die Software an gewisse Anwender besser „verkaufen“ konnten, insbesondere an Geschäftskunden. Der Begriff „Open Source“ wird als ein Weg angeboten, der weiter in diese Richtung führt – ein Weg, für Geschäftskunden „akzeptabler“ zu sein. Die Anschauungen und Wertvorstellungen der Open-Source-Bewegung gehen auf diese Entscheidung zurück.

Dieser Ansatz hat sich bewährt – gemäß seinen eigenen Begriffen. Viele Menschen wechseln heutzutage aus rein praktischen Erwägungen heraus zu freier Software. Dies ist, für sich genommen, eine gute Sache, aber damit ist nicht alles getan! Das Gewinnen von Anwendern freier Software ist nicht die vollständige Aufgabe, sondern nur der erste Schritt.

Früher oder später wird man diese Anwender dazu einladen, wegen gewisser praktischer Vorteile zu propritärer Software zurückzuwechseln. Zahllose Unternehmen streben danach, solche Versuchungen anzubieten. Warum sollten die Anwender sie zurückweisen? Nur deswegen, weil sie gelernt haben, *die Freiheit wertzuschätzen*, die ihnen freie Software gibt, um der Freiheit selbst willen. Es liegt an uns, dieses Konzept zu verbreiten, – und um das zu tun, müssen wir über Freiheit reden. Ein gewisses Maß des „Schweigen ist Gold“-Ansatzes kann für die Gemeinschaft nützlich sein, aber wir brauchen darüberhinaus reichlich Reden über Freiheit.

Zur Zeit haben wir reichlich Schweigen, aber nicht genug Reden über Freiheit. Die meisten Leute, die mit freier Software zu tun haben, sprechen wenig über Freiheit – typischerweise, weil sie „akzeptabler für Geschäftskunden“ sein wollen. Insbesondere Software-Distributoren zeigen solche Verhaltensweisen. Manche Distributionen des GNU/Linux-Betriebssystems fügen dem freien Basis-System proprietäre Software-Pakete hinzu und fordern die Anwender auf, darin einen Vorteil zu sehen – anstelle eines Rückschritts weg von der Freiheit.

Es gelingt uns nicht, mit dem Zustrom von Anwendern freier Software mitzuhalten. Sie kommen so schnell, daß wir es nicht schaffen, sie über die Freiheit und unsere Gemeinschaft zu unterrichten. Deshalb findet nicht-freie Software (was Qt war, als es populär wurde) und teilweise unfreie Betriebssystem-Distributionen derart fruchtbaren Boden. Jetzt damit aufzuhören, das Wort „frei“ zu verwenden, wäre ein Fehler. Wir brauchen mehr, nicht weniger Reden über Freiheit.

Wenn diejenigen, die den Begriff „Open Source“ benutzen, mehr Anwender in unsere Gemeinschaft holen, ist dies ein Beitrag, aber der Rest von uns muß dann um so härter daran arbeiten, diese Anwendern auf das Konzept der Freiheit aufmerksam zu machen. Wir müssen den Satz „Es ist freie Software, und sie gibt dir Freiheit!“ häufiger und lauter sagen ale je zuvor.

Würde eine Marke helfen?

Die Befürworter von „Open-Source-Software“ versuchten, daraus eine Marke zu machen, und sagten, dies würde sie in die Lage versetzen, Mißbrauch zu verhindern. Diese Initiative wurde später fallengelassen; der Begriff sei zu sehr beschreibend, um sich als Marke zu eignen. Daher ist der rechtliche Status von „Open Source“ derselbe wie der von „freie Software“: Es gibt keine *rechlichen* Beschränkungen für seinen Gebrauch. Ich habe Berichte von etlichen Fällen gehört, in denen Unternehmen Software-Pakete als „Open Source“ bezeichnet haben, obwohl sie nicht der offiziellen Definition genügten. Einige solche Fälle habe ich selbst beobachtet.

Aber würde es einen großen Unterschied ausmachen, einen Begriff zu verwenden, der eine Marke ist? Nicht notwendigerweise.

Unternehmen haben Ankündigungen herausgegeben, die den Eindruck erweckten, ein Computerprogramm sei „Open-Source-Software“, ohne dies ausdrücklich zu behaupten. Zum Beispiel lautete eine Ankündigung von IBM zu einem Programm, das nicht der offiziellen Definition genügte, folgendermaßen:

Wie in der Open-Source-Gemeinschaft üblich, werden Anwender der …-Technologie auch mit IBM kooperieren können …

Hier wird nicht tatsächlich behauptet, die Software *sei* „Open Source“, aber dieses Detail nahmen viele Leser nicht zur Kenntnis. (Ich sollte erwähnen, daß IBM aufrichtig versucht hat, dieses Programm zu freier

Software zu machen, und daß IBM später eine neue Lizenz eingeführt hat, die daraus freie Software und „Open Source“ macht, aber als die Ankündigung herauskam, qualifizierte sich das Programm als keins von beidem.)

Und hier sehen wir, wie die Firma Cygnus Solutions, die als Firma für freie Software gegründet worden war und später zu proprietärer Software (um es so auszudrücken) verzweigte, ein proprietäres Software-Produkt bewarb:

Cygnus Solutions ist auf dem Open-Source-Markt ein führendes Unternehmen und hat soeben zwei neue Produkte in den [GNU/]Linux-Markt eingeführt.

Im Gegensatz zu IBM hat Cygnus nicht versucht, diese Software-Pakete zu freier Software zu machen; sie waren weit davon entfernt, sich zu qualifizieren. Aber Cygnus hat nicht wirklich behauptet, es handele sich um „Open-Source-Software“; man hat nur den Begriff benutzt, um bei unachtsamen Lesern diesen Eindruck zu erwecken.

Diese Beobachtungen legen nahe, daß eine Marke die Verwirrung, die mit dem Begriff „Open Source“ einhergeht, nicht wirklich verhindert hätte.

Mißverständnisse(?) von „Open Source“

Die Open-Source-Definition ist eindeutig genug, und es ist klar, daß typische nicht-freie Programme ihr nicht genügen. Sie vermuten jetzt wahrscheinlich, daß „Open-Source-Unternehmen“ ein Unternehmen bezeichnet, dessen Produkte freie Software (oder nahe daran) sind, nicht wahr? Leider versuchen viele Unternehmen, dem Begriff eine andere Bedeutung zu geben.

Auf der Konferenz „Open Source Developers Day“ im August 1998 sagten viele der eingeladenen kommerziellen Entwickler, daß sie nur einen Teil ihrer Arbeit zu freier Software (oder „Open Source“) machen würden. Ihr Kerngeschäft sei es, proprietäre Zusätze (Software oder Handbücher) zu entwickeln, um diese dann den Benutzern freier Software zu verkaufen. Sie forderten uns auf, dies als legitim zu betrachten – als Teil unserer Gemeinschaft –, weil ein Teil des Geldes für die Entwicklung freier Software gespendet wird.

In Wirklichkeit bemühen sich diese Unternehmen darum, das Gütesiegel „Open Source“ für ihre proprietären Software-Produkte zu erlangen – obwohl diese gar keine „Open-Source-Software“ sind –, weil diese irgendeinen Bezug zu freier Software haben oder weil dasselbe Unternehmen auch eine freie Software pflegt. (Ein Unternehmensgründer sagte ganz explizit, daß sie in das freie Software-Paket, das sie unterstützen, so wenig Arbeit hineinsteckten, wie es die Gemeinschaft noch tolerieren würde.)

Über die Jahre haben viele Unternehmen zur Entwicklung freier Software beigetragen. Manche dieser Unternehmen entwickelten in erster Linie nicht-freie Software, aber diese beiden Aktivitäten waren getrennt; dadurch konnten wir die nicht-freien Produkte des Unternehmens ignorieren und mit ihm gemeinsam an Freie-Software-Projekten arbeiten. Wir konnten uns dann ehrlich bei dem Unternehmen für seinen Beitrag zur freien Software bedanken, ohne darüber zu reden, was das Unternehmen ansonsten tat.

Dasselbe können wir bei diesen neuen Unternehmen nicht machen, weil diese uns nicht lassen. Sie laden die Öffentlichkeit aktiv ein, alle ihre Aktivitäten in einen Topf zu werfen. Sie wollen, daß wir ihre nicht-freie Software genauso positiv betrachten wie einen echten Beitrag zu freier Software, obwohl sie keiner ist. Sie präsentieren sich selbst als „Open-Source-Unternehmen“ und hoffen, daß wir kuschelig-warme Gedanken für sie entwickeln und diese mit einem kuschelig-laschen Verstand anwenden werden.

Diese manipulative Praxis wäre nicht weniger schädlich, wenn man dafür den Begriff „freie Software“ verwenden würde. Allerdings verwenden Unternehmen den Begriff „freie Software“ nicht in dieser Weise; vielleicht macht seine Verbindung mit Idealismus den Begriff ungeeignet. Der Begriff „Open Source“ öffnete dieser Praxis die Tür.

Ende 1998, auf einer Messe zu dem Betriebssystem, das häufig „Linux“ genannt wird, war ein Direktor eines bekannten Software-Unternehmens als Sprecher geladen – vermutlich wegen der Entscheidung seines Unternehmens, dieses Betriebssystem zu „unterstützen“. Leider besteht dessen Art der „Unterstützung“ darin, nicht-freie Software herauszubringen, die auf dem Betriebssystem läuft – mit anderen Worten: darin, unsere Gemeinschaft als Markt zu erschließen, aber nicht zu ihr beizutragen.

Er sagte: „Wir werden keineswegs unser Produkt zu Open Source machen, aber vielleicht zu ‚internem' Open Source. Wenn wir unseren Support-Mitarbeitern Zugriff auf den Quelltext geben, können diese für die Kunden Fehler beheben, und wir können ein besseres Produkt und besseren Service anbieten." (Dies ist kein exaktes Zitat, weil ich seine Worte nicht aufgeschrieben habe, aber es trifft das Wesentliche.)

Zuhörer sagten mir danach: „Er trifft einfach nicht den Punkt." Aber ist das wirklich so? Welchen Punkt hat er nicht getroffen?

Den Zielpunkt der Open-Source-Bewegung hat er nicht verfehlt. Diese Bewegung sagt nicht, daß die Anwender Freiheit haben sollten, sondern nur, daß es eine schnellere und bessere Produktentwicklung fördert, wenn man mehr Leuten erlaubt, den Quelltext einzusehen und zu verbessern. Diesen Punkt hat der Direktor voll und ganz begriffen. Er wollte den Open-Source-Ansatz nicht zur Gänze – einschließlich Anwender – umsetzen und überlegte, ihn teilweise – innerhalb der Firma – zu implementieren.

Der Punkt, den er verfehlt hat, ist der Punkt, den „Open Source" von der Anlage her gar nicht aufwerfen soll: den Punkt, daß Benutzer Freiheit *verdienen*.

Den Gedanken der Freiheit zu verbreiten, ist eine große Aufgabe, die Ihre Mitarbeit benötigt. Deshalb halten wir im GNU-Projekt an dem Begriff „freie Software" fest, damit wir bei dieser Aufgabe helfen können. Wenn Sie das Gefühl haben, daß Freiheit und die Gemeinschaft für sich selbst genommen wichtig sind – nicht nur wegen der Bequemlichkeiten, die sie uns bringen –, dann schließen Sie sich bitte uns an und verwenden Sie den Begriff „freie Software".

* * *

Joe Barr hat einen Artikel Live and let license geschrieben, der seine Ansichten zu diesem Thema wiedergibt.

Ein Fachartikel über die Motivation der Entwickler freier Software von Lakhani und Wolf besagt, daß ein nennenswerter Anteil durch den Gedanken motiviert wird, daß Software frei sein sollte. Zu diesem Ergebnis kamen sie trotz der Tatsache, daß sie Software-Entwickler bei SourceForge befragt haben – einem Betrieb, der die Anschaung nicht teilt, daß es sich hierbei um eine ethische Frage handelt.

* * *

GNU PUBLIC LICENSE

Version 3
Version 2

GNU PUBLIC LISENZ Version 3

Deutsch
Englisch

.

Deutsche Übersetzung der Version 3, 29. Juni 2007
Peter Gerwinski, 5.7.2007

Den offiziellen englischen Originaltext finden Sie unter http://www.gnu.org/licenses/gpl.html.

Dies ist eine inoffizielle deutsche Übersetzung der GNU General Public License, die nicht von der Free Software Foundation herausgegeben wurde. Es handelt sich hierbei *nicht* um eine rechtsgültige Festlegung der Bedingungen für die Weitergabe von Software, die der GNU GPL unterliegt; dies leistet nur der englische Originaltext. Wir hoffen jedoch, daß diese Übersetzung deutschsprachigen Lesern helfen wird, die GNU GPL besser zu verstehen.

This is an unofficial translation of the GNU General Public License into German. It was not published by the Free Software Foundation, and does not legally state the distribution terms for software that uses the GNU GPL—only the original English text of the GNU GPL does that. However, we hope that this translation will help German speakers understand the GNU GPL better.

GNU General Public License

Diese Übersetzung ist kein rechtskräftiger Ersatz für die englischsprachige Originalversion!

Vorwort

Die GNU General Public License – die Allgemeine Öffentliche GNU-Lizenz – ist eine freie Copyleft-Lizenz für Software und andere Arten von Werken.

Die meisten Lizenzen für Software und andere nutzbaren Werke sind daraufhin entworfen worden, Ihnen die Freiheit zu nehmen, die Werke mit anderen zu teilen und zu verändern. Im Gegensatz dazu soll Ihnen die *GNU General Public License* die Freiheit garantieren, alle Versionen eines Programms zu teilen und zu verändern. Sie soll sicherstellen, daß die Software für alle ihre Benutzer frei bleibt. Wir, die Free Software Foundation, nutzen die GNU General Public License für den größten Teil unserer Software; sie gilt außerdem für jedes andere Werk, dessen Autoren es auf diese Weise freigegeben haben. Auch Sie können diese Lizenz auf Ihre Programme anwenden.

Wenn wir von freier Software sprechen, so beziehen wir uns auf Freiheit, nicht auf den Preis. Unsere Allgemeinen Öffentlichen Lizenzen sind darauf angelegt, sicherzustellen, daß Sie die Freiheit haben, Kopien freier Software zu verbreiten (und dafür etwas zu berechnen, wenn Sie möchten), die Möglichkeit, daß Sie die Software als Quelltext erhalten oder den Quelltext auf Wunsch bekommen, daß Sie die Software ändern oder Teile davon in neuen freien Programmen verwenden dürfen und daß Sie wissen, daß Sie dies alles tun dürfen.

Um Ihre Rechte zu schützen, müssen wir andere daran hindern, Ihnen diese Rechte zu verweigern oder Sie aufzufordern, auf diese Rechte zu verzichten. Aus diesem Grunde tragen Sie eine Verantwortung, wenn Sie Kopien der Software verbreiten oder die Software verändern: die Verantwortung, die Freiheit anderer zu respektieren.

Wenn Sie beispielsweise die Kopien eines solchen Programms verbreiten – kostenlos oder gegen Bezahlung – müssen Sie an die Empfänger dieselben Freiheiten weitergeben, die Sie selbst erhalten haben. Sie müssen sicherstellen, daß auch die Empfänger die Software im Quelltext erhalten bzw. den

Quelltext erhalten können. Und Sie müssen ihnen diese Bedingungen zeigen, damit sie ihre Rechte kennen.

Software-Entwickler, die die GNU GPL nutzen, schützen Ihre Rechte in zwei Schritten: (1) Sie machen ihr Urheberrecht (Copyright) auf die Software geltend, und (2) sie bieten Ihnen diese Lizenz an, die Ihnen das Recht gibt, die Software zu vervielfältigen, zu verbreiten und/oder zu verändern.

Um die Entwickler und Autoren zu schützen, stellt die GPL darüberhinaus klar, daß für diese freie Software keinerlei Garantie besteht. Um sowohl der Anwender als auch der Autoren Willen erfordert die GPL, daß modifizierte Versionen der Software als solche gekennzeichnet werden, damit Probleme mit der modifizierten Software nicht fälschlicherweise mit den Autoren der Originalversion in Verbindung gebracht werden.

Manche Geräte sind daraufhin entworfen worden, ihren Anwendern zu verweigern, modifizierte Versionen der darauf laufenden Software zu installieren oder laufen zu lassen, wohingegen der Hersteller diese Möglichkeit hat. Dies ist grundsätzlich unvereinbar mit dem Ziel, die Freiheit der Anwender zu schützen, die Software zu modifizieren. Derartige gezielte mißbräuchliche Verhaltensmuster finden auf dem Gebiet persönlicher Gebrauchsgegenstände statt – also genau dort, wo sie am wenigsten akzeptabel sind. Aus diesem Grunde wurde diese Version der GPL daraufhin entworfen, diese Praxis für diese Produkte zu verbieten. Sollten derartige Probleme substantiell auf anderen Gebieten auftauchen, sind wir bereit, diese Regelung auf diese Gebiete auszudehnen, soweit dies notwendig ist, um die Freiheit der Benutzer zu schützen.

Schließlich und endlich ist jedes Computerprogramm permanent durch Software-Patente bedroht. Staaten sollten es nicht zulassen, daß Patente die Entwicklung und Anwendung von Software für allgemein einsetzbare Computer einschränken, aber in Staaten, wo dies geschieht, wollen wir die spezielle Gefahr vermeiden, daß Patente dazu verwendet werden, ein freies Programm im Endeffekt proprietär zu machen. Um dies zu verhindern, stellt die GPL sicher, daß Patente nicht verwendet werden können, um das Programm nicht-frei zu machen.

Es folgen die präzisen Bedingungen für das Kopieren, Verbreiten und Modifizieren.

LIZENZBEDINGUNGEN

0. Definitionen

„Diese Lizenz“ bezieht sich auf die Version 3 der GNU General Public License.

Mit „Urheberrecht“ sind auch urheberrechtähnliche Rechte gemeint, die auf andere Arten von Werken Anwendung finden, beispielsweise auf Fotomasken in der Halbleitertechnologie.

„Das Programm“ bezeichnet jedes urheberrechtlich schützbare Werk, das unter diese Lizenz gestellt wurde. Jeder Lizenznehmer wird als „Sie“ angeredet. „Lizenznehmer“ und „Empfänger“ können natürliche oder rechtliche Personen sein.

Ein Werk zu „modifizieren“ bedeutet, aus einem Werk zu kopieren oder es ganz oder teilweise auf eine Weise umzuarbeiten, die eine urheberrechtliche Erlaubnis erfordert und kein Eins-zu-eins-Kopieren darstellt. Das daraus hervorgehende Werk wird als „modifizierte Version“ des früheren Werks oder als auf dem früheren Werk „basierendes“ Werk bezeichnet.

Ein „betroffenes Werk“ bezeichnet entweder das unmodifizierte Programm oder ein auf dem Programm basierendes Werk.
Ein Werk zu „propagieren“ bezeichnet jedwede Handlung mit dem Werk, für die man, wenn unerlaubt begangen, wegen Verletzung anwendbaren Urheberrechts direkt oder indirekt zur Verantwortung gezogen würde, ausgenommen das Ausführen auf einem Computer oder das Modifizieren einer privaten Kopie. Unter das Propagieren eines Werks fallen Kopieren, Weitergeben (mit oder ohne Modifikationen), öffentliches Zugänglichmachen und in manchen Staaten noch weitere Tätigkeiten.

Ein Werk zu „übertragen“ bezeichnet jede Art von Propagation, die es Dritten ermöglicht, das Werk zu kopieren oder Kopien zu erhalten. Reine Interaktion mit einem Benutzer über ein Computer-Netzwerk ohne Übergabe einer Kopie ist keine Übertragung.

Eine interaktive Benutzerschnittstelle zeigt „angemessene rechtliche Hinweise“ in dem Umfang, daß sie eine zweckdienliches und deutlich sichtbare Funktion bereitstellt, die (1) einen angemessenen Copyright-Vermerk zeigt und (2) dem Benutzer mitteilt, daß keine Garantie für das Werk besteht (ausgenommen in dem Umfang, in dem Garantie gewährt wird), daß Lizenznehmer das Werk gemäß dieser Lizenz übertragen dürfen und wie man ein Exemplar dieser Lizenz zu Gesicht bekommen kann. Wenn die Benutzerschnittstelle eine Liste von Benutzerkommandos oder Optionen anzeigt, zum Beispiel ein Menü, dann erfüllt ein deutlich sichtbarer Punkt in dieser Liste dieses Kriterium.

1. Quelltext

Der „Quelltext“ eines Werkes bezeichnet diejenige Form des Werkes, die für Bearbeitungen vorzugsweise verwendet wird. „Objekt-Code“ bezeichnet jede Nicht-Quelltext-Form eines Werks.

Eine „Standardschnittstelle“ bezeichnet eine Schnittstelle, die entweder ein offizieller Standard eines anerkannten Standardisierungsgremiums ist oder – im Falle von Schnittstellen, die für eine spezielle Programmiersprache spezifiziert wurden – eine Schnittstelle, die unter Entwicklern, die in dieser Programmiersprache arbeiten, weithin gebräuchlich ist.

Die „Systembibliotheken“ eines ausführbaren Werks enthalten alles, ausgenommen das Werk als Ganzes, was (a) normalerweise zum Lieferumfang einer Hauptkomponente gehört, aber selbst nicht die Hauptkomponente ist, und (b) ausschließlich dazu dient, das Werk zusammen mit der Hauptkomponente benutzen zu können oder eine Standardschnittstelle zu implementieren, für die eine Implementation als Quelltext öffentlich erhältlich ist. Eine „Hauptkomponente“ bezeichnet in diesem Zusammenhang eine größere wesentliche Komponente (Betriebssystemkern, Fenstersystem usw.) des spezifischen Betriebssystems (soweit vorhanden), auf dem das ausführbare Werk läuft, oder des Compilers, der zur Erzeugung des Objekt-Codes eingesetzt wurde, oder des für die Ausführung verwendeten Objekt-Code-Interpreters.

Der „korrespondierende Quelltext“ eines Werks in Form von Objekt-Code bezeichnet den vollständigen Quelltext, der benötigt wird, um das Werk zu erzeugen, es zu installieren, um (im Falle eines ausführbaren Werks) den Objekt-Code auszuführen und um das Werk zu modifizieren, einschließlich der Skripte zur Steuerung dieser Aktivitäten. Er schließt jedoch nicht die Systembibliotheken, allgemein einsetzbare Werkzeuge oder allgemein erhältliche freie Computerprogramme mit ein, die in unmodifizierter Form verwendet werden, um die o.a. Tätigkeiten durchzuführen, die aber nicht Teil des Werks sind. Zum Beispiel enthält der korrespondierende Quelltext die zum Programmquelltext gehörenden Schnittstellendefinitionsdateien sowie die Quelltexte von dynamisch eingebundenen Bibliotheken und Unterprogrammen, auf die das Werk konstruktionsbedingt angewiesen ist, beispielsweise durch komplexe Datenkommunikation oder Ablaufsteuerung zwischen diesen Unterprogrammen und anderen Teilen des Werks.

Der korrespondierende Quelltext braucht nichts zu enthalten, das der Anwender aus anderen Teilen des korrespondierenden Quelltextes automatisch regenerieren kann.

Der korrespondierende Quelltext eines Werks in Quelltextform ist das Werk selbst.

2. Grundlegende Genehmigungen

Alle unter dieser Lizenz gewährten Rechte werden gewährt auf Grundlage des Urheberrechts an dem Programm, und sie sind unwiderruflich, solange die festgelegten Bedingungen erfüllt sind. Diese Lizenz erklärt ausdrücklich Ihr uneingeschränktes Recht zur Ausführung des unmodifizierten Programms. Die beim Ausführen eines betroffenen Werks erzeugten Ausgabedaten fallen unter diese Lizenz nur dann, wenn sie, in Anbetracht ihres Inhalts, ein betroffenes Werk darstellen. Diese Lizenz erkennt Ihr im Urheberrecht vorgesehenes Recht auf angemessene Benutzung – oder seine Entsprechung – an.

Sie dürfen betroffene Werke, die Sie nicht übertragen, uneingeschränkt erzeugen, ausführen und propagieren, solange Ihre Lizenz ansonsten in Kraft bleibt. Sie dürfen betroffene Werke an Dritte übertragen für den einzigen Zweck, Modifikationen exklusiv für Sie durchzuführen oder Einrichtungen für Sie bereitzustellen, um diese Werke auszuführen, vorausgesetzt, Sie erfüllen alle Bedingungen dieser Lizenz für das Übertragen von Material, dessen Urheberrecht nicht bei Ihnen liegt. Diejenigen, die auf

diese Weise betroffene Werke für Sie anfertigen oder ausführen, müssen dies ausschließlich in Ihrem Namen tun, unter Ihrer Anleitung und Kontrolle und unter Bedingungen, die ihnen verbieten, außerhalb ihrer Beziehung zu Ihnen weitere Kopien Ihres urheberrechtlich geschützten Materials anzufertigen.

Übertragung ist in jedem Fall ausschließlich unter den unten aufgeführten Bedingungen gestattet. Unterlizensierung ist nicht gestattet, ist aber wegen §10 unnötig.

3. Schutz von Anwenderrechten vor Umgehungsverbotgesetzen

Kein betroffenes Werk darf als Teil eines wirksamen technischen Mechanismus' unter jedwedem anwendbarem Recht betrachtet werden, das die Auflagen von Artikel 11 des am 20. Dezember 1996 verabschiedeten WIPO-Urheberrechtsvertrags oder unter vergleichbaren Gesetzen erfüllt, die die Umgehung derartiger Mechanismen verbietet oder einschränkt.

Wenn Sie ein betroffenes Werk übertragen, verzichten Sie auf jedes Recht, die Umgehung technischer Mechanismen zu verbieten, insoweit diese Umgehung durch die Ausübung der von dieser Lizenz gewährten Rechte in bezug auf das betroffene Werk herbeigeführt wird, und Sie weisen jede Absicht von sich, die Benutzung oder Modifikation des Werks zu beschränken, um Ihre Rechtsansprüche oder Rechtsansprüche Dritter zum Verbot der Umgehung technischer Mechanismen gegen die Anwender des Werks durchzusetzen.

4. Unveränderte Kopien

Sie dürfen auf beliebigen Medien unveränderte Kopien des Quelltextes des Programms, wie sie ihn erhalten, übertragen, sofern Sie auf deutliche und angemessene Weise auf jeder Kopie einen angemessenen Urheberrechts-Vermerk veröffentlichen, alle Hinweise intakt lassen, daß diese Lizenz und sämtliche gemäß §7 hinzugefügten Einschränkungen auf den Quelltext anwendbar sind, alle Hinweise auf das Nichtvorhandensein einer Garantie intakt lassen und allen Empfängern gemeinsam mit dem Programm ein Exemplar dieser Lizenz zukommen lassen.

Sie dürfen für jede übertragene Kopie ein Entgelt – oder auch kein Entgelt – verlangen, und Sie dürfen Kundendienst- oder Garantieleistungen gegen Entgelt anbieten.

5. Übertragung modifizierter Quelltextversionen

Sie dürfen ein auf dem Programm basierendes Werk oder die nötigen Modifikationen, um es aus dem Programm zu g
und übertragen in Form von Quelltext unter den Bestimmungen von §4, vorausgesetzt, daß Sie zusätzlich alle im fo
Bedingungen erfüllen:

a) Das veränderte Werk muß auffällige Vermerke tragen, die besagen, daß Sie es modifiziert haben, und die ein
Datum angeben.

b) Das veränderte Werk muß auffällige Vermerke tragen, die besagen, daß es unter dieser Lizenz einschließli
hinzugefügten Bedingungen herausgegeben wird. Diese Anforderung wandelt die Anforderung aus §4 ab, „alle
lassen“.

c) Sie müssen das Gesamtwerk als Ganzes gemäß dieser Lizenz an jeden lizensieren, der in den Besitz einer Ko
Lizenz wird daher – ggf. einschließlich zusätzlicher Bedingungen gemäß §7 – für das Werk als Ganzes und alle
unabhängig davon, wie diese zusammengepackt werden. Diese Lizenz erteilt keine Erlaubnis, das Werk in irg
Weise zu lizensieren, setzt aber eine derartige Erlaubnis nicht außer Kraft, wenn Sie sie diese gesondert erhalten

d) Wenn das Werk über interaktive Benutzerschnittstellen verfügt, müssen diese jeweils angemessene rechtliche H
Wenn allerdings das Programm interaktive Benutzerschnittstellen hat, die keine angemessenen rechtlichen H
braucht Ihr Werk nicht dafür zu sorgen, daß sie dies tun.

Die Zusammenstellung eines betroffenen Werks mit anderen gesonderten und unabhängigen Werken, die nicht
Erweiterungen des betroffenen Werks sind und die nicht mit ihm in einer Weise kombiniert sind, um ein größeres Prog
oder auf einem Speicher- oder Verbreitungsmedium wird als „Aggregat“ bezeichnet, wenn die Zusammenstellung u

ergebende Urheberrecht nicht dazu verwendet werden, den Zugriff oder die Rechte der Benutzer der Zusam einzuschränken, als dies die einzelnen Werke erlauben. Die Aufnahme des betroffenen Werks in ein Aggregat sorgt nic Lizenz auf die anderen Teile des Aggregats wirke.

6. Übertragung in Nicht-Quelltext-Form

Sie dürfen ein betroffenes Werk in Form von Objekt-Code unter den Bedingungen der Paragraphen 4 und 5 kopiere vorausgesetzt, daß Sie außerdem den maschinenlesbaren korrespondierenden Quelltext unter den Bedingungen diese auf eine der folgenden Weisen:

a) Sie übertragen den Objekt-Code in einem physikalischen Produkt (einschließlich ein physikalisches Speicherme mit dem korrespondierenden Quelltext, der sich unveränderlich auf einem haltbaren physikalischen Med üblicherweise für den Austausch von Software verwendet wird.

b) Sie übertragen den Objekt-Code in einem physikalischen Produkt (einschließlich ein physikalisches Speicherme mit einem schriftlichen Angebot, das mindestens drei Jahre lang gültig sein muß und so lange, wie Sie Ersatzteile für dieses Produktmodell anbieten, jedem, der im Besitz des Objekt-Codes ist, entweder (1) eine Kopie des k Quelltextes der gesamten Software, die in dem Produkt enthalten und von dieser Lizenz betroffen ist, zur Verfügu einem haltbaren physikalischen Medium, das üblicherweise für den Austausch von Software verwendet wird, un Kosten als denen, die begründbar durch den physikalischen Vorgang der Übertragung des Quelltextes anfallen, od Zugriff, um den korrespondierenden Quelltext von einem Netzwerk-Server zu kopieren.

c) Sie übertragen Kopien des Objekt-Codes gemeinsam mit einer Kopie des schriftlichen Angebots, den korrespond zur Verfügung zu stellen. Diese Alternative ist nur für gelegentliche, nicht-kommerzielle Übertragung zulässig und Objekt-Code als mit einem entsprechenden Angebot gemäß Absatz 6b erhalten haben.

d) Sie übertragen den Objekt-Code dadurch, daß Sie Zugriff auf eine dafür vorgesehene Stelle gewähren, und bie Zugriff auf den korrespondierenden Quelltext auf gleichem Weg auf dieselbe Stelle und ohne zusätzliche Kosten von den Empfängern verlangen, den korrespondierenden Quelltext gemeinsam mit dem Objekt-Code zu kopieren der für das Kopieren vorgesehenen Stelle um einen Netzwerk-Server handelt, darf sich der korrespondierende C anderen Server befinden (von Ihnen oder von einem Dritten betrieben), der gleichwertige Kopiermöglichke vorausgesetzt Sie legen dem Objekt-Code klare Anleitungen bei, die besagen, wo der korrespondierende Que Unabhängig davon, welcher Netzwerk-Server den korrespondierenden Quelltext beherbergt, bleiben Sie verpflichte daß dieser lange genug bereitgestellt wird, um diesen Bedingungen zu genügen.

e) Sie übertragen den Objekt-Code unter Verwendung von Peer-To-Peer-Übertragung – vorausgesetzt, Sie informieren darüber, wo der Objekt-Code und der korrespondierende Quelltext des Werks unter den Bedingungen von Absat kostenfrei angeboten werden.

Ein abtrennbarer Anteil des Objekt-Codes, dessen Quelltext von dem korrespondierenden Quelltext als Systembibliothek ausgeschlossen ist, braucht bei der Übertragung des Werks als Objekt-Code nicht miteinbezogen zu werden.

Ein „Benutzerprodukt“ ist entweder (1) ein „Endbenutzerprodukt“, womit ein materieller persönlicher Besitz gemeint ist, der normalerweise für den persönlichen oder familiären Gebrauch oder im Haushalt eingesetzt wird, oder (2) alles, was für den Einbau in eine Wohnung hin entworfen oder dafür verkauft wird. Bei der Entscheidung, ob ein Produkt ein Endbenutzerprodukt ist, sollen Zweifelsfälle als erfaßt gelten. Wenn ein spezieller Anwender ein spezielles Produkt erhält, bezeichnet „normalerweise einsetzen“ eine typische oder weitverbreitete Anwendung dieser Produktklasse, unabhängig vom Status des speziellen Anwenders oder der Art und Weise, wie der spezielle Anwender des spezielle Produkt tatsächlich einsetzt oder wie von ihm erwartet wird, daß er es einsetzt. Ein Produkt gilt als Endbenutzerprodukt unabhängig davon, ob es substantiellen kommerziellen, industriellen oder nicht-endbenutzerspezifischen Nutzen hat, es sei denn, dieser Nutzen stellt das einzige signifikante Anwendungsgebiet des Produkts dar.

Mit „Installationsinformationen“ für ein Benutzerprodukt sind jedwede Methoden, Prozeduren, Berechtigungsschlüssel oder andere informationen gemeint, die notwendig sind, um modifizierte Versionen eines betroffenen Werks, die aus einer modifizierten Version seines korrespondierenden Quelltextes hervorgegangen sind, auf dem Produkt zu installieren und auszuführen. Die Informationen

müssen ausreichen, um sicherzustellen, daß das Weiterfunktionieren des modifizierten Objekt-Codes in keinem Fall verhindert oder gestört wird aus dem einzigen Grunde, weil Modifikationen vorgenommen worden sind.

Wenn Sie Objekt-Code gemäß diesem Paragraphen innerhalb oder zusammen mit oder speziell für den Gebrauch innerhalb eines Benutzerprodukts übertragen und die Übertragung als Teil einer Transaktion stattfindet, in der das Recht auf den Besitz und die Benutzung des Benutzerprodukts dauerhaft auf den Empfänger übergeht (unabhängig davon, wie diese Transaktion charakterisiert ist), müssen dem gemäß diesem Paragraphen mitübertragenen korrespondierenden Quelltext die Installationsinformationen beiliegen. Diese Anforderung gilt jedoch nicht, wenn weder Sie noch irgendeine Drittpartei die Möglichkeit behält, modifizierten Objekt-Code auf dem Benutzerprodukt zu installieren (zum Beispiel, wenn das Werk in einem ROM installiert wurde).

Die Anforderung, Installationsinformationen bereitzustellen, schließt keine Anforderung mit ein, weiterhin Kundendienst, Garantie oder Updates für ein Werk bereitzustellen, das vom Empfänger modifiziert oder installiert worden ist, oder für das Benutzerprodukt, in dem das Werk modifiziert oder installiert worden ist. Der Zugriff auf ein Computer-Netzwerk darf verweigert werden, wenn die Modifikation selbst die Funktion des Netzwerks grundlegend nachteilig beeinflußt oder wenn sie die Regeln und Protokolle für die Kommunikation über das Netzwerk verletzt.

Der korrespondierende Quelltext und die Installationsinformationen, die in Übereinstimmung mit diesem Paragraphen übertragen werden, müssen in einem öffentlich dokumentierten Format vorliegen (für das eine Implementation in Form von Quelltext öffentlich zugänglich ist), und sie dürfen keine speziellen Passwörter oder Schlüssel für das Auspacken, Lesen oder Kopieren erfordern.

7. Zusätzliche Bedingungen

„Zusätzliche Genehmigungen“ sind Bedingungen, die die Bedingungen dieser Lizenz ergänzen, indem sie Ausnahm mehreren Auflagen zulassen. Zusätzliche Genehmigungen zur Anwendung auf das gesamte Programm sollen so bet wären sie in dieser Lizenz enthalten, soweit dies unter anwendbarem Recht zulässig ist. Wenn zusätzliche Genehmigu Teil des Programms gelten, darf dieser Teil separat unter diesen Genehmigungen verwendet werden; das gesamte unterliegt weiterhin dieser Lizenz ohne Beachtung der zusätzlichen Genehmigungen.

Wenn Sie eine Kopie eines betroffenen Werks übertragen, dürfen Sie, wenn Sie es wünschen, jegliche zusätzliche G dieser Kopie oder jedem Teil der Kopie entfernen. (Zusätzliche Genehmigungen dürfen so verfaßt sein, daß sie in bestim Sie das Werk modifizieren, entfernt werden müssen.) Sie dürfen Material, das Sie einem betroffenen Werk hinzufügen Urheberrecht besitzen oder in entsprechender Form gewähren dürfen, mit zusätzlichen Genehmigungen ausstatten.

Ungeachtet jeglicher anderer Regelungen dieser Lizenz dürfen Sie für Material, das Sie einem betroffenen Werk hinzufüg die Urheberrechtsinhaber dieses Materials autorisiert sind), die Bedingungen dieser Lizenz um folgendes ergänzen:

a) Gewährleistungsausschluß oder Haftungsbegrenzung abweichend von §§15 und 16 dieser Lizenz oder

b) die Anforderung, spezifizierte sinnvolle rechtliche Hinweise oder Autorenschaftshinweise in diesem Mate angemessenen rechtlichen Hinweisen, die von den sie enthaltenen Werken angezeigt werden, zu erhalten, oder

c) das Verbot, die Herkunft des Materials falsch darzustellen oder die Anforderung, daß modifizierte Versionen angemessens Weise als vom Original verschieden markiert werden, oder

d) Begrenzung der Verwendung der Namen von Lizenzgebern oder Autoren des Materials für Werbezwecke oder

e) das Zurückweisen der Einräumung von Rechten gemäß dem Markenrecht zur Benutzung gewisser Produktnam Service-Marken oder

f) die Erfordernis der Freistellung des Lizenznehmers und der Autoren des Materials durch jeden, der die Software Versionen davon) überträgt, mit vertraglichen Prämissen der Verantwortung gegenüber dem Empfänger für jede V diese vertraglichen Prämissen diesen Lizenzgebern und Autoren direkt auferlegen.

Alle anderen hinzugefügten einschränkenden Bedingungen werden als „zusätzliche Einschränkungen“ im Sinne von §10 betrachtet. Wenn das Programm, wie Sie es erhalten haben, oder ein Teil davon dieser Lizenz untersteht zuzüglich einer weiteren Bedingung, die eine zusätzliche Einschränkung darstellt, dürfen Sie diese Bedingung entfernen. Wenn ein Lizenzdokument eine zusätzliche

Einschränkung enthält, aber die Relizensierung unter dieser Lizenz erlaubt, dürfen Sie dem betroffenen Werk Material hinzufügen, das den Bedingungen jenes Lizenzdokuments unterliegt, unter der Voraussetzung, daß die zusätzlichen Einschränkungen bei einer derartigen Relizensierung oder Übertragung verfallen.

Wenn Sie einem betroffenen Werk in Übereinstimmung mit diesem Paragraphen Bedingungen hinzufügen, müssen Sie in den betroffenen Quelltextdateien eine Aufstellung der zusätzlichen Bedingungen plazieren, die auf diese Quelltextdatei Anwendung finden, oder einen Hinweis darauf, wo die Zusätzlichen Bedingungen zu finden sind.

Zusätzliche Bedingungen, seien es Genehmigungen oder Einschränkungen, dürfen in Form einer separaten schriftlichen Lizenz oder in Form von Ausnahmen festgelegt werden; die o.a. Anforderungen gelten in jedem Fall.

8. Kündigung

Sie dürfen das Programm nicht verbreiten oder modifizieren, sofern es nicht durch diese Lizenz ausdrücklich gestattet ist. Jeder anderweitige Versuch der Verbreitung oder Modifizierung ist nichtig und beendet automatisch Ihre Rechte unter dieser Lizenz (einschließlich aller Patentlizenzen gemäß §11 Abs. 3).

Wenn Sie jedoch alle Verletzungen dieser Lizenz beenden, wird Ihre Lizenz durch einen speziellen Urheberrechtsinhaber wiederhergestellt, und zwar (a) vorübergehend, solange nicht bzw. bis der Rechteinhaber Ihre Lizenz ausdrücklich und endgültig kündigt, und (b) dauerhaft, sofern es der Rechteinhaber versäumt, Sie auf sinnvolle Weise auf die Lizenzverletzung innerhalb von 60 Tagen ab deren Beendigung hinzuweisen.

Darüberhinaus wird Ihre Lizenz durch einen speziellen Urheberrechtsinhaber permanent wiederhergestellt, wenn Sie der Rechteinhaber auf sinnvolle Weise auf die Verletzung hinweist, wenn außerdem dies das erste Mal ist, daß Sie auf die Verletzung dieser Lizenz (für jedes Werk) des Rechteinhabers hingewiesen werden, und wenn Sie die Verletzung innerhalb von 30 Tagen ab dem Eingang des Hinweises einstellen.

Die Beendigung Ihrer Rechte unter dieser Lizenz beendet nicht die Lizenzen Dritter, die von Ihnen Kopien oder Rechte unter dieser Lizenz erhalten haben. Wenn Ihre Rechte beendet und nicht dauerhaft wiederhergestellt worden sind, sind Sie nicht berechtigt, neue Lizenzen für dasselbe Material gemäß §10 zu erhalten.

9. Annahme der Lizenz keine Voraussetzung für den Besitz von Kopien

Um eine Kopie des Programms auszuführen, ist es nicht erforderlich, daß Sie diese Lizenz annehmen. Die nebenbei stattfindende Verbreitung eines betroffenen Werks, die sich ausschließlich als Konsequenz der Teilnahme an einer Peer-To-Peer-Datenübertragung ergibt, um eine Kopie entgegennehmen zu können, erfordert ebenfalls keine Annahme dieser Lizenz. Jedoch gibt Ihnen nichts außer dieser Lizenz die Erlaubnis, das Programm oder jedes betroffene Werk zu verbreiten oder zu verändern. Diese Handlungen verstoßen gegen das Urheberrecht, wenn Sie diese Lizenz nicht anerkennen. Indem Sie daher ein betroffenes Werk verändern oder propagieren, erklären Sie Ihr Einverständnis mit dieser Lizenz, die Ihnen diese Tätigkeiten erlaubt.

10. Automatische Lizensierung nachgeordneter Anwender

Jedesmal, wenn Sie ein betroffenes Werk übertragen, erhält der Empfänger automatisch vom ursprünglichen Lizenzgeber die Lizenz, das Werk auszuführen, zu verändern und zu propagieren – in Übereinstimmung mit dieser Lizenz. Sie sind nicht dafür verantwortlich, die Einhaltung dieser Lizenz durch Dritte durchzusetzen.

Eine „Organisations-Transaktion“ ist entweder eine Transaktion, bei der die Kontrolle über eine Organisation oder das im wesentlichen gesamte Kapital einer solchen, übertragen wird, oder sie ist die Aufteilung einer Organisation in mehrere oder die Fusion mehrerer Organisationen zu einer. Wenn die

Propagation eines betroffenen Werks durch eine Organisations-Transaktion erfolgt, erhält jeder an der Transaktion Beteiligte, der eine Kopie des Werks erhält, zugleich jedwede Lizenz an dem Werk, die der Interessenvorgänger des Beteiligten hatte, sowie das Recht auf den Besitz des korrespondierenden Quelltextes des Werks vom Interessenvorgänger, wenn dieser ihn hat oder mit vertretbarem Aufwand beschaffen kann.

Sie dürfen keine zusätzlichen Einschränkungen bzgl. der Ausübung der unter dieser Lizenz gewährten oder zugesicherten Rechte vornehmen. Beispielsweise dürfen Sie keine Lizenzgebühr oder sonstige Gebühr für die Ausübung der unter dieser Lizenz gewährten Rechte verlangen, und Sie dürfen keine Rechtsstreitigkeit beginnen (eingeschlossen Kreuz- oder Gegenansprüche in einem Gerichtsverfahren), in der Sie unterstellen, daß irgendein Patentanspruch durch Erzeugung, Anwendung, Verkauf, Verkaufsangebot oder Import des Programms oder irgendeines Teils davon verletzt wurde.

11. Patente

Ein „Kontributor“ ist ein Urheberrechtsinhaber, der die Benutzung des Programms oder eines auf dem Programm basierenden Werks unter dieser Lizenz erlaubt. Das auf diese Weise lizensierte Werk bezeichnen wir als die „Kontributor-Version“ des Kontributors.

Die „wesentlichen Patentansprüche“ eines Kontributors sind all diejenigen Patentansprüche, die der Kontributor besitzt oder kontrolliert, ob bereits erworben oder erst in Zukunft zu erwerben, die durch irgendeine Weise des gemäß dieser Lizenz erlaubten Erzeugens, Ausführens oder Verkaufens seiner Kontributor-Version verletzt würden. Dies schließt keine Patentansprüche ein, die erst als Konsequenz weiterer Modifizierung seiner Kontributor-Version entstünden. Für den Zweck dieser Definition schließt "Kontrolle" das Recht mit ein, Unterlizenzen für ein Patent zu erteilen auf eine Weise, die mit den Erfordernissen dieser Lizenz vereinbar ist.

Jeder Kontributor gewährt Ihnen eine nicht-exklusive, weltweite und gebührenfreie Patentlizenz gemäß den wesentlichen Patentansprüchen des Kontributors, den Inhalt seiner Kontributor-Version zu erzeugen, zu verkaufen, zum Verkauf anzubieten, zu importieren und außerdem auszuführen, zu modifizieren und zu propagieren.

In den folgenden drei Absätzen ist eine „Patentlizenz“ jedwede ausdrückliche Vereinbarung oder Verpflichtung, wie auch immer benannt, ein Patent nicht geltend zu machen (beispielsweise eine ausdrückliche Erlaubnis, ein Patent zu nutzen oder eine Zusicherung, bei Patentverletzung nicht zu klagen). Jemandem eine solche Patentlizenz zu „erteilen“ bedeutet, eine solche Vereinbarung oder Verpflichtung zu beschließen, ein Patent nicht gegen ihn durchzusetzen.

Wenn Sie ein betroffenes Werk übertragen, das wissentlich auf eine Patentlizenz angewiesen ist, und wenn der korrespondierende Quelltext nicht für jeden zum Kopieren zur Verfügung gestellt wird – kostenlos, unter den Bedingungen dieser Lizenz und über einen öffentlich zugänglichen Netzwerk-Server oder andere leicht zugängliche Mittel –, dann müssen Sie entweder (1) dafür sorgen, daß der korrespondierende Quelltext auf diese Weise verfügbar gemacht wird oder (2) dafür sorgen, daß Ihnen selbst die Vorteile der Patentlizenz für dieses spezielle Werk entzogen werden oder (3) in einer mit den Erfordernissen dieser Lizenz vereinbaren Weise bewirken, daß die Patentlizenz auf nachgeordnete Empfänger ausgedehnt wird. „Wissentlich angewiesen sein“ bedeutet, daß Sie tatsächliches Wissen darüber haben, daß – außer wegen der Patentlizenz – Ihre Übertragung des betroffenen Werks in einen Staat oder die Benutzung des betroffenen Werks durch Ihren Empfänger in einem Staat, eins oder mehrere identifizierbare Patente in diesem Staat verletzen würden, deren Gültigkeit Ihnen glaubhaft erscheint.

Wenn Sie, als Folge von oder in Verbindung mit einer einzelnen Transaktion oder Vereinbarung, ein betroffenes Werk übertragen oder durch Vermittlung einer Übertragung propagieren, und Sie gewähren einigen Empfängern eine Patentlizenz, die ihnen das Benutzen, Propagieren, Modifizieren und Übertragen einer speziellen Kopie des betroffenen Werks gestatten, dann wird die von Ihnen gewährte Patentlizenz automatisch auf alle Empfänger des betroffenen Werks und darauf basierender Werke ausgedehnt.

Eine Patentlizenz ist „diskriminierend“, wenn sie in ihrem Gültigkeitsbereich die speziell unter dieser Lizenz gewährten Rechte nicht einschließt, wenn sie die Ausübung dieser Rechte verbietet oder wenn sie die Nichtausübung einer oder mehrerer dieser Rechte zur Bedingung hat. Sie dürfen ein betroffenes

Werk nicht übertragen, wenn Sie Partner in einem Vertrag mit einer Drittpartei sind, die auf dem Gebiet der Verbreitung von Software geschäftlich tätig ist, gemäß dem Sie dieser Drittpartei Zahlungen leisten, die auf dem Maß Ihrer Aktivität des Übertragens des Werks basieren, und gemäß dem die Drittpartei eine diskriminierende Patentlizenz all denjenigen gewährt, die das Werk von Ihnen erhielten, (a) in Verbindung mit von Ihnen übertragenen Kopien des betroffenen Werks (oder Kopien dieser Kopien) oder (b) hauptsächlich für und in Verbindung mit spezifischen Produkten oder Zusammenstellungen, die das betroffene Werk enthalten, es sei denn, Sie sind in diesen Vertrag vor dem 28. März 2007 eingetreten oder die Patentlizenz wurde vor diesem Datum erteilt.

Nichts in dieser Lizenz soll in einer Weise ausgelegt werden, die irgendeine implizite Lizenz oder sonstige Abwehr gegen Rechtsverletzung ausschließt oder begrenzt, die Ihnen ansonsten gemäß anwendbarem Patentrecht zustünde.

12. Keine Preisgabe der Freiheit Dritter

Sollten Ihnen (durch Gerichtsbeschluß, Vergleich oder anderweitig) Bedingungen auferlegt werden, die den Bedingungen dieser Lizenz widersprechen, so befreien Sie diese Umstände nicht von den Bestimmungen dieser Lizenz. Wenn es Ihnen nicht möglich ist, ein betroffenes Werk unter gleichzeitiger Beachtung der Bedingungen in dieser Lizenz und Ihrer anderweitigen Verpflichtungen zu übertragen, dann dürfen Sie als Folge das Programm überhaupt nicht übertragen. Wenn Sie zum Beispiel Bedingungen akzeptieren, die Sie dazu verpflichten, von denen, denen Sie das Programm übertragen haben, eine Gebühr für die weitere Übertragung einzufordern, dann besteht der einzige Weg, sowohl jene Bedingungen als auch diese Lizenz zu befolgen darin, ganz auf die Übertragung des Programms zu verzichten.

13. Nutzung zusammen mit der GNU Affero General Public License

Ungeachtet anderer Regelungen dieser Lizenz, ist es Ihnen gestattet, ein betroffenes Werk mit einem Werk zu einem einzelnen, kombinierten Werk zu verbinden (linken) oder zu kombinieren, das unter Version 3 der GNU Affero General Public License steht, und das Ergebnis zu übertragen. Die Bedingungen dieser Lizenz bleiben weiterhin auf denjenigen Teil anwendbar, der das betroffene Werk darstellt, aber die speziellen Anforderungen der GNU Affero General Public License, §13, die sich auf Interaktion über ein Computer-Netzwerk beziehen, werden auf die Kombination als solche anwendbar.

14. Überarbeitungen dieser Lizenz

Die Free Software Foundation kann von Zeit zu Zeit überarbeitete und/oder neue Versionen der *General Public License* veröffentlichen. Solche neuen Versionen werden vom Grundprinzip her der gegenwärtigen entsprechen, können aber im Detail abweichen, um neuen Problemen und Anforderungen gerecht zu werden.

Jede Version dieser Lizenz hat eine eindeutige Versionsnummer. Wenn in einem Programm angegeben wird, daß es dieser Lizenz in einer bestimmten Versionsnummer „oder jeder späteren Version“ *(“or any later version”)* unterliegt, so haben Sie die Wahl, entweder den Bestimmungen der genannten Version zu folgen oder denen jeder beliebigen späteren Version, die von der Free Software Foundation veröffentlicht wurde. Wenn das Programm keine Versionsnummer angibt, können Sie eine beliebige Version wählen, die je von der Free Software Foundation veröffentlicht wurde.

15. Gewährleistungsausschluß

Es besteht keinerlei Gewährleistung für das Programm, soweit dies gesetzlich zulässig ist. Sofern nicht anderweitig schriftlich bestätigt, stellen die Urheberrechtsinhaber und/oder Dritte das Programm so zur Verfügung, „wie es ist“, ohne irgendeine Gewährleistung, weder ausdrücklich noch implizit, einschließlich – aber nicht begrenzt auf – die implizite Gewährleistung der Marktreife oder der Verwendbarkeit für einen bestimmten Zweck. Das volle Risiko bezüglich Qualität und Leistungsfähigkeit des Programms liegt bei Ihnen. Sollte

sich das Programm als fehlerhaft herausstellen, liegen die Kosten für notwendigen Service, Reparatur oder Korrektur bei Ihnen.

16. Haftungsbegrenzung

In keinem Fall, außer wenn durch geltendes Recht gefordert oder schriftlich zugesichert, ist irgendein Urheberrechtsinhaber oder irgendein Dritter, der das Programm wie oben erlaubt modifiziert oder übertragen hat, Ihnen gegenüber für irgendwelche Schäden haftbar, einschließlich jeglicher allgemeiner oder spezieller Schäden, Schäden durch Seiteneffekte (Nebenwirkungen) oder Folgeschäden, die aus der Benutzung des Programms oder der Unbenutzbarkeit des Programms folgen (einschließlich – aber nicht beschränkt auf – Datenverluste, fehlerhafte Verarbeitung von Daten, Verluste, die von Ihnen oder anderen getragen werden müssen, oder dem Unvermögen des Programms, mit irgendeinem anderen Programm zusammenzuarbeiten), selbst wenn ein Urheberrechtsinhaber oder Dritter über die Möglichkeit solcher Schäden unterrichtet worden war.

17. Interpretation von §§ 15 und 16

Sollten der o.a. Gewährleistungsausschluß und die o.a. Haftungsbegrenzung aufgrund ihrer Bedingungen gemäß lokalem Recht unwirksam sein, sollen Bewertungsgerichte dasjenige lokale Recht anwenden, das einer absoluten Aufhebung jeglicher zivilen Haftung in Zusammenhang mit dem Programm am nächsten kommt, es sei denn, dem Programm lag eine entgeltliche Garantieerklärung oder Haftungsübernahme bei.

ENDE DER LIZENZBEDINGUNGEN

Wie Sie diese Bedingungen auf Ihre eigenen, neuen Programme anwenden können

Wenn Sie ein neues Programm entwickeln und wollen, daß es vom größtmöglichen Nutzen für die Allgemeinheit ist, dann erreichen Sie das am besten, indem Sie es zu freier Software machen, die jeder unter diesen Bestimmungen weiterverbreiten und verändern kann.

Um dies zu erreichen, fügen Sie die folgenden Vermerke zu Ihrem Programm hinzu. Am sichersten ist es, sie an den Anfang einer jeden Quelldatei zu stellen, um den Gewährleistungsausschluß möglichst deutlich darzustellen; zumindest aber sollte jede Datei die „Copyright“-Zeile besitzen sowie einen kurzen Hinweis darauf, wo die vollständigen Vermerke zu finden sind.

[eine Zeile mit dem Programmnamen und einer kurzen Beschreibung]
Copyright (C) [Jahr] [Name des Autors]

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 3 of the License, 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.

You should have received a copy of the GNU General Public License along with this program; if not, see <http://www.gnu.org/licenses/>.

Auf Deutsch:

[eine Zeile mit dem Programmnamen und einer kurzen Beschreibung]

Copyright (C) [Jahr] [Name des Autors]

Dieses Programm ist freie Software. Sie können es unter den Bedingungen der GNU General Public License, wie von der Free Software Foundation veröffentlicht, weitergeben und/oder modifizieren, entweder gemäß Version 3 der Lizenz oder (nach Ihrer Option) jeder späteren Version.

Die Veröffentlichung dieses Programms erfolgt in der Hoffnung, daß es Ihnen von Nutzen sein wird, aber OHNE IRGENDEINE GARANTIE, sogar ohne die implizite Garantie der MARKTREIFE oder der VERWENDBARKEIT FÜR EINEN BESTIMMTEN ZWECK. Details finden Sie in der GNU General Public License.

Sie sollten ein Exemplar der GNU General Public License zusammen mit diesem Programm erhalten haben. Falls nicht, siehe <http://www.gnu.org/licenses/>.

Fügen Sie auch einen kurzen Hinweis hinzu, wie Sie elektronisch und per Brief erreichbar sind.

Wenn Ihr Programm interaktive Befehle in einem Terminal entgegennimmt, sorgen Sie dafür, daß es nach dem Start einen kurzen Vermerk ausgibt:

[Programm] Copyright (C) [Jahr] [Name des Autors]
This program comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
This is free software, and you are welcome to redistribute it under certain conditions; type ‘show c’ for details.

Auf Deutsch:

[Programm] Copyright (C) [Jahr] [Name des Autors] Für dieses Programm besteht KEINERLEI GARANTIE; geben Sie "show w" für Details ein.
Dies ist freie Software, die Sie unter bestimmten Bedingungen weitergeben dürfen; geben Sie "show c" für Details ein.

Die hypothetischen Kommandos „show w“ und „show c“ sollten die entsprechenden Teile der GNU-GPL anzeigen. Natürlich können die von Ihnen verwendeten Kommandos auch anders lauten; für ein Programm mit graphischer Benutzeroberfläche werden Sie sicherlich eine „About-Box“ verwenden.

Soweit vorhanden, sollten Sie auch Ihren Arbeitgeber (wenn Sie als Programmierer arbeiten) oder Ihre Schule einen Urheberrechteverzicht für das Programm unterschreiben lassen. Für weitere Informationen darüber und wie Sie die GNU GPL anwenden und befolgen, siehe http://www.gnu.org/licenses/.

Diese *General Public License* gestattet nicht die Einbindung Ihres Programms in proprietäre Programme. Wenn Ihr Programm eine Funktionsbibliothek ist, dann kann es sinnvoller sein, das Linken proprietärer Programme mit dieser Bibliothek zu gestatten. Wenn dies Ihre Absicht ist, sollten Sie die GNU Lesser General Public License anstelle dieser Lizenz verwenden. Lesen Sie aber bitte vorher http://www.gnu.org/philosophy/why-not-lgpl.html.

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarante your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

“This License” refers to version 3 of the GNU General Public License.
“Copyright” also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.
“The Program” refers to any copyrightable work licensed under this License. Each licensee is addressed as

“you”. “Licensees” and “recipients” may be individuals or organizations.

To “modify” a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version” of the earlier work or a work “based on” the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate” a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.
An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.
A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works,

provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to “keep intact all notices”.
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an “aggregate” if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these

ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and

protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- **a)** Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- **b)** Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- **c)** Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- **d)** Limiting the use for publicity purposes of names of licensors or authors of the material; or
- **e)** Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- **f)** Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your

license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's “contributor version”.

A contributor's “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license

for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year>  <name of author>

 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 3 of the License, 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.


You should have received a copy of the GNU General Public License
 along with this program.  If not, see <https://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
<program>  Copyright (C) <year>  <name of author>
 This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
 under certain conditions; type `show c' for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public

License. Of course, your program's commands might be different; for a GUI interface, you would use an “about box”.

You should also get your employer (if you work as a programmer) or school, if any, to sign a “copyright disclaimer” for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <https://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <https://www.gnu.org/licenses/why-not-lgpl.html>.

GNU PUBLIC LISENZ Version 2

Deutsch
Englisch

Deutsche Übersetzung der Version 2, Juni 1991

Den offiziellen englischen Originaltext finden Sie unter http://www.gnu.org/licenses/gpl-2.0.html.

Diese Übersetzung wurde ursprünglich erstellt von Katja Lachmann Übersetzungen im Auftrag der S.u.S.E. GmbH –http://www.suse.de.
Sie wurde überarbeitet von Peter Gerwinski, G-N-U GmbH – http://www.g-n-u.de (31. Oktober 1996, 4. Juni 2000, 29. Januar 2011).

Diese Übersetzung wird mit der Absicht angeboten, das Verständnis der GNU General Public License (GNU GPL) zu erleichtern. Es handelt sich jedoch nicht um eine offizielle oder im rechtlichen Sinne anerkannte Übersetzung.

Die Free Software Foundation (FSF) ist nicht der Herausgeber dieser Übersetzung, und sie hat diese Übersetzung auch nicht als rechtskräftigen Ersatz für die Original-GNU-GPL anerkannt. Da die Übersetzung nicht sorgfältig von Anwälten überprüft wurde, können die Übersetzer nicht garantieren, daß die Übersetzung die rechtlichen Aussagen der GNU GPL exakt wiedergibt. Wenn Sie sichergehen wollen, daß von Ihnen geplante Aktivitäten im Sinne der GNU GPL gestattet sind, halten Sie sich bitte an die ***englischsprachige Originalversion****.*

Die Übersetzer und die Free Software Foundation möchten Sie darum bitten, diese Übersetzung nicht als offizielle Lizenzbedingungen für von Ihnen geschriebene Programme zu verwenden. Bitte benutzen Sie hierfür stattdessen die von der Free Software Foundation herausgegebene ***englischsprachige Originalversion****.*

This is a translation of the GNU General Public License into German. This translation is distributed in the hope that it will facilitate understanding, but it is not an official or legally approved translation.

The Free Software Foundation is not the publisher of this translation and has not approved it as a legal substitute for the authentic GNU General Public License. The translation has not been reviewed carefully by lawyers, and therefore the translator cannot be sure that it exactly represents the legal meaning of the GNU General Public License. If you wish to be sure whether your planned activities are permitted by the GNU General Public License, please refer to the authentic English version.

The translators and the Free Software Foundation strongly urge you not to use this translation as the official distribution terms for your programs; instead, please use the authentic English version published by the Free Software Foundation.

GNU General Public License

Deutsche Übersetzung der Version 2, Juni 1991

Diese Übersetzung ist kein rechtskräftiger Ersatz für die englischsprachige Originalversion!

Vorwort

Die meisten Softwarelizenzen sind daraufhin entworfen worden, Ihnen die Freiheit zu nehmen, die Software weiterzugeben und zu verändern. Im Gegensatz dazu soll Ihnen die *GNU General Public License*, die Allgemeine Öffentliche GNU-Lizenz, ebendiese Freiheit garantieren. Sie soll sicherstellen, daß die Software für alle Benutzer frei ist. Diese Lizenz gilt für den Großteil der von der *Free Software Foundation* herausgegebenen Software und für alle anderen Programme, deren Autoren ihr Werk dieser Lizenz unterstellt haben. Auch Sie können diese Möglichkeit der Lizenzierung für Ihre Programme

anwenden. (Ein anderer Teil der Software der *Free Software Foundation* unterliegt stattdessen der *GNU Lesser General Public License*, der Kleineren Allgemeinen Öffentlichen GNU-Lizenz.)

Die Bezeichnung *„freie" Software* bezieht sich auf Freiheit, nicht auf den Preis. Unsere Lizenzen sollen Ihnen die Freiheit garantieren, Kopien freier Software zu verbreiten (und etwas für diesen Service zu berechnen, wenn Sie möchten), die Möglichkeit, die Software im Quelltext zu erhalten oder den Quelltext auf Wunsch zu bekommen. Die Lizenzen sollen garantieren, daß Sie die Software ändern oder Teile davon in neuen freien Programmen verwenden dürfen – und daß Sie wissen, daß Sie dies alles tun dürfen.

Um Ihre Rechte zu schützen, müssen wir Einschränkungen machen, die es jedem verbieten, Ihnen diese Rechte zu verweigern oder Sie aufzufordern, auf diese Rechte zu verzichten. Aus diesen Einschränkungen folgen bestimmte Verantwortlichkeiten für Sie, wenn Sie Kopien der Software verbreiten oder sie verändern.

Beispielsweise müssen Sie den Empfängern alle Rechte gewähren, die Sie selbst haben, wenn Sie – kostenlos oder gegen Bezahlung – Kopien eines solchen Programms verbreiten. Sie müssen sicherstellen, daß auch die Empfänger den Quelltext erhalten bzw. erhalten können. Und Sie müssen ihnen diese Bedingungen zeigen, damit sie ihre Rechte kennen.

Wir schützen Ihre Rechte in zwei Schritten: (1) Wir stellen die Software unter ein Urheberrecht (Copyright), und (2) wir bieten Ihnen diese Lizenz an, die Ihnen das Recht gibt, die Software zu vervielfältigen, zu verbreiten und/oder zu verändern.

Um die Autoren und uns zu schützen, wollen wir darüberhinaus sicherstellen, daß jeder erfährt, daß für diese freie Software keinerlei Garantie besteht. Wenn die Software von jemand anderem modifiziert und weitergegeben wird, möchten wir, daß die Empfänger wissen, daß sie nicht das Original erhalten haben, damit irgendwelche von anderen verursachte Probleme nicht den Ruf des ursprünglichen Autors schädigen.

Schließlich und endlich ist jedes freie Programm permanent durch Software-Patente bedroht. Wir möchten die Gefahr ausschließen, daß Distributoren eines freien Programms individuell Patente lizensieren – mit dem Ergebnis, daß das Programm proprietär würde. Um dies zu verhindern, haben wir klargestellt, daß jedes Patent entweder für freie Benutzung durch jedermann lizenziert werden muß oder überhaupt nicht lizenziert werden darf.

Es folgen die genauen Bedingungen für die Vervielfältigung, Verbreitung und Bearbeitung:

Allgemeine Öffentliche GNU-Lizenz

Bedingungen für die Vervielfältigung, Verbreitung und Bearbeitung

§0. Diese Lizenz gilt für jedes Programm und jedes andere Werk, in dem ein entsprechender Vermerk des Copyright-Inhabers darauf hinweist, daß das Werk unter den Bestimmungen dieser *General Public License* verbreitet werden darf. Im folgenden wird jedes derartige Programm oder Werk als „das Programm" bezeichnet; die Formulierung „auf dem Programm basierendes Werk" bezeichnet das Programm sowie jegliche Bearbeitung des Programms im urheberrechtlichen Sinne, also ein Werk, welches das Programm, auch auszugsweise, sei es unverändert oder verändert und/oder in eine andere Sprache übersetzt, enthält. (Im folgenden wird die Übersetzung ohne Einschränkung als „Bearbeitung" eingestuft.) Jeder Lizenznehmer wird im folgenden als „Sie" angesprochen.

Andere Handlungen als Vervielfältigung, Verbreitung und Bearbeitung werden von dieser Lizenz nicht berührt; sie fallen nicht in ihren Anwendungsbereich. Der Vorgang der Ausführung des Programms wird nicht eingeschränkt, und die Ausgaben des Programms unterliegen dieser Lizenz nur, wenn der Inhalt ein auf dem Programm basierendes Werk darstellt (unabhängig davon, daß die Ausgabe durch die Ausführung des Programmes erfolgte). Ob dies zutrifft, hängt von den Funktionen des Programms ab.

§1. Sie dürfen auf beliebigen Medien unveränderte Kopien des Quelltextes des Programms, wie sie ihn erhalten haben, anfertigen und verbreiten. Voraussetzung hierfür ist, daß Sie mit jeder Kopie einen entsprechenden Copyright-Vermerk sowie einen Haftungsausschluß veröffentlichen, alle Vermerke, die sich auf diese Lizenz und das Fehlen einer Garantie beziehen, unverändert lassen und desweiteren allen anderen Empfängern des Programms zusammen mit dem Programm eine Kopie dieser Lizenz zukommen lassen.

Sie dürfen für den physikalischen Vorgang des Zugänglichmachens einer Kopie eine Gebühr verlangen.

Wenn Sie es wünschen, dürfen Sie auch gegen Entgelt eine Garantie für das Programm anbieten.

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

1. Sie müssen die veränderten Dateien mit einem auffälligen Vermerk versehen, der auf die von Ihnen vorgenommene Modifizierung und das Datum jeder Änderung hinweist.
2. 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.
3. 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 Werk auch keine solche Meldung ausgeben).

Diese Anforderungen gelten für das bearbeitete Werk als Ganzes. Wenn identifizierbare Teile des Werkes nicht von dem Programm abgeleitet sind und vernünftigerweise als unabhängige und eigenständige Werke für sich selbst zu betrachten sind, dann gelten diese Lizenz und ihre Bedingungen nicht für die betroffenen Teile, wenn Sie diese als eigenständige Werke weitergeben. Wenn Sie jedoch dieselben Abschnitte als Teil eines Ganzen weitergeben, das ein auf dem Programm basierendes Werk 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 Werke in Anspruch zu nehmen oder Ihnen die Rechte für Werke streitig zu machen, die komplett von Ihnen geschrieben wurden; vielmehr ist es die Absicht, die Rechte zur Kontrolle der Verbreitung von Werken, die auf dem Programm basieren oder unter seiner auszugsweisen Verwendung zusammengestellt worden sind, auszuüben.

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

§3. Sie dürfen das Programm (oder ein darauf basierendes Werk gemäß Paragraph 2) als Objectcode oder in ausführbarer Form unter den Bedingungen der Paragraphen 1 und 2 kopieren und weitergeben – vorausgesetzt, daß Sie außerdem eine der folgenden Leistungen erbringen:

1. Liefern Sie das Programm zusammen mit dem vollständigen zugehörigen maschinenlesbaren Quelltext auf einem für den Datenaustausch üblichen Medium aus, wobei die Verteilung unter den Bedingungen der Paragraphen 1 und 2 erfolgen muß. Oder:
2. Liefern Sie das Programm zusammen mit einem mindestens drei Jahre lang gültigen schriftlichen Angebot aus, jedem Dritten eine vollständige maschinenlesbare Kopie des Quelltextes zur Verfügung zu stellen – zu nicht höheren Kosten als denen, die durch das physikalische Zugänglichmachen des Quelltextes anfallen –, wobei der Quelltext unter den Bedingungen der Paragraphen 1 und 2 auf einem für den Datenaustausch üblichen Medium weitergegeben wird. Oder:
3. Liefern Sie das Programm zusammen mit dem schriftlichen Angebot der Zurverfügungstellung des Quelltextes aus, das Sie selbst erhalten haben. (Diese Alternative ist nur für nicht-kommerzielle Verbreitung zulässig und nur, wenn Sie das Programm als Objectcode oder in ausführbarer Form mit einem entsprechenden Angebot gemäß Absatz b erhalten haben.)

Unter dem Quelltext eines Werkes wird diejenige Form des Werkes verstanden, die für Bearbeitungen vorzugsweise verwendet wird. Für ein ausführbares Programm bedeutet „der komplette Quelltext“: Der Quelltext aller im Programm enthaltenen Module einschließlich aller zugehörigen Modulschnittstellen-Definitionsdateien sowie der zur Compilation und Installation verwendeten Skripte. Als besondere Ausnahme jedoch braucht der verteilte Quelltext nichts von dem zu enthalten, was üblicherweise (entweder als Quelltext oder in binärer Form) zusammen mit den Hauptkomponenten des Betriebssystems (Kernel, Compiler usw.) geliefert wird, unter dem das Programm läuft – es sei denn, diese Komponente selbst gehört

zum ausführbaren Programm.

Wenn die Verbreitung eines ausführbaren Programms oder von Objectcode dadurch erfolgt, daß der Kopierzugriff auf eine dafür vorgesehene Stelle gewährt wird, so gilt die Gewährung eines gleichwertigen Kopierzugriffs auf den Quelltext von derselben Stelle als Verbreitung des Quelltextes, auch wenn Dritte nicht dazu gezwungen sind, den Quelltext zusammen mit dem Objectcode zu kopieren.

§4. Sie dürfen das Programm nicht vervielfältigen, verändern, weiter lizenzieren oder verbreiten, sofern es nicht durch diese Lizenz ausdrücklich gestattet ist. Jeder anderweitige Versuch der Vervielfältigung, Modifizierung, Weiterlizenzierung und Verbreitung ist nichtig und beendet automatisch Ihre Rechte unter dieser Lizenz. Jedoch werden die Lizenzen Dritter, die von Ihnen Kopien oder Rechte unter dieser Lizenz erhalten haben, nicht beendet, solange diese die Lizenz voll anerkennen und befolgen.

§5. Sie sind nicht verpflichtet, diese Lizenz anzunehmen, da Sie sie nicht unterzeichnet haben. Jedoch gibt Ihnen nichts anderes die Erlaubnis, das Programm oder von ihm abgeleitete Werke zu verändern oder zu verbreiten. Diese Handlungen sind gesetzlich verboten, wenn Sie diese Lizenz nicht anerkennen. Indem Sie das Programm (oder ein darauf basierendes Werk) verändern oder verbreiten, erklären Sie Ihr Einverständnis mit dieser Lizenz und mit allen ihren Bedingungen bezüglich der Vervielfältigung, Verbreitung und Veränderung des Programms oder eines darauf basierenden Werks.

§6. Jedesmal, wenn Sie das Programm (oder ein auf dem Programm basierendes Werk) weitergeben, erhält der Empfänger automatisch vom ursprünglichen Lizenzgeber die Lizenz, das Programm entsprechend den hier festgelegten Bestimmungen zu vervielfältigen, zu verbreiten und zu verändern. Sie dürfen keine weiteren Einschränkungen der Durchsetzung der hierin zugestandenen Rechte des Empfängers vornehmen. Sie sind nicht dafür verantwortlich, die Einhaltung dieser Lizenz durch Dritte durchzusetzen.

§7. Sollten Ihnen infolge eines Gerichtsurteils, des Vorwurfs einer Patentverletzung oder aus einem anderen Grunde (nicht auf Patentfragen begrenzt) Bedingungen (durch Gerichtsbeschluß, Vergleich oder anderweitig) auferlegt werden, die den Bedingungen dieser Lizenz widersprechen, so befreien Sie diese Umstände nicht von den Bestimmungen dieser Lizenz. Wenn es Ihnen nicht möglich ist, das Programm unter gleichzeitiger Beachtung der Bedingungen in dieser Lizenz und Ihrer anderweitigen Verpflichtungen zu verbreiten, dann dürfen Sie als Folge das Programm überhaupt nicht verbreiten. Wenn zum Beispiel ein Patent nicht die gebührenfreie Weiterverbreitung des Programms durch diejenigen erlaubt, die das Programm direkt oder indirekt von Ihnen erhalten haben, dann besteht der einzige Weg, sowohl das Patentrecht als auch diese Lizenz zu befolgen, darin, ganz auf die Verbreitung des Programms zu verzichten.

Sollte sich ein Teil dieses Paragraphen als ungültig oder unter bestimmten Umständen nicht durchsetzbar erweisen, so soll dieser Paragraph seinem Sinne nach angewandt werden; im übrigen soll dieser Paragraph als Ganzes gelten.

Zweck dieses Paragraphen ist nicht, Sie dazu zu bringen, irgendwelche Patente oder andere Eigentumsansprüche zu verletzen oder die Gültigkeit solcher Ansprüche zu bestreiten; dieser Paragraph hat einzig den Zweck, die Integrität des Verbreitungssystems der freien Software zu schützen, das durch die Praxis öffentlicher Lizenzen verwirklicht wird. Viele Leute haben großzügige Beiträge zu dem großen Angebot der mit diesem System verbreiteten Software im Vertrauen auf die konsistente Anwendung dieses Systems geleistet; es liegt am Autor/Geber, zu entscheiden, ob er die Software mittels irgendeines anderen Systems verbreiten will; ein Lizenznehmer hat auf diese Entscheidung keinen Einfluß.

Dieser Paragraph ist dazu gedacht, deutlich klarzustellen, was als Konsequenz aus dem Rest dieser Lizenz betrachtet wird.

§8. Wenn die Verbreitung und/oder die Benutzung des Programms in bestimmten Staaten entweder durch Patente oder durch urheberrechtlich geschützte Schnittstellen eingeschränkt ist, kann der Urheberrechtsinhaber, der das Programm unter diese Lizenz gestellt hat, eine explizite geographische Begrenzung der Verbreitung angeben, in der diese Staaten ausgeschlossen werden, so daß die Verbreitung nur innerhalb und zwischen den Staaten erlaubt ist, die nicht ausgeschlossen sind. In einem solchen Fall beinhaltet diese Lizenz die Beschränkung, als wäre sie in diesem Text niedergeschrieben.

§9. Die *Free Software Foundation* kann von Zeit zu Zeit überarbeitete und/oder neue Versionen der *General Public License* veröffentlichen. Solche neuen Versionen werden vom Grundprinzip her der gegenwärtigen entsprechen, können aber im Detail abweichen, um neuen Problemen und Anforderungen gerecht zu werden.

Jede Version dieser Lizenz hat eine eindeutige Versionsnummer. Wenn in einem Programm angegeben wird, daß es dieser Lizenz in einer bestimmten Versionsnummer oder „jeder späteren Version“ *(“any later*

version") unterliegt, so haben Sie die Wahl, entweder den Bestimmungen der genannten Version zu folgen oder denen jeder beliebigen späteren Version, die von der *Free Software Foundation* veröffentlicht wurde. Wenn das Programm keine Versionsnummer angibt, können Sie eine beliebige Version wählen, die je von der *Free Software Foundation* veröffentlicht wurde.

§10. Wenn Sie den Wunsch haben, Teile des Programms in anderen freien Programmen zu verwenden, deren Bedingungen für die Verbreitung anders sind, schreiben Sie an den Autor, um ihn um die Erlaubnis zu bitten. Für Software, die unter dem Copyright der *Free Software Foundation* steht, schreiben Sie an die *Free Software Foundation*; wir machen zu diesem Zweck gelegentlich Ausnahmen. Unsere Entscheidung wird von den beiden Zielen geleitet werden, zum einen den freien Status aller von unserer freien Software abgeleiteten Werke zu erhalten und zum anderen das gemeinschaftliche Nutzen und Wiederverwenden von Software im allgemeinen zu fördern.

Keine Gewährleistung

§11. Da das Programm ohne jegliche Kosten lizenziert wird, besteht keinerlei Gewährleistung für das Programm, soweit dies gesetzlich zulässig ist. Sofern nicht anderweitig schriftlich bestätigt, stellen die Copyright-Inhaber und/oder Dritte das Programm so zur Verfügung, „wie es ist", ohne irgendeine Gewährleistung, weder ausdrücklich noch implizit, einschließlich – aber nicht begrenzt auf – Marktreife oder Verwendbarkeit für einen bestimmten Zweck. Das volle Risiko bezüglich Qualität und Leistungsfähigkeit des Programms liegt bei Ihnen. Sollte sich das Programm als fehlerhaft herausstellen, liegen die Kosten für notwendigen Service, Reparatur oder Korrektur bei Ihnen.

§12. In keinem Fall, außer wenn durch geltendes Recht gefordert oder schriftlich zugesichert, ist irgendein Copyright-Inhaber oder irgendein Dritter, der das Programm wie oben erlaubt modifiziert oder verbreitet hat, Ihnen gegenüber für irgendwelche Schäden haftbar, einschließlich jeglicher allgemeiner oder spezieller Schäden, Schäden durch Seiteneffekte (Nebenwirkungen) oder Folgeschäden, die aus der Benutzung des Programms oder der Unbenutzbarkeit des Programms folgen (einschließlich – aber nicht beschränkt auf – Datenverluste, fehlerhafte Verarbeitung von Daten, Verluste, die von Ihnen oder anderen getragen werden müssen, oder dem Unvermögen des Programms, mit irgendeinem anderen Programm zusammenzuarbeiten), selbst wenn ein Copyright-Inhaber oder Dritter über die Möglichkeit solcher Schäden unterrichtet worden war.

Ende der Bedingungen

Wie Sie diese Bedingungen auf Ihre eigenen, neuen Programme anwenden können

Wenn Sie ein neues Programm entwickeln und wollen, daß es vom größtmöglichen Nutzen für die Allgemeinheit ist, dann erreichen Sie das am besten, indem Sie es zu freier Software machen, die jeder unter diesen Bestimmungen weiterverbreiten und verändern kann.

Um dies zu erreichen, fügen Sie die folgenden Vermerke zu Ihrem Programm hinzu. Am sichersten ist es, sie an den Anfang einer jeden Quelldatei zu stellen, um den Gewährleistungsausschluß möglichst deutlich darzustellen; zumindest aber sollte jede Datei eine Copyright-Zeile besitzen sowie einen kurzen Hinweis darauf, wo die vollständigen Vermerke zu finden sind.

```
[eine Zeile mit dem Programmnamen und einer kurzen Beschreibung]
Copyright (C) [Jahr]  [Name des Autors]

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 of the License, 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.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110, USA

Auf Deutsch:

[eine Zeile mit dem Programmnamen und einer kurzen Beschreibung]
Copyright (C) [Jahr] [Name des Autors]

Dieses Programm ist freie Software. Sie können es unter den Bedingungen der GNU General Public License, wie von der Free Software Foundation veröffentlicht, weitergeben und/oder modifizieren, entweder gemäß Version 2 der Lizenz oder (nach Ihrer Option) jeder späteren Version.

*Die Veröffentlichung dieses Programms erfolgt in der Hoffnung, daß es Ihnen von Nutzen sein wird, aber **OHNE IRGENDEINE GARANTIE**, sogar ohne die implizite Garantie der **MARKTREIFE** oder der **VERWENDBARKEIT FÜR EINEN BESTIMMTEN ZWECK**. Details finden Sie in der GNU General Public License.*

Sie sollten ein Exemplar der GNU General Public License zusammen mit diesem Programm erhalten haben. Falls nicht, schreiben Sie an die Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110, USA.

Fügen Sie auch einen kurzen Hinweis hinzu, wie Sie elektronisch und per Brief erreichbar sind.

Wenn Ihr Programm interaktiv ist, sorgen Sie dafür, daß es nach dem Start einen kurzen Vermerk ausgibt:

version 69, Copyright (C) [Jahr] [Name des Autors]
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’. This is free software, and you are welcome to redistribute it under certain conditions; type ‘show c’ for details.

Auf Deutsch:

Version 69, Copyright (C) [Jahr] [Name des Autors] Für Gnomovision besteht KEINERLEI GARANTIE; geben Sie "show w" für Details ein. Gnonovision ist freie Software, die Sie unter bestimmten Bedingungen weitergeben dürfen; geben Sie "show c" für Details ein.

Die hypothetischen Kommandos „show w“ und „show c“ sollten die entsprechenden Teile der GNU-GPL anzeigen. Natürlich können die von Ihnen verwendeten Kommandos anders heißen als „show w“ und „show c“; es könnten auch Mausklicks oder Menüpunkte sein – was immer am besten in Ihr Programm paßt.

Soweit vorhanden, sollten Sie auch Ihren Arbeitgeber (wenn Sie als Programmierer arbeiten) oder Ihre Schule einen Copyright-Verzicht für das Programm unterschreiben lassen. Hier ein Beispiel. Die Namen müssen Sie natürlich ändern.

Yoyodyne, Inc., hereby disclaims all copyright interest in the program ‘Gnomovision’ (which makes passes at compilers) written by James Hacker.

[Unterschrift von Ty Coon], 1 April 1989
Ty Coon, President of Vice

Auf Deutsch:

*Die Yoyodyne GmbH erhebt **keinen** urheberrechtlichen Anspruch auf das von James Hacker geschriebene Programm "Gnomovision" (einem Schrittmacher für Compiler).*

[Unterschrift von Ty Coon], 1. April 1989
Ty Coon, Vizepräsident

Diese *General Public License* gestattet nicht die Einbindung des Programms in proprietäre Programme. Ist Ihr Programm eine Funktionsbibliothek, so kann es sinnvoller sein, das Binden proprietärer Programme mit dieser Bibliothek zu gestatten. Wenn Sie dies tun wollen, sollten Sie die GNU Lesser General Public License anstelle dieser Lizenz verwenden.

GNU Lesser PUBLIC LISENZ Version 3

Version 3
Version 2.1

Warum man die Lesser GPL nicht für die nächste Bibliothek verwenden sollte

***Bitte beachten Sie für weitere allgemeine Empfehlungen über die Wahl einer Lizenz für Ihr Werk unseren** Lizenzratgeber.*

Das GNU-Projekt verfügt über zwei wesentliche Lizenzen zur Verwendung mit Bibliotheken. Eine ist die *GNU Lesser General Public License* (LGPL), die andere ist die normale*GNU General Public License* (GPL). Die Wahl der Lizenz macht einen großen Unterschied: mithilfe der GNU LGPL wird die Nutzung der Bibliothek in proprietären Programmen erlaubt, mithilfe der GNU GPL ist sie ausschließlich für freie Programme nutzbar.

Welche Lizenz am besten für eine bestimmte Bibliothek geeignet ist, ist eine Frage der Strategie und situationsabhängig. Derzeit werden die meisten GNU-Bibliotheken von der GNU LGPL abgedeckt, und das bedeutet, es wird nur eine dieser zwei Strategien verwendet, die andere wird vernachlässigt. Deshalb suchen wir nun weitere Bibliotheken, um sie **unter der normalen GNU GPL** freizugeben.

Entwickler proprietärer Software haben den Vorteil des Geldes; Entwickler freier Software müssen Vorteile für einander schaffen. Das Verwenden der GNU GPL für eine Bibliothek gibt Freie-Software-Entwicklern gegenüber proprietären Entwicklern einen Vorteil: eine Bibliothek, die sie nutzen können, während proprietäre Entwickler sie nicht nutzen können.

Die Verwendung der normalen GNU GPL ist nicht für jede Bibliothek vorteilhaft. Es gibt Gründe, in denen es in bestimmten Fällen besser sein kann, die GNU LGPL zu verwenden. Häufigster Fall ist, wenn die Funktionen einer freien Bibliothek für proprietäre Software durch andere Bibliotheken leicht zugänglich sind. In diesem Fall kann die Bibliothek freier Software keinen besonderen Vorteil geben, so dass es besser ist für diese Bibliothek die GNU LGPL zu verwenden.

Deshalb haben wir die GNU LGPL für die GNU C-Bibliothek eingesetzt. Immerhin gibt es viele andere C-Bibliotheken. Mithilfe der GNU GPL wären Entwickler proprietärer Software gezwungen gewesen, eine andere zu verwenden - kein Problem für sie, nur für uns.

Wenn eine Bibliothek jedoch eine signifikante einzigartige Fähigkeit enthält, wie beispielsweise **GNU Readline**, ist das etwas ganz anderes. Die Readline-Bibliothek implementiert Eingabeergänzung und -verlauf für interaktive Programme, und das ist eine woanders nicht allgemein verfügbare Funktion. Die Freigabe unter GNU GPL - und damit die ausschließliche Nutzung in freien Programmen - gibt unserer Gemeinschaft einen richtigen Schub. Mindestens ein Anwendungsprogramm ist heute insbesondere freie Software, weil das für die Nutzung von Readline notwendig war.

Wenn wir eine Auswahl leistungsfähiger, unter GPL lizenzierten Bibliotheken ansammeln, die keine parallelen für proprietäre Software haben, liefern sie eine Reihe nützlicher Module, die als Bausteine in neuen freien Programmen dienen. Dies wird ein signifikanter Vorteil für die weitere Entwicklung freier Software sein, und einige Projekte werden sich dazu entscheiden, Software frei zu machen, um diese Bibliotheken zu nutzen. Universitäre Projekte können leicht beeinflusst werden. Heutzutage, wo Unternehmen beginnen daran zu denken Software frei zu machen, können sogar einige kommerzielle Projekte auf diese Weise beeinflusst werden.

Entwickler proprietärer Software - bestrebt, dem freien Wettbewerb einen wichtigen Vorteil zu versagen - werden versuchen Autoren davon zu überzeugen, keine GPL-lizenzierten Bibliotheken der Sammlung beizutragen. Beispielsweise könnten sie an das Ego appellieren, wenn sie *mehr Benutzer für diese Bibliothek* versprechen, wenn wir sie den Quellcode in proprietären Softwareprodukten verwenden lassen. Popularität ist verlockend und für einen Entwickler einer Bibliothek die Idee leicht zu rationalisieren, dass die Erhöhung der Popularität dieser einen Bibliothek zu steigern das ist, was die Gemeinschaft vor allem braucht.

Aber wir sollten nicht auf diese Versuchungen hören, weil wir viel mehr erreichen können, wenn wir zusammenstehen. Wir Freie-Software-Entwickler sollten einander unterstützen. Durch die Freigabe von Bibliotheken, die nur auf freie Software begrenzt sind, helfen wir einander, dass die eigenen Freie-Software-Pakete die proprietären Pendants übertreffen. Die gesamte Freie-Software-Bewegung wird mehr Popularität haben, weil sich freie Software als Ganzes besser gegen den Wettbewerb abheben wird.

FAQ - Question and Answer

1.1. *Basic questions about the GNU Project, the Free Software Foundation, and its licenses*

2.1. *General understanding of the GNU licenses*

7.1. *Questions about violations of the GNU licenses*

1. Basic questions about the GNU Project, the Free Software Foundation, and its licenses

---

1.1. What does GPL stand for?
ssss
ssss

1.2. Does free software mean using the GPL?

1.3. Why should I use the GNU GPL rather than other free software licenses?

1.4. Does all GNU software use the GNU GPL as its license?

1.5. Does using the GPL for a program make it GNU software?

1.6. Can I use the GPL for something other than software?

1.7. Why don't you use the GPL for manuals?

1.8. Are there translations of the GPL into other languages?

1.9. Why are some GNU libraries released under the ordinary GPL rather than the Lesser GPL?

1.10. Who has the power to enforce the GPL?

1.11. Why does the FSF require that contributors to FSF-copyrighted programs assign copyright to the FSF? If I hold copyright on a GPLed program, should I do this, too? If so, how?

1.12. Can I modify the GPL and make a modified license?

1.13. Why did you decide to write the GNU Affero GPLv3 as a separate license?

2.1. *General understanding of the GNU licenses*

kinds of "modified versions"?

2.1.26. When it comes to determining whether two pieces of software form a single work, does the fact that the code is in one or more containers have any effect?

2.1.27. Why does the FSF require that contributors to FSF-copyrighted programs assign copyright to the FSF?
If I hold copyright on a GPLed program, should I do this, too?
If so, how?

2.1.28. If I use a piece of software that has been obtained under the GNU GPL, am I allowed to modify the original code into a new program, then distribute and sell that new program commercially?

2.1.29. Can I use the GPL for something other than software?

2.1.30. I'd like to license my code under the GPL, but I'd also like to make it clear that it can't be used for military and/or commercial uses.
Can I do this?

2.1.31. Can I use the GPL to license hardware?

2.1.32. Does prelinking a GPLed binary to various libraries on the system, to optimize its performance, count as modification?

2.1.33. How does the LGPL work with Java?

2.1.34. Why did you invent the new terms "propagate" and "convey" in GPLv3?

2.1.35. Is “convey” in GPLv3 the same thing as what GPLv2 means by "distribute"?

2.1.36. If I only make copies of a GPL-covered program and run them, without distributing or conveying them to others, what does the license require of me?

2.1.37. GPLv3 gives "making available to the public" as an example of propagation. What does this mean?
Is making available a form of conveying?

2.1.38. Since distribution and making available to the public are forms of propagation that are also conveying in GPLv3, what are some examples of propagation that do not constitute conveying?

2.1.39. How does GPLv3 make BitTorrent distribution easier?

2.1.40. What is tivoization? How does GPLv3 prevent it?

2.1.41. Does GPLv3 prohibit DRM?

2.1.42. Does GPLv3 require that voters be able to modify the software running in a voting machine?

2.1.43. Does GPLv3 have a "patent retaliation clause"?

2.1.44. In GPLv3 and AGPLv3, what does it mean when it says "notwithstanding any other provision of this License"?

2.1.45. In AGPLv3, what counts as "interacting with [the software] remotely through a computer network"?

2.1.46. How does GPLv3's concept of "you" compare to the definition of "Legal Entity" in the Apache License 2.0?

2.1.47. In GPLv3, what does "the Program" refer to?
Is it every program ever released under GPLv3?

2.1.48. If some network client software is released under AGPLv3, does it have to be able to provide source to the servers it interacts with?

2.1.49. For software that runs a proxy server licensed under the

AGPL, how can I provide an offer of source to users interacting with that code?

2.1.1. *Why does the GPL permit users to publish their modified versions?*

2.1.2. Does the GPL require that source code of modified versions be posted to the public?

2.1.3. Can I have a GPL-covered program and an unrelated nonfree program on the same computer?

2.1.4. If I know someone has a copy of a GPL-covered program, can I demand they give me a copy?

2.1.5. What does "written offer valid for any third party" mean in GPLv2? Does that mean everyone in the world can get the source to any GPLed program no matter what?

2.1.6. The GPL says that modified versions, if released, must be "licensed to all third parties." Who are these third parties?

2.1.7. Does the GPL allow me to sell copies of the program for money?

2.1.8. Does the GPL allow me to charge a fee for downloading the program from my distribution site?

2.1.9. Does the GPL allow me to require that anyone who receives the software must pay me a fee and/or notify me?

2.1.10. If I distribute GPLed software for a fee, am I required to also make it available to the public without a charge?

2.1.11. Does the GPL allow me to distribute a copy under a nondisclosure agreement?

2.1.12. Does the GPL allow me to distribute a modified or beta version under a nondisclosure agreement?

2.1.13. Does the GPL allow me to develop a modified version under a nondisclosure agreement?

2.1.14. Why does the GPL require including a copy of the GPL with every copy of the program?

2.1.15. What if the work is not very long?

2.1.16. Am I required to claim a copyright on my modifications to a GPL-covered program?

2.1.17. What does the GPL say about translating some code to a different programming language?

2.1.18. If a program combines public-domain code with GPL-covered code, can I take the public-domain part and use it as public domain code?

2.1.19. I want to get credit for my work. I want people to know what I wrote. Can I still get credit if I use the GPL?

2.1.20. Does the GPL allow me to add terms that would require citation or acknowledgment in research papers which use

3.1 How do I upgrade from (L)GPLv2 to (L)GPLv3?

3.2 Could you give me step by step instructions on how to apply the GPL to my program?

3.3 Why should I use the GNU GPL rather than other free software licenses?

3.4 Why does the GPL require including a copy of the GPL with every copy of the program?

3.5 Is putting a copy of the GNU GPL in my repository enough to apply the GPL?

3.6 Why should I put a license notice in each source file?

3.7 What if the work is not very long?

3.8 Can I omit the preamble of the GPL, or the instructions for how to use it on your own programs, to save space?

3.9 How do I get a copyright on my program in order to release it under the GPL?

3.10 What if my school might want to make my program into its own proprietary software product?

3.11 I would like to release a program I wrote under the GNU GPL, but I would like to use the same code in nonfree programs.

4. Distribution of programs released under the GNU licenses

4.1 Can I release a modified version of a GPL-covered program in binary form only?
4.2 I downloaded just the binary from the net. If I distribute copies, do I have to get the source and distribute that too?
4.3 I want to distribute binaries via physical media without accompanying sources. Can I provide source code by FTP instead of by mail order?
4.4 My friend got a GPL-covered binary with an offer to supply source, and made a copy for me. Can I use the offer to obtain the source?
4.5 Can I put the binaries on my Internet server and put the source on a different Internet site?
4.6 I want to distribute an extended version of a GPL-covered program in binary form. Is it enough to distribute the source for the original version?
4.7 I want to distribute binaries, but distributing complete source is inconvenient. Is it ok if I give users the diffs from the “standard” version along with the binaries?
4.8 Can I make binaries available on a network server, but send sources only to people who order them?
4.9 How can I make sure each user who downloads the binaries also gets the source?
4.10 Does the GPL require me to provide source code that can be built to match the exact hash of the binary I am distributing?
4.11 Can I release a program with a license which says that you can distribute modified versions of it under the GPL but you can't distribute the original itself under the GPL?
4.12 I just found out that a company has a copy of a GPLed program, and it costs money to get it. Aren't they violating the GPL by not making it available on the Internet?
4.13 A company is running a modified version of a GPLed program on a web site. Does the GPL say they must release their modified sources?
4.14 A company is running a modified version of a program licensed under the GNU Affero GPL (AGPL) on a web site. Does the AGPL say they must release their modified sources?
4.15 Is use within one organization or company “distribution”?
4.16 If someone steals a CD containing a version of a GPL-covered program, does the GPL give him the right to redistribute that version?
4.17 What if a company distributes a copy of some other developers' GPL-covered work to me as a trade secret?
4.18 What if a company distributes a copy of its own GPL-covered work to me as a trade secret?
4.19 Do I have “fair use” rights in using the source code of a GPL-covered program?
4.20 Does moving a copy to a majority-owned, and controlled, subsidiary constitute distribution?
4.21 Can software installers ask people to click to agree to the GPL? If I get some software under the GPL, do I have to agree to anything?
4.22 I would like to bundle GPLed software with some sort of installation software. Does that installer need to have a GPL-compatible license?
4.23 Does a distributor violate the GPL if they require me to “represent and warrant” that I am located in the US, or that I intend to distribute the software in compliance with relevant export control laws?
4.24 The beginning of GPLv3 section 6 says that I can convey a covered work in object code form “under the terms of sections 4 and 5” provided I also meet the conditions of section 6. What does that mean?
4.25 My company owns a lot of patents. Over the years we've contributed code to projects under “GPL version 2 or any later version”, and the project itself has been distributed under the same terms. If a user decides to take the project's code (incorporating my contributions) under GPLv3, does that mean I've automatically granted GPLv3's explicit patent license to that user?
4.26 If I distribute a GPLv3-covered program, can I provide a warranty that is voided if the user modifies the program?

4.27 If I give a copy of a GPLv3-covered program to a coworker at my company, have I “conveyed” the copy to that coworker?
4.28 Am I complying with GPLv3 if I offer binaries on an FTP server and sources by way of a link to a source code repository in a version control system, like CVS or Subversion?
4.29 Can someone who conveys GPLv3-covered software in a User Product use remote attestation to prevent a user from modifying that software?
4.30 What does “rules and protocols for communication across the network” mean in GPLv3?
4.31 Distributors that provide Installation Information under GPLv3 are not required to provide “support service” for the product. What kind of “support service” do you mean?

---

4. Distribution of programs released under the GNU licenses

4.1 Can I release a modified version of a GPL-covered program in binary form only?

4.2 I downloaded just the binary from the net. If I distribute copies, do I have to get the source and distribute that too?

4.3 I want to distribute binaries via physical media without accompanying sources. Can I provide source code by FTP instead of by mail order?

4.4 My friend got a GPL-covered binary with an offer to supply source, and made a copy for me. Can I use the offer to obtain the source?

4.5 Can I put the binaries on my Internet server and put the source on a different Internet site?

4.6 I want to distribute an extended version of a GPL-covered program in binary form. Is it enough to distribute the source for the original version?

4.7 I want to distribute binaries, but distributing complete source is inconvenient. Is it ok if I give users the diffs from the “standard” version along with the binaries?

4.8 Can I make binaries available on a network server, but send sources only to people who order them?

4.9 How can I make sure each user who downloads the binaries also gets the source?

4.10 Does the GPL require me to provide source code that can be built to match the exact hash of the binary I am distributing?

4.11 Can I release a program with a license which says that you can distribute modified versions of it under the GPL but you can't distribute the original itself under the GPL?

4.12 I just found out that a company has a copy of a GPLed program, and it costs money to get it. Aren't they violating the GPL by not making it available on the Internet?

4.13 A company is running a modified version of a GPLed program on a web site. Does the GPL say they must release their modified sources?

4.14 A company is running a modified version of a program licensed under the GNU Affero GPL (AGPL) on a web site. Does the AGPL say they must release their modified sources?

4.15 Is use within one organization or company “distribution”?

4.16 If someone steals a CD containing a version of a GPL-covered program, does the GPL give him the right to redistribute that version?

4.17 What if a company distributes a copy of some other developers' GPL-covered work to me as a trade secret?

5. Using programs released under the GNU licenses when writing other programs

---

5.1 Can I have a GPL-covered program and an unrelated nonfree program on the same computer?

5.2 Can I use GPL-covered editors such as GNU Emacs to develop nonfree programs?

5.3 Can I use GPL-covered tools such as GCC to compile them?

5.4 Is there some way that I can GPL the output people get from use of my program? For example, if my program is used to develop hardware designs, can I require that these designs must be free?

5.5 In what cases is the output of a GPL program covered by the GPL too?

5.6 If I port my program to GNU/Linux, does that mean I have to release it as free software under the GPL or some other free software license?

5.7 I'd like to incorporate GPL-covered software in my proprietary system. I have no permission to use that software except what the GPL gives me. Can I do this?

5.8 If I distribute a proprietary program that links against an LGPLv3-covered library that I've modified, what is the “contributor version” for purposes of determining the scope of the explicit patent license grant I'm making—is it just the library, or is it the whole combination?

5.9 Under AGPLv3, when I modify the Program under section 13, what Corresponding Source does it have to offer?

5.10 Where can I learn more about the GCC Runtime Library Exception?

---

6. Combining work with code released under the GNU licenses

6.1 Is GPLv3 compatible with GPLv2?

6.2 Does GPLv2 have a requirement about delivering installation information?

6.3 How are the various GNU licenses compatible with each other?

6.4 What is the difference between an “aggregate” and other kinds of “modified versions”?

6.5 Do I have “fair use” rights in using the source code of a GPL-covered program?

6.6 Can the US Government release improvements to a GPL-covered program?

6.7 Does the GPL have different requirements for statically vs dynamically linked modules with a covered work?

6.8 Does the LGPL have different requirements for statically vs dynamically linked modules with a covered work?

6.9 If a library is released under the GPL (not the LGPL), does that mean that any software which uses it has to be under the GPL or a GPL-compatible license?

6.10 You have a GPLed program that I'd like to link with my code to build a proprietary program. Does the fact that I link with your program mean I have to GPL my program?

6.11 If so, is there any chance I could get a license of your program under the Lesser GPL?

6.12 If a programming language interpreter is released under the GPL, does that mean programs written to be interpreted by it must be under GPL-compatible licenses?

6.13 If a programming language interpreter has a license that is incompatible with the GPL, can I run GPL-covered programs on it?

6.14 If I add a module to a GPL-covered program, do I have to use the GPL as the license for my module?

6.15 When is a program and its plug-ins considered a single combined program?

6.16 If I write a plug-in to use with a GPL-covered program, what requirements does that impose on the licenses I can use for distributing my plug-in?

6.17 Can I apply the GPL when writing a plug-in for a nonfree program?

6.18 Can I release a nonfree program that's designed to load a GPL-covered plug-in?

6.19 I'd like to incorporate GPL-covered software in my proprietary system. I have no permission to use that software except what the GPL gives me. Can I do this?

6.20 Using a certain GNU program under the GPL does not fit our project to make proprietary software. Will you make an exception for us? It would mean more users of that program.

6.21 I'd like to incorporate GPL-covered software in my proprietary system. Can I do this by putting a “wrapper” module, under a GPL-compatible lax permissive license (such as the X11 license) in between the GPL-covered part and the proprietary part?

6.22 Can I write free software that uses nonfree libraries?

6.23 Can I link a GPL program with a proprietary system library?

6.24 In what ways can I li^nk or combine AGPLv3-covered and GPLv3-covered code?

6.25 What legal issues come up if I use GPL-incompatible libraries with GPL software?

6.26 I'm writing a Windows application with Microsoft Visual C++ and I will be releasing it under the GPL. Is dynamically linking my program with the Visual C++ runtime library permitted under the GPL?

6.27 I'd like to modify GPL-covered programs and link them with the portability libraries from Money Guzzler Inc. I cannot distribute the source code for these libraries, so any user who wanted to change these versions would have to obtain those libraries separately. Why doesn't the GPL permit this?

6.28 If license for a module Q has a requirement that's incompatible with the GPL, but the requirement applies only when Q is distributed by itself, not when Q is included in a larger program, does that make the license GPL-compatible? Can I combine or link Q with a GPL-covered program?

6.29 In an object-oriented language such as Java, if I use a class that is GPLed without modifying, and subclass it, in what way does the GPL affect the larger program?

6.30 Does distributing a nonfree driver meant to link with the kernel Linux violate the GPL?

6.31 How can I allow linking of proprietary modules with my GPL-covered library under a controlled interface only?

6.32 Consider this situation: 1) X releases V1 of a project under the GPL. 2) Y contributes to the development of V2 with changes and new code based on V1. 3) X wants to convert V2 to a non-GPL license. Does X need Y's permission?

6.33 I have written an application that links with many different components, that have different licenses. I am very confused as to what licensing requirements are placed on my program. Can you please tell me what licenses I may use?

6.34 Can I use snippets of GPL-covered source code within documentation that is licensed under some license that is incompatible with the GPL?

7. Questions about violations of the GNU licenses

7.1 What should I do if I discover a possible violation of the GPL?

7.2 Who has the power to enforce the GPL?

7.3 I heard that someone got a copy of a GPLed program under another license. Is this possible?

7.4 Is the developer of a GPL-covered program bound by the GPL? Could the developer's actions ever be a violation of the GPL?

7.5 I just found out that a company has a copy of a GPLed program, and it costs money to get it. Aren't they violating the GPL by not making it available on the Internet?

7.6 Can I use GPLed software on a device that will stop operating if customers do not continue paying a subscription fee?

7.7 What does it mean to “cure” a violation of GPLv3?

7.8 If someone installs GPLed software on a laptop, and then lends that laptop to a friend without providing source code for the software, have they violated the GPL?

7.9 Suppose that two companies try to circumvent the requirement to provide Installation Information by having one company release signed software, and the other release a User Product that only runs signed software from the first company. Is this a violation of GPLv3?

Frequently Asked Questions about the GNU Licenses

Basic questions about the GNU Project, the Free Software Foundation, and its licenses

- What does “GPL” stand for?
- Does free software mean using the GPL?
- Why should I use the GNU GPL rather than other free software licenses?
- Does all GNU software use the GNU GPL as its license?
- Does using the GPL for a program make it GNU software?
- Can I use the GPL for something other than software?
- Why don't you use the GPL for manuals?
- Are there translations of the GPL into other languages?
- Why are some GNU libraries released under the ordinary GPL rather than the Lesser GPL?
- Who has the power to enforce the GPL?
- Why does the FSF require that contributors to FSF-copyrighted programs assign copyright to the FSF? If I hold copyright on a GPLed program, should I do this, too? If so, how?
- Can I modify the GPL and make a modified license?
- Why did you decide to write the GNU Affero GPLv3 as a separate license?

General understanding of the GNU licenses

- Why does the GPL permit users to publish their modified versions?
- Does the GPL require that source code of modified versions be posted to the public?
- Can I have a GPL-covered program and an unrelated nonfree program on the same computer?

- If I know someone has a copy of a GPL-covered program, can I demand they give me a copy?
- What does “written offer valid for any third party” mean in GPLv2? Does that mean everyone in the world can get the source to any GPLed program no matter what?
- The GPL says that modified versions, if released, must be “licensed … to all third parties.” Who are these third parties?
- Does the GPL allow me to sell copies of the program for money?
- Does the GPL allow me to charge a fee for downloading the program from my distribution site?
- Does the GPL allow me to require that anyone who receives the software must pay me a fee and/or notify me?
- If I distribute GPLed software for a fee, am I required to also make it available to the public without a charge?
- Does the GPL allow me to distribute a copy under a nondisclosure agreement?
- Does the GPL allow me to distribute a modified or beta version under a nondisclosure agreement?
- Does the GPL allow me to develop a modified version under a nondisclosure agreement?
- Why does the GPL require including a copy of the GPL with every copy of the program?
- What if the work is not very long?
- Am I required to claim a copyright on my modifications to a GPL-covered program?
- What does the GPL say about translating some code to a different programming language?
- If a program combines public-domain code with GPL-covered code, can I take the public-domain part and use it as public domain code?
- I want to get credit for my work. I want people to know what I wrote. Can I still get credit if I use the GPL?
- Does the GPL allow me to add terms that would require citation or acknowledgment in research papers which use the GPL-covered software or its output?
- Can I omit the preamble of the GPL, or the instructions for how to use it on your own programs, to save space?
- What does it mean to say that two licenses are “compatible”?
- What does it mean to say a license is “compatible with the GPL”?
- Why is the original BSD license incompatible with the GPL?

- What is the difference between an “aggregate” and other kinds of “modified versions”?
- When it comes to determining whether two pieces of software form a single work, does the fact that the code is in one or more containers have any effect?
- Why does the FSF require that contributors to FSF-copyrighted programs assign copyright to the FSF? If I hold copyright on a GPLed program, should I do this, too? If so, how?
- If I use a piece of software that has been obtained under the GNU GPL, am I allowed to modify the original code into a new program, then distribute and sell that new program commercially?
- Can I use the GPL for something other than software?
- I'd like to license my code under the GPL, but I'd also like to make it clear that it can't be used for military and/or commercial uses. Can I do this?
- Can I use the GPL to license hardware?
- Does prelinking a GPLed binary to various libraries on the system, to optimize its performance, count as modification?
- How does the LGPL work with Java?
- Why did you invent the new terms “propagate” and “convey” in GPLv3?
- Is “convey” in GPLv3 the same thing as what GPLv2 means by “distribute”?
- If I only make copies of a GPL-covered program and run them, without distributing or conveying them to others, what does the license require of me?
- GPLv3 gives “making available to the public” as an example of propagation. What does this mean? Is making available a form of conveying?
- Since distribution and making available to the public are forms of propagation that are also conveying in GPLv3, what are some examples of propagation that do not constitute conveying?
- How does GPLv3 make BitTorrent distribution easier?
- What is tivoization? How does GPLv3 prevent it?
- Does GPLv3 prohibit DRM?
- Does GPLv3 require that voters be able to modify the software running in a voting machine?
- Does GPLv3 have a “patent retaliation clause”?
- In GPLv3 and AGPLv3, what does it mean when it says “notwithstanding any other provision of this License”?
- In AGPLv3, what counts as “ interacting with [the software] remotely through a computer network?”

- How does GPLv3's concept of “you” compare to the definition of “Legal Entity” in the Apache License 2.0?
- In GPLv3, what does “the Program” refer to? Is it every program ever released under GPLv3?
- If some network client software is released under AGPLv3, does it have to be able to provide source to the servers it interacts with?
- For software that runs a proxy server licensed under the AGPL, how can I provide an offer of source to users interacting with that code?

Using GNU licenses for your programs

- How do I upgrade from (L)GPLv2 to (L)GPLv3?
- Could you give me step by step instructions on how to apply the GPL to my program?
- Why should I use the GNU GPL rather than other free software licenses?
- Why does the GPL require including a copy of the GPL with every copy of the program?
- Is putting a copy of the GNU GPL in my repository enough to apply the GPL?
- Why should I put a license notice in each source file?
- What if the work is not very long?
- Can I omit the preamble of the GPL, or the instructions for how to use it on your own programs, to save space?
- How do I get a copyright on my program in order to release it under the GPL?
- What if my school might want to make my program into its own proprietary software product?
- I would like to release a program I wrote under the GNU GPL, but I would like to use the same code in nonfree programs.
- Can the developer of a program who distributed it under the GPL later license it to another party for exclusive use?
- Can the US Government release a program under the GNU GPL?
- Can the US Government release improvements to a GPL-covered program?
- Why should programs say “Version 3 of the GPL or any later version”?
- Is it a good idea to use a license saying that a certain program can be used only under the latest version of the GNU GPL?
- Is there some way that I can GPL the output people get from use of my program? For example, if my program is used to develop hardware designs, can I require that these designs must be free?

- Why don't you use the GPL for manuals?
- How does the GPL apply to fonts?
- What license should I use for website maintenance system templates?
- Can I release a program under the GPL which I developed using nonfree tools?
- I use public key cryptography to sign my code to assure its authenticity. Is it true that GPLv3 forces me to release my private signing keys?
- Does GPLv3 require that voters be able to modify the software running in a voting machine?
- The warranty and liability disclaimers in GPLv3 seem specific to U.S. law. Can I add my own disclaimers to my own code?
- My program has interactive user interfaces that are non-visual in nature. How can I comply with the Appropriate Legal Notices requirement in GPLv3?

Distribution of programs released under the GNU licenses

- Can I release a modified version of a GPL-covered program in binary form only?
- I downloaded just the binary from the net. If I distribute copies, do I have to get the source and distribute that too?
- I want to distribute binaries via physical media without accompanying sources. Can I provide source code by FTP instead of by mail order?
- My friend got a GPL-covered binary with an offer to supply source, and made a copy for me. Can I use the offer to obtain the source?
- Can I put the binaries on my Internet server and put the source on a different Internet site?
- I want to distribute an extended version of a GPL-covered program in binary form. Is it enough to distribute the source for the original version?
- I want to distribute binaries, but distributing complete source is inconvenient. Is it ok if I give users the diffs from the “standard” version along with the binaries?
- Can I make binaries available on a network server, but send sources only to people who order them?
- How can I make sure each user who downloads the binaries also gets the source?
- Does the GPL require me to provide source code that can be built to match the exact hash of the binary I am distributing?
- Can I release a program with a license which says that you can distribute modified versions of it under the GPL but you can't distribute the original itself under the GPL?

- I just found out that a company has a copy of a GPLed program, and it costs money to get it. Aren't they violating the GPL by not making it available on the Internet?
- A company is running a modified version of a GPLed program on a web site. Does the GPL say they must release their modified sources?
- A company is running a modified version of a program licensed under the GNU Affero GPL (AGPL) on a web site. Does the AGPL say they must release their modified sources?
- Is use within one organization or company “distribution”?
- If someone steals a CD containing a version of a GPL-covered program, does the GPL give him the right to redistribute that version?
- What if a company distributes a copy of some other developers' GPL-covered work to me as a trade secret?
- What if a company distributes a copy of its own GPL-covered work to me as a trade secret?
- Do I have “fair use” rights in using the source code of a GPL-covered program?
- Does moving a copy to a majority-owned, and controlled, subsidiary constitute distribution?
- Can software installers ask people to click to agree to the GPL? If I get some software under the GPL, do I have to agree to anything?
- I would like to bundle GPLed software with some sort of installation software. Does that installer need to have a GPL-compatible license?
- Does a distributor violate the GPL if they require me to “represent and warrant” that I am located in the US, or that I intend to distribute the software in compliance with relevant export control laws?
- The beginning of GPLv3 section 6 says that I can convey a covered work in object code form “under the terms of sections 4 and 5” provided I also meet the conditions of section 6. What does that mean?
- My company owns a lot of patents. Over the years we've contributed code to projects under “GPL version 2 or any later version”, and the project itself has been distributed under the same terms. If a user decides to take the project's code (incorporating my contributions) under GPLv3, does that mean I've automatically granted GPLv3's explicit patent license to that user?
- If I distribute a GPLv3-covered program, can I provide a warranty that is voided if the user modifies the program?
- If I give a copy of a GPLv3-covered program to a coworker at my company, have I “conveyed” the copy to that coworker?
- Am I complying with GPLv3 if I offer binaries on an FTP server and sources by way of a link to a source code repository in a version control system, like CVS or Subversion?

- Can someone who conveys GPLv3-covered software in a User Product use remote attestation to prevent a user from modifying that software?
- What does “rules and protocols for communication across the network” mean in GPLv3?
- Distributors that provide Installation Information under GPLv3 are not required to provide “support service” for the product. What kind of “support service” do you mean?

Using programs released under the GNU licenses when writing other programs

- Can I have a GPL-covered program and an unrelated nonfree program on the same computer?
- Can I use GPL-covered editors such as GNU Emacs to develop nonfree programs? Can I use GPL-covered tools such as GCC to compile them?
- Is there some way that I can GPL the output people get from use of my program? For example, if my program is used to develop hardware designs, can I require that these designs must be free?
- In what cases is the output of a GPL program covered by the GPL too?
- If I port my program to GNU/Linux, does that mean I have to release it as free software under the GPL or some other free software license?
- I'd like to incorporate GPL-covered software in my proprietary system. I have no permission to use that software except what the GPL gives me. Can I do this?
- If I distribute a proprietary program that links against an LGPLv3-covered library that I've modified, what is the “contributor version” for purposes of determining the scope of the explicit patent license grant I'm making—is it just the library, or is it the whole combination?
- Under AGPLv3, when I modify the Program under section 13, what Corresponding Source does it have to offer?
- Where can I learn more about the GCC Runtime Library Exception?

Combining work with code released under the GNU licenses

- Is GPLv3 compatible with GPLv2?
- Does GPLv2 have a requirement about delivering installation information?
- How are the various GNU licenses compatible with each other?
- What is the difference between an “aggregate” and other kinds of “modified versions”?
- Do I have “fair use” rights in using the source code of a GPL-covered program?

- Can the US Government release improvements to a GPL-covered program?
- Does the GPL have different requirements for statically vs dynamically linked modules with a covered work?
- Does the LGPL have different requirements for statically vs dynamically linked modules with a covered work?
- If a library is released under the GPL (not the LGPL), does that mean that any software which uses it has to be under the GPL or a GPL-compatible license?
- You have a GPLed program that I'd like to link with my code to build a proprietary program. Does the fact that I link with your program mean I have to GPL my program?
- If so, is there any chance I could get a license of your program under the Lesser GPL?
- If a programming language interpreter is released under the GPL, does that mean programs written to be interpreted by it must be under GPL-compatible licenses?
- If a programming language interpreter has a license that is incompatible with the GPL, can I run GPL-covered programs on it?
- If I add a module to a GPL-covered program, do I have to use the GPL as the license for my module?
- When is a program and its plug-ins considered a single combined program?
- If I write a plug-in to use with a GPL-covered program, what requirements does that impose on the licenses I can use for distributing my plug-in?
- Can I apply the GPL when writing a plug-in for a nonfree program?
- Can I release a nonfree program that's designed to load a GPL-covered plug-in?
- I'd like to incorporate GPL-covered software in my proprietary system. I have no permission to use that software except what the GPL gives me. Can I do this?
- Using a certain GNU program under the GPL does not fit our project to make proprietary software. Will you make an exception for us? It would mean more users of that program.
- I'd like to incorporate GPL-covered software in my proprietary system. Can I do this by putting a “wrapper” module, under a GPL-compatible lax permissive license (such as the X11 license) in between the GPL-covered part and the proprietary part?
- Can I write free software that uses nonfree libraries?
- Can I link a GPL program with a proprietary system library?
- In what ways can I link or combine AGPLv3-covered and GPLv3-covered

code?

- What legal issues come up if I use GPL-incompatible libraries with GPL software?
- I'm writing a Windows application with Microsoft Visual C++ and I will be releasing it under the GPL. Is dynamically linking my program with the Visual C++ runtime library permitted under the GPL?
- I'd like to modify GPL-covered programs and link them with the portability libraries from Money Guzzler Inc. I cannot distribute the source code for these libraries, so any user who wanted to change these versions would have to obtain those libraries separately. Why doesn't the GPL permit this?
- If license for a module Q has a requirement that's incompatible with the GPL, but the requirement applies only when Q is distributed by itself, not when Q is included in a larger program, does that make the license GPL-compatible? Can I combine or link Q with a GPL-covered program?
- In an object-oriented language such as Java, if I use a class that is GPLed without modifying, and subclass it, in what way does the GPL affect the larger program?
- Does distributing a nonfree driver meant to link with the kernel Linux violate the GPL?
- How can I allow linking of proprietary modules with my GPL-covered library under a controlled interface only?
- Consider this situation: 1) X releases V1 of a project under the GPL. 2) Y contributes to the development of V2 with changes and new code based on V1. 3) X wants to convert V2 to a non-GPL license. Does X need Y's permission?
- I have written an application that links with many different components, that have different licenses. I am very confused as to what licensing requirements are placed on my program. Can you please tell me what licenses I may use?
- Can I use snippets of GPL-covered source code within documentation that is licensed under some license that is incompatible with the GPL?

Questions about violations of the GNU licenses

- What should I do if I discover a possible violation of the GPL?
- Who has the power to enforce the GPL?
- I heard that someone got a copy of a GPLed program under another license. Is this possible?
- Is the developer of a GPL-covered program bound by the GPL? Could the developer's actions ever be a violation of the GPL?
- I just found out that a company has a copy of a GPLed program, and it costs money to get it. Aren't they violating the GPL by not making it

available on the Internet?

- Can I use GPLed software on a device that will stop operating if customers do not continue paying a subscription fee?
- What does it mean to “cure” a violation of GPLv3?
- If someone installs GPLed software on a laptop, and then lends that laptop to a friend without providing source code for the software, have they violated the GPL?
- Suppose that two companies try to circumvent the requirement to provide Installation Information by having one company release signed software, and the other release a User Product that only runs signed software from the first company. Is this a violation of GPLv3?

What does “GPL” stand for? (#WhatDoesGPLStandFor)

“GPL” stands for “General Public License”. The most widespread such license is the GNU General Public License, or GNU GPL for short. This can be further shortened to “GPL”, when it is understood that the GNU GPL is the one intended.

Does free software mean using the GPL? (#DoesFreeSoftwareMeanUsingTheGPL)

Not at all—there are many other free software licenses. We have an incomplete list. Any license that provides the user certain specific freedoms is a free software license.

Why should I use the GNU GPL rather than other free software licenses? (#WhyUseGPL)

Using the GNU GPL will require that all the released improved versions be free software. This means you can avoid the risk of having to compete with a proprietary modified version o your own work. However, in some special situations it can be better to use a more permissive license.

Does all GNU software use the GNU GPL as its license? (#DoesAllGNUSoftwareUseTheGNUGPLAsItsLicense)

Most GNU software packages use the GNU GPL, but there are a few GNU programs (and parts of programs) that use looser licenses, such as the Lesser GPL. When we do this, it is a matter of strategy.

Does using the GPL for a program make it GNU software? (#DoesUsingTheGPLForAProgramMakeItGNUSoftware)

Anyone can release a program under the GNU GPL, but that does not make it a GNU package.

Making the program a GNU software package means explicitly contributing to the GNU Project. This happens when the program's developers and the GNU Project agree to do it. If you are interested in contributing a program to the GNU Project, please write to <maintainers@gnu.org>.

What should I do if I discover a possible violation of the GPL? (#ReportingViolation)

You should report it. First, check the facts as best you can. Then tell the publisher or copyright holder of the specific GPL-covered program. If that is the Free Software Foundation, write to <license-violation@gnu.org>. Otherwise, the program's maintainer may be the copyright holder, or else could tell you how to contact the copyright holder, so report it to the maintainer.

Why does the GPL permit users to publish their modified versions? (#WhyDoesTheGPLPermitUsersToPublishTheirModifiedVersions)

A crucial aspect of free software is that users are free to cooperate. It is absolutely essential to permit users who wish to help each other to share their bug fixes and improvements with other users.

Some have proposed alternatives to the GPL that require modified versions to go through the original author. As long as the original author keeps up with the need for maintenance, this may work well in practice, but if the author stops (more or less) to do something else or does not attend to all the users' needs, this scheme falls down. Aside from the practical problems, this scheme does not allow users to help each other.

Sometimes control over modified versions is proposed as a means of preventing confusion between various versions made by users. In our experience, this confusion is not a major problem. Many versions of Emacs have been made outside the GNU Project, but users can tell them apart. The GPL requires the maker of a version to place his or her name on it, to distinguish it from other versions and to protect the reputations of other maintainers.

Does the GPL require that source code of modified versions be posted to the public? (#GPLRequireSourcePostedPublic)

The GPL does not require you to release your modified version, or any part of it. You are free to make modifications and use them privately, without ever releasing them. This applies to organizations (including companies), too; an organization can make a modified version and use it

internally without ever releasing it outside the organization.

But *if* you release the modified version to the public in some way, the GPL requires you to make the modified source code available to the program's users, under the GPL.

Thus, the GPL gives permission to release the modified program in certain ways, and not in other ways; but the decision of whether to release it is up to you.

Can I have a GPL-covered program and an unrelated nonfree program on the same computer? (#GPLAndNonfreeOnSameMachine)

Yes.

If I know someone has a copy of a GPL-covered program, can I demand they give me a copy? (#CanIDemandACopy)

No. The GPL gives a person permission to make and redistribute copies of the program *if and when that person chooses to do so*. That person also has the right not to choose to redistribute the program.

What does “written offer valid for any third party” mean in GPLv2? Does that mean everyone in the world can get the source to any GPLed program no matter what? (#WhatDoesWrittenOfferValid)

If you choose to provide source through a written offer, then anybody who requests the source from you is entitled to receive it.

If you commercially distribute binaries not accompanied with source code, the GPL says you

must provide a written offer to distribute the source code later. When users non-commercially redistribute the binaries they received from you, they must pass along a copy of this written offer. This means that people who did not get the binaries directly from you can still receive copies of the source code, along with the written offer.

The reason we require the offer to be valid for any third party is so that people who receive the binaries indirectly in that way can order the source code from you.

GPLv2 says that modified versions, if released, must be "licensed … to all third parties." Who are these third parties? (#TheGPLSaysModifiedVersions)

Section 2 says that modified versions you distribute must be licensed to all third parties under the GPL. "All third parties" means absolutely everyone—but this does not require you to *do* anything physically for them. It only means they have a license from you, under the GPL, for your version.

Am I required to claim a copyright on my modifications to a GPL-covered program? (#RequiredToClaimCopyright)

You are not required to claim a copyright on your changes. In most countries, however, that happens automatically by default, so you need to place your changes explicitly in the public domain if you do not want them to be copyrighted.

Whether you claim a copyright on your changes or not, either way you must release the modified version, as a whole, under the GPL (if you release your modified version at all).

What does the GPL say about translating some code to a different programming language? (#TranslateCode)

Under copyright law, translation of a work is considered a kind of modification. Therefore, what the GPL says about modified versions applies also to translated versions. The translation is covered by the copyright on the original program.

If the original program carries a free license, that license gives permission to translate it. How you can use and license the translated program is determined by that license. If the original program is licensed under certain versions of the GNU GPL, the translated program must be covered by the same versions of the GNU GPL.

If a program combines public-domain code with GPL-covered code, can I take the public-domain part and use it as public domain code? (#CombinePublicDomainWithGPL)

You can do that, if you can figure out which part is the public domain part and separate it from the rest. If code was put in the public domain by its developer, it is in the public domain no matter where it has been.

Does the GPL allow me to sell copies of the program for money? (#DoesTheGPLAllowMoney)

Yes, the GPL allows everyone to do this. The right to sell copies is part of the definition of free software. Except in one special situation, there is no limit on what price you can charge. (The one exception is the required written offer to provide source code that must accompany binary-only release.)

Does the GPL allow me to charge a fee for downloading the program from my distribution site? (#DoesTheGPLAllowDownloadFee)

Yes. You can charge any fee you wish for distributing a copy of the program. Under GPLv2, i

you distribute binaries by download, you must provide “equivalent access” to download the source—therefore, the fee to download source may not be greater than the fee to download the binary. If the binaries being distributed are licensed under the GPLv3, then you must offer equivalent access to the source code in the same way through the same place at no further charge.

Does the GPL allow me to require that anyone who receives the software must pay me a fee and/or notify me? (#DoesTheGPLAllowRequireFee)

No. In fact, a requirement like that would make the program nonfree. If people have to pay when they get a copy of a program, or if they have to notify anyone in particular, then the program is not free. See the definition of free software.

The GPL is a free software license, and therefore it permits people to use and even redistribute the software without being required to pay anyone a fee for doing so.

You *can* charge people a fee to get a copy *from you*. You can't require people to pay you when they get a copy *from someone else.*

If I distribute GPLed software for a fee, am I required to also make it available to the public without a charge? (#DoesTheGPLRequireAvailabilityToPublic)

No. However, if someone pays your fee and gets a copy, the GPL gives them the freedom to release it to the public, with or without a fee. For example, someone could pay your fee, and then put her copy on a web site for the general public.

Does the GPL allow me to distribute copies under a nondisclosure agreement? (#DoesTheGPLAllowNDA)

No. The GPL says that anyone who receives a copy from you has the right to redistribute copies, modified or not. You are not allowed to distribute the work on any more restrictive basis.

If someone asks you to sign an NDA for receiving GPL-covered software copyrighted by the FSF, please inform us immediately by writing to license-violation@fsf.org.

If the violation involves GPL-covered code that has some other copyright holder, please inform that copyright holder, just as you would for any other kind of violation of the GPL.

Does the GPL allow me to distribute a modified or beta version under a nondisclosure agreement? (#DoesTheGPLAllowModNDA)

No. The GPL says that your modified versions must carry all the freedoms stated in the GPL. Thus, anyone who receives a copy of your version from you has the right to redistribute copies (modified or not) of that version. You may not distribute any version of the work on a more restrictive basis.

Does the GPL allow me to develop a modified version under a nondisclosure agreement? (#DevelopChangesUnderNDA)

Yes. For instance, you can accept a contract to develop changes and agree not to release *your changes* until the client says ok. This is permitted because in this case no GPL-covered code is being distributed under an NDA.

You can also release your changes to the client under the GPL, but agree not to release them to anyone else unless the client says ok. In this case, too, no GPL-covered code is being distributed under an NDA, or under any additional restrictions.

The GPL would give the client the right to redistribute your version. In this scenario, the client will probably choose not to exercise that right, but does *have* the right.

I want to get credit for my work. I want people to know what I wrote. Can I still get credit if I use the GPL? (#IWantCredit)

You can certainly get credit for the work. Part of releasing a program under the GPL is writing a copyright notice in your own name (assuming you are the copyright holder). The GPL requires all copies to carry an appropriate copyright notice.

Does the GPL allow me to add terms that would require citation or acknowledgment in research papers which use the GPL-covered software or its output? (#RequireCitation)

No, this is not permitted under the terms of the GPL. While we recognize that proper citation is an important part of academic publications, citation cannot be added as an additional requirement to the GPL. Requiring citation in research papers which made use of GPLed software goes beyond what would be an acceptable additional requirement under section 7(b) of GPLv3, and therefore would be considered an additional restriction under Section 7 of the GPL. And copyright law does not allow you to place such a requirement on the output of software, regardless of whether it is licensed under the terms of the GPL or some other license.

Why does the GPL require including a copy of the GPL with every copy of the program? (#WhyMustIInclude)

Including a copy of the license with the work is vital so that everyone who gets a copy of the program can know what their rights are.

It might be tempting to include a URL that refers to the license, instead of the license itself. But you cannot be sure that the URL will still be valid, five years or ten years from now. Twenty

years from now, URLs as we know them today may no longer exist.

The only way to make sure that people who have copies of the program will continue to be able to see the license, despite all the changes that will happen in the network, is to include a copy the license in the program.

Is it enough just to put a copy of the GNU GPL in my repository? (#LicenseCopyOnly)

Just putting a copy of the GNU GPL in a file in your repository does not explicitly state that the code in the same repository may be used under the GNU GPL. Without such a statement, it's not entirely clear that the permissions in the license really apply to any particular source file. An explicit statement saying that eliminates all doubt.

A file containing just a license, without a statement that certain other files are covered by that license, resembles a file containing just a subroutine which is never called from anywhere else. The resemblance is not perfect: lawyers and courts might apply common sense and conclude that you must have put the copy of the GNU GPL there because you wanted to license the code that way. Or they might not. Why leave an uncertainty?

This statement should be in each source file. A clear statement in the program's README file is legally sufficient *as long as that accompanies the code,* but it is easy for them to get separated. Why take a risk of uncertainty about your code's license?

This has nothing to do with the specifics of the GNU GPL. It is true for any free license.

Why should I put a license notice in each source file? (#NoticeInSourceFile)

You should put a notice at the start of each source file, stating what license it carries, in order to avoid risk of the code's getting disconnected from its license. If your repository's README says that source file is under the GNU GPL, what happens if someone copies that file to another

program? That other context may not show what the file's license is. It may appear to have some other license, or no license at all (which would make the code nonfree).

Adding a copyright notice and a license notice at the start of each source file is easy and makes such confusion unlikely.

This has nothing to do with the specifics of the GNU GPL. It is true for any free license.

What if the work is not very long? (#WhatIfWorkIsShort)

If a whole software package contains very little code—less than 300 lines is the benchmark we use—you may as well use a lax permissive license for it, rather than a copyleft license like the GNU GPL. (Unless, that is, the code is specially important.) We recommend the Apache License 2.0 for such cases.

Can I omit the preamble of the GPL, or the instructions for how to use it on your own programs, to save space? (#GPLOmitPreamble)

The preamble and instructions are integral parts of the GNU GPL and may not be omitted. In fact, the GPL is copyrighted, and its license permits only verbatim copying of the entire GPL. (You can use the legal terms to make another license but it won't be the GNU GPL.)

The preamble and instructions add up to some 1000 words, less than 1/5 of the GPL's total size. They will not make a substantial fractional change in the size of a software package unless the package itself is quite small. In that case, you may as well use a simple all-permissive license rather than the GNU GPL.

What does it mean to say that two licenses are “compatible”? (#WhatIsCompatible)

In order to combine two programs (or substantial parts of them) into a larger work, you need to have permission to use both programs in this way. If the two programs' licenses permit this, they are compatible. If there is no way to satisfy both licenses at once, they are incompatible.

For some licenses, the way in which the combination is made may affect whether they are compatible—for instance, they may allow linking two modules together, but not allow merging their code into one module.

If you just want to install two separate programs in the same system, it is not necessary that their licenses be compatible, because this does not combine them into a larger work.

What does it mean to say a license is "compatible with the GPL?" (#WhatDoesCompatMean)

It means that the other license and the GNU GPL are compatible; you can combine code released under the other license with code released under the GNU GPL in one larger program.

All GNU GPL versions permit such combinations privately; they also permit distribution of such combinations provided the combination is released under the same GNU GPL version. The other license is compatible with the GPL if it permits this too.

GPLv3 is compatible with more licenses than GPLv2: it allows you to make combinations with code that has specific kinds of additional requirements that are not in GPLv3 itself. Section 7 has more information about this, including the list of additional requirements that are permitted.

Can I write free software that uses nonfree libraries? (#FSWithNFLibs)

If you do this, your program won't be fully usable in a free environment. If your program depends on a nonfree library to do a certain job, it cannot do that job in the Free World. If it depends on a nonfree library to run at all, it cannot be part of a free operating system such as GNU; it is entirely off limits to the Free World.

So please consider: can you find a way to get the job done without using this library? Can you write a free replacement for that library?

If the program is already written using the nonfree library, perhaps it is too late to change the decision. You may as well release the program as it stands, rather than not release it. But please mention in the README that the need for the nonfree library is a drawback, and suggest the task of changing the program so that it does the same job without the nonfree library. Please suggest that anyone who thinks of doing substantial further work on the program first free it from dependence on the nonfree library.

Note that there may also be legal issues with combining certain nonfree libraries with GPL-covered free software. Please see the question on GPL software with GPL-incompatible libraries for more information.

Can I link a GPL program with a proprietary system library? (#SystemLibraryException)

Both versions of the GPL have an exception to their copyleft, commonly called the system library exception. If the GPL-incompatible libraries you want to use meet the criteria for a system library, then you don't have to do anything special to use them; the requirement to distribute source code for the whole program does not include those libraries, even if you distribute a linked executable containing them.

The criteria for what counts as a “system library” vary between different versions of the GPL. GPLv3 explicitly defines “System Libraries” in section 1, to exclude it from the definition of “Corresponding Source.” GPLv2 deals with this issue slightly differently, near the end of section 3.

In what ways can I link or combine AGPLv3-covered and GPLv3-covered code? (#AGPLGPL)

Each of these licenses explicitly permits linking with code under the other license. You can

always link GPLv3-covered modules with AGPLv3-covered modules, and vice versa. That is true regardless of whether some of the modules are libraries.

What legal issues come up if I use GPL-incompatible libraries with GPL software? (#GPLIncompatibleLibs)

If you want your program to link against a library not covered by the system library exception, you need to provide permission to do that. Below are two example license notices that you can use to do that; one for GPLv3, and the other for GPLv2. In either case, you should put this text in each file to which you are granting this permission.

Only the copyright holders for the program can legally release their software under these terms. If you wrote the whole program yourself, then assuming your employer or school does not claim the copyright, you are the copyright holder—so you can authorize the exception. But if you want to use parts of other GPL-covered programs by other authors in your code, you cannot authorize the exception for them. You have to get the approval of the copyright holders of those programs.

When other people modify the program, they do not have to make the same exception for their code—it is their choice whether to do so.

If the libraries you intend to link with are nonfree, please also see the section on writing Free Software which uses nonfree libraries.

If you're using GPLv3, you can accomplish this goal by granting an additional permission under section 7. The following license notice will do that. You must replace all the text in brackets with text that is appropriate for your program. If not everybody can distribute source for the libraries you intend to link with, you should remove the text in braces; otherwise, just remove the braces themselves.

Copyright (C) [years] [name of copyright holder]

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 3 of the License, 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.

You should have received a copy of the GNU General Public License along with this program; if not, see <https://www.gnu.org/licenses>.

Additional permission under GNU GPL version 3 section 7

If you modify this Program, or any covered work, by linking or combining it with [name of library] (or a modified version of that library), containing parts covered by the terms of [name of library's license], the licensors of this Program grant you additional permission to convey the resulting work. {Corresponding Source for a non-source form of such a combination shall include the source code for the parts of [name of library] used as well as that of the covered work.}

If you're using GPLv2, you can provide your own exception to the license's terms. The following license notice will do that. Again, you must replace all the text in brackets with text that is appropriate for your program. If not everybody can distribute source for the libraries you intend to link with, you should remove the text in braces; otherwise, just remove the braces themselves.

Copyright (C) [years] [name of copyright holder]

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 of the License, 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.

You should have received a copy of the GNU General Public License along with this program; if not, see <https://www.gnu.org/licenses>.

Linking [name of your program] statically or dynamically with other modules is making a combined work based on [name of your program]. Thus, the terms and conditions of the GNU General Public License cover the whole combination.

In addition, as a special exception, the copyright holders of [name of your program] give you permission to combine [name of your program] with free software programs or libraries that are released under the GNU LGPL and with code included in the standard release of [name of library] under the [name of library's license] (or modified versions of such code, with unchanged license). You may copy and distribute such a system following the terms of the GNU GPL for [name of your program] and the licenses of the other code concerned{, provided that you include the source code of that other code when and as the GNU GPL requires distribution of source code}.

Note that people who make modified versions of [name of your program] are not obligated to grant this special exception for their modified versions; it is their choice whether to do so. The GNU General Public License gives permission to release a modified version without this exception; this exception also makes it possible to release a modified version which carries forward this exception.

How do I get a copyright on my program in order to release it under the GPL? (#HowIGetCopyright)

Under the Berne Convention, everything written is automatically copyrighted from whenever it is put in fixed form. So you don't have to do anything to “get” the copyright on what you write—as long as nobody else can claim to own your work.

However, registering the copyright in the US is a very good idea. It will give you more clout in dealing with an infringer in the US.

The case when someone else might possibly claim the copyright is if you are an employee or student; then the employer or the school might claim you did the job for them and that the copyright belongs to them. Whether they would have a valid claim would depend on circumstances such as the laws of the place where you live, and on your employment contract and what sort of work you do. It is best to consult a lawyer if there is any possible doubt.

If you think that the employer or school might have a claim, you can resolve the problem clearly by getting a copyright disclaimer signed by a suitably authorized officer of the company or school. (Your immediate boss or a professor is usually NOT authorized to sign such a disclaimer.)

What if my school might want to make my program into its own proprietary software product? (#WhatIfSchool)

Many universities nowadays try to raise funds by restricting the use of the knowledge and information they develop, in effect behaving little different from commercial businesses. (See “The Kept University”, Atlantic Monthly, March 2000, for a general discussion of this problem and its effects.)

If you see any chance that your school might refuse to allow your program to be released as free software, it is best to raise the issue at the earliest possible stage. The closer the program is to working usefully, the more temptation the administration might feel to take it from you and finish it without you. At an earlier stage, you have more leverage.

So we recommend that you approach them when the program is only half-done, saying, “If you will agree to releasing this as free software, I will finish it.” Don't think of this as a bluff. To prevail, you must have the courage to say, “My program will have liberty, or never be born.”

Could you give me step by step instructions on how to apply the GPL to my program? (#CouldYouHelpApplyGPL)

See the page of GPL instructions.

I heard that someone got a copy of a GPLed program under another license. Is this possible? (#HeardOtherLicense)

The GNU GPL does not give users permission to attach other licenses to the program. But the copyright holder for a program can release it under several different licenses in parallel. One o them may be the GNU GPL.

The license that comes in your copy, assuming it was put in by the copyright holder and that you got the copy legitimately, is the license that applies to your copy.

I would like to release a program I wrote under the GNU GPL, but I would like to use the same code in nonfree programs. (#ReleaseUnderGPLAndNF)

To release a nonfree program is always ethically tainted, but legally there is no obstacle to your doing this. If you are the copyright holder for the code, you can release it under various different non-exclusive licenses at various times.

Is the developer of a GPL-covered program bound by the GPL? Could the developer's actions ever be a violation of the GPL? (#DeveloperViolate)

Strictly speaking, the GPL is a license from the developer for others to use, distribute and change the program. The developer itself is not bound by it, so no matter what the developer does, this is not a “violation” of the GPL.

However, if the developer does something that would violate the GPL if done by someone else,

the developer will surely lose moral standing in the community.

Can the developer of a program who distributed it under the GPL later license it to another party for exclusive use? (#CanDeveloperThirdParty)

No, because the public already has the right to use the program under the GPL, and this right cannot be withdrawn.

Can I use GPL-covered editors such as GNU Emacs to develop nonfree programs? Can I use GPL-covered tools such as GCC to compile them? (#CanIUseGPLToolsForNF)

Yes, because the copyright on the editors and tools does not cover the code you write. Using them does not place any restrictions, legally, on the license you use for your code.

Some programs copy parts of themselves into the output for technical reasons—for example, Bison copies a standard parser program into its output file. In such cases, the copied text in the output is covered by the same license that covers it in the source code. Meanwhile, the part of the output which is derived from the program's input inherits the copyright status of the input.

As it happens, Bison can also be used to develop nonfree programs. This is because we decided to explicitly permit the use of the Bison standard parser program in Bison output files without restriction. We made the decision because there were other tools comparable to Bison which already permitted use for nonfree programs.

Do I have “fair use” rights in using the source code of a GPL-covered program? (#GPLFairUse)

Yes, you do. “Fair use” is use that is allowed without any special permission. Since you don't need the developers' permission for such use, you can do it regardless of what the developers

said about it—in the license or elsewhere, whether that license be the GNU GPL or any other free software license.

Note, however, that there is no world-wide principle of fair use; what kinds of use are considered “fair” varies from country to country.

Can the US Government release a program under the GNU GPL? (#GPLUSGov)

If the program is written by US federal government employees in the course of their employment, it is in the public domain, which means it is not copyrighted. Since the GNU GPL is based on copyright, such a program cannot be released under the GNU GPL. (It can still be free software, however; a public domain program is free.)

However, when a US federal government agency uses contractors to develop software, that is a different situation. The contract can require the contractor to release it under the GNU GPL. (GNU Ada was developed in this way.) Or the contract can assign the copyright to the government agency, which can then release the software under the GNU GPL.

Can the US Government release improvements to a GPL-covered program? (#GPLUSGovAdd)

Yes. If the improvements are written by US government employees in the course of their employment, then the improvements are in the public domain. However, the improved version, as a whole, is still covered by the GNU GPL. There is no problem in this situation.

If the US government uses contractors to do the job, then the improvements themselves can be GPL-covered.

Does the GPL have different requirements for statically vs dynamically linked modules with a covered work? (#GPLStaticVsDynamic)

No. Linking a GPL covered work statically or dynamically with other modules is making a combined work based on the GPL covered work. Thus, the terms and conditions of the GNU General Public License cover the whole combination. See also What legal issues come up if I use GPL-incompatible libraries with GPL software?

Does the LGPL have different requirements for statically vs dynamically linked modules with a covered work? (#LGPLStaticVsDynamic)

For the purpose of complying with the LGPL (any extant version: v2, v2.1 or v3):

> *(1) If you statically link against an LGPLed library, you must also provide your application in an object (not necessarily source) format, so that a user has the opportunity to modify the library and relink the application.*

> *(2) If you dynamically link against an LGPLed library already present on the user's computer, you need not convey the library's source. On the other hand, if you yourself convey the executable LGPLed library along with your application, whether linked with statically or dynamically, you must also convey the library's sources, in one of the ways for which the LGPL provides.*

Is there some way that I can GPL the output people get from use of my program? For example, if my program is used to develop hardware designs, can I require that these designs must be free? (#GPLOutput)

In general this is legally impossible; copyright law does not give you any say in the use of the output people make from their data using your program. If the user uses your program to enter or convert her own data, the copyright on the output belongs to her, not you. More generally, when a program translates its input into some other form, the copyright status of the output inherits that of the input it was generated from.

So the only way you have a say in the use of the output is if substantial parts of the output are copied (more or less) from text in your program. For instance, part of the output of Bison (see above) would be covered by the GNU GPL, if we had not made an exception in this specific case.

You could artificially make a program copy certain text into its output even if there is no technical reason to do so. But if that copied text serves no practical purpose, the user could simply delete that text from the output and use only the rest. Then he would not have to obey the conditions on redistribution of the copied text.

In what cases is the output of a GPL program covered by the GPL too? (#WhatCaseIsOutputGPL)

The output of a program is not, in general, covered by the copyright on the code of the program. So the license of the code of the program does not apply to the output, whether you pipe it into a file, make a screenshot, screencast, or video.

The exception would be when the program displays a full screen of text and/or art that comes from the program. Then the copyright on that text and/or art covers the output. Programs that output audio, such as video games, would also fit into this exception.

If the art/music is under the GPL, then the GPL applies when you copy it no matter how you copy it. However, fair use may still apply.

Keep in mind that some programs, particularly video games, can have artwork/audio that is licensed separately from the underlying GPLed game. In such cases, the license on the artwork/audio would dictate the terms under which video/streaming may occur. See also: Can I use the GPL for something other than software?

If I add a module to a GPL-covered program, do I have to use the GPL as the license for my module? (#GPLModuleLicense)

The GPL says that the whole combined program has to be released under the GPL. So your module has to be available for use under the GPL.

But you can give additional permission for the use of your code. You can, if you wish, release your module under a license which is more lax than the GPL but compatible with the GPL. The license list page gives a partial list of GPL-compatible licenses.

If a library is released under the GPL (not the LGPL), does that mean that any software which uses it has to be under the GPL or a GPL-compatible license? (#IfLibraryIsGPL)

Yes, because the program actually links to the library. As such, the terms of the GPL apply to the entire combination. The software modules that link with the library may be under various GPL compatible licenses, but the work as a whole must be licensed under the GPL. See also: What does it mean to say a license is “compatible with the GPL”?

If a programming language interpreter is released under the GPL, does that mean programs written to be interpreted by it must be under GPL-compatible licenses? (#IfInterpreterIsGPL)

When the interpreter just interprets a language, the answer is no. The interpreted program, to the interpreter, is just data; a free software license like the GPL, based on copyright law, cannot limit what data you use the interpreter on. You can run it on any data (interpreted program), any way you like, and there are no requirements about licensing that data to anyone.

However, when the interpreter is extended to provide “bindings” to other facilities (often, but not necessarily, libraries), the interpreted program is effectively linked to the facilities it uses through these bindings. So if these facilities are released under the GPL, the interpreted program that uses them must be released in a GPL-compatible way. The JNI or Java Native Interface is an example of such a binding mechanism; libraries that are accessed in this way are linked dynamically with the Java programs that call them. These libraries are also linked with the interpreter. If the interpreter is linked statically with these libraries, or if it is designed to link

dynamically with these specific libraries, then it too needs to be released in a GPL-compatible way.

Another similar and very common case is to provide libraries with the interpreter which are themselves interpreted. For instance, Perl comes with many Perl modules, and a Java implementation comes with many Java classes. These libraries and the programs that call them are always dynamically linked together.

A consequence is that if you choose to use GPLed Perl modules or Java classes in your program, you must release the program in a GPL-compatible way, regardless of the license used in the Perl or Java interpreter that the combined Perl or Java program will run on.

I'm writing a Windows application with Microsoft Visual C++ (or Visual Basic) and I will be releasing it under the GPL. Is dynamically linking my program with the Visual C++ (or Visual Basic) runtime library permitted under the GPL? (#WindowsRuntimeAndGPL)

You may link your program to these libraries, and distribute the compiled program to others. When you do this, the runtime libraries are “System Libraries” as GPLv3 defines them. That means that you don't need to worry about including their source code with the program's Corresponding Source. GPLv2 provides a similar exception in section 3.

You may not distribute these libraries in compiled DLL form with the program. To prevent unscrupulous distributors from trying to use the System Library exception as a loophole, the GPL says that libraries can only qualify as System Libraries as long as they're not distributed with the program itself. If you distribute the DLLs with the program, they won't be eligible for this exception anymore; then the only way to comply with the GPL would be to provide their source code, which you are unable to do.

It is possible to write free programs that only run on Windows, but it is not a good idea. These programs would be “trapped” by Windows, and therefore contribute zero to the Free World.

Why is the original BSD license incompatible with the GPL? (#OrigBSD)

Because it imposes a specific requirement that is not in the GPL; namely, the requirement on advertisements of the program. Section 6 of GPLv2 states:

> *You may not impose any further restrictions on the recipients' exercise of the rights granted herein.*

GPLv3 says something similar in section 10. The advertising clause provides just such a further restriction, and thus is GPL-incompatible.

The revised BSD license does not have the advertising clause, which eliminates the problem.

When is a program and its plug-ins considered a single combined program? (#GPLPlugins)

It depends on how the main program invokes its plug-ins. If the main program uses fork and exec to invoke plug-ins, and they establish intimate communication by sharing complex data structures, or shipping complex data structures back and forth, that can make them one single combined program. A main program that uses simple fork and exec to invoke plug-ins and does not establish intimate communication between them results in the plug-ins being a separate program.

If the main program dynamically links plug-ins, and they make function calls to each other and share data structures, we believe they form a single combined program, which must be treated as an extension of both the main program and the plug-ins. If the main program dynamically links plug-ins, but the communication between them is limited to invoking the ‘main’ function of the plug-in with some options and waiting for it to return, that is a borderline case.

Using shared memory to communicate with complex data structures is pretty much equivalent to

dynamic linking.

If I write a plug-in to use with a GPL-covered program, what requirements does that impose on the licenses I can use for distributing my plug-in? (#GPLAndPlugins)

Please see this question for determining when plug-ins and a main program are considered a single combined program and when they are considered separate works.

If the main program and the plugins are a single combined program then this means you must license the plug-in under the GPL or a GPL-compatible free software license and distribute it with source code in a GPL-compliant way. A main program that is separate from its plug-ins makes no requirements for the plug-ins.

Can I apply the GPL when writing a plug-in for a nonfree program? (#GPLPluginsInNF)

Please see this question for determining when plug-ins and a main program are considered a single combined program and when they are considered separate programs.

If they form a single combined program this means that combination of the GPL-covered plug-in with the nonfree main program would violate the GPL. However, you can resolve that legal problem by adding an exception to your plug-in's license, giving permission to link it with the nonfree main program.

See also the question I am writing free software that uses a nonfree library.

Can I release a nonfree program that's designed to load a GPL-covered plug-in? (#NFUseGPLPlugins)

Please see this question for determining when plug-ins and a main program are considered a single combined program and when they are considered separate programs.

If they form a single combined program then the main program must be released under the GPL or a GPL-compatible free software license, and the terms of the GPL must be followed when the main program is distributed for use with these plug-ins.

However, if they are separate works then the license of the plug-in makes no requirements about the main program.

See also the question I am writing free software that uses a nonfree library.

You have a GPLed program that I'd like to link with my code to build a proprietary program. Does the fact that I link with your program mean I have to GPL my program? (#LinkingWithGPL)

Not exactly. It means you must release your program under a license compatible with the GPL (more precisely, compatible with one or more GPL versions accepted by all the rest of the code in the combination that you link). The combination itself is then available under those GPL versions.

If so, is there any chance I could get a license of your program under the Lesser GPL? (#SwitchToLGPL)

You can ask, but most authors will stand firm and say no. The idea of the GPL is that if you want to include our code in your program, your program must also be free software. It is supposed to put pressure on you to release your program in a way that makes it part of our community.

You always have the legal alternative of not using our code.

Does distributing a nonfree driver meant to link with the kernel Linux violate the GPL? (#NonfreeDriverKernelLinux)

Linux (the kernel in the GNU/Linux operating system) is distributed under GNU GPL version 2. Does distributing a nonfree driver meant to link with Linux violate the GPL?

Yes, this is a violation, because effectively this makes a larger combined work. The fact that the user is expected to put the pieces together does not really change anything.

Each contributor to Linux who holds copyright on a substantial part of the code can enforce the GPL and we encourage each of them to take action against those distributing nonfree Linux-drivers.

How can I allow linking of proprietary modules with my GPL-covered library under a controlled interface only? (#LinkingOverControlledInterface)

Add this text to the license notice of each file in the package, at the end of the text that says the file is distributed under the GNU GPL:

Linking ABC statically or dynamically with other modules is making a combined work based on ABC. Thus, the terms and conditions of the GNU General Public License cover the whole combination.

As a special exception, the copyright holders of ABC give you permission to combine ABC program with free software programs or libraries that are released under the GNU LGPL and with independent modules that communicate with ABC solely through the ABCDEF interface. You may copy and distribute such a system following the terms of the GNU GPL for ABC and the licenses of the other code

concerned, provided that you include the source code of that other code when and as the GNU GPL requires distribution of source code and provided that you do not modify the ABCDEF interface.

Note that people who make modified versions of ABC are not obligated to grant this special exception for their modified versions; it is their choice whether to do so. The GNU General Public License gives permission to release a modified version without this exception; this exception also makes it possible to release a modified version which carries forward this exception. If you modify the ABCDEF interface, this exception does not apply to your modified version of ABC, and you must remove this exception when you distribute your modified version.

This exception is an additional permission under section 7 of the GNU General Public License, version 3 (“GPLv3”)

This exception enables linking with differently licensed modules over the specified interface (“ABCDEF”), while ensuring that users would still receive source code as they normally would under the GPL.

Only the copyright holders for the program can legally authorize this exception. If you wrote the whole program yourself, then assuming your employer or school does not claim the copyright, you are the copyright holder—so you can authorize the exception. But if you want to use parts of other GPL-covered programs by other authors in your code, you cannot authorize the exception for them. You have to get the approval of the copyright holders of those programs.

I have written an application that links with many different components, that have different licenses. I am very confused as to what licensing requirements are placed on my program. Can you please tell me what licenses I may use? (#ManyDifferentLicenses)

To answer this question, we would need to see a list of each component that your program uses, the license of that component, and a brief (a few sentences for each should suffice) describing how your library uses that component. Two examples would be:

- To make my software work, it must be linked to the FOO library, which is available under the Lesser GPL.
- My software makes a system call (with a command line that I built) to run the BAR program, which is licensed under “the GPL, with a special exception allowing for linking with QUUX”.

What is the difference between an “aggregate” and other kinds of “modified versions”? (#MereAggregation)

An “aggregate” consists of a number of separate programs, distributed together on the same CD-ROM or other media. The GPL permits you to create and distribute an aggregate, even when the licenses of the other software are nonfree or GPL-incompatible. The only condition is that you cannot release the aggregate under a license that prohibits users from exercising rights that each program's individual license would grant them.

Where's the line between two separate programs, and one program with two parts? This is a legal question, which ultimately judges will decide. We believe that a proper criterion depends both on the mechanism of communication (exec, pipes, rpc, function calls within a shared address space, etc.) and the semantics of the communication (what kinds of information are interchanged).

If the modules are included in the same executable file, they are definitely combined in one program. If modules are designed to run linked together in a shared address space, that almost surely means combining them into one program.

By contrast, pipes, sockets and command-line arguments are communication mechanisms normally used between two separate programs. So when they are used for communication, the modules normally are separate programs. But if the semantics of the communication are intimate enough, exchanging complex internal data structures, that too could be a basis to consider the two parts as combined into a larger program.

When it comes to determining whether two pieces of software form a single work, does the

fact that the code is in one or more containers have any effect? (#AggregateContainers)

No, the analysis of whether they are a single work or an aggregate is unchanged by the involvement of containers.

Why does the FSF require that contributors to FSF-copyrighted programs assign copyright to the FSF? If I hold copyright on a GPLed program, should I do this, too? If so, how? (#AssignCopyright)

Our lawyers have told us that to be in the best position to enforce the GPL in court against violators, we should keep the copyright status of the program as simple as possible. We do this by asking each contributor to either assign the copyright on contributions to the FSF, or disclaim copyright on contributions.

We also ask individual contributors to get copyright disclaimers from their employers (if any) so that we can be sure those employers won't claim to own the contributions.

Of course, if all the contributors put their code in the public domain, there is no copyright with which to enforce the GPL. So we encourage people to assign copyright on large code contributions, and only put small changes in the public domain.

If you want to make an effort to enforce the GPL on your program, it is probably a good idea for you to follow a similar policy. Please contact <licensing@gnu.org> if you want more information.

Can I modify the GPL and make a modified license? (#ModifyGPL)

It is possible to make modified versions of the GPL, but it tends to have practical consequences.

You can legally use the GPL terms (possibly modified) in another license provided that you call your license by another name and do not include the GPL preamble, and provided you modify the instructions-for-use at the end enough to make it clearly different in wording and not mention GNU (though the actual procedure you describe may be similar).

If you want to use our preamble in a modified license, please write to <licensing@gnu.org> for permission. For this purpose we would want to check the actual license requirements to see if we approve of them.

Although we will not raise legal objections to your making a modified license in this way, we hope you will think twice and not do it. Such a modified license is almost certainly incompatible with the GNU GPL, and that incompatibility blocks useful combinations of modules. The mere proliferation of different free software licenses is a burden in and of itself.

Rather than modifying the GPL, please use the exception mechanism offered by GPL version 3.

If I use a piece of software that has been obtained under the GNU GPL, am I allowed to modify the original code into a new program, then distribute and sell that new program commercially? (#GPLCommercially)

You are allowed to sell copies of the modified program commercially, but only under the terms of the GNU GPL. Thus, for instance, you must make the source code available to the users of the program as described in the GPL, and they must be allowed to redistribute and modify it as described in the GPL.

These requirements are the condition for including the GPL-covered code you received in a program of your own.

Can I use the GPL for something other than software? (#GPLOtherThanSoftware)

You can apply the GPL to any kind of work, as long as it is clear what constitutes the “source code” for the work. The GPL defines this as the preferred form of the work for making changes in it.

However, for manuals and textbooks, or more generally any sort of work that is meant to teach a subject, we recommend using the GFDL rather than the GPL.

How does the LGPL work with Java? (#LGPLJava)

See this article for details. It works as designed, intended, and expected.

Consider this situation: 1) X releases V1 of a project under the GPL. 2) Y contributes to the development of V2 with changes and new code based on V1. 3) X wants to convert V2 to a non-GPL license. Does X need Y's permission? (#Consider)

Yes. Y was required to release its version under the GNU GPL, as a consequence of basing it on X's version V1. Nothing required Y to agree to any other license for its code. Therefore, X must get Y's permission before releasing that code under another license.

I'd like to incorporate GPL-covered software in my proprietary system. I have no permission to use that software except what the GPL gives me. Can I do this? (#GPLInProprietarySystem)

You cannot incorporate GPL-covered software in a proprietary system. The goal of the GPL is to grant everyone the freedom to copy, redistribute, understand, and modify a program. If you could incorporate GPL-covered software into a nonfree system, it would have the effect of making the GPL-covered software nonfree too.

A system incorporating a GPL-covered program is an extended version of that program. The

GPL says that any extended version of the program must be released under the GPL if it is released at all. This is for two reasons: to make sure that users who get the software get the freedom they should have, and to encourage people to give back improvements that they make.

However, in many cases you can distribute the GPL-covered software alongside your proprietary system. To do this validly, you must make sure that the free and nonfree programs communicate at arms length, that they are not combined in a way that would make them effectively a single program.

The difference between this and “incorporating” the GPL-covered software is partly a matter substance and partly form. The substantive part is this: if the two programs are combined so that they become effectively two parts of one program, then you can't treat them as two separate programs. So the GPL has to cover the whole thing.

If the two programs remain well separated, like the compiler and the kernel, or like an editor and a shell, then you can treat them as two separate programs—but you have to do it properly. The issue is simply one of form: how you describe what you are doing. Why do we care about this? Because we want to make sure the users clearly understand the free status of the GPL-covered software in the collection.

If people were to distribute GPL-covered software calling it “part of” a system that users know is partly proprietary, users might be uncertain of their rights regarding the GPL-covered software. But if they know that what they have received is a free program plus another program, side by side, their rights will be clear.

Using a certain GNU program under the GPL does not fit our project to make proprietary software. Will you make an exception for us? It would mean more users of that program. (#WillYouMakeAnException)

Sorry, we don't make such exceptions. It would not be right.

Maximizing the number of users is not our aim. Rather, we are trying to give the crucial freedoms

to as many users as possible. In general, proprietary software projects hinder rather than help the cause of freedom.

We do occasionally make license exceptions to assist a project which is producing free software under a license other than the GPL. However, we have to see a good reason why this will advance the cause of free software.

We also do sometimes change the distribution terms of a package, when that seems clearly the right way to serve the cause of free software; but we are very cautious about this, so you will have to show us very convincing reasons.

I'd like to incorporate GPL-covered software in my proprietary system. Can I do this by putting a "wrapper" module, under a GPL-compatible lax permissive license (such as the X11 license) in between the GPL-covered part and the proprietary part? (#GPLWrapper)

No. The X11 license is compatible with the GPL, so you can add a module to the GPL-covered program and put it under the X11 license. But if you were to incorporate them both in a larger program, that whole would include the GPL-covered part, so it would have to be licensed *as a whole* under the GNU GPL.

The fact that proprietary module A communicates with GPL-covered module C only through X11-licensed module B is legally irrelevant; what matters is the fact that module C is included in the whole.

Where can I learn more about the GCC Runtime Library Exception? (#LibGCCException)

The GCC Runtime Library Exception covers libgcc, libstdc++, libfortran, libgomp, libdecnumber, and other libraries distributed with GCC. The exception is meant to allow people to distribute programs compiled with GCC under terms of their choice, even when parts of these libraries are included in the executable as part of the compilation process. To learn more, please

read our FAQ about the GCC Runtime Library Exception.

I'd like to modify GPL-covered programs and link them with the portability libraries from Money Guzzler Inc. I cannot distribute the source code for these libraries, so any user who wanted to change these versions would have to obtain those libraries separately. Why doesn't the GPL permit this? (#MoneyGuzzlerInc)

There are two reasons for this. First, a general one. If we permitted company A to make a proprietary file, and company B to distribute GPL-covered software linked with that file, the effect would be to make a hole in the GPL big enough to drive a truck through. This would be carte blanche for withholding the source code for all sorts of modifications and extensions to GPL-covered software.

Giving all users access to the source code is one of our main goals, so this consequence is definitely something we want to avoid.

More concretely, the versions of the programs linked with the Money Guzzler libraries would not really be free software as we understand the term—they would not come with full source code that enables users to change and recompile the program.

If the license for a module Q has a requirement that's incompatible with the GPL, but the requirement applies only when Q is distributed by itself, not when Q is included in a larger program, does that make the license GPL-compatible? Can I combine or link Q with a GPL-covered program? (#GPLIncompatibleAlone)

If a program P is released under the GPL that means *any and every part of it* can be used under the GPL. If you integrate module Q, and release the combined program P+Q under the GPL, that means any part of P+Q can be used under the GPL. One part of P+Q is Q. So releasing P+Q under the GPL says that Q any part of it can be used under the GPL. Putting it in other words, a user who obtains P+Q under the GPL can delete P, so that just Q remains, still

under the GPL.

If the license of module Q permits you to give permission for that, then it is GPL-compatible. Otherwise, it is not GPL-compatible.

If the license for Q says in no uncertain terms that you must do certain things (not compatible with the GPL) when you redistribute Q on its own, then it does not permit you to distribute Q under the GPL. It follows that you can't release P+Q under the GPL either. So you cannot link or combine P with Q.

Can I release a modified version of a GPL-covered program in binary form only? (#ModifiedJustBinary)

No. The whole point of the GPL is that all modified versions must be free software—which means, in particular, that the source code of the modified version is available to the users.

I downloaded just the binary from the net. If I distribute copies, do I have to get the source and distribute that too? (#UnchangedJustBinary)

Yes. The general rule is, if you distribute binaries, you must distribute the complete corresponding source code too. The exception for the case where you received a written offer for source code is quite limited.

I want to distribute binaries via physical media without accompanying sources. Can I provide source code by FTP? (#DistributeWithSourceOnInternet)

Version 3 of the GPL allows this; see option 6(b) for the full details. Under version 2, you're certainly free to offer source via FTP, and most users will get it from there. However, if any o

them would rather get the source on physical media by mail, you are required to provide that.

If you distribute binaries via FTP, you should distribute source via FTP.

My friend got a GPL-covered binary with an offer to supply source, and made a copy for me. Can I use the offer myself to obtain the source? (#RedistributedBinariesGetSource)

Yes, you can. The offer must be open to everyone who has a copy of the binary that it accompanies. This is why the GPL says your friend must give you a copy of the offer along with a copy of the binary—so you can take advantage of it.

Can I put the binaries on my Internet server and put the source on a different Internet site? (#SourceAndBinaryOnDifferentSites)

Yes. Section 6(d) allows this. However, you must provide clear instructions people can follow to obtain the source, and you must take care to make sure that the source remains available for as long as you distribute the object code.

I want to distribute an extended version of a GPL-covered program in binary form. Is it enough to distribute the source for the original version? (#DistributeExtendedBinary)

No, you must supply the source code that corresponds to the binary. Corresponding source means the source from which users can rebuild the same binary.

Part of the idea of free software is that users should have access to the source code for *the programs they use.* Those using your version should have access to the source code for your version.

A major goal of the GPL is to build up the Free World by making sure that improvement to a free program are themselves free. If you release an improved version of a GPL-covered program, you must release the improved source code under the GPL.

I want to distribute binaries, but distributing complete source is inconvenient. Is it ok if I give users the diffs from the “standard” version along with the binaries? (#DistributingSourceIsInconvenient)

This is a well-meaning request, but this method of providing the source doesn't really do the job.

A user that wants the source a year from now may be unable to get the proper version from another site at that time. The standard distribution site may have a newer version, but the same diffs probably won't work with that version.

So you need to provide complete sources, not just diffs, with the binaries.

Can I make binaries available on a network server, but send sources only to people who order them? (#AnonFTPAndSendSources)

If you make object code available on a network server, you have to provide the Corresponding Source on a network server as well. The easiest way to do this would be to publish them on the same server, but if you'd like, you can alternatively provide instructions for getting the source from another server, or even a version control system. No matter what you do, the source should be just as easy to access as the object code, though. This is all specified in section 6(d) of GPLv3.

The sources you provide must correspond exactly to the binaries. In particular, you must make sure they are for the same version of the program—not an older version and not a newer version.

How can I make sure each user who downloads the binaries also gets the source? (#HowCanIMakeSureEachDownloadGetsSource)

You don't have to make sure of this. As long as you make the source and binaries available so that the users can see what's available and take what they want, you have done what is required of you. It is up to the user whether to download the source.

Our requirements for redistributors are intended to make sure the users can get the source code, not to force users to download the source code even if they don't want it.

Does the GPL require me to provide source code that can be built to match the exact hash of the binary I am distributing? (#MustSourceBuildToMatchExactHashOfBinary)

Complete corresponding source means the source that the binaries were made from, but that does not imply your tools must be able to make a binary that is an exact hash of the binary you are distributing. In some cases it could be (nearly) impossible to build a binary from source with an exact hash of the binary being distributed — consider the following examples: a system might put timestamps in binaries; or the program might have been built against a different (even unreleased) compiler version.

A company is running a modified version of a GPLed program on a web site. Does the GPL say they must release their modified sources? (#UnreleasedMods)

The GPL permits anyone to make a modified version and use it without ever distributing it to others. What this company is doing is a special case of that. Therefore, the company does not have to release the modified sources. The situation is different when the modified program is licensed under the terms of the GNU Affero GPL.

Compare this to a situation where the web site contains or links to separate GPLed programs

that are distributed to the user when they visit the web site (often written in JavaScript, but other languages are used as well). In this situation the source code for the programs being distributed must be released to the user under the terms of the GPL.

A company is running a modified version of a program licensed under the GNU Affero GPL (AGPL) on a web site. Does the AGPL say they must release their modified sources? (#UnreleasedModsAGPL)

The GNU Affero GPL requires that modified versions of the software offer all users interacting with it over a computer network an opportunity to receive the source. What the company is doing falls under that meaning, so the company must release the modified source code.

Is making and using multiple copies within one organization or company "distribution"? (#InternalDistribution)

No, in that case the organization is just making the copies for itself. As a consequence, a company or other organization can develop a modified version and install that version through its own facilities, without giving the staff permission to release that modified version to outsiders.

However, when the organization transfers copies to other organizations or individuals, that is distribution. In particular, providing copies to contractors for use off-site is distribution.

If someone steals a CD containing a version of a GPL-covered program, does the GPL give the thief the right to redistribute that version? (#StolenCopy)

If the version has been released elsewhere, then the thief probably does have the right to make copies and redistribute them under the GPL, but if thieves are imprisoned for stealing the CD, they may have to wait until their release before doing so.

If the version in question is unpublished and considered by a company to be its trade secret, then publishing it may be a violation of trade secret law, depending on other circumstances. The GPL does not change that. If the company tried to release its version and still treat it as a trade secret, that would violate the GPL, but if the company hasn't released this version, no such violation has occurred.

What if a company distributes a copy of some other developers' GPL-covered work to me as a trade secret? (#TradeSecretRelease)

The company has violated the GPL and will have to cease distribution of that program. Note how this differs from the theft case above; the company does not intentionally distribute a copy when a copy is stolen, so in that case the company has not violated the GPL.

What if a company distributes a copy of its own GPL-covered work to me as a trade secret? (#TradeSecretRelease2)

If the program distributed does not incorporate anyone else's GPL-covered work, then the company is not violating the GPL (see "Is the developer of a GPL-covered program bound by the GPL?" for more information). But it is making two contradictory statements about what you can do with that program: that you can redistribute it, and that you can't. It would make sense to demand clarification of the terms for use of that program before you accept a copy.

Why are some GNU libraries released under the ordinary GPL rather than the Lesser GPL? (#WhySomeGPLAndNotLGPL)

Using the Lesser GPL for any particular library constitutes a retreat for free software. It means we partially abandon the attempt to defend the users' freedom, and some of the requirements to share what is built on top of GPL-covered software. In themselves, those are changes for the

worse.

Sometimes a localized retreat is a good strategy. Sometimes, using the LGPL for a library might lead to wider use of that library, and thus to more improvement for it, wider support for free software, and so on. This could be good for free software if it happens to a large extent. But how much will this happen? We can only speculate.

It would be nice to try out the LGPL on each library for a while, see whether it helps, and change back to the GPL if the LGPL didn't help. But this is not feasible. Once we use the LGPL for a particular library, changing back would be difficult.

So we decide which license to use for each library on a case-by-case basis. There is a long explanation of how we judge the question.

Why should programs say "Version 3 of the GPL or any later version"? (#VersionThreeOrLater)

From time to time, at intervals of years, we change the GPL—sometimes to clarify it, sometimes to permit certain kinds of use not previously permitted, and sometimes to tighten up a requirement. (The last two changes were in 2007 and 1991.) Using this "indirect pointer" in each program makes it possible for us to change the distribution terms on the entire collection GNU software, when we update the GPL.

If each program lacked the indirect pointer, we would be forced to discuss the change at length with numerous copyright holders, which would be a virtual impossibility. In practice, the chance of having uniform distribution terms for GNU software would be nil.

Suppose a program says "Version 3 of the GPL or any later version" and a new version of the GPL is released. If the new GPL version gives additional permission, that permission will be available immediately to all the users of the program. But if the new GPL version has a tighter requirement, it will not restrict use of the current version of the program, because it can still be used under GPL version 3. When a program says "Version 3 of the GPL or any later version",

users will always be permitted to use it, and even change it, according to the terms of GPL version 3—even after later versions of the GPL are available.

If a tighter requirement in a new version of the GPL need not be obeyed for existing software, how is it useful? Once GPL version 4 is available, the developers of most GPL-covered programs will release subsequent versions of their programs specifying “Version 4 of the GPL or any later version”. Then users will have to follow the tighter requirements in GPL version 4, for subsequent versions of the program.

However, developers are not obligated to do this; developers can continue allowing use of the previous version of the GPL, if that is their preference.

Is it a good idea to use a license saying that a certain program can be used only under the latest version of the GNU GPL? (#OnlyLatestVersion)

The reason you shouldn't do that is that it could result some day in withdrawing automatically some permissions that the users previously had.

Suppose a program was released in 2000 under “the latest GPL version”. At that time, people could have used it under GPLv2. The day we published GPLv3 in 2007, everyone would have been suddenly compelled to use it under GPLv3 instead.

Some users may not even have known about GPL version 3—but they would have been required to use it. They could have violated the program's license unintentionally just because they did not get the news. That's a bad way to treat people.

We think it is wrong to take back permissions already granted, except due to a violation. If your freedom could be revoked, then it isn't really freedom. Thus, if you get a copy of a program version under one version of a license, you should *always* have the rights granted by that version of the license. Releasing under “GPL version N or any later version” upholds that principle.

Why don't you use the GPL for manuals? (#WhyNotGPLForManuals)

It is possible to use the GPL for a manual, but the GNU Free Documentation License (GFDL) is much better for manuals.

The GPL was designed for programs; it contains lots of complex clauses that are crucial for programs, but that would be cumbersome and unnecessary for a book or manual. For instance, anyone publishing the book on paper would have to either include machine-readable “source code” of the book along with each printed copy, or provide a written offer to send the “source code” later.

Meanwhile, the GFDL has clauses that help publishers of free manuals make a profit from selling copies—cover texts, for instance. The special rules for Endorsements sections make it possible to use the GFDL for an official standard. This would permit modified versions, but they could not be labeled as “the standard”.

Using the GFDL, we permit changes in the text of a manual that covers its technical topic. It is important to be able to change the technical parts, because people who change a program ought to change the documentation to correspond. The freedom to do this is an ethical imperative.

Our manuals also include sections that state our political position about free software. We mark these as “invariant”, so that they cannot be changed or removed. The GFDL makes provisions for these “invariant sections”.

How does the GPL apply to fonts? (#FontException)

Font licensing is a complex issue which needs serious consideration. The following license exception is experimental but approved for general use. We welcome suggestions on this subject —please see this this explanatory essay and write to licensing@gnu.org.

To use this exception, add this text to the license notice of each file in the package (to the extent

possible), at the end of the text that says the file is distributed under the GNU GPL:

> *As a special exception, if you create a document which uses this font, and embed this font or unaltered portions of this font into the document, this font does not by itself cause the resulting document to be covered by the GNU General Public License. This exception does not however invalidate any other reasons why the document might be covered by the GNU General Public License. If you modify this font, you may extend this exception to your version of the font, but you are not obligated to do so. If you do not wish to do so, delete this exception statement from your version.*

I am writing a website maintenance system (called a "content management system" by some), or some other application which generates web pages from templates. What license should I use for those templates? (#WMS)

Templates are minor enough that it is not worth using copyleft to protect them. It is normally harmless to use copyleft on minor works, but templates are a special case, because they are combined with data provided by users of the application and the combination is distributed. So, we recommend that you license your templates under simple permissive terms.

Some templates make calls into JavaScript functions. Since Javascript is often non-trivial, it is worth copylefting. Because the templates will be combined with user data, it's possible that template+user data+JavaScript would be considered one work under copyright law. A line needs to be drawn between the JavaScript (copylefted), and the user code (usually under incompatible terms).

Here's an exception for JavaScript code that does this:

As a special exception to the GPL, any HTML file which merely makes function calls to this code, and for that purpose includes it by reference shall be deemed a separate work for copyright law purposes. In addition, the copyright holders of this code give you permission to combine this code with free software libraries that are released under the GNU LGPL. You may copy and distribute such a system following the terms of the GNU GPL for this code and the LGPL for the libraries. If you modify this code, you may extend this exception to your version of the code, but you are not obligated to do so. If you do not wish to do so, delete this exception statement from your version.

Can I release a program under the GPL which I developed using nonfree tools? (#NonFreeTools)

Which programs you used to edit the source code, or to compile it, or study it, or record it, usually makes no difference for issues concerning the licensing of that source code.

However, if you link nonfree libraries with the source code, that would be an issue you need to deal with. It does not preclude releasing the source code under the GPL, but if the libraries don't fit under the “system library” exception, you should affix an explicit notice giving permission to link your program with them. The FAQ entry about using GPL-incompatible libraries provides more information about how to do that.

Are there translations of the GPL into other languages? (#GPLTranslations)

It would be useful to have translations of the GPL into languages other than English. People have even written translations and sent them to us. But we have not dared to approve them as officially valid. That carries a risk so great we do not dare accept it.

A legal document is in some ways like a program. Translating it is like translating a program from one language and operating system to another. Only a lawyer skilled in both languages can do it —and even then, there is a risk of introducing a bug.

If we were to approve, officially, a translation of the GPL, we would be giving everyone permission to do whatever the translation says they can do. If it is a completely accurate translation, that is fine. But if there is an error in the translation, the results could be a disaster which we could not fix.

If a program has a bug, we can release a new version, and eventually the old version will more or less disappear. But once we have given everyone permission to act according to a particular translation, we have no way of taking back that permission if we find, later on, that it had a bug.

Helpful people sometimes offer to do the work of translation for us. If the problem were a matter of finding someone to do the work, this would solve it. But the actual problem is the risk of error, and offering to do the work does not avoid the risk. We could not possibly authorize a translation written by a non-lawyer.

Therefore, for the time being, we are not approving translations of the GPL as globally valid and binding. Instead, we are doing two things:

- Referring people to unofficial translations. This means that we permit people to write translations of the GPL, but we don't approve them as legally valid and binding.

  An unapproved translation has no legal force, and it should say so explicitly. It should be marked as follows:

  > *This translation of the GPL is informal, and not officially approved by the Free Software Foundation as valid. To be completely sure of what is permitted, refer to the original GPL (in English).*

  But the unapproved translation can serve as a hint for how to understand the English GPL. For many users, that is sufficient.

However, businesses using GNU software in commercial activity, and people doing public ftp distribution, should need to check the real English GPL to make sure of what it permits.

- Publishing translations valid for a single country only.

We are considering the idea of publishing translations which are officially valid only for one country. This way, if there is a mistake, it will be limited to that country, and the damage will not be too great.

It will still take considerable expertise and effort from a sympathetic and capable lawyer to make a translation, so we cannot promise any such translations soon.

If a programming language interpreter has a license that is incompatible with the GPL, can I run GPL-covered programs on it? (#InterpreterIncompat)

When the interpreter just interprets a language, the answer is yes. The interpreted program, to the interpreter, is just data; the GPL doesn't restrict what tools you process the program with.

However, when the interpreter is extended to provide “bindings” to other facilities (often, but not necessarily, libraries), the interpreted program is effectively linked to the facilities it uses through these bindings. The JNI or Java Native Interface is an example of such a facility; libraries that are accessed in this way are linked dynamically with the Java programs that call them.

So if these facilities are released under a GPL-incompatible license, the situation is like linking in any other way with a GPL-incompatible library. Which implies that:

1. If you are writing code and releasing it under the GPL, you can state an explicit exception giving permission to link it with those GPL-incompatible facilities.
2. If you wrote and released the program under the GPL, and you designed it specifically to work with those facilities, people can take that as an implicit exception permitting them to link it with those facilities. But if that is what you intend, it is better to say so explicitly.
3. You can't take someone else's GPL-covered code and use it that way, or add such

exceptions to it. Only the copyright holders of that code can add the exception.

Who has the power to enforce the GPL? (#WhoHasThePower)

Since the GPL is a copyright license, it can be enforced by the copyright holders of the software. If you see a violation of the GPL, you should inform the developers of the GPL-covered software involved. They either are the copyright holders, or are connected with the copyright holders.

In addition, we encourage the use of any legal mechanism available to users for obtaining complete and corresponding source code, as is their right, and enforcing full compliance with the GNU GPL. After all, we developed the GNU GPL to make software free for all its users.

In an object-oriented language such as Java, if I use a class that is GPLed without modifying, and subclass it, in what way does the GPL affect the larger program? (#OOPLang)

Subclassing is creating a derivative work. Therefore, the terms of the GPL affect the whole program where you create a subclass of a GPLed class.

If I port my program to GNU/Linux, does that mean I have to release it as free software under the GPL or some other Free Software license? (#PortProgramToGPL)

In general, the answer is no—this is not a legal requirement. In specific, the answer depends on which libraries you want to use and what their licenses are. Most system libraries either use the GNU Lesser GPL, or use the GNU GPL plus an exception permitting linking the library with anything. These libraries can be used in nonfree programs; but in the case of the Lesser GPL, it does have some requirements you must follow.

Some libraries are released under the GNU GPL alone; you must use a GPL-compatible license

to use those libraries. But these are normally the more specialized libraries, and you would not have had anything much like them on another platform, so you probably won't find yourself wanting to use these libraries for simple porting.

Of course, your software is not a contribution to our community if it is not free, and people who value their freedom will refuse to use it. Only people willing to give up their freedom will use your software, which means that it will effectively function as an inducement for people to lose their freedom.

If you hope some day to look back on your career and feel that it has contributed to the growth of a good and free society, you need to make your software free.

I just found out that a company has a copy of a GPLed program, and it costs money to get it. Aren't they violating the GPL by not making it available on the Internet? (#CompanyGPLCostsMoney)

No. The GPL does not require anyone to use the Internet for distribution. It also does not require anyone in particular to redistribute the program. And (outside of one special case), even if someone does decide to redistribute the program sometimes, the GPL doesn't say he has to distribute a copy to you in particular, or any other person in particular.

What the GPL requires is that he must have the freedom to distribute a copy to you *if he wishes to*. Once the copyright holder does distribute a copy of the program to someone, that someone can then redistribute the program to you, or to anyone else, as he sees fit.

Can I release a program with a license which says that you can distribute modified versions of it under the GPL but you can't distribute the original itself under the GPL? (#ReleaseNotOriginal)

No. Such a license would be self-contradictory. Let's look at its implications for me as a user.

Suppose I start with the original version (call it version A), add some code (let's imagine it is 1000 lines), and release that modified version (call it B) under the GPL. The GPL says anyone can change version B again and release the result under the GPL. So I (or someone else) can delete those 1000 lines, producing version C which has the same code as version A but is under the GPL.

If you try to block that path, by saying explicitly in the license that I'm not allowed to reproduce something identical to version A under the GPL by deleting those lines from version B, in effect the license now says that I can't fully use version B in all the ways that the GPL permits. In other words, the license does not in fact allow a user to release a modified version such as B under the GPL.

Does moving a copy to a majority-owned, and controlled, subsidiary constitute distribution? (#DistributeSubsidiary)

Whether moving a copy to or from this subsidiary constitutes “distribution” is a matter to be decided in each case under the copyright law of the appropriate jurisdiction. The GPL does not and cannot override local laws. US copyright law is not entirely clear on the point, but appears not to consider this distribution.

If, in some country, this is considered distribution, and the subsidiary must receive the right to redistribute the program, that will not make a practical difference. The subsidiary is controlled by the parent company; rights or no rights, it won't redistribute the program unless the parent company decides to do so.

Can software installers ask people to click to agree to the GPL? If I get some software under the GPL, do I have to agree to anything? (#ClickThrough)

Some software packaging systems have a place which requires you to click through or otherwise indicate assent to the terms of the GPL. This is neither required nor forbidden. With or without a click through, the GPL's rules remain the same.

Merely agreeing to the GPL doesn't place any obligations on you. You are not required to agree to anything to merely use software which is licensed under the GPL. You only have obligation you modify or distribute the software. If it really bothers you to click through the GPL, nothing stops you from hacking the GPLed software to bypass this.

I would like to bundle GPLed software with some sort of installation software. Does that installer need to have a GPL-compatible license? (#GPLCompatInstaller)

No. The installer and the files it installs are separate works. As a result, the terms of the GPL do not apply to the installation software.

Some distributors of GPLed software require me in their umbrella EULAs or as part of their downloading process to "represent and warrant" that I am located in the US or that I intend to distribute the software in compliance with relevant export control laws. Why are they doing this and is it a violation of those distributors' obligations under GPL? (#ExportWarranties)

This is not a violation of the GPL. Those distributors (almost all of whom are commercial businesses selling free software distributions and related services) are trying to reduce their own legal risks, not to control your behavior. Export control law in the United States *might* make them liable if they knowingly export software into certain countries, or if they give software to parties they know will make such exports. By asking for these statements from their customers and others to whom they distribute software, they protect themselves in the event they are later asked by regulatory authorities what they knew about where software they distributed was going to wind up. They are not restricting what you can do with the software, only preventing themselves from being blamed with respect to anything you do. Because they are not placing additional restrictions on the software, they do not violate section 10 of GPLv3 or section 6 of GPLv2.

The FSF opposes the application of US export control laws to free software. Not only are such laws incompatible with the general objective of software freedom, they achieve no reasonable governmental purpose, because free software is currently and should always be available from

parties in almost every country, including countries that have no export control laws and which do not participate in US-led trade embargoes. Therefore, no country's government is actually deprived of free software by US export control laws, while no country's citizens *should* be deprived of free software, regardless of their governments' policies, as far as we are concerned. Copies of all GPL-licensed software published by the FSF can be obtained from us without making any representation about where you live or what you intend to do. At the same time, the FSF understands the desire of commercial distributors located in the US to comply with US laws. They have a right to choose to whom they distribute particular copies of free software; exercise of that right does not violate the GPL unless they add contractual restrictions beyond those permitted by the GPL.

Can I use GPLed software on a device that will stop operating if customers do not continue paying a subscription fee? (#SubscriptionFee)

No. In this scenario, the requirement to keep paying a fee limits the user's ability to run the program. This is an additional requirement on top of the GPL, and the license prohibits it.

How do I upgrade from (L)GPLv2 to (L)GPLv3? (#v3HowToUpgrade)

First, include the new version of the license in your package. If you're using LGPLv3 in your project, be sure to include copies of both GPLv3 and LGPLv3, since LGPLv3 is now written as a set of additional permissions on top of GPLv3.

Second, replace all your existing v2 license notices (usually at the top of each file) with the new recommended text available on the GNU licenses howto. It's more future-proof because it no longer includes the FSF's postal mailing address.

Of course, any descriptive text (such as in a README) which talks about the package's license should also be updated appropriately.

How does GPLv3 make BitTorrent distribution easier? (#BitTorrent)

Because GPLv2 was written before peer-to-peer distribution of software was common, it is difficult to meet its requirements when you share code this way. The best way to make sure you are in compliance when distributing GPLv2 object code on BitTorrent would be to include all the corresponding source in the same torrent, which is prohibitively expensive.

GPLv3 addresses this problem in two ways. First, people who download this torrent and send the data to others as part of that process are not required to do anything. That's because section 9 says “Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance [of the license].”

Second, section 6(e) of GPLv3 is designed to give distributors—people who initially seed torrents—a clear and straightforward way to provide the source, by telling recipients where it is available on a public network server. This ensures that everyone who wants to get the source can do so, and it's almost no hassle for the distributor.

What is tivoization? How does GPLv3 prevent it? (#Tivoization)

Some devices utilize free software that can be upgraded, but are designed so that users are not allowed to modify that software. There are lots of different ways to do this; for example, sometimes the hardware checksums the software that is installed, and shuts down if it doesn't match an expected signature. The manufacturers comply with GPLv2 by giving you the source code, but you still don't have the freedom to modify the software you're using. We call this practice tivoization.

When people distribute User Products that include software under GPLv3, section 6 requires that they provide you with information necessary to modify that software. User Products is a term specially defined in the license; examples of User Products include portable music players, digital video recorders, and home security systems.

Does GPLv3 prohibit DRM? (#DRMProhibited)

It does not; you can use code released under GPLv3 to develop any kind of DRM technology you like. However, if you do this, section 3 says that the system will not count as an effective technological “protection” measure, which means that if someone breaks the DRM, she will be free to distribute her software too, unhindered by the DMCA and similar laws.

As usual, the GNU GPL does not restrict what people do in software, it just stops them from restricting others.

Can I use the GPL to license hardware? (#GPLHardware)

Any material that can be copyrighted can be licensed under the GPL. GPLv3 can also be used to license materials covered by other copyright-like laws, such as semiconductor masks. So, as an example, you can release a drawing of a physical object or circuit under the GPL.

In many situations, copyright does not cover making physical hardware from a drawing. In these situations, your license for the drawing simply can't exert any control over making or selling physical hardware, regardless of the license you use. When copyright does cover making hardware, for instance with IC masks, the GPL handles that case in a useful way.

I use public key cryptography to sign my code to assure its authenticity. Is it true that GPLv3 forces me to release my private signing keys? (#GiveUpKeys)

No. The only time you would be required to release signing keys is if you conveyed GPLed software inside a User Product, and its hardware checked the software for a valid cryptographic signature before it would function. In that specific case, you would be required to provide anyone who owned the device, on demand, with the key to sign and install modified software on the device so that it will run. If each instance of the device uses a different key, then you need

only give each purchaser a key for that instance.

Does GPLv3 require that voters be able to modify the software running in a voting machine? (#v3VotingMachine)

No. Companies distributing devices that include software under GPLv3 are at most required to provide the source and Installation Information for the software to people who possess a copy of the object code. The voter who uses a voting machine (like any other kiosk) doesn't get possession of it, not even temporarily, so the voter also does not get possession of the binary software in it.

Note, however, that voting is a very special case. Just because the software in a computer is free does not mean you can trust the computer for voting. We believe that computers cannot be trusted for voting. Voting should be done on paper.

Does GPLv3 have a "patent retaliation clause"? (#v3PatentRetaliation)

In effect, yes. Section 10 prohibits people who convey the software from filing patent suits against other licensees. If someone did so anyway, section 8 explains how they would lose their license and any patent licenses that accompanied it.

Can I use snippets of GPL-covered source code within documentation that is licensed under some license that is incompatible with the GPL? (#SourceCodeInDocumentation)

If the snippets are small enough that you can incorporate them under fair use or similar laws, then yes. Otherwise, no.

The beginning of GPLv3 section 6 says that I can convey a covered work in object code form "under the terms of sections 4 and 5" provided I also meet the conditions of section 6. What does that mean? (#v3Under4and5)

> This means that all the permissions and conditions you have to convey source code also apply when you convey object code: you may charge a fee, you must keep copyright notices intact, and so on.

My company owns a lot of patents. Over the years we've contributed code to projects under "GPL version 2 or any later version", and the project itself has been distributed under the same terms. If a user decides to take the project's code (incorporating my contributions) under GPLv3, does that mean I've automatically granted GPLv3's explicit patent license to that user? (#v2OrLaterPatentLicense)

> No. When you convey GPLed software, you must follow the terms and conditions of one particular version of the license. When you do so, that version defines the obligations you have. If users may also elect to use later versions of the GPL, that's merely an additional permission they have—it does not require you to fulfill the terms of the later version of the GPL as well.

> Do not take this to mean that you can threaten the community with your patents. In many countries, distributing software under GPLv2 provides recipients with an implicit patent license to exercise their rights under the GPL. Even if it didn't, anyone considering enforcing their patents aggressively is an enemy of the community, and we will defend ourselves against such an attack.

If I distribute a proprietary program that links against an LGPLv3-covered library that I've modified, what is the "contributor version" for purposes of determining the scope of the explicit patent license grant I'm making—is it just the library, or is it the whole combination? (#LGPLv3ContributorVersion)

The “contributor version” is only your version of the library.

Is GPLv3 compatible with GPLv2? (#v2v3Compatibility)

No. Many requirements have changed from GPLv2 to GPLv3, which means that the precise requirement of GPLv2 is not present in GPLv3, and vice versa. For instance, the Termination conditions of GPLv3 are considerably more permissive than those of GPLv2, and thus different from the Termination conditions of GPLv2.

Due to these differences, the two licenses are not compatible: if you tried to combine code released under GPLv2 with code under GPLv3, you would violate section 6 of GPLv2.

However, if code is released under GPL “version 2 or later,” that is compatible with GPLv3 because GPLv3 is one of the options it permits.

Does GPLv2 have a requirement about delivering installation information? (#InstInfo)

GPLv3 explicitly requires redistribution to include the full necessary “Installation Information.” GPLv2 doesn't use that term, but it does require redistribution to include “scripts used to control compilation and installation of the executable” with the complete and corresponding source code. This covers part, but not all, of what GPLv3 calls “Installation Information.” Thus, GPLv3's requirement about installation information is stronger.

What does it mean to “cure” a violation of GPLv3? (#Cure)

To cure a violation means to adjust your practices to comply with the requirements of the license.

The warranty and liability disclaimers in GPLv3 seem specific to U.S. law. Can I add my own disclaimers to my own code? (#v3InternationalDisclaimers)

Yes. Section 7 gives you permission to add your own disclaimers, specifically 7(a).

My program has interactive user interfaces that are non-visual in nature. How can I comply with the Appropriate Legal Notices requirement in GPLv3? (#NonvisualLegalNotices)

All you need to do is ensure that the Appropriate Legal Notices are readily available to the user in your interface. For example, if you have written an audio interface, you could include a command that reads the notices aloud.

If I give a copy of a GPLv3-covered program to a coworker at my company, have I "conveyed" the copy to that coworker? (#v3CoworkerConveying)

As long as you're both using the software in your work at the company, rather than personally, then the answer is no. The copies belong to the company, not to you or the coworker. This copying is propagation, not conveying, because the company is not making copies available to others.

If I distribute a GPLv3-covered program, can I provide a warranty that is voided if the user modifies the program? (#v3ConditionalWarranty)

Yes. Just as devices do not need to be warranted if users modify the software inside them, you are not required to provide a warranty that covers all possible activities someone could undertake with GPLv3-covered software.

Why did you decide to write the GNU Affero GPLv3 as a separate license? (#SeparateAffero)

Early drafts of GPLv3 allowed licensors to add an Affero-like requirement to publish source in section 7. However, some companies that develop and rely upon free software consider this requirement to be too burdensome. They want to avoid code with this requirement, and expressed concern about the administrative costs of checking code for this additional requirement. By publishing the GNU Affero GPLv3 as a separate license, with provisions in it and GPLv3 to allow code under these licenses to link to each other, we accomplish all of our original goals while making it easier to determine which code has the source publication requirement.

Why did you invent the new terms “propagate” and “convey” in GPLv3? (#WhyPropagateAndConvey)

The term “distribute” used in GPLv2 was borrowed from United States copyright law. Over the years, we learned that some jurisdictions used this same word in their own copyright laws, but gave it different meanings. We invented these new terms to make our intent as clear as possible no matter where the license is interpreted. They are not used in any copyright law in the world, and we provide their definitions directly in the license.

I'd like to license my code under the GPL, but I'd also like to make it clear that it can't be used for military and/or commercial uses. Can I do this? (#NoMilitary)

No, because those two goals contradict each other. The GNU GPL is designed specifically to prevent the addition of further restrictions. GPLv3 allows a very limited set of them, in section 7, but any other added restriction can be removed by the user.

More generally, a license that limits who can use a program, or for what, is not a free software license.

Is “convey” in GPLv3 the same thing as what GPLv2 means by “distribute”? (#ConveyVsDistribute)

Yes, more or less. During the course of enforcing GPLv2, we learned that some jurisdictions used the word “distribute” in their own copyright laws, but gave it different meanings. We invented a new term to make our intent clear and avoid any problems that could be caused by these differences.

GPLv3 gives “making available to the public” as an example of propagation. What does this mean? Is making available a form of conveying? (#v3MakingAvailable)

One example of “making available to the public” is putting the software on a public web or FTP server. After you do this, some time may pass before anybody actually obtains the software from you—but because it could happen right away, you need to fulfill the GPL's obligations right away as well. Hence, we defined conveying to include this activity.

Since distribution and making available to the public are forms of propagation that are also conveying in GPLv3, what are some examples of propagation that do not constitute conveying? (#PropagationNotConveying)

Making copies of the software for yourself is the main form of propagation that is not conveying. You might do this to install the software on multiple computers, or to make backups.

Does prelinking a GPLed binary to various libraries on the system, to optimize its performance, count as modification? (#Prelinking)

No. Prelinking is part of a compilation process; it doesn't introduce any license requirements

above and beyond what other aspects of compilation would. If you're allowed to link the program to the libraries at all, then it's fine to prelink with them as well. If you distribute prelinked object code, you need to follow the terms of section 6.

If someone installs GPLed software on a laptop, and then lends that laptop to a friend without providing source code for the software, have they violated the GPL? (#LaptopLoan)

No. In the jurisdictions where we have investigated this issue, this sort of loan would not count as conveying. The laptop's owner would not have any obligations under the GPL.

Suppose that two companies try to circumvent the requirement to provide Installation Information by having one company release signed software, and the other release a User Product that only runs signed software from the first company. Is this a violation of GPLv3? (#TwoPartyTivoization)

Yes. If two parties try to work together to get around the requirements of the GPL, they can both be pursued for copyright infringement. This is especially true since the definition of convey explicitly includes activities that would make someone responsible for secondary infringement.

Am I complying with GPLv3 if I offer binaries on an FTP server and sources by way of a link to a source code repository in a version control system, like CVS or Subversion? (#SourceInCVS)

This is acceptable as long as the source checkout process does not become burdensome or otherwise restrictive. Anybody who can download your object code should also be able to check out source from your version control system, using a publicly available free software client. Users should be provided with clear and convenient instructions for how to get the source for the exact object code they downloaded—they may not necessarily want the latest

development code, after all.

Can someone who conveys GPLv3-covered software in a User Product use remote attestation to prevent a user from modifying that software? (#RemoteAttestation)

No. The definition of Installation Information, which must be provided with source when the software is conveyed inside a User Product, explicitly says: “The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.” If the device uses remote attestation in some way, the Installation Information must provide you some means for your modified software to report itself as legitimate.

What does “rules and protocols for communication across the network” mean in GPLv3? (#RulesProtocols)

This refers to rules about traffic you can send over the network. For example, if there is a limit on the number of requests you can send to a server per day, or the size of a file you can upload somewhere, your access to those resources may be denied if you do not respect those limits.

These rules do not include anything that does not pertain directly to data traveling across the network. For instance, if a server on the network sent messages for users to your device, your access to the network could not be denied merely because you modified the software so that it did not display the messages.

Distributors that provide Installation Information under GPLv3 are not required to provide “support service” for the product. What kind of “support service”do you mean? (#SupportService)

This includes the kind of service many device manufacturers provide to help you install, use, or troubleshoot the product. If a device relies on access to web services or similar technology to function properly, those should normally still be available to modified versions, subject to the terms in section 6 regarding access to a network.

In GPLv3 and AGPLv3, what does it mean when it says “notwithstanding any other provision of this License”? (#v3Notwithstanding)

This simply means that the following terms prevail over anything else in the license that may conflict with them. For example, without this text, some people might have claimed that you could not combine code under GPLv3 with code under AGPLv3, because the AGPL's additional requirements would be classified as “further restrictions” under section 7 of GPLv3. This text makes clear that our intended interpretation is the correct one, and you can make the combination.

This text only resolves conflicts between different terms of the license. When there is no conflict between two conditions, then you must meet them both. These paragraphs don't grant you carte blanche to ignore the rest of the license—instead they're carving out very limited exceptions.

Under AGPLv3, when I modify the Program under section 13, what Corresponding Source does it have to offer? (#AGPLv3CorrespondingSource)

“Corresponding Source” is defined in section 1 of the license, and you should provide what it lists. So, if your modified version depends on libraries under other licenses, such as the Expat license or GPLv3, the Corresponding Source should include those libraries (unless they are System Libraries). If you have modified those libraries, you must provide your modified source code for them.

The last sentence of the first paragraph of section 13 is only meant to reinforce what most

people would have naturally assumed: even though combinations with code under GPLv3 are handled through a special exception in section 13, the Corresponding Source should still include the code that is combined with the Program this way. This sentence does not mean that you *only* have to provide the source that's covered under GPLv3; instead it means that such code is *not* excluded from the definition of Corresponding Source.

In AGPLv3, what counts as “interacting with [the software] remotely through a computer network?” (#AGPLv3InteractingRemotely)

If the program is expressly designed to accept user requests and send responses over a network, then it meets these criteria. Common examples of programs that would fall into this category include web and mail servers, interactive web-based applications, and servers for games that are played online.

If a program is not expressly designed to interact with a user through a network, but is being run in an environment where it happens to do so, then it does not fall into this category. For example, an application is not required to provide source merely because the user is running it over SSH, or a remote X session.

How does GPLv3's concept of “you” compare to the definition of “Legal Entity” in the Apache License 2.0? (#ApacheLegalEntity)

They're effectively identical. The definition of “Legal Entity” in the Apache License 2.0 is very standard in various kinds of legal agreements—so much so that it would be very surprising if a court did not interpret the term in the same way in the absence of an explicit definition. We fully expect them to do the same when they look at GPLv3 and consider who qualifies as a licensee.

In GPLv3, what does “the Program” refer to? Is it every program ever released under GPLv3? (#v3TheProgram)

The term “the Program” means one particular work that is licensed under GPLv3 and is received by a particular licensee from an upstream licensor or distributor. The Program is the particular work of software that you received in a given instance of GPLv3 licensing, as you received it.

“The Program” cannot mean “all the works ever licensed under GPLv3”; that interpretation makes no sense for a number of reasons. We've published an analysis of the term “the Program” for those who would like to learn more about this.

If I only make copies of a GPL-covered program and run them, without distributing or conveying them to others, what does the license require of me? (#NoDistributionRequirements)

Nothing. The GPL does not place any conditions on this activity.

If some network client software is released under AGPLv3, does it have to be able to provide source to the servers it interacts with? (#AGPLv3ServerAsUser)

AGPLv3 requires a program to offer source code to “all users interacting with it remotely through a computer network.” It doesn't matter if you call the program a “client” or a “server,” the question you need to ask is whether or not there is a reasonable expectation that a person will be interacting with the program remotely over a network.

For software that runs a proxy server licensed under the AGPL, how can I provide an offer of source to users interacting with that code? (#AGPLProxy)

For software on a proxy server, you can provide an offer of source through a normal method o delivering messages to users of that kind of proxy. For example, a Web proxy could use a landing page. When users initially start using the proxy, you can direct them to a page with the

offer of source along with any other information you choose to provide.

The AGPL says you must make the offer to “all users.” If you know that a certain user has already been shown the offer, for the current version of the software, you don't have to repeat it to that user again.

How are the various GNU licenses compatible with each other? (#AllCompatibility)

The various GNU licenses enjoy broad compatibility between each other. The only time you may not be able to combine code under two of these licenses is when you want to use code that's *only* under an older version of a license with code that's under a newer version.

Below is a detailed compatibility matrix for various combinations of the GNU licenses, to provide an easy-to-use reference for specific cases. It assumes that someone else has written some software under one of these licenses, and you want to somehow incorporate code from that into a project that you're releasing (either your own original work, or a modified version o someone else's software). Find the license for your project in a column at the top of the table, and the license for the other code in a row on the left. The cell where they meet will tell you whether or not this combination is permitted.

When we say “copy code,” we mean just that: you're taking a section of code from one source, with or without modification, and inserting it into your own program, thus forming a work based on the first section of code. “Use a library” means that you're not copying any source directly, but instead interacting with it through linking, importing, or other typical mechanisms that bind the sources together when you compile or run the code.

Each place that the matrix states GPLv3, the same statement about compatibility is true for AGPLv3 as well.

	I want to license my code under:					
	GPLv2 only	**GPLv2 or later**	**GPLv3 or later**	**LGPLv2.1 only**	**LGPLv2.1 or later**	**LGPLv3 or later**

I want to copy code under:	**GPLv2 only**	OK	OK [2]	NO	OK: Combination is under GPLv2 only [7]	OK: Combination is under GPLv2 only [7][2]	NO
	GPLv2 or later	OK [1]	OK	OK	OK: Combination is under GPLv2 or later [7]	OK: Combination is under GPLv2 or later [7]	OK: Combination is under GPLv3 [8]
	GPLv3	NO	OK: Combination is under GPLv3 [3]	OK	OK: Combination is under GPLv3 [7]	OK: Combination is under GPLv3 [7]	OK: Combination is under GPLv3 [8]
	LGPLv2.1 only	OK: Convey copied code under GPLv2 [7]	OK: Convey copied code under GPLv2 or later [7]	OK: Convey copied code under GPLv3 or later [7]	OK	OK [6]	OK: Convey copied code under GPLv3 [7] [8]
	LGPLv2.1 or later	OK: Convey copied code under GPLv2 [7] [1]	OK: Convey copied code under GPLv2 or later [7]	OK: Convey code under GPLv3 or later [7]	OK [5]	OK	OK
	LGPLv3	NO	OK: Combination is under GPLv3 [8] [3]	OK: Combination is under GPLv3 [8]	OK: Combination is under GPLv3 [7] [8]	OK: Combination is under LGPLv3 [4]	OK: Combination is under LGPLv3
I want to use a library under:	**GPLv2 only**	OK	OK [2]	NO	OK: Combination is under GPLv2 only [7]	OK: Combination is under GPLv2 only [7][2]	NO
	GPLv2 or later	OK [1]	OK	OK	OK: Combination is under GPLv2 or later [7]	OK: Combination is under GPLv2 or later [7]	OK: Combination is under GPLv3 [8]
	GPLv3	NO	OK: Combinatio	OK	OK: Combinatio	OK: Combinatio	OK: Combinatio

			n is under GPLv3 [3]		n is under GPLv3 [7]	n is under GPLv3 [7]	n is under GPLv3 [8]
	LGPLv2.1 only	OK	OK	OK	OK	OK	OK
	LGPLv2.1 or later	OK	OK	OK	OK	OK	OK
	LGPLv3	NO	OK: Combination is under GPLv3 [9]	OK	OK	OK	OK

Skip footnotes

1: You must follow the terms of GPLv2 when incorporating the code in this case. You cannot take advantage of terms in later versions of the GPL.

2: While you may release under GPLv2-or-later both your original work, and/or modified versions of work you received under GPLv2-or-later, the GPLv2-only code that you're using must remain under GPLv2 only. As long as your project depends on that code, you won't be able to upgrade the license of your own code to GPLv3-or-later, and the work as a whole (any combination of both your project and the other code) can only be conveyed under the terms of GPLv2.

3: If you have the ability to release the project under GPLv2 or any later version, you can choose to release it under GPLv3 or any later version—and once you do that, you'll be able to incorporate the code released under GPLv3.

4: If you have the ability to release the project under LGPLv2.1 or any later version, you can choose to release it under LGPLv3 or any later version—and once you do that, you'll be able to incorporate the code released under LGPLv3.

5: You must follow the terms of LGPLv2.1 when incorporating the code in this case. You cannot take advantage of terms in later versions of the LGPL.

6: If you do this, as long as the project contains the code released under LGPLv2.1 only, you will not be able to upgrade the project's license to LGPLv3 or later.

7: LGPLv2.1 gives you permission to relicense the code under any version of the GPL since GPLv2. If you can switch

the LGPLed code in this case to using an appropriate version of the GPL instead (as noted in the table), you can make this combination.

8: LGPLv3 is GPLv3 plus extra permissions that you can ignore in this case.

9: Because GPLv2 does not permit combinations with LGPLv3, you must convey the project under GPLv3's terms in this case, since it will allow that combination.

GNU Lesser PUBLIC LISENZ Version 3

Deutsch
Englisch

Deutsche Übersetzung der Version 3, 29. Juni 2007

Den offiziellen englischen Originaltext finden Sie unter http://www.gnu.org/licenses/lgpl-3.0.html.

Deutsche Übersetzung: Peter Gerwinski, *31.7.2007*

Dies ist eine inoffizielle deutsche Übersetzung der GNU Lesser General Public License, die nicht von der Free Software Foundation herausgegeben wurde. Es handelt sich hierbei nicht um eine rechtsgültige Festlegung der Bedingungen für die Weitergabe von Software, die der GNU LGPL unterliegt; dies leistet nur der englische Originaltext. Wir hoffen jedoch, daß diese Übersetzung deutschsprachigen Lesern helfen wird, die GNU LGPL besser zu verstehen.

This is an unofficial translation of the GNU Lesser General Public License into German. It was not published by the Free Software Foundation, and does not legally state the distribution terms for software that uses the GNU LGPL—only the original English text of the GNU LGPL does that. However, we hope that this translation will help German speakers understand the GNU LGPL better.

GNU Lesser General Public License

Deutsche Übersetzung der Version 3, 29. Juni 2007

Diese Übersetzung ist kein rechtskräftiger Ersatz für die englischsprachige Originalversion!

Diese Version der GNU Lesser General Public License umfaßt die Bedingungen von Version 3 der GNU General Public License, ergänzt um die unten aufgelisteten zusätzlichen Genehmigungen.

0. Zusätzliche Definitionen.

Nachstehend bezeichnet „diese Lizenz“ die GNU Lesser General Public License, Version 3, und „GNU GPL“ die GNU General Public License, Version 3.

„Die Bibliothek“ steht für ein betroffenes Werk unter dieser Lizenz, bei dem es nicht um eine Anwendung oder um ein kombiniertes Werk im Sinne der untenstehenden Definitionen handelt.

Eine „Anwendung“ ist irgendein Werk, das eine von der Bibliothek bereitgestellte Schnittstelle nutzt, ansonsten aber nicht auf der Bibliothek basiert. Die Definition einer abgeleiteten Klasse einer von der Bibliothek bereitgestellten Klasse wird als eine Weise betrachtet, eine von der Bibliothek bereitgestellte Schnittstelle zu nutzen.

Ein „kombiniertes Werk“ ist ein Werk, das durch das Kombinieren oder Linken einer Anwendung mit der Bibliothek erzeugt wurde. Die spezifische Version der Bibliothek, mit der zusammen das kombinierte Werk erzeugt wurde, wird auch „gelinkte Version“ genannt.

Der „Minimalquelltext“ eines kombinierten Werks bezeichnet den korrespondierenden Quelltext des kombinierten Werks, ausgenommen den Quelltext von Teilen des kombinierten Werks, die, einzeln betrachtet, auf der Anwendung basieren und nicht auf der gelinkten Version.

Der „korrespondierende Anwendungs-Code“ eines kombinierten Werks bezeichnet den Objekt-Code und/oder Quelltext der Anwendung einschließlich aller Daten und Hilfsprogramme, die benötigt werden, um das kombinierte Werk anhand der Anwendung zu reproduzieren, mit Ausnahme der Systembibliotheken des kombinierten Werks.

1. Ausnahmen von §3 der GNU GPL

Sie dürfen ein betroffenes Werk gemäß §§3 und 4 dieser Lizenz übertragen, ohne an §3 der GNU GPL gebunden zu sein.

2. Übertragung modifizierter Versionen

Wenn Sie ein Exemplar der Bibliothek modifizieren und sich eine Routine *(“facility”)* innerhalb dieser Modifikationen auf eine Funktion oder auf Daten bezieht, die von einer Anwendung bereitgestellt werden, die die Bibliothek nutzt (auf eine andere Weise als in Gestalt eines Arguments, das beim Aufruf der Routine übergeben wird), dann dürfen Sie eine Kopie der modifizierten Version folgendermaßen übertragen:

a) gemäß dieser Lizenz, sofern Sie sich in gutem Glauben darum bemühen, sicherzustellen, daß die Routine weiterh und denjenigen Teil ihres Zweckes, der sinnvoll bleibt, weiterhin ausführt, oder

b) gemäß der GNU GPL, wobei keine der zusätzlichen Genehmigungen dieser Lizenz bei dieser Kopie greifen.

3. Objekt-Code, der Material aus Bibliotheks-Header-Dateien enthält

Die Objekt-Code-Form einer Anwendung darf Material aus einer Header-Datei enthalten, die Teil der Bibliothek ist. Sie dürfen derartigen Objekt-Code gemäß Bedingungen Ihrer Wahl übertragen, vorausgesetzt – sofern das enthaltene Material nicht auf numerische Parameter, Datenstrukturanordnungen und -zugriffsfunktionen oder kleine Makros, Inline-Funktionen und Templates (zehn oder weniger Zeilen lang) beschränkt ist –, Sie führen die beiden folgenden Handlungen aus:

a) Versehen Sie jedes Exemplar des Objekt-Codes mit einem prominenten Hinweis, daß die Bibliothek darin verwend abgedeckt werden.

b) Legen Sie dem Objekt-Code ein Exemplar der GNU GPL und dieses Lizenzdokuments bei.

4. Kombinierte Werke

Sie dürfen ein betroffenes Werk unter Bedingungen Ihrer Wahl übertragen, die insgesamt die Modifikation der in dem kombinierten Werk enthaltenen Teile der Bibliothek und das Zurückbilden *(“reverse engineering”)*, um derartige Modifikationen von Fehlern zu bereinigen, nicht wirksam einschränken, wenn Sie außerdem alle folgenden Handlungen ausführen:

a) Versehen Sie jedes Exemplar des kombinierten Werks mit einem prominenten Hinweis, daß die Bibliothek darin ve abgedeckt werden.

b) Legen Sie dem kombinierten Werk ein Exemplar der GNU GPL und dieses Lizenzdokuments bei.

c) Für ein kombiniertes Werk, das bei Ausführung Copyright-Hinweise anzeigt, fügen Sie den Copyright-Hinweis für d den Exemplaren der GNU GPL und dieses Lizenzdokuments führt.

d) Führen Sie eine der folgenden Handlungen aus:

0. Übertragen Sie den korrespondierenden Minimalquelltext gemäß den Bedingungen dieser Lizenz und den ko die den Anwender erlauben, die Anwendung mit einer modifizierten Version der gelinkten Version neu zu kon erzeugen, auf eine Weise, wie sie in §6 der GNU GPL spezifiziert ist, um korrespondierenden Quelltext zu üb
1. Verwenden Sie einen geeigneten Shared-Library-Mechanismus, um mit der Bibliothek zu linken. Ein geeigne Bibliothek verwendet, das sich bereits auf dem Computer des Anwenders befindet, und (b) mit einer modifizie schnittstellenkompatibel ist, korrekt arbeiten wird.

e) Stellen Sie Installationsinformationen zur Verfügung – allerdings nur dann, wenn Sie dazu ansonsten gemäß §6 de Informationen benötigt werden, um eine modifizierte Version des kombinierten Werks installieren und ausführen zu modifizierten Version der gelinkten Version erzeugt wurde. (Wenn Sie Option 4d0 verwenden, müssen die Installati Code beiliegen. Wenn Sie Option 4d1 verwenden, müssen Sie die Installationsinformationen in einer Form zur Verf korrespondierenden Quelltextes spezifiziert wurde.)

5. Kombinierte Bibliotheken

Sie dürfen Routinen aus der Bibliothek, die ein auf der Bibliothek basierendes Werk darstellen, mit anderen Bibliotheks-Routinen, die keine Anwendungen sind und die nicht unter dieser Lizenz stehen, in einer einzelnen Bibliothek nebeneinanderstellen und eine derartige kombinierte Bibliothek unter Bedingungen Ihrer Wahl übertragen,

wenn Sie die beiden folgenden Handlungen ausführen:

- **a)** Legen Sie der kombinierten Bibliothek ein Exemplar desselben auf der Bibliothek basierenden Werks bei, das von Bedingungen dieser Lizenz übertragen wird.
- **b**) Versehen Sie die kombinierte Bibliothek mit einem prominenten Hinweis, daß Teile davon ein auf der Bibliothek ba Form desselben Werks zu finden ist.

6. Überarbeitungen der GNU Lesser General Public License

Die Free Software Foundation kann von Zeit zu Zeit überarbeitete und/oder neue Versionen der *GNU Lesser General Public License* veröffentlichen. Solche neuen Versionen werden vom Grundprinzip her der gegenwärtigen entsprechen, können aber im Detail abweichen, um neuen Problemen und Anforderungen gerecht zu werden.

Jede Version hat eine eindeutige Versionsnummer. Wenn eine Bibliothek, wie Sie sie erhalten haben, angibt, daß eine bestimmte Versionsnummer der GNU Lesser General Public License „oder irgendeine spätere Version“*(“or any later version”)* darauf anwendbar ist, so haben Sie die Wahl, entweder den Bestimmungen der genannten Version zu folgen oder denen jeder beliebigen späteren Version, die von der Free Software Foundation veröffentlicht wurde. Wenn die Bibliothek, wie Sie sie erhalten haben, keine Versionsnummer angibt, können Sie irgendeine Version der GNU Lesser General Public License wählen, die je von der Free Software Foundation veröffentlicht wurde.

Wenn die Bibliothek, wie Sie sie erhalten haben, bestimmt, daß ein Bevollmächtigter entscheiden kann, ob zukünftige Versionen der GNU Lesser General Public License anwendbar sein sollen, dann ist eine öffentliche Stellungnahme der Akzeptanz einer beliebigen Version für Sie eine permanente Erlaubnis, diese Version für die Bibliothek auszuwählen.

* * *

GNU LESSER GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

0. Additional Definitions.

As used herein, "this License" refers to version 3 of the GNU Lesser General Public License, and the "GNU GPL" refers to version 3 of the GNU General Public License.
"The Library" refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.
An "Application" is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library. Defining a subclass of a class defined by the Library is deemed a mode of using an interface provided by the Library.

A "Combined Work" is a work produced by combining or linking an Application with the Library. The particular version of the Library with which the Combined Work was made is also called the "Linked Version".

The "Minimal Corresponding Source" for a Combined Work means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version.

The "Corresponding Application Code" for a Combined Work means the object code and/or source code for the Application, including any data and utility programs needed for reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work.

1. Exception to Section 3 of the GNU GPL.

You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL.

2. Conveying Modified Versions.

If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:

a) under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or

- b) under the GNU GPL, with none of the additional permissions of this License applicable to that copy.

3. Object Code Incorporating Material from Library Header Files.

The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:

- a) Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License.
- b) Accompany the object code with a copy of the GNU GPL and this license document.

4. Combined Works.

You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:

- a) Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License.
- b) Accompany the Combined Work with a copy of the GNU GPL and this license document.
- c) For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document.
- d) Do one of the following:
    - 0) Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.
    - 1) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user's computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version.
- e) Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)

5. Combined Libraries.

You may place library facilities that are a work based on the Library side by side in a single library together with other library facilities that are not Applications and are not covered by this License, and convey such a combined library under terms of your choice, if you do both of the following:

- a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities, conveyed under the terms of this License.

- b) Give prominent notice with the combined library that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

6. Revised Versions of the GNU Lesser General Public License.

The Free Software Foundation may publish revised and/or new versions of the GNU Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library as you received it specifies that a certain numbered version of the GNU Lesser General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that published version or of any later version published by the Free Software Foundation. If the Library as you received it does not specify a version number of the GNU Lesser General Public License, you may choose any version of the GNU Lesser General Public License ever published by the Free Software Foundation.

If the Library as you received it specifies that a proxy can decide whether future versions of the GNU Lesser General Public License shall apply, that proxy's public statement of acceptance of any version is permanent authorization for you to choose that version for the Library.

GNU PUBLIC LISENZ Version 2.1

Deutsch
Englisch

MIT-Lizenzen / MIT Licenses

MIT-Lizenz / MIT License

Deutsche Übersetzung (inoffiziell)

Original English Version

Traduction française (traduction non officielle)

Traducción al español (traducción no oficial)

Tłumaczenie na język polski (nieoficjalne tłumaczenie)

Русский перевод (неофициальный перевод)

HESSENBERG, GERHARD
Grundbegriffe der Mengenlehre

Aufgearbeitet von Jens Kallup
Januar: 2026

Liste der Tabellen

Vorwort

Das vorliegende Referat über das Unendliche in der Mathematik war ursprünglich als Fortsetzung des im ersten Heft erschienenen Berichts gedacht. Es sollte zeigen, daß mit der Ausschaltung aktual unendlicher Größen aus den Grenzmethoden, insbesondere aus der Infinitesimalrechnung, die Mathematik keineswegs auf die Betrachtung des aktual Unendlichen überhaupt verzichtet. Vielmehr sollte das Beispiel der Nichtabzählbarkeit des Kontinuums die Möglichkeit der Unterscheidung verschiedener unendlicher Mächtigkeiten, und der daraus folgende Cantorsche Beweis der Existenz transzendenter Zahlen die praktische Bedeutung dieser Unterscheidung dartun.

Das Referat wuchs aber während der Ausarbeitung dauernd; das augenblicklich stark zunehmende Interesse an mengentheoretischen Untersuchungen veranlaßte mich schließlich, den Bericht geradezu zu einer Einleitung in die Grundbegriffe des betrachteten Gebietes auszugestalten. Darüber hinauszugehen und etwa noch die mathematischen Anwendungen in größerem Umfange darzustellen verbot aber die Rücksicht auf den nicht ausschließlich mathematischen Leserkreis dieser Zeitschrift. Andererseits liegt hierüber der ausführliche Schoenfliesche Bericht in den Jahresberichten der Deutschen Mathematikervereinigung vor, und endlich hätte der unvermeidliche Literaturnachweis die Fertigstellung der Arbeit ins Unangemessene verzögert.

Den mengentheoretischen Kalkül wollte ich ursprünglich nicht in den Umfang des Berichtes einbeziehen. Es zwangen mich aber zwei Gründe dazu: Einmal die Paradoxie der Menge aller Ordnungszahlen, die mir die klarste und schärfste Fassung des ultrafiniten Paradoxons zu sein scheint, das unter anderen von Russell in so zahlreichen Formen gebracht worden ist. Da hierfür Darstellung bedarf man immerhin des Begriffs der Ordnung; und damit der Widerspruch nicht in diesem Begriff selbst gesucht werde, muß gezeigt werden, welche umfangreiches Gebiet des widerspruchsfreien Kalküls durch ihn eröffnet wird.

Der zweite Grund, der mich zur Darstellung des Kalküls veranlaßte, war die Frage der Erzeugungsprinzipien. Bei diesen ist der Satz von Bedeutung, daß jede Mächtigkeit, die eine unmittelbar vorangehende besitzt, eine neue Prinzip verlangt; und hierfür muß man zeigen können, daß das Quadrat jeder Mächtigkeit ihr selbst gleich ist. Ausgesprochen ist dieser Satz nach einer Mitteilung von Herrn Bernstein zuerst von Herrn Georg Cantor. Ob der in dieser Mitteilung flüchtig skizzierte Beweis derselbe ist, den ich hier darstelle, vermag ich nicht zu beurteilen. Da ferner die Sätze über den Kalkül mit transfiniter Ordnungszahlen von Herrn Cantor nur für die zweite Zahlklasse ausgesprochen sind und auch Herr Schoenflies in der *Encyklopädie der mathematischen Wissenschaften* sich ausschließlich mit dieser beschäftigt, hielt ich es für angebracht, die Gültigkeit des ganzen Kalküls für jede Zahlklasse zum Ausdruck zu bringen; ich verhehle mir nicht, daß dieser Teil über den transfiniten Kalkül in erster Linie nur für Mathematiker Interesse haben kann, und habe mich daher bemüht, die späteren Kapitel nach Möglichkeit unabhängig von ihm zu gestalten, so daß der mathematisch ungeschulte Leser ihn überschlagen kann.

Wer auf Konsequenz und Geschlossenheit der Darstellung Wert legt, wird nicht damit einverstanden sein, daß die endlichen Zahlen zunächst als etwas Bekanntes angenommen werden und die Theorie der unendlichen Mengen vielfach auf sie gestützt wird. Wäre dieser Bericht bloß für Mathematiker bestimmt, so hätte ich, wenn ich ihn dann überhaupt zu schreiben für nötig gehalten hätte, vielleicht anders angeordnet und die im letzten Teil gegebenen Theorien der endlichen Zahlen vorangestellt. Ich hätte auf diesem Wege auch nicht die schönen Dedekindschen Betrachtungen in verschiedene Kapitel zu zerstreuen brauchen. Daß ich eine andere Anordnung zu Grunde gelegt habe, geschah aus der festen Überzeugung heraus, daß dadurch das Verständnis des schwierigen Gegenstandes wesentlich erleichtert wird.

Zu besonderem Danke bin ich Herrn Zermelo verpflichtet für die Durchsicht der Korrekturen; vor allem aber für die Mitteilung eigener, noch unveröffentlichter Untersuchungen und die Erlaubnis, von ihrem Gebrauch zu machen. Zu diesen gehört auch der schöne Satz XX, Seite 639, bei dem im Text versehentlich der Hinweis auf den Urheber unterblieben ist. Ferner machte mich Herr Zermelo darauf aufmerksam, daß der Beweis des § 119 (ebenso wie der entsprechende bei Dedekind, „Was sind und was sollen die Zahlen?“) in versteckter Weise von dem in § 102 und § 137 besprochenen Auswahlpostulat Gebrauch macht; es war leider nicht mehr möglich, dies im Text hervorzuheben.

Vorwort

Das vorliegende Referat über das Unendliche in der Mathematik war ursprünglich als Fortsetzung des im ersten Heft erschienenen Berichts gedacht. Es sollte zeigen, daß mit der Ausschaltung aktual unendlicher Größen aus den Grenzmethoden, insbesondere aus der Infinitesimalrechnung, die Mathematik keineswegs

auf die Betrachtung des aktual Unendlichen überhaupt verzichtet. Vielmehr sollte das Beispiel der Nichtabzählbarkeit des Kontinuums die Möglichkeit der Unterscheidung verschiedener unendlicher Mächtigkeiten, und der daraus folgende Cantorsche Beweis der Existenz transzendenter Zahlen die praktische Bedeutung dieser Unterscheidung dartun.

Das Referat wuchs aber während der Ausarbeitung dauernd; das augenblicklich stark zunehmende Interesse an mengentheoretischen Untersuchungen veranlaßte mich schließlich, den Bericht geradezu zu einer Einleitung in die Grundbegriffe des betrachteten Gebietes auszugestalten. Darüber hinauszugehen und etwa noch die mathematischen Anwendungen in größerem Umfange darzustellen verbot aber die Rücksicht auf den nicht ausschließlich mathematischen Leserkreis dieser Zeitschrift. Andererseits liegt hierüber der ausführliche Schoenfliesche Bericht in den Jahresberichten der Deutschen Mathematikervereinigung vor, und endlich hätte der unvermeidliche Literaturnachweis die Fertigstellung der Arbeit ins Unangemessene verzögert.

Den mengentheoretischen Kalkül wollte ich ursprünglich nicht in den Umfang des Berichtes einbeziehen. Es zwangen mich aber zwei Gründe dazu: Einmal die Paradoxie der Menge aller Ordnungszahlen, die mir die klarste und schärfste Fassung des ultrafiniten Paradoxons zu sein scheint, das unter anderen von Russell in so zahlreichen Formen gebracht worden ist. Da hierfür Darstellung bedarf man immerhin des Begriffs der Ordnung; und damit der Widerspruch nicht in diesem Begriff selbst gesucht werde, muß gezeigt werden, welche umfangreiches Gebiet des widerspruchsfreien Kalküls durch ihn eröffnet wird.

Der zweite Grund, der mich zur Darstellung des Kalküls veranlaßte, war die Frage der Erzeugungsprinzipien. Bei diesen ist der Satz von Bedeutung, daß jede Mächtigkeit, die eine unmittelbar vorangehende besitzt, eine neue Prinzip verlangt; und hierfür muß man zeigen können, daß das Quadrat jeder Mächtigkeit ihr selbst gleich ist. Ausgesprochen ist dieser Satz nach einer Mitteilung von Herrn Bernstein zuerst von Herrn Georg Cantor. Ob der in dieser Mitteilung flüchtig skizzierte Beweis derselbe ist, den ich hier darstelle, vermag ich nicht zu beurteilen. Da ferner die Sätze über den Kalkül mit transfiniter Ordnungszahlen von Herrn Cantor nur für die zweite Zahlklasse ausgesprochen sind und auch Herr Schoenflies in der *Encyklopädie der mathematischen Wissenschaften* sich ausschließlich mit dieser beschäftigt, hielt ich es für angebracht, die Gültigkeit des ganzen Kalküls für jede Zahlklasse zum Ausdruck zu bringen; ich verhehle mir nicht, daß dieser Teil über den transfiniten Kalkül in erster Linie nur für Mathematiker Interesse haben kann, und habe mich daher bemüht, die späteren Kapitel nach Möglichkeit unabhängig von ihm zu gestalten, so daß der mathematisch ungeschulte Leser ihn überschlagen kann.

Wer auf Konsequenz und Geschlossenheit der Darstellung Wert legt, wird nicht damit einverstanden sein, daß die endlichen Zahlen zunächst als etwas Bekanntes angenommen werden und die Theorie der unendlichen Mengen vielfach auf sie gestützt wird. Wäre dieser Bericht bloß für Mathematiker bestimmt, so hätte ich, wenn ich ihn dann überhaupt zu schreiben für nötig gehalten hätte, vielleicht anders angeordnet und die im letzten Teil gegebenen Theorien der endlichen Zahlen vorangestellt. Ich hätte auf diesem Wege auch nicht die schönen Dedekindschen Betrachtungen in verschiedene Kapitel zu zerstreuen brauchen. Daß ich eine andere Anordnung zu Grunde gelegt habe, geschah aus der festen Überzeugung heraus, daß dadurch das Verständnis des schwierigen Gegenstandes wesentlich erleichtert wird.

Zu besonderem Danke bin ich Herrn Zermelo verpflichtet für die Durchsicht der Korrekturen; vor allem aber für die Mitteilung eigener, noch unveröffentlichter Untersuchungen und die Erlaubnis, von ihrem Gebrauch zu machen. Zu diesen gehört auch der schöne Satz XX, Seite 639, bei dem im Text versehentlich der Hinweis auf den Urheber unterblieben ist. Ferner machte mich Herr Zermelo darauf aufmerksam, daß der Beweis des § 119 (ebenso wie der entsprechende bei Dedekind, „Was sind und was sollen die Zahlen?“) in versteckter Weise von dem in § 102 und § 137 besprochenen Auswahlpostulat Gebrauch macht; es war leider nicht mehr möglich, dies im Text hervorzuheben.

Erster Teil
Die Grundbegriffe der Teilung, Vergleichung und Ordnung.

I.
Das Paradoxon der Winkelvergleichung.

1. Im ersten Hefte dieser Zeitschrift ist in einem Referat über das Unendliche in der Mathematik dargetan worden, daß weder in den elementaren noch in den als „Infinitesimalrechnung“ bezeichneten Kapiteln der Mathematik eine wirklich unendliche „Größe“ auftritt; daß vielmehr das Wort „unendlich“ lediglich zur abkürzenden Beschreibung wichtiger Tatsachen des endlichen benutzt wird.

Dort bot sich im zweiten Kapitel bei Gelegenheit der Winkelmessung ein Beispiel einer Paradoxie des Unendlichen, die dadurch ausgeschaltet wurde, daß Winkel lediglich durch Scheitelstrahlen verglichen wurden. Dieses Paradoxon der Winkelmessung soll hier ausführlich und daher wieder als Ausgangspunkt unserer Betrachtung dienen.

Es sei PQPQPQ ein beliebiger, spitzer oder stumpfer Winkel, RRR ein Punkt des Schenkels OPOPOP. Wir ziehen durch RRR in das Innere des Winkels eine Parallele zu RSRSRS an OQOQOQ. Sie zerlegt die Fläche des Winkels in den Streifen PQRSPQRSPQRS und in den Winkel

(Abbildung der Winkelteilung. S. 84.)

§ 1. PRSPRSPRS, der dem Winkel POQPOQPOQ gleich ist und durch Verschieben mit ihm zur Deckung gebracht werden kann.

Nach dem Grundsatz „Der Teil kleiner als das Ganze“ ist andererseits der Winkel PRSPRSPRS kleiner als der Winkel POQPOQPOQ. Da die Begriffe „kleiner“ und „gleich“ sich ausschließen, entsteht ein Widerspruch.

§ 2. Es wird vielfach versucht, den Widerspruch durch folgende Argumentation zu beseitigen: Das Ebenenstück POQPOQPOQ ist unendlich, also kein Ganzes. Demnach kann der Grundsatz von Teil und Ganzem nicht angewandt werden.

Diese Argumentation hat den Fehler, daß sie zuviel beweist. Da sie nämlich im weiteren von Gebrauch macht, daß die verlagernde Halbstrahl RSRSRS am Punkte OOO vorbeigeht, würde sie auch auf den Fall der Zerlegung durch Scheitelstrahlen treffen, und es wäre damit nachgewiesen, daß Winkel überhaupt nicht verglichen werden können. Außerdem kollidiert dieser Gebrauch der Begriffe Teil und Ganzes mit dem vulgären Sprachgebrauch: Teil und Ganzes sind korrelative Begriffe, und da das Ebenenstück POQPOQPOQ offenbar Teile besitzt, ist es selbst das Ganze, von dem diese Teile genommen werden.

Der Mathematiker im besonderen hat eine tiefe Abneigung gegen solche Argumentation mit Allgemeinbegriffen. Wenn überhaupt irgend ein Gegenstand „ganz“ ist, so genügt es dafür, daß er sein festes, was zu ihm gehört und was nicht. Das Zugehören selbst ist aber wieder ein Allgemeinbegriff, und es sei daher an speziellen Beispielen ausdrücklich festgestellt, daß von jedem Punkt der Ebene feststeht, ob er im Innern, im Äußern oder auf der Grenze von POQPOQPOQ liegt. Darum gilt POQPOQPOQ mathematisch als Ganzes. Da weiterhin jeder Punkt PRSPRSPRS ein Punkt von POQPOQPOQ, aber nicht jeder Punkt von POQPOQPOQ einer von PRSPRSPRS ist, gilt PRSPRSPRS als Teil von POQPOQPOQ.

§ 3. Wollen wir also den falschen Schluß, der unseren Widerspruch verursacht, wirklich scharf herauspräparieren, so müssen wir die allgemeinen Begriffe, mit denen wir operieren, näher untersuchen und finden drei Grundbeziehungen: die Vergleichung, der das Wort „gleich“ entspricht, die Ordnung, der das Wort „kleiner“ entspricht, und endlich die Beziehung des Teilens zum Ganzen. Die Bedeutung dieser drei Beziehungen und ihre Verknüpfung untereinander reicht natürlich weit über das Paradoxon der Winkelvergleichung hinaus; die Ausführlichkeit der folgenden Betrachtungen ist nicht durch das einzelne Beispiel geboten.

Wir werden bei der Untersuchung unserer drei Grundbeziehungen auf eine Definition derselben zunächst verzichten, da eine solche, falls sie überhaupt möglich ist, an Stelle der zu prüfenden lediglich neue Allgemeinbegriffe von gleicher oder größerer Verschwommenheit zum Ausgangspunkt nehmen würde. Dagegen fragen wir, der kritisch-mathematischen Methode folgend, bei jedem unserer Begriffe nach den Grundsätzen, in denen er auftritt, nach denjenigen Eigenschaften, die eine Beziehung besitzen muß, damit der Name einer Vergleichung, Ordnung oder Teilung zukommen kann.

II.

Die Teilung und die Vergleichung.

§ 4. Welcher Art die Beziehung auch sei, die wir mit dem Wort „A ist ein Teil von B“ bezeichnen, sie wird folgenden Grundsätzen genügen müssen:

Ia. Ist AAA ein Teil von BBB, BBB ein Teil von CCC, so ist AAA ein Teil von CCC.

Da man zwischen den Dingen AAA mit einem eigenen Teil von CCC unendlich vielen oder „unechten“ Teilen hinzunimmt, wollen wir jeden mit AAA nicht identischen Teil von AAA einen „eigentlichen“ oder „echten“ Teil nennen. Diese Trennung ist für Satz Ia nicht erforderlich, wohl aber für folgende zwei Sätze:

IIa. Ist AAA ein eigentlicher Teil von BBB, so ist BBB kein Teil von AAA.

IIIa. Ist AAA mit BBB identisch, so ist BBB kein eigentlicher Teil von AAA.

Satz IIa handelt von der Umkehrung, Satz IIIa von dem Verhältnis zur Identität.

Ganz analoge Sätze gelten von der Vergleichung. Da aber keineswegs alle Vergleichungen durch eine Zeichen- und das Wort „gleich“ ausgedrückt werden, wollen wir hierfür das mengentheoretische Zeichen „~“ und den allgemeinen Terminus „äquivalent“ oder „gleichwertig“ gebrauchen.

Wir haben alsdann folgende Grundsätze:

Ib. Ist A~BA \sim BA~B, B~CB \sim CB~C, so ist A~CA \sim CA~C.

IIb. Ist A~BA \sim BA~B, so ist B~AB \sim AB~A.

IIIb. Ist AAA mit BBB identisch, so ist A~BA \sim BA~B.

Diesen Bedingungen einer Vergleichung genügen in der Geometrie unter anderem die Kongruenz und die Ähnlichkeit von Figuren, die Parallelität von Geraden u. a. Da wir allgemein bestrebt sind, jede Vergleichung als Identität gewisser Eigenschaften darzustellen, konzentrieren wir vielfach solche Eigenschaften auf einen Zweck. So spricht man bei parallelen Geraden von „gleichen Richtungen“, bei ähnlichen Figuren von „gleicher Form“.

---

§ 5. Die Beziehungen der Teilung und der Vergleichung können an der gleichen Gruppe von Dingen auftreten. Von einer Verknüpfung der beiden Beziehungen kann aber dann erst die Rede sein, wenn die Teile der verglichenen Gegenstände auch der Vergleichung unterworfen sind. Z. B. die Gegenstände ebene Polygone, die hinsichtlich ihrer Flächeninhalte vergleichen werden, ihre Teile aber die Punkte des Innern, so findet die Vergleichung auf diese Teile keine Anwendung. Ebenso liegt der Fall beim Winkelmessen. Die Vergleichung bezieht sich auf aneinandergelegte Schenkel. Bei jeder Zerlegung nun, die nicht durch Scheitelstrahlen erzeugt wird, ist mindestens ein Teil unvergleichbar mit dem Ganzen.

Definieren wir dagegen die Teilung der Polygone durch geradlinige Zerschneidung, so sind die Teile wieder Polygone, also wieder vergleichbar. Definieren wir ebenso die Vergleichung der Winkel durch Zerschneiden in kongruente Teile ohne Beschränkung auf Scheitelstrahlen, so sind alle Teile, die durch geradlinige Zerschneidung entstehen, demselben Vergleichungsprinzip zugänglich.

Wir betrachten nun lediglich diesen Fall, da die Teile dem Vergleichungsprinzip zugänglich sind, und stellen folgenden Postulat, dessen Gültigkeit im speziellen Fall erst nachzuweisen ist:

IVa. (Postulat der äquivalenten Teile.) Ist AAA zu BBB äquivalent, und enthält AAA einen Teil A1A_1A1, so enthält BBB einen zu A1A_1A1 äquivalenten Teil B1B_1B1.

Für die weiteren Ausführungen werden wir lediglich Sätze I—IVa und b und IVa verwenden. Die eingestreuten Beispiele sollen nur zur Erläuterung und zum Nachweise dienen, daß dem logischen Formalismus der Sätze wirkliche Beziehungen entsprechen. In jedem einzelnen Anwendungsfalle wird man sich selbstverständlich immer erst zu überzeugen haben, daß die vorhandenen Beziehungen wirklich umgekehrt dem Formalismus I—IVa genügen.

Da es von Wert ist, daß tatsächlich nur die Sätze I—IVa zur Verwendung gelangen, werde ich die nächstfolgenden Beweise zum größeren Teil in ihre Syllogismen zerlegen.

III.

Die Ordnung.

§ 6. Als „Ordnung“ werden alle diejenigen Beziehungen bezeichnet, die wir mit dem Zeichen „<“ und mit Worten wie „größer, kleiner“, „früher, später“, „vor, nach“, „über, unter“, „rechts, links“ ansprechen.

Eine Ordnung muß folgenden Grundsätzen genügen:

Ia. Ist A<BA < BA<B, B<CB < CB<C, so ist A<CA < CA<C.

IIa. Ist A<BA < BA<B, so ist es nicht B<AB < AB<A.

IIIa. Ist AAA mit BBB identisch, so ist es nicht A<BA < BA<B.

Diese Bedingungen unterscheiden sich zunächst nicht durch die Bezeichnung von den der ersten der Teilung, Ia bis IIIa. Der Unterschied zwischen Ordnung und Teilung tritt erst zu Tage, wenn wir die charakteristischen Sätze aufstellen, die die Ordnung mit der Vergleichung verknüpfen. Der erste von diesen schließt wegen IIIb den Satz IIIa ein und lautet:

V. Ist A~BA \sim BA~B, so ist es nicht A<BA < BA<B.

Satz V und die Behauptung unterscheiden die Ausschließung der Begriffe „kleiner, größer“ und „gleich“. Diese Ausschließung ist

* Es ist hier stillschweigend angenommen, daß A<BA < BA<B und B>AB > AB>A verschiedene Ausdrucksweisen für die gleiche Beziehung sind. eine Konzession an den Sprachgebrauch: Wir wollen die Worte „größer“, „kleiner“ nur für Begriffe brauchen, die Gleichheit ausschließen.

Der nächste Satz kann als Satz der äquivalenten Ordnung bezeichnet werden und lautet:

VI. Ist A~A′A \sim A'A~A′, B~B′B \sim B'B~B′ und A<BA < BA<B, so ist auch A′<B′A' < B'A′<B′.

Man kann ihn in zwei Sätze zerlegen, deren jeder aus VI folgt und die zusammen wieder VI ergeben:

VIa. Ist A~BA \sim BA~B, B<CB < CB<C, so ist A<CA < CA<C.

VIb. Ist A<BA < BA<B, B~CB \sim CB~C, so ist A<CA < CA<C.

Gelten VIIa und VIIb, so folgt aus dem zusammengesetzten A~A′A \sim A'A~A′, A<BA < BA<B zunächst A′ <BA' < BA′<B nach VIa, hieraus und aus B~B′B \sim B'B~B′ nach VIb die Behauptung von VII. Nimmt man umgekehrt A~A′A \sim A'A~A′ als identisch, so ergibt Satz VI im übrigen den speziellen Fall von VI, in dem der Identität von BBB mit B′B'B′ entsteht.

Zu diesen Sätzen kommt als letzter ein Satz von prinzipieller Bedeutung:

VIII. (Satz der Trichotomie.) Entweder ist A<BA < BA<B, oder B<AB < AB<A, oder A~BA \sim BA~B.

In § 7. Sätze, die die Ordnung mit Teilung in Beziehung setzen, sind zunächst nicht aufgestellt; denn gerade diese Sätze, wie z. B. der Grundsatz, der Teil sei kleiner als das Ganze, sollen Gegenstand weiterer Untersuchungen sein.

Ehe wir zu diesen übergehen, beweisen wir an einem einfachen Beispiel, daß der Satz von der Trichotomie unabhängig ist von allen vorhergehenden Sätzen der Vergleichung und Ordnung.

Zu diesem Nachweis wählen wir als Gegenstände der Vergleichung die Punkte des Raumes und als Vergleichung die Identität. A=BA = BA=B heißt also, daß AAA und BBB denselben Punkt bezeichnen.

Um die Punkte zu ordnen, greifen wir zunächst ein beliebigen Punkt OOO heraus und setzen fest, daß O<AO < AO<A für jeden von OOO verschiedenen Punkt AAA sein soll. Sind nun AAA und BBB zwei voneinander und von OOO verschiedene Punkte, so schreiben wir A<BA < BA<B, wenn die Strecke OAOAOA kürzer als OBOBOB ist. Wir überzeugen uns sofort von der Gültigkeit der Sätze Ib bis IIIb. Um weiter V und VI zu beweisen, beachten wir, daß aus A=BA = BA=B die Gleichheit der Strecken OAOAOA und OBOBOB folgt. Daß VII aber nicht gilt, erkennt man sofort daran, daß aus der gleichen Länge der Strecken OAOAOA und OBOBOB keineswegs die Identität der Punkte A,BA, BA,B folgt.

Die Unabhängigkeit des Satzes VII läßt sich nicht nachweisen, vielmehr ist VII eine Folge von VIII. Betrachten wir, um dies einzusehen, zunächst VIIb, d. h. die beiden Voraussetzungen:

(1) A~BA \sim BA~B (2) B~CB \sim CB~C.

Die Annahme C<AC < AC<A würde mit (2) nach Ia zu (1) führen, was (1) widerspricht. Die Annahme C~AC \sim AC~A ergäbe mit (1) nach Ib C~BC \sim BC~B, was (2) widerspricht. Der Widerspruch entspringt in beiden Fällen aus V. Da weder C<AC < AC<A noch C~AC \sim AC~A sein kann, folgt aus VIII, daß A<CA < CA<C sein muß. Ebenso beweist man VIa.

IV.

Das Problem der Trichotomie.

§ 8. Wenn wir in das System der bisherigen Sätze noch den Grundsatz hinzunehmen, daß der Teil kleiner als das Ganze sei, so folgt mit Satz VI sofort, daß A<BA < BA<B auch dann statthat, wenn AAA zwar nicht selbst ein Teil von BBB, wohl aber einen Teil von BBB enthält.

* Es kann sich hier im folgenden natürlich nur um eigentliche Teile handeln.

äquivalent ist. Bei den üblichen Vergleichungsmethoden teilbarer Größen gilt aber auch die Umkehrung: Wenn A<BA < BA<B ist, so gibt es in BBB einen zu AAA äquivalenten Teil. Danach liegt es nahe, den Begriff des kleineren überhaupt so zu definieren: AAA heißt kleiner wie BBB, wenn es einen Teil von BBB äquivalent ist.

Trotzdem nun unsere bisherige Entwicklung noch lange nicht alle für den Messung und der Größen charakteristischen Tatsachen enthalten, sind wir bereits in der Lage, die Unzulässigkeit einer solchen Definition zu erkennen. Sie führt nämlich nicht zu einer dreifachen, sondern zu einer vierfachen Disjunktion.

Benutzen wir lediglich das Kriterium, ob AAA einen Teil von BBB äquivalent ist (ohne es mit dem Zeichen A<BA < BA<B auszudrücken), so erhalten wir die Dichotomie: Entweder gibt es einen Teil B1B_1B1 von BBB, so daß A~B1A \sim B_1A~B1, oder es gibt keinen solchen Teil.

Vertauschen wir die Rollen von AAA und BBB, so entsteht eine zweite Dichotomie. Beide Dichotomien vereinigt man leicht zu dem bekannten logischen Schema zu folgender vierfachen Disjunktion:

VIII. Eines der folgenden vier Fälle muß stets zutreffen, und jeder schließt die drei anderen aus:

(A) AAA ist einem Teil von BBB äquivalent.
(B) AAA ist einem Teil von BBB kein Teil von AAA äquivalent.
(C) AAA ist keinem Teil von BBB äquivalent.
(D) AAA ist keinem Teil von BBB, BBB keinem Teil von AAA äquivalent.

Es sind hierbei stets eigentliche Teile gemeint.

§ 9. Wir beweisen zunächst folgenden Satz:
Wird das Eintreten des Falles (B) mit A<BA < BA<B oder B<AB < AB<A bezeichnet, so erfüllt diese so definierte Beziehung alle Anforderungen der Ordnung mit Ausnahme des Satzes VIII von der Trichotomie.

Zunächst weisen wir die Gültigkeit des Postulates V nach:
Ist A~BA \sim BA~B, so ist es nicht A<BA < BA<B. Wäre nämlich A<BA < BA<B, so hieße dies nach VIII (B):

(1) BBB besitzt einen Teil B1B_1B1,
(2) B1~AB_1 \sim AB1A. Im Verein mit
(3) A~BA \sim BA~B
folgt hieraus nach IV:

(4) B1~BB_1 \sim BB1B.

(4) aber besagt, daß BBB einen zu sich selbst äquivalenten echten Teil besitzt, was unmöglich ist.

Aus der Äquivalenz
A~BA \sim BA~B, B~AB \sim AB~A, A~BA \sim BA~B
folgt nun nach Ib, daß A~BA \sim BA~B äquivalent ist, im Widerspruch mit der zweiten Aussage von (B).

Das Postulat IIIa ist ein spezieller Fall von V, da die Identität nach IIIb ein spezieller Fall der Äquivalenz ist.

Das Umkehren des Falles (B) geht aus (B) durch Vertauschung von AAA mit BBB hervor. Das Eintreten dieses Falles ist also durch A>BA > BA>B oder B<AB < AB<A zu bezeichnen. Da (B) und (C) sich logisch ausschließen, ist hiermit das Postulat IIa als erfüllt nachgewiesen.

Das Postulat Ia beweisen wir in zwei Schritten. Wir entnehmen zunächst aus A<BA < BA<B die Voraussetzungen:

(1) BBB enthält einen Teil B1B_1B1, (2) B1~AB_1 \sim AB1A,
und ebenso aus B<CB < CB<C:

(3) CCC enthält einen Teil C1C_1C1, (4) C1~BC_1 \sim BC1B.

Aus (1) und (4) folgt nach IV:

(5) C1C_1C1 enthält einen Teil C2C_2C2, (6) C2~AC_2 \sim AC2A.

Aus (3) und (5) nach Ia:

(7) CCC enthält einen Teil C2C_2C2.

Endlich aus (2) und (6) nach Ib:

(8) C2~AC_2 \sim AC2A.

Damit ist der erste Teil der Behauptung A<CA < CA<C erwiesen. Noch einfacher ist sein Beweis unter den Voraussetzungen der Sätze VIb oder VIa. Im ersten Falle hat man:

(9) A~BA \sim BA~B, (10) CCC enthält einen Teil C1C_1C1, (11) C1~BC_1 \sim BC1B.

Aus (9) und (11) folgt nach Ib:

(12) C1~AC_1 \sim AC1A.

Bei Satz VIa ist vorausgesetzt:

(13) BBB enthält einen Teil B1B_1B1, (14) B1~AB_1 \sim AB1A, (15) C~BC \sim BC~B.

Aus (13) und (15) folgt nach IV:

(16) CCC enthält einen Teil C1C_1C1, (17) C1~B1C_1 \sim B_1C1B1.

und aus (17) und (14) nach Ib:

(18) C1~AC_1 \sim AC1A.

Sicher enthält also CCC einen zu AAA äquivalenten Teil, und es bleibt zu zeigen, daß das Umgekehrte nicht statthat. Wir nehmen das Gegenteil an:

(19) AAA enthält einen Teil A1A_1A1, (20) A1~CA_1 \sim CA1C.

Mit (3) und (14) hieraus, daß A1A_1A1 also auch AAA einen zu BBB äquivalenten Teil enthielte. Das gleiche folgt aus (18) noch direkter und steht im Widerspruch zu der Voraussetzung A<BA < BA<B, die den Satz von der VI reglementiert.

Aus (9) aber würde folgen, daß auch BBB als zu AAA äquivalent einen mit CCC äquivalenten Teil besäße, und das widerspricht B<CB < CB<C, der in Ie und VI enthaltenen Voraussetzung.

Hiermit sind auch die Postulate Ie und VI als erfüllt nachgewiesen.

---

§ 10. Wenn AAA mit BBB äquivalent ist, so kann, wie gezeigt war, weder die Beziehung (B) noch die Beziehung (C) bestehen, d. h. es muß entweder (A) oder (D) eintreten. Da beide sich logisch ausschließen, kann auch nur eines von beiden zutreffen. Als das

„**Problem der Trichotomie**“ kann man folgende Aufgabe bezeichnen:

Es sei für gewisse Dinge A,B,…A, B, \ldotsA,B,… eine Teilung und eine Vergleichung gegeben, die zusammen das Postulat IV erfüllen, so daß die Disjunktion VIII möglich ist. Es sollen folgende Fragen beantwortet werden:

1. Sind die Fälle B, D möglich?
2. Welche der Beziehungen A~BA \sim BA~B hat im Falle der Äquivalenz statt?
3. Ist diese im Falle der Äquivalenz auftretende Beziehung auch eine hinreichende Bedingung der Äquivalenz?
4. Ist eine der Beziehungen A<BA < BA<B überhaupt ausgeschlossen?

Auf diese Fragen werden wir in den verschiedenen Fällen die verschiedensten Antworten erhalten. Zunächst beachten wir die Antwort, die der Satz vom Teil und Ganzen gibt: Er beantwortet nämlich die Frage (4) dahin, daß (A) unmöglich ist. Wenn die Beziehung (A) gilt, d. h. wenn A~BA \sim BA~B, BBB einen Teil von

AAA hat, so folgt aus B~AB \sim AB~A, daß auch AAA einen Teil A1A_1A1 besitzt, und daß A1~BA_1 \sim BA1B. Danach hat aber erstens A1A_1A1 als Teil von AAA auch Teil von AAA, zweitens A1~BA_1 \sim BA1B, B~AB \sim AB~A, also A1~AA_1 \sim AA1A. Dies widerspricht dem Postulat vom Teil und Ganzen, welches die Beziehung (A) ausschließt.

Hiermit hat also die Frage (1) und der Satz vom Teil und Ganzen schließlich die Beziehung (A) ausgeschlossen.

§ 10. — 499 —

Satz vom Teil und Ganzen nicht erledigt und bedarf einer gesonderten Behandlung, nämlich, sofern sie bejaht werden soll, des Nachweises folgender Sätze:

IX. Ist kein Teil von AAA zu BBB und kein Teil von BBB zu AAA äquivalent, so ist A~BA \sim BA~B äquivalent.

Wenn dieser Nachweis geführt ist, ist das Problem der Trichotomie in dem gewünschten Sinn gelöst, die Disjunktion VIII ist trichotom und der von (B, C) verschiedene Fall stimmt mit der Äquivalenz überein.

Wir wollen aber auch die Möglichkeit in Betracht ziehen, daß der Satz vom Teil und Ganzen nicht gilt, d. h. daß AAA einen seiner Teile A1A_1A1, äquivalent nicht ist. Ist dann auch A~BA \sim BA~B, so ist B~A1B \sim A_1B~A1. Andererseits enthält IV BBB einen Teil B1B_1B1, der A1A_1A1 äquivalent ist. Die Beziehung (A) bleibt also in Frage (?) beantwortet. Auf die anderen Fragen erhalten wir keine Antwort, denn mangels Charakter der Voraussetzungen entsprechen, von der wir ausgegangen.

Soll die Frage (3) im Falle der Nichtgültigkeit des Satzes A1<AA_1 < AA1<A beantwortet werden, so muß folgender Satz bewiesen werden:

X. (Äquivalenzsatz.) Ist AAA ein Teil von AAA, BBB von BBB, A1~B1A_1 \sim B_1A1B1, A0A_0A0 so ist A~BA \sim BA~B.

Diesen Satz kann folgende einfachere Fassung gegeben werden:

Xa. Ist A1A_1A1 ein Teil von AAA, AAA von AAA, und A1~AA_1 \sim AA1A, so ist A~AA \sim AA~A.

Aus dieser Fassung folgt wieder der zweite Äquivalenzsatz X. Denn es wäre gezeigt, daß aus A1~BA_1 \sim BA1B, BBB von AAA, die Existenz eines Teils A1A_1A1 von AAA folgt, der zu AAA äquivalent ist. Nach Xa ist demnach A1~AA_1 \sim AA1A und wegen A1~BA_1 \sim BA1B auch A~BA \sim BA~B.

Nimmt man umgekehrt in Xa, es sei BBB mit A1A_1A1 nicht nur äquivalent, sondern auch identisch, so erhält man Xa. Dieser Satz ist also in der Tat eine engere Fassung des Äquivalenzsatzes, und zwar auch verallgemeinert, weil er ein Äquivalent in den Voraussetzungen einführt.

Dies hieß recht abstrakt dem Leser vielleicht noch nicht vollkommen erscheinen, und Anführung möge noch durch einige Beispiele erläutert werden. Bei der üblichen gleichmäßigen Messung gilt der Satz vom Teil und Ganzen, gleiches auch Satz IX, so daß Satz VII zutrifft. Auch die Winkelmessung durch Scheitelstrahlenteilung verhält sich so. Sie operiert mit einer speziellen Teilung, die die Postulate Ia, IIa, IV erfüllt. Läßt man aber andere Teilungen zu — gerade der Winkelfläche, so gilt der Satz vom Teil und Ganzen nicht mehr. Daß damit das gesuchte Paradoxon auf Grund unserer bisherigen Ausführungen keinen logischen Widerspruch einschließt, dürfte dargetan sein. Man könnte nun immerhin geneigt sein, wenigstens die Unbrauchbarkeit der allgemeinen Teilung auf die Nichtgültigkeit des Satzes vom Teil und Ganzen zurückzuführen. Einmal aber werden wir in den Ausführungen über die Mengenlehre sehen, daß diese Nichtgültigkeit eine Teilung durchaus nicht unbrauchbar für die Definition einer Ordnung macht; zum andern aber können wir jetzt die Ursache der Unbrauchbarkeit der allgemeinen Zerlegung am Beispiel des Winkels entdecken. Es war nämlich in erster Linie gerade die Zerlegung des Winkels durch Zerschneiden in jeden andern verwandten Winkel, d. h. das zwischen zwei beliebigen Winkeln stets eine Beziehung (A) besteht. Es ist daher allerdings die Frage (4) für (A) zu bejahen, ebenso ist (3) zu bejahen, aber die Frage (1) ist zu verneinen: Die Fälle (B, C) sind unmöglich. Hierin liegt der wahre Grund der Unbrauchbarkeit der freien Zerschneidung: in der **Unmöglichkeit von (B) und (C)**, nicht aber in der **Möglichkeit von (A)**.

Zweiter Teil
Äquivalenz, Teilmenge und Mächtigkeit.

V.

Vergleichung und Teilung der Mengen.

§ 11. Den Begriff der Menge in völlig einwandfreier Weise zu definieren ist bisher nicht gelungen. Wahrscheinlich darf auch hier eine Definition nicht verlangt werden, sondern nur ein Axiomensystem. Aber selbst das fehlt bis jetzt noch. Die üblichen Definitionen der Menge gestatten keinerlei brauchbare Schlüsse zu ziehen; andererseits aber passen unter sie auch paradoxe Mengen, auf die wir weiter unten zurückkommen werden. Da es andererseits widerspruchsfreie unendliche Mengen zu geben scheint, muß eine richtige Definition oder ein korrektes Axiomensystem paradoxe Bildungen ausschließen, wenn es zu einem brauchbaren System von Folgerungen Anlaß geben will.

Wir entwickeln die Grundbegriffe zunächst an den endlichen Mengen. Denken wir uns als ganz konkretes Beispiel etwa einen Korb Äpfel und einen Korb Birnen. Um zu prüfen, ob beide gleichviel Stücke enthalten, können wir so verfahren: Wir entfernen mit der linken Hand einen Apfel, mit der rechten eine Birne und legen jedes Stück aus seinem Korb heraus. Dieses Verfahren wiederholen wir, solange es geht. Es wird nach einer endlichen Anzahl von Wiederholungen zu einem Ende führen, und

zwar entweder dadurch, daß keine Äpfel mehr da sind, oder dadurch, daß keine Birnen mehr vorhanden sind, oder endlich dadurch, daß weder Birnen noch Äpfel mehr übrig bleiben. Im ersten Fall ist die Anzahl der Birnen, im zweiten die der Äpfel die größere, im dritten Fall sind Anzahlen einander gleich.

Das angewendete Verfahren besteht in einer Zuordnung der Elemente zweier Mengen, da je ein Apfel und je eine Birne zu einem Paar vereinigt werden, und zwar ist jedem der beiden Stücke das andere eindeutig zugeordnet. Wir nennen eine solche Zuordnung „umkehrbar eindeutig“. Wir können nun das Kriterium der gleichen Anzahl formal so aussprechen, daß die Voraussetzung der Gleichheit der verglichenen Mengen nicht darin auftritt. Da wir uns ja auch tatsächlich über diese Voraussetzung nicht ausweisen, sollen sogleich die Termini „Anzahl“ und „gleich“ durch die mengentheoretischen „Mächtigkeit“ und „äquivalent“ ersetzt und das Kriterium in die Form einer Definition gesetzt werden:

Zwei Mengen heißen „äquivalent“ oder von „gleicher Mächtigkeit“, wenn die Dinge der einen den Dingen der anderen umkehrbar eindeutig zugeordnet werden können.

§ 12. Es ist natürlich noch nicht gesagt, daß dieser Definition auch für unendliche Mengen ein vernünftiger Sinn zukommt. Vielmehr ist bereits eines zu beweisen: daß willkürliche Zuordnungsverfahren, wie es bei dem Beispiel der Äpfel und Birnen angewandt ist,

* Dedekind gebraucht hierfür das Wort „ähnlich“. Doch hat sich nach dem Vorgang von Georg Cantor dieses Wort für eine engere Bedeutung eingebürgert. Siehe Teil III.

wurde, ist für unendliche Mengen unbrauchbar, da es zu keinem Ende führt. Dagegen können solche Zuordnungen sehr wohl durch irgendwelche Gesetze hergestellt werden. Wie auch die Zuordnung herzustellen sei, jedenfalls läßt sich folgende Bezeichnungsweise anwenden: Ist mmm ein Ding der Menge MMM und nnn das zugeordnete eine äquivalente Menge NNN, so schreiben wir die Tatsache der Zuordnung mit Zeichen n=φ(m)n = \varphi(m)n=φ(m), und sprechen von der Zuordnung φ\varphiφ. Daß umgekehrt zu nnn das zugeordnete mmm zweckmäßig durch ein anderes Zeichen, etwa ψ(n)\psi(n)ψ(n), ausgedrückt wird.

Die Frage, ob es prinzipiell zulässig ist, eine Menge von unendlich vielen Dingen als ein abgeschlossenes Ganzes zu betrachten, berührt die Zulässigkeit unserer Definition der Äquivalenz nicht, sondern findet in ihrer praktischen Anwendung. Die Behauptung, daß ist ein Ding der Menge aller ganzen Zahlen, enthält lediglich die Aussage: „nnn ist eine ganze Zahl“.

Daß es aber auf diesen Standpunkt zweifellos verfehlt wäre, die eine Menge durch gänzliche Iteration, um von der Identität zweier Mengen zu sprechen, so wollen wir folgende Definition ausdrücklich aufstellen, die ein Spezialfall der sogleich zu gebenden Definition der Teilung ist:

Zwei Mengen heißen identisch, wenn jedes Ding der einen auch ein Ding der anderen ist und umgekehrt.

Die Zuordnung eines Dinges zu sich selbst ist umkehrbar eindeutig. Daher ist das Postulat IIIb erfüllt, wonach identische Mengen auch äquivalent sein sollen. Von den beiden anderen Postulaten der Äquivalenz ist IIIb in der Forderung der Umkehrbarkeit der Eindeutigkeit enthalten. Und daß auch Ib erfüllt ist, ergibt sich durch folgende Überlegung:

Wenn a~ba \sim ba~b, b~cb \sim cb~c zugeordnet ist, so ist damit eine Zuord-

(Abbildung des Paradoxons. S. 35.)

nung zwischen aaa und ccc ausgesprochen; und wenn die Zuordnungen zwischen a,ba, ba,b und zwischen b,cb, cb,c umkehrbar eindeutig sind, so entsprechen sich auch a,ca, ca,c eindeutig. Der Beweis bedarf wohl keiner näheren Ausführung, zumal uns die Zuordnung zweier Mengen über eine dritte aus dem allgemeinen Lehren geläufig ist. Die Probe auf die gleiche Stückzahl in den oben erwähnten Obstkörben wird man nämlich lieber auf dem abstrakten Wege des Abzählens der Äpfel einerseits, der Birnen andererseits ausführen, als durch die zuerst beschriebene konkrete Methode des Ergreifens. Dieses Abzählen bedeutet aber nichts anderes, als daß man die Äpfel für sich und ebenso die Birnen den Dingen einer dritten gedachten Menge 1,2,3,4,5,…1, 2, 3, 4, 5, \ldots1,2,3,4,5,… zuordnet.

§ 13. Ebenso wie wir die Vergleichung definierten, ohne über die Existenz unendlicher Mengen etwas auszusagen, können wir auch die Teilung einführen:

Eine Menge M1M_1M1 heißt Teil einer Menge MMM, wenn jedes Ding von M1M_1M1 ein Ding von MMM ist. Sie heißt eigentlicher Teil, wenn nicht jedes Ding von MMM auch Ding von M1M_1M1 ist.

Den Charakter dieser Definition als reiner Wortdefinition erkennt man leicht aus folgenden Beispielen: Die Menge aller geraden Zahlen ist eine eigentliche Teilmenge der Menge aller ganzen Zahlen. Dieser Satz spricht die völlig einwandfreie Behauptung aus, daß jede gerade Zahl ganz, aber nicht jede ganze Zahl gerade ist.

Die Beweise zu den Postulaten Ia bis IIIa bedürfen keiner Ausführung. Bei dem Postulat IV ist folgendes nachzuweisen:

Ist MMM zu NNN äquivalent, M1M_1M1 eine Teilmenge von MMM, so besitzt NNN eine zu M1M_1M1 äquivalente Teilmenge N1N_1N1. In der Tat überzeugt man

sich leicht, daß die den Elementen von M1M_1M1 entsprechenden Elemente von NNN eine Teilmenge N1N_1N1 von NNN bilden, die zu M1M_1M1 äquivalent ist. Die Definition von N1N_1N1 lautet also: Ein Ding nnn von NNN wird als Ding von N1N_1N1 bezeichnet, wenn n=φ(m)n = \varphi(m)n=φ(m) für ein mmm aus M1M_1M1 ist. Hierin bedeutet φ\varphiφ die Zuordnung von MMM zu NNN. N1N_1N1 wird man zweckmäßig mit φ(M1)\varphi(M_1)φ(M1) bezeichnen.

Wenn M1M_1M1 eine eigentliche Teilmenge von MMM ist, so muß es nicht immer von NNN geben, die nicht in M1M_1M1 sind. Dieser Elemente bilden die zu M1M_1M1 komplementäre Teilmenge von MMM, jedes Ding von MMM entweder in M1M_1M1 oder in M\M1M \setminus M_1M\M1 enthalten. Diese komplementären Teilmengen entsprechen also eine vollständige Disjunktion. Entsprechend kann man komplementäre Systeme mehrerer Teilmengen bilden: Die Teilmengen M1,M2,…,MkM_1, M_2, \ldots, M_kM1,M2,…,Mk von MMM bilden ein komplementäres System, wenn jedes Element von MMM in einer und nur einer von ihnen enthalten ist. Zum Beispiel sind die Mengen der geraden und der ungeraden Zahlen komplementäre Teilmengen der Menge der ganzen Zahlen, womit die vollständige Aussage ausgesprochen wird, daß jede ganze Zahl entweder gerade oder ungerade ist. Die ganzen, gebrochenen und irrationalen Zahlen bilden ein komplementäres System von drei Teilmengen der Menge aller Zahlen. Komplementäre Systeme aus unendlich vielen Teilmengen werden wir späterhin betrachten, können aber vorläufig davon absehen. — Ist M1M_1M1 eine Teilmenge von MMM, so bezeichnet man zweckmäßig ihre komplementäre Menge mit M−M1M - M_1M−M1.

§ 14. Nach dem Beweis des Postulates I–IV können die Betrachtungen des ersten Kapitels einsetzen, durch die wir zu der vierfachen Disjunktion VIII gelangten.

Es ist nun ohne weiteres klar, daß für endliche Mengen der

— 506 — § 14.

Fall (A) ausscheidet und der Satz IX gültig ist. Ob und wie dies zu beweisen ist, d. h. auf einfachere Tatsachen zurückzuführen werden kann, das darf zunächst außer Betracht bleiben, da über die Richtigkeit anderer Behauptung niemals ein Zweifel bestanden hat. Ja, man hat sie als Grundsatz vielfach dahin ausgesprochen, daß das Resultat des Abzählens einer endlichen unabhängigen Reihenfolge das Abzählen aller Dinge einer Menge ist, ohne daß ein Rechtfertigung bedürfe, wenn überhaupt der Versuch unternommen wird, die Eigenschaften endlicher Mengen durch eine weitausholende Betrachtung zu begründen, wie dies Dedekind getan hat.

Wir werden im nächsten Kapitel zunächst an dem wichtigsten Beispiel, den unendlichen Mengen, zeigen, daß unendliche Mengen gibt, die äquivalente Teile besitzen. Daran schließt sich naturgemäß die Frage an, ob jede unendliche Menge äquivalente Teile besitzt. Auf diese Frage ist aber eine einwandfreie Antwort nur möglich, wenn der Begriff der unendlichen Menge definiert wird. Die Definition unendlicher Mengen hängt jedoch wieder von der Frage ab, ob nicht der Begriff „endlich“ hinreichend oder wenigstens durch ein Axiomensystem hinreichend charakterisiert ist. Denn um einen Schluß ziehen zu können, bedarf man zweier Obersätze; die Behauptung, eine Menge sei nicht endlich, gibt nur einen, der andere muß entweder aus der Definition des Endlichen abgeleitet sein oder aus dem Grundbegriff der Endlichkeit als Axiom entspringen. Dies ist die Schwierigkeit, die zu einer Kritik des Begriffs der endlichen Menge geführt hat.

Wir übergehen nun diese Schwierigkeit dadurch, daß wir

* „Was sind und was sollen die Zahlen?“ Braunschweig 1888.

§§ 14, 15. — 507 —

zunächst nur solche unendliche Mengen betrachten, die äquivalente eigene Teile besitzen. Solche Mengen nennen wir mit Georg Cantor **transfinite Mengen**. Da endliche Mengen keine äquivalenten Teile haben, steht zunächst fest, daß eine transfinite Menge nicht endlich sein kann. Die Frage umgekehrt, ob jede nicht endliche Menge transfinit ist, vertagen wir bis zum letzten Teil und verwenden solange auf einer Herleitung der bekannten Eigenschaften endlicher Mengen. Diesen Standpunkt der Kritik des Endlichen gegenüber kann man wohl treffend als den „naiven Standpunkt“ bezeichnen, womit natürlich kein Werturteil ausgesprochen werden soll.

§ 15. Nachdem wir so die schwierige Frage, ob auch jede unendliche Menge Teile besitzt, die ihr äquivalent sind, vorläufig beiseite geschoben haben, beweisen wir sogleich einen Satz, der das Schema des üblichen Nachweises der Unendlichkeit einer Menge enthält. Er lautet:

XI. Eine Menge, die eine transfinite Teilmenge enthält, ist selbst transfinit.

Hier ist eine solche Menge nicht endlich, denn eine endliche Menge enthält keine unendliche, also auch keine transfinite Teilmenge. Unser Satz behauptet aber mehr, nämlich die Existenz einer Zuordnung, die die Menge MMM auf eine Teilmenge abbildet.

Nach Voraussetzung besitzt M1M_1M1 eine Teilmenge M2M_2M2, die selbst wieder eine eigentliche Teilmenge M1M_1M1 ist, natürlich als äquivalente Teilmenge von M1M_1M1 gedacht; der Satz nennt zwar richtig, aber trivialerweise: Ihre komplementäre Menge M1−M2M_1 - M_2M1−M2 heiße M3M_3M3. Die zu M1M_1M1 komplementäre Teilmenge von MMM, heiße M4M_4M4. Da M2M_2M2 komplementäre Teilmengen M3,M4M_3, M_4M3,M4 nach Voraussetzung existiert eine Zuordnung

— 508 — §§ 15, 16.

φ\varphiφ, die jedem Element xxx von M1M_1M1 (d. h. also von MMM oder M2M_2M2) ein Element φ(x) \varphi(x)φ(x) von M1M_1M1 zuordnet. Diese Zuordnung ergibt sich dadurch von selbst, daß wir jedes Element von M1M_1M1 sich selbst zuordnen. Die Zuordnung φ(x)=x\varphi(x) = xφ(x)=x ist also die Identität. Ist dagegen xxx in M2M_2M2 oder M3M_3M3, so bedeutet φ(x)\varphi(x)φ(x) bereits als definiert vorausgesetzt Element φ(x)\varphi(x)φ(x) von M1M_1M1.

Diese Zuordnung bildet MMM auf eine Teilmenge ab, und zwar eine eigentliche, nämlich auf die beiden Mengen M1M_1M1 und M2M_2M2, die zusammen die komplementäre Menge von M3M_3M3 bilden. Demnach ist MMM transfinit, was zu beweisen war.

Ebenso leicht beweisen wir folgenden Satz:

XII. Ist MMM eine transfinite, NNN eine zu MMM äquivalente Menge, so ist NNN transfinit.

Nach Voraussetzung ist M1~NM_1 \sim NM1N, M1⊂MM_1 \subset MM1M. Nach Postulat IV ist daher N1~M1N_1 \sim M_1N1M1, N1⊂NN_1 \subset NN1N. Nach Ib folgt aus N1~M1N_1 \sim M_1N1M1 und M1⊂MM_1 \subset MM1M, daß N1N_1N1 eine eigentliche Teilmenge von NNN ist. w. z. b. w.

Die weiteren Aufgaben, die sich nun zur Behandlung bieten, entsprechen den vier Fragen des Trichotomieproblems. Nachdem die Frage (2) durch den Nachweis transfinitter Mengen beantwortet, die Frage (2) durch den Nachweis der Äquivalenzsatz Xa (Kap. VII), endlich beantwortet wird (1) durch die Aufweisung von Mengen verschiedener Mächtigkeit (Kap. VIII). Die Frage (4) wird ihre Behandlung und vorläufige Erledigung erst im Kapitel XXX finden.

VI.

Die abzählbaren Mengen.

§ 16. Offenbar ist die Menge N\mathbb{N}N aller ganzen positiven Zahlen 1,2,3,…1, 2, 3, \ldots1,2,3,… transfinit, denn die Gleichung n=n+1n = n + 1n=n+1 ordnet jeder

ganzen Zahl nnn eine ganze Zahl n+1n+1n+1 umkehrbar eindeutig zu. Unter den Zahlen nnn findet sich indessen die Zahl 111 nicht vor; sie wird also auf die eigentliche Teilmenge 2,3,4,…2, 3, 4, \ldots2,3,4,… abgebildet.

Andere Abbildungen, die in gleicher Art zum Nachweise benutzt werden, sind beispielsweise φ(n) =2n\varphi(n) = 2nφ(n)=2n, ψ(n)=n2\psi(n) = n^2ψ(n)=n2, allgemeiner φ(n)=n+a\varphi(n) = n + aφ(n)=n+a, worin nnn eine ganze positive Zahl ist.

Die Mächtigkeit von N\mathbb{N}N ist die kleinste unendliche Mächtigkeit überhaupt und wird daher eine besondere Bezeichnung haben: man nennt die Mächtigkeit ℵ0\aleph_0ℵ0 (aleph null). Das ℵ0\aleph_0ℵ0 ist insbesondere von allen transfiniten Mengen die kleinste Mächtigkeit besitzt, geht aus folgendem Satz hervor:

XIII. Jede Teilmenge von N\mathbb{N}N ist endlich oder abzählbar (d. h. ~N\sim \mathbb{N}~N).

XIV. Jede transfinite Menge enthält eine abzählbare Teilmenge.

Zum Beweise des ersten Satzes benutzen wir die Ordnung der ganzen Zahlen nach ihrer natürlichen Größe. Zwischen zwei ganzen Zahlen und auch unterhalb einer ganzen Zahl gibt es wohl endlich viele, eventuell gar keine ganze Zahlen. Daraus folgt zunächst, daß jede Teilmenge von N\mathbb{N}N, in der es eine größte Zahl gibt, endlich ist. Danach ist nun noch der Fall möglich, daß eine Teilmenge von N\mathbb{N}N keine größte Zahl enthält. Dann muß es aber zu jeder Zahl nnn dieser Teilmenge nichtgrößere unter den Zahlen von N\mathbb{N}N geben, die wir φ(n)\varphi(n)φ(n) nennen wollen. Diese Zuordnung φ\varphiφ ordnet jedem Element von MMM ein anderes zu. Ferner ist klar, daß MMM die kleinste ganze Zahl enthält, nämlich 111. Nun bilden wir die Zuordnung φ\varphiφ auf folgende Art: φ(1)=1\varphi(1) = 1φ(1)=1, φ(n)=φ(φ(n))\varphi(n) = \varphi(\varphi(n))φ(n)=φ(φ(n)), so erkennt man leicht, daß jedem Element von N\mathbb{N}N ein Element von MMM eindeutig zugeordnet wird.

Die Abzählbarkeit einer Menge MMM kann man mit Hilfe der Ordnung von N\mathbb{N}N leicht schematisch zum Ausdruck bringen, indem man die Elemente von MMM hintereinander, nötigenfalls über oder unter die ganzen Zahlen schreibt:

a1, a2, a3, a4, …1, 2, 3, 4, …\begin{aligned} a_1,\; a_2,\; a_3,\; a_4,\; \ldots \\ 1,\; 2,\; 3,\; 4,\; \ldots \end{aligned}a1,a2,a3,a4,...1,2,3,4,...
Ein solches Schema muß aber immer die Bildungsgesetze der abzählbaren Menge zum Ausdruck bringen oder als bekannt voraussetzen. Beispiele hierfür bieten die unendlichen Reihen, wie

a=1+11!+12!+13!+⋯a = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdotsa=1+1!1+2!1+3!1+⋯
π=1−13+15−17+19−\pi = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdotsπ=1−31+51−71+91−
17=0,142857 142857 142857…\frac{1}{7} = 0{,}142857\,142857\,142857\ldots71=0,142857142857142857…
während Ausdrücke wie
a=3,14159…a = 3{,}14159\ldotsa=3,14159…, 2=1,414…\sqrt{2} = 1{,}414\ldots2=1,414…
kein Bildungsgesetz anklingen lassen, es vielmehr als bekannt voraussetzen, sofern die Gleichungen richtig und keine approximativen sein sollen.

Der Satz XIV setzt voraus, daß MMM transfinit sei, d. h. daß jedem Element von MMM ein Element φ(m) \varphi(m)φ(m) einer Teilmenge M1M_1M1 von MMM zugeordnet werde. Sei nun aaa ein Element von M−M1M - M_1M−M1, so gibt es, da φ(m)\varphi(m)φ(m) stets in M1M_1M1 ist, kein Element xxx, für welches φ(x)=a\varphi(x) = aφ(x)=a ist.

Aus aaa bilden wir die Elemente φ(a),φ(φ(a)),…\varphi(a), \varphi(\varphi(a)), \ldotsφ(a),φ(φ(a)),…, φn(a),… \varphi^n(a), \ldotsφn(a),…. Wenn wir zeigen können, daß alle diese Elemente von einander verschieden sind, so ist unser Satz bewiesen; denn sie sind alle in M1M_1M1, bilden also eine Teilmenge von MMM, und da diese Teilmenge abzählbar ist, gilt aus dem Inklusio hervor, die bereits eine Zuordnung zu

N\mathbb{N}N ausdrücken.

Sicher ist φ(a)\varphi(a)φ(a) von allen Elementen φ(x)\varphi(x)φ(x) verschieden, da aaa in M−M1M - M_1M−M1, zu φ(a)\varphi(a)φ(a), aber sicher in M1M_1M1 enthalten ist. Ist nun xxx von φ(x)\varphi(x)φ(x) verschieden, so folgt aus der umkehrbaren Eindeutigkeit der Zuordnung, daß auch φ(x)\varphi(x)φ(x) von φ(φ(x))\varphi(\varphi(x))φ(φ(x)) verschieden sein muß. Demnach ist φn(a)\varphi^n(a)φn(a) d. h. φ\varphiφ von φn(a)\varphi^n(a)φn(a) für jedes kkk verschieden, daher wieder φn(a)\varphi^{n}(a)φn(a) von φn+1(a) \varphi^{n+1}(a)φn+1(a), u. s. f. Damit ist der Beweis unseres Satzes erbracht.

Aus diesem Satz geht hervor, daß eine transfinite Menge nur von gleicher oder höherer Mächtigkeit sein kann, als N\mathbb{N}N. Daß keine unendliche Menge eine Mächtigkeit unter ℵ0\aleph_0ℵ0 haben kann, folgt schon aus XIII, doch ist damit noch nicht gesagt, daß sie von gleicher oder höherer Mächtigkeit sein muß, solange nicht die Unmöglichkeit des Falles (A) feststeht.

§ 17. Um zu weiteren speziellen abzählbaren Mengen zu gelangen, beweisen wir folgenden allgemeinen Satz:

XV. MMM sei eine abzählbare Menge, m1,m2,m3,…m_1, m_2, m_3, \ldotsm1,m2,m3,… seien ihre Elemente, und jedes dieser Elemente sei wieder eine endliche Menge. Von den Dingen xxx dieser Mengen kommen keines in zwei oder mehreren der Mengen m1,m2,…m_1, m_2, \ldotsm1,m2,… vor. Dann ist die Menge der Dinge xxx selbst abzählbar.

Wenn der Sinn des Satzes richtig verstanden ist, kann er auch in folgenden kürzeren Fassungen ausgesprochen werden:

Eine abzählbare Menge endlicher Mengen ist abzählbar; oder:

Ersetzt man in einer abzählbaren Menge jedes Element durch eine endliche Anzahl von Elementen, so entsteht wieder eine abzählbare Menge.

Der Beweis des Satzes ist fast trivial zu nennen, wenn man die letzte Fassung beachtet. Sind m1,m2,m3,…m_1, m_2, m_3, \ldotsm1,m2,m3,… die Dinge der Menge m1m_1m1 und m2m_2m2 ihre Anzahl, so schreibe man die Dinge xxx in der durch die Indizes gegebenen Reihenfolge an:

m1: x1(1), x2(1), …, xn1(1)m2: x1(2), x2(2), …, xn2(2):\begin{aligned} m_1:\; & x^{(1)}_1,\; x^{(1)}_2,\; \ldots,\; x^{(1)}_{n_1} \\ m_2:\; & x^{(2)}_1,\; x^{(2)}_2,\; \ldots,\; x^{(2)}_{n_2} \\ & \vdots \\ \end{aligned}m1 :m2:x1(1),x2(1),...,xn1(1)x1(2),x2(2),...,xn2(2)
Man kann ohne weiteres die Ordnung der Dinge xxx den ganzen Zahlen angeben. Der Menge m1m_1m1 gehen die Mengen m2,m3,…m_2, m_3, \ldotsm2,m3,… voran; sie liefern n1+n2+⋯n_1 + n_2 + \cdotsn1+n2 +⋯ Elemente. Den Elementen xi(k)x^{(k)}_ixi(k) kommen daher die Ordnungszahlen

n1+n2+⋯+nk−1+in_1 + n_2 + \cdots + n_{k-1} + in1+n2+⋯+nk−1+i
zu, womit die Abzählbarkeit erwiesen ist.

Eine stillschweigende Voraussetzung bei diesem Beweise ist die, daß an die Elemente von mim_imi, die unteren Indizes 1,2,…1, 2, \ldots1,2,… in irgend einer Art ersetzt werden. Dies kann für die unendliche mim_imi, \ldots, nur dadurch eingesehen werden.

Wie weit die Annahme der Existenz eines solchen Gesetzes zulässig ist, wird später noch untersucht werden, da die Frage von prinzipieller Bedeutung ist. Wir begnügen uns hier mit der Feststellung, daß bei allen Anwendungen des Satzes XV ein solches Gesetz als vorausgesetzt werden kann.

§ 18. Eine erste Anwendung des Satzes XV ist das folgende Theorem:

XVI. *(Satz von der endlichen Bezeichnung.)* Es sei MMM eine unendliche, ZZZ eine endliche Menge, deren Dinge z1,z2,…,znz_1, z_2, \ldots, z_nz1,z2,…,zn seien. Wir zeichnen MMM eine wohlgeordnete Reihe von Dingen aus ZZZ in der jedes zzz mehrmals auftreten darf, soll eine „Bezeichnung“ genannt werden. Wenn jedem Ding mmm von MMM eine Bezeichnung φ(m)\varphi(m)φ(m) umkehrbar eindeutig zugeordnet ist, so ist MMM abzählbar.

In der Definition der Bezeichnungen ist die Reihenfolge betont; es sollen also BBB, AAA, BBB und AAA verschiedene Bezeichnungen sein.

Der Beweis ist ziemlich einfach. Die Menge der Bezeichnungen ist abzählbar; die Menge der den Dingen von MMM zugeordneten Bezeichnungen φ(m)\varphi(m)φ(m) ist daher eine Teilmenge der Menge aller Bezeichnungen nach Satz XV. Selbstfalls abzählbar, also ist MMM äquivalent der Menge der Bezeichnungen.

Daß aber die Menge der Bezeichnungen abzählbar ist, folgt aus Satz XV. Sie zerfällt nämlich in eine abzählbare Reihe

Θ1, Θ2, Θ3, …\Theta_1,\ \Theta_2,\ \Theta_3,\ \ldotsΘ1, Θ2, Θ3, …
von Mengen; die erste enthält alle Bezeichnungen, die aus einem Element von ZZZ bestehen:

z1, z2, …, zn;z_1,\ z_2,\ \ldots,\ z_n;z1, z2, …, zn;
die zweite alle zweigliedrigen Bezeichnungen:

z1z1, z1z2, …, znzn;z_1z_1,\ z_1z_2,\ \ldots,\ z_nz_n;z1z1, z1z2, …, znzn;
die dritte alle dreigliedrigen, die kkk-te alle kkk-gliedrigen. Die Mengen Θ1,Θ2,…\Theta_1, \Theta_2, \ldotsΘ1,Θ2,… sind endlich; Θk\Theta_kΘk enthält genau nkn^knk Elemente. Diese können nach einer einheitlichen Gesetz geordnet werden, z. B. nach dem lexikographischen, sofern z1,z2,…,znz_1, z_2, \ldots, z_nz1,z2,…,zn willkürlich geordnet hat. Damit folgt aus Satz XV die Abzählbarkeit von Θ\ThetaΘ und daraus die von MMM. — Die Ordnungsnummer, die zu einer Bezeichnung zi1zi2…zikz_{i_1} z_{i_2} \ldots z_{i_k}zi1zi2…zik gehört, läßt sich auch …

— 514 — § 18.

explizit angeben. Ist die Bezeichnung *n*-gliedrig, so ist ihre Ordnung gleich

$a_0 + b_1 + b_2 + \ldots + b_n + p.$

Die prinzipielle Bedeutung des Satzes XVI geht daraus hervor, daß das Alphabet eine endliche Menge von Zeichen enthält, daß also die Menge aller Wortbildungen, insbesondere der sinnvollen, abzählbar ist. Eine daraus gegründete Paradoxie wird später zu erledigen sein (Kap. XXXIII).

Aber auch die Menge der Ziffern 0, 1, 2, bis 9 ist eine endliche; mit ihr bezeichnen wir alle ganzen Zahlen, deren Menge der der Tat abzählbar ist. Ferner erweist sich die Menge aller Dezimalzahlen (d. h. Brüche, deren Nenner eine Potenz von 10 ist) als abzählbar, denn zur Bezeichnung dienen die zehn Ziffern und das Komma. Die Dezimalzahlen sind eine Teilmenge der Menge aller Brüche überhaupt. Da aber jeder Bruch mit den zehn Ziffern und einem Doppelpunkt (:) bezeichnet werden kann, ist die Menge aller Brüche, d. i. der Rationalzahlen, abzählbar. Aber auch die Menge aller algebraischen Zahlen läßt sich abzählen. Algebraisch heißt eine Zahl, wenn sie einer Gleichung von folgender Form genügt:

$a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n = 0,$

in der $a_0, a_1, \ldots, a_n$ ganze Zahlen, positive oder negative sind. Unter allen Gleichungen, denen eine algebraische Zahl genügt, gibt es eine bevorzugte, die irreduzible. Sie ist von nicht höherem Grade als jede andere, und die Zahlen $a_0, a_1, \ldots, a_n$ haben keinen gemeinsamen Teiler; a_n ist positiv. Die Wurzeln dieser Gleichung sind alle voneinander verschieden und, was hier übrigens belanglos ist, die Gleichung ist für jede ihrer Wurzeln die irreduzible Gleichung. Die *n* Wurzeln lassen sich auch, wenn sie …

§ 18, 19. — 515 —

imaginär sind, nach einem einheitlichen Prinzip ordnen; als erste, zweite, … *n*-te.

Demnach ist eine algebraische Zahl eindeutig definiert durch Angabe der Zahlen $a_0, a_1, \ldots, a_n$, d. h. der Koeffizienten ihrer irreduziblen Gleichung, und der Ordnungsnummer *k*, die ihr unter den Wurzeln dieser Gleichung zukommt. Schreiben wir dafür etwa

α=[a0,a1,a2,…,an,k],\alpha = [a_0, a_1, a_2, \ldots, a_n, k],α=[a0,a1,a2,…,an,k],
so ist *α* eindeutig beschrieben durch *n + 2* Zeichen: die 10 Ziffern, das Komma und den Minusstrich von den negativen Koeffizienten. Z. B. wäre

α=[−3,0,1,2]\alpha = [-3, 0, 1, 2]α=[−3,0,1,2]
die zweite Wurzel der (irreduziblen) Gleichung

−3+x2=0,-3 + x^2 = 0,−3+x2=0,
d. h., wenn wir der Größe nach ordnen, die positive Wurzel aus 3, d. i. $\sqrt{3} = 1{,}732$.

Die angegebene Bezeichnungsweise ist eine endliche; damit folgt aus Satz XVI die Abzählbarkeit der Menge der algebraischen Zahlen.

§ 19. Eine weitere Folgerung aus Satz XVI ist:

XVII. Eine abzählbare Menge abzählbarer Mengen ist abzählbar.

Dem Satz pflegt man auch, seiner geometrischen Bedeutung wegen, das Bild eines von der Abzählbarkeit des Punktgitters zu entlehnen.

Nach Voraussetzung ist eine Reihe $M_1, M_2, M_3, \ldots$ von Mengen gegeben, und jeder Menge ist umkehrbar eindeutig eine ganze Zahl zugeordnet, die wir ihren Index nennen wollen, und …

— 516 — § 19.

ihr auch als Index anhängen. Indem wir alle diese Mengen zusammenfassen, entsteht eine Menge M, und jedes Element m von M ist in einer bestimmten Menge M_k enthalten. Den Index dieser Menge setzen wir dem Element als ersten Index an. Da weiter die Abzählbarkeit von M_k angenommen, existiert eine Zuordnung, die dem Element m in M_k eine bestimmte Zahl n zuordnet. Diese nennen wir den zweiten Index von m und bezeichnen ihn kurz durch die beiden Indices: $m = (k, n)$. Der erste gibt die Teilmenge M_k an, in der m zu suchen ist, der zweite gibt die Stelle an, die m in M_k einnimmt.

Die Abzählbarkeit von M folgt nun unmittelbar nach XVI daraus, daß jedes Indexpaar (k, n) durch die zehn Ziffern und das Komma bezeichnet wird.

Um die Abzählbarkeit der Menge M aller Indexpaaren (k, n) noch auf einen zweiten Weg nachzuweisen, bilden wir die Summe $k + n$ der beiden Indices des Paares. Es gibt nur eine endliche Anzahl von Paaren, die eine vorgeschriebene Summe liefern, nämlich $(1, s - 1), (2, s - 2), (3, s - 3), \ldots, (s - 1, 1)$. Ihre Anzahl ist $s - 1$; diese Zahlen wollen wir *Höhe* des Indexpaares (k, n). Sie ist eine endliche Zahl.

Es ist jetzt N_s die Menge aller Paare von der Höhe s. Sie ist endlich. Die Menge der Mengen N_s, die wir der Indexreihe nach abzählen, ist abzählbar. Die Menge aller Elemente der Mengen N_s ist aber die Menge M identisch, also ist M nach Satz XVI abzählbar.

Die bereits erwähnte geometrische Interpretation erhält man, wenn man die beiden Indices nach den Methoden der analytischen Geometrie als Abszisse und Ordinate darstellt. In der Figur sind dann die Einheiten der beiden Achsen gleich groß gewählt. Man erhält für die ersten Indices die Teilpunkte 1, 2, 3, etc. auf der horizon-

§ 19. — 517 —

talen Axe, in denen Lote nach oben errichtet werden. Den zweiten Indices entsprechen die Punkte der vertikalen Axe mit den auf ihr nach rechts errichteten Loten. Der Schnitt zweier Lote wird dem Element mit den beiden Indices zugeordnet. Jedes Lot auf der horizontalen Axe enthält die Punkte, die einer der Mengen M_k angehören. Die Lote selbst bilden ersichtlich eine abzählbare Menge.

Denkt man sich die Schnittpunkte markiert und die Lote entfernt, so entsteht die in Punktgittern bezeichnete Figur, deren Abzählbarkeit wir nachgewiesen haben. Fassen wir in ihr die Punkte zusammen, die Indexpaaren von gleicher Höhe entsprechen, so erhalten wir in der nächsten Figur eingezeichneten Diagonalen, die sich durch die vertikalen und horizontalen Verbindungslinienstücke in eine einzige Linienzug vereinigen lassen, der jeden Gitterpunkt trifft. Diesem Linienzug entspricht eine Anordnung aller Indicespaare von gleicher Höhe, bei der alle Paare von gleicher gerader Höhe nach dem hinteren Index geordnet sind. Natürlich gibt es auch noch andere Ordnungen, die aber geometrisch weniger einfach aussehen; umgekehrt gibt es geometrische …

— 518 — § 19.

… ganze Ordnungen, die abstrakt schwieriger zu beschreiben sind, z. B. die folgenden: *(1,1), (2,1), (2,2), (1,2), (1,3), (2,3), (3,3), (3,2), (3,1), (4,1), (4,2), (4,3), (4,4), (3,4), (2,4), (1,4), (1,5), (2,5), (3,5), (4,5), (5,5), (5,4), (5,3), (5,2), (5,1), (6,1), (6,2), (6,3), (6,4), (6,5), (6,6), …*

Auf diesem Wege gelangt man zu einer hübschen Abzählung der rationalen Zahlen. Jede rationale Zahl wird in einfacher Weise als Bruch $a : b$ zweier teilerfremder Zahlen dargestellt. Den beiden Zahlen a, b entspricht ein Punkt des Gitters, und jedem Gitterpunkt, dessen Indices (Koordinaten) teilerfremd sind, entspricht eine rationale Zahl in ihrer Normaldarstellung. Damit werden die rationalen Zahlen auf die nebenstehend dargestellte Teilmenge des Punktgitters abgebildet und nach dem Höhenverfahren in eine Reihe gebracht, die so beginnt:

1 : 1, 1 : 2, 2 : 1, 1 : 3, 2 : 2, 3 : 1, 1 : 4, 2 : 3, 3 : 2, 4 : 1, …

Die Figur des Punktgitters enthält auch die Anordnung der rationalen Zahlen nach ihrer natürlichen Größe. Zeichnet man die beiden Achsen als Strahlen nach den Gitterpunkten, so ordnen sich diese nach demselben Gesetz, wie die rationalen Zahlen: liegt von drei Rationalzahlen *a : b*, *c : d*, *e : f* die Größe nach zwischen *c : d* und *e : f*, so liegt auch der entsprechende Schnittstrahl zwischen den zu *c : d* und *e : f* gehörigen.

Aus Satz XVII folgt nun unmittelbar, daß auch die Menge …

§ 19, 20. — 519 —

… aller Indicestripel *(a, β, y)* abzählbar ist; denn zunächst ist die Menge aller Tripel mit gleichem vorderen Index *a* abzählbar, da jedes Element in ihr durch das Paar *(β, y)* eindeutig charakterisiert ist. Die Menge aller Mengen M_a von gleichem vorderen Index *a* ist aber ersichtlich selbst abzählbar, woraus nach XVII die Abzählbarkeit aller Tripel folgt. Den Indicestripeln entspricht die abstrische ein räumlichen Punktgittern.

Durch Schluß von *n* auf *n + 1* beweisen wir weiter, daß jede Menge abzählbar ist, in der jedes Element durch *n* Indices aus der Reihe der ganzen Zahlen eindeutig charakterisiert ist.

Von da an endlich folgt, daß eine Menge *M* auch dann abzählbar ist, wenn jedes Element durch eine endliche, aber nicht beliebig große Zahl von Indices bestimmt wird. Denn eine solche Menge zerfällt in die Teilmengen $M_1, M_2, M_3, \ldots, M_\square \ldots$, von denen $M_\square$ alle Elemente mit einem, $M_\square$ mit ***k*** Indices enthält. Jede dieser Mengen ist abzählbar, da die Abzählbarkeit folgt. Diese letzte Resultat zeigt wiederum die Abzählbarkeit aller algebraischen Zahlen, von denen jede durch die *n + 2* Indices $a_0, a_1, \ldots, a_{\square k}$ charakterisiert ist, wobei *n* endlich, aber keiner oberen Schranke unterworfen ist.

§ 20. Die speziellen hier angeführten Beispiele der rationalen, dezimalen und algebraischen Zahlen sind bekannt dafür, daß sie einen völlig überraschenden Eindruck bei jedem hervorrufen, der sie zum ersten Male kennen lernt. Das Überraschende liegt darin, daß zwischen zwei beliebigen rationalen Zahlen stets …

Das hierbei auch negative Indices auftreten, ist belanglos, da auch diese Reihen der ganzen Zahlen wie folgt abzählbar ist:

0, 1, −1, 2, −2, 3, −3, …

Anmerkungen zur philosophischen Schule. I. Bd.
36