Por qué debería utilizar una licencia de estilo BSD para su proyecto de Código Abierto

Tabla de contenidos

1. Introducción

Este documento justifica el uso de una licencia de estilo BSD para software y datos; específicamente recomienda utilizar una licencia de estilo BSD en lugar de la GPL. También se puede leer como una introducción y resumen de la licencia de código abierto BSD versus GPL.

2. Breve historia del Código Abierto

Mucho antes de que el término "Open Source" fuera utilizado, el software era desarrollado por diferentes asociaciones de programadores y compartido libremente. A comienzos de los años 50, organizaciones como SHARE y DECUS desarrollaron mucho del software que las compañías de hardware incluían en sus ofertas de equipos. En aquella época las compañías informáticas estaban en el negocio del hardware; cualquier cosa que redujera el coste del software e hiciera los programas más disponibles hacía que las compañías de hardware fueran más competitivas.

Este modelo cambió en los años 60. En 1965 la ADR desarrolló el primer producto de software con licencia que era independiente de una compañía de hardware. ADR estaba compitiendo con un

paquete gratuito de IBM desarrollado originalmente por los clientes de IBM. ADR patentó su software en 1968. Para dejar de compartir programas, los suministraron bajo un contrato de arrendamiento del equipo en el que el pago se extendía durante toda la vida del producto. ADR retenía así la propiedad y podía controlar la reventa y la reutilización.

En 1969, el Departamento de Justicia de los Estados Unidos acusó a IBM de destruir negocios al combinar software libre con su hardware. Como resultado de este demanda, IBM quitó su software; es decir, el software se convirtió en un producto independiente y separado del hardware.

En 1968, la informática introdujo la primera aplicación asesina comercial y estableció rápidamente el concepto del producto de software, la empresa de software y tasas de retorno muy altas. La informática desarrolló la licencia perpetua que ahora es estándar en toda la industria informática, en la que la propiedad nunca se transfiere al cliente.

3. Unix desde una perspectiva de Licencia de BSD

AT&T, que era propietario de la implementación original de Unix, era un monopolio regulado públicamente vinculado a un tribunal antimonopolio; legalmente no podía vender un producto en el mercado de software. Sin embargo, pudo proporcionarlo a instituciones académicas por el precio de los medios.

Las universidades adoptaron rápidamente Unix después de que una conferencia sobre sistemas operativos anunciara su disponibilidad. Fue extremadamente útil que Unix se ejecutara en el PDP-11, una computadora de 16 bits muy asequible, y estuviera codificado en un lenguaje de alto nivel que era demostrablemente bueno para la programación de sistemas. El DEC PDP-11 tenía, en efecto, una interfaz de hardware abierta diseñada para facilitar a los clientes la escritura de su propio sistema operativo, lo cual era común. Como proclamó el fundador de DEC, Ken Olsen, "el software viene del cielo cuando tienes un buen hardware".

El autor de Unix Ken Thompson regresó a su alma máter, la Universidad de California Berkeley (UCB), en 1975 y enseñó el núcleo línea por línea. Esto finalmente resultó en un sistema en evolución conocido como BSD (Distribución estándar de Berkeley). UCB convirtió Unix a 32 bits, agregó memoria virtual e implementó la versión de la pila TCP / IP sobre la que se construyó Internet esencialmente. UCB puso a disposición BSD por el costo de los medios, bajo lo que se conoció como "la licencia BSD". Un cliente compraba Unix a AT&T y después encargaba una cinta con BSD a la UCB.

A mediados de la década de 1980, un caso antimonopolio del gobierno contra AT&T terminó con su desintegración. AT&T todavía era dueña de Unix y podía venderlo. AT&T se embarcó en un esfuerzo agresivo de licenciamiento y la mayoría de los sistemas Unix comerciales de la época llegaron a ser derivados de los de AT&T.

Al principio de los 90 AT&T demandó a UCB por violaciones de la licencia relacionada con BSD. La UCB descubrió que AT&T había incorporado, sin conocimiento ni retribución, muchas mejoras de BSD en los productos de AT&T, y tuvo lugar un largo proceso judicial, principalmente entre AT&T y la UCB. Durante este periodo algunos de los programadores de la UCB se embarcaron en un

proyecto para reescribir cualquier código de AT&T asociado con BSD. Este proyecto resultó en un sistema llamado BSD 4.4-lite ("lite" o ligero, porque no era un sistema completo; carecía de 6 archivos clave de AT&T).

Una larga serie de artículos publicados un poco más tarde en la revista Dr. Dobbs describe una versión 386 de Unix para PC derivada de BSD, con archivos de reemplazo con licencia BSD para los 6 archivos 4.4 lite que faltan. Este sistema, llamado 386BSD, se debió al ex programador de UCB William Jolitz. Se convirtió en la base original de todos los BSD de PC que se utilizan en la actualidad.

A mediados de la década de 1990, Novell compró AT&T Se llegó a los derechos Unix y a un acuerdo (entonces secreto) para terminar la demanda. UCB pronto terminó su apoyo a BSD.

4. El estado actual de las licencias FreeBSD y BSD

La llamada nueva licencia de BSD aplicada a FreeBSD en los últimos años es efectivamente una declaración de que puedes hacer cualquier cosa con el programa o su fuente, pero no tienes ninguna garantía y ninguno de los autores tiene ninguna responsabilidad (básicamente, no puedes demandar a nadie). Esta nueva licencia BSD está destinada a fomentar la comercialización de productos. Cualquier código BSD puede venderse o incluirse en productos patentados sin ninguna restricción sobre la disponibilidad de su código o su comportamiento futuro.

No confunda la nueva licencia BSD con "dominio públic". Mientras que en el dominio público un apartado establece el uso libre para todos, no establece propietario.

5. Los orígenes de la GPL

Si bien el futuro de Unix había sido tan confuso a fines de la década de 1980 y principios de la de 1990, la GPL, otro desarrollo con importantes consideraciones de licencia, alcanzó sus frutos.

Richard Stallman, el desarrollador de Emacs, era miembro del personal del MIT cuando su laboratorio cambió de sistemas propios a propietarios. Stallman se molestó cuando descubrió que legalmente no podía agregar mejoras menores al sistema. (Muchos de los compañeros de trabajo de Stallman se habían marchado para formar dos empresas basadas en software desarrollado en MIT y con licencia de MIT; parece que hubo desacuerdo sobre el acceso al código fuente de este software). Stallman ideó una alternativa a la licencia de software comercial y la llamó GPL o "Licencia pública GNU". También inició una fundación sin fines de lucro, la Free Software Foundation (FSF), que pretendía desarrollar un sistema operativo completo, incluidos todos software asociado, que no estaría sujeto a licencias de propiedad. Este sistema se llamó GNU, porque "GNU no es Unix".

La GPL fue diseñada para ser la antítesis de la licencia propietaria estándar. Con este fin, se requería que cualquier modificación que se hiciera a un programa GPL se devolviera a la comunidad GPL (requiriendo que el código fuente del programa estuviera disponible para el usuario) y se requería cualquier programa que usara o vinculara el código GPL estar bajo la GPL. La GPL estaba destinada a evitar que el software se convirtiera en propietario. Como dice el último

párrafo de la GPL:

"Esta Licencia Pública General no permite la incorporación de tu programa en programas propietarios."[1]

La GPL es una licencia compleja así que aquí hay algunas reglas básicas cuando se utiliza la GPL:

- puede cobrar tanto como desee por distribuir, respaldar o documentar el software, pero no puede vender el software en sí.
- la regla general establece que si se requiere la fuente GPL para que un programa se compile, el programa debe estar bajo la GPL. La vinculación estática a una biblioteca GPL requiere que un programa esté bajo la GPL.
- la GPL requiere que cualquier patente asociada con el software con licencia GPL debe tener licencia para uso gratuito de todos.
- simplemente agregar software, como cuando se colocan varios programas en un disco, no cuenta como la inclusión de programas con GPL en programas sin GPL.
- la salida de un programa no cuenta como trabajo derivado. Esto permite utilizar el compilador gcc en entornos comerciales sin problemas legales.
- dado que el kernel de Linux está bajo la GPL, cualquier código enlazado estáticamente con el kernel de Linux debe ser GPL. Este requisito se puede eludir vinculando dinámicamente módulos del kernel cargables. Esto permite a las empresas distribuir controladores binarios, pero a menudo tiene la desventaja de que solo funcionarán para versiones particulares del kernel de Linux.

Debido en parte a su complejidad, en muchas partes del mundo hoy en día se ignoran las legalidades de la GPL con respecto a Linux y software relacionado. Las ramificaciones a largo plazo de esto no están claras.

6. Los orígenes de Linux y de la licencia GPL

Mientras se desarrollaban las guerras comerciales de Unix, el kernel de Linux se desarrolló como un clon de Unix para PC. Linus Torvalds acredita la existencia del compilador GNU C y las herramientas GNU asociadas para la existencia de Linux. Puso el kernel de Linux bajo la GPL.

Recuerde que la licencia GPL requiere que cualquier cosa que se enlace estáticamente a cualquier código bajo la licencia GPL también se pondrá bajo la licencia GPL. El código fuente debe estar disponible para el usuario del programa. El enlace dinámico, sin embargo, no se considera una violación de la licencia GPL. La presión para poner aplicaciones propietarias en Linux llegó a ser abrumadora. Tales aplicaciones deben enlazarse a menudo con las bibliotecas del sistema. Esto resultó en una versión modificada de la GPL denominada LGPL ("Library", rebautizada como menor ("Lesser"), GPL). La licencia LGPL permite al código propietario estar enlazado a la biblioteca C de GNU, glibc. No hay que liberar el código fuente que ha sido enlazado dinámicamente a una biblioteca bajo una licencia LGPL.

Si enlaza estáticamente una aplicación con glibc, como suele ser necesario en los sistemas integrados, no puede mantener la propiedad de la aplicación, es decir, debe liberar la fuente. Tanto

la GPL como la LGPL requieren cualquier modificación al código directamente bajo la licencia para ser liberadas.

7. Las licencias de Código Abierto y el Problema de la Orfandad

Uno de los graves problemas asociados con el software privativo se conoce como "orfandad". Esto ocurre cuando un solo fallo de negocio o un cambio en la estrategia del producto hace que una enorme pirámide de sistemas y empresas dependientes falle por razones que están más allá de su control. Décadas de experiencia han mostrado que el tamaño o éxito momentáneos de un proveedor de software no es garantía de que su software permanecerá disponible, ya que las condiciones y estrategias actuales del mercado pueden cambiar rápidamente.

La licencia GPL intenta prevenir la orfandad mediante el corte del vínculo con la propiedad intelectual privativa.

Una licencia BSD le da a una pequeña empresa el equivalente al software en custodia sin complicaciones ni costos legales. Si un programa con licencia BSD queda huérfano, una empresa puede simplemente hacerse cargo, de manera patentada, del programa del que depende. Una situación aún mejor ocurre cuando un pequeño consorcio informal mantiene una base de código BSD, ya que el proceso de desarrollo no depende de la supervivencia de una sola empresa o línea de productos. La capacidad de supervivencia del equipo de desarrollo cuando están mentalmente en la zona es mucho más importante que la simple disponibilidad física del código fuente.

8. Lo que no puede hacer una licencia

Ninguna licencia puede garantizar la disponibilidad futura del software. Aunque un titular de derechos de autor tradicionalmente puede cambiar los términos de un derecho de autor en cualquier momento, la presunción en la comunidad de BSD es que tal intento simplemente hace que la fuente se bifurque.

La GPL no permite explícitamente revocar la licencia. Sin embargo, ha ocurrido que una empresa (Mattel) compró un copyright GPL (cphack), revocó todo el copyright, acudió a los tribunales y prevaleció [2]. Es decir, revocaron legalmente toda la distribución y todos los trabajos derivados basados en los derechos de autor. Si esto podría suceder con una distribución más amplia y dispersa es una pregunta abierta; También existe cierta confusión sobre si el software estaba realmente bajo la GPL.

En otro ejemplo, Red Hat compró Cygnus, una empresa de ingeniería que se había hecho cargo del desarrollo de las herramientas del compilador FSF. Cygnus pudo hacerlo porque habían desarrollado un modelo de negocio en el que vendían soporte para software GNU. Esto les permitió emplear a unos 50 ingenieros e impulsar la dirección de los programas contribuyendo con la preponderancia de las modificaciones. Como afirma Donald Rosenberg "los proyectos que utilizan licencias como la GPL ... viven bajo la amenaza constante de que alguien se haga cargo del proyecto produciendo una mejor versión del código y haciéndolo más rápido que los propietarios originales". [3]

9. Ventajas y Desventajas de la licencia GPL

Una razón común para usar la GPL es cuando se modifica o se extiende el compilador gcc. Esto es particularmente adecuado cuando se trabaja con CPU especiales únicas en entornos donde es probable que todos los costos de software se consideren gastos generales, con expectativas mínimas de que otros usen el compilador resultante.

La GPL también es atractiva para las pequeñas empresas que venden CD en un entorno en el que "comprar barato, vender caro" puede ofrecer al usuario final un producto muy económico. También es atractivo para las empresas que esperan sobrevivir proporcionando diversas formas de soporte técnico, incluida la documentación, para el mundo de la propiedad intelectual con GPL.

Un uso menos publicitado y no intencionado de la GPL es que es muy favorable para las grandes empresas que quieren socavar a las empresas de software. En otras palabras, la GPL es adecuada para su uso como arma de marketing, reduciendo potencialmente el beneficio económico general y contribuyendo al comportamiento monopolístico.

La GPL puede presentar un problema real para quienes deseen comercializar y beneficiarse del software. Por ejemplo, la GPL se suma a la dificultad que tendrá un estudiante de posgrado para formar directamente una empresa para comercializar sus resultados de investigación, o la dificultad que tendrá un estudiante para unirse a una empresa en el supuesto de que se comercializará un proyecto de investigación prometedor.

Para aquellos que deben trabajar con implementaciones vinculadas estáticamente de múltiples estándares de software, la GPL es a menudo una licencia deficiente, porque excluye el uso de implementaciones propietarias de los estándares. Por tanto, la GPL minimiza el número de programas que se pueden construir utilizando un estándar GPL. La GPL no tenía la intención de proporcionar un mecanismo para desarrollar un estándar sobre el cual se diseñan productos patentados. (Esto no se aplica a las aplicaciones de Linux porque no se enlazan estáticamente, sino que utilizan una API basada en trampas.)

La GPL intenta hacer que los programadores contribuyan a un conjunto de programas en evolución y luego competir en la distribución y el soporte de este conjunto. Esta situación no es realista para muchos estándares de sistemas centrales requeridos, que pueden aplicarse en entornos muy variados que requieren personalización comercial o integración con estándares heredados bajo licencias existentes (no GPL). Los sistemas en tiempo real a menudo están vinculados estáticamente, por lo que muchas empresas de sistemas integrados consideran definitivamente problemas potenciales a la GPL y la LGPL.

La GPL es un intento de mantener los esfuerzos, independientemente de la demanda, en las etapas de investigación y desarrollo. Esto maximiza los beneficios para los investigadores y desarrolladores, a un costo desconocido para aquellos que se beneficiarían de una distribución más amplia.

La GPL fue diseñada para evitar que los resultados de la investigación se conviertan en productos patentados. A menudo se asume que este paso es el último en el proceso de transferencia de tecnología tradicional y, por lo general, es bastante difícil en las mejores circunstancias; la GPL estaba destinada a hacerlo imposible.

10. Ventajas de BSD

Una licencia de estilo BSD es una buena opción para investigaciones de larga duración u otros proyectos que necesitan un entorno de desarrollo que:

- Tiene un coste casi nulo
- evolucionará durante un largo período de tiempo
- permite que cualquiera conserve la opción de comercializar los resultados finales con un mínimo de problemas legales.

Esta consideración final a menudo puede ser la dominante, como lo fue cuando el proyecto Apache decidió su licencia:

"Este tipo de licencia es ideal para promover el uso de un código de referencia que implemente un protocolo para un servicio común. Esta es otra razón por la cual la escogimos para el grupo Apache - muchos de nosotros queríamos ver a HTTP sobrevivir y convertirse en un verdadero estándar comunitario, y no nos habría importado en lo más mínimo si Microsoft o Netscape hubieran elegido incorporar nuestro motor HTTP o cualquier otro componente de nuestro código en sus productos, si esto ayudaba además al objetivo de mantener a HTTP comunitario... Todo esto significa que, estratégicamente hablando, el proyecto necesita mantener el impulso suficiente, y que los participantes se den cuenta del valor de contribuir con su código al proyecto, incluso código que hubiera tenido valor si se mantuviera como propietario."

Los desarrolladores tienden a encontrar atractiva la licencia BSD, ya que evita los problemas legales y les permite hacer lo que quieran con el código. En contraste, aquellos que esperan principalmente usar un sistema en lugar de programarlo, o esperan que otros desarrollen el código, o que no esperan ganarse la vida con su trabajo asociado con el sistema (como los empleados del gobierno), encuentran la GPL atractivo, porque obliga a que se les proporcione código desarrollado por otros y evita que su empleador retenga los derechos de autor y, por lo tanto, potencialmente "entierre" o deje huérfano al software. Si quiere obligar a sus competidores a que le ayuden, la GPL es atractiva.

Una licencia BSD no es simplemente un regalo. La cuestión "¿por qué deberíamos ayudar a nuestros competidores o dejarles que roben nuestro trabajo?" surge frecuentemente en relación con la licencia BSD. Bajo una licencia BSD, si una compañia llega a dominar un nicho de mercado que otras consideran estratégico, las otras compañías pueden, con un mínimo esfuerzo, formar un mini-consorcio destinado a restablecer la igualdad contribuyendo a una variante de BSD competitiva que aumente la competencia y la equidad en el mercado. Esto permite a cada compañía tener confianza en que podrá beneficiarse de algunas ventajas, a la vez que contribuyen a la flexibilidad y eficiencia económica. Cuanto más rápido y fácil puedan hacerlo los miembros que están cooperando, más éxito tendrán. Una licencia BSD es esencialmente una licencia mínimamente complicada que permite tal comportamiento.

Un efecto clave de la GPL, hacer que un sistema de código abierto completo y competitivo esté ampliamente disponible al costo de los medios, es un objetivo razonable. Una licencia de estilo BSD, junto con consorcios ad-hoc de individuos, puede lograr este objetivo sin destruir los supuestos económicos construidos en torno al final de la implementación del proceso de transferencia de tecnología.

11. Recomendaciones específicas para usar una licencia BSD

- La licencia BSD es preferible para transferir los resultados de la investigación de una manera que se implementará ampliamente y beneficiará más a la economía. Como tal, las agencias de financiación de la investigación, como NSF, ONR y DARPA, deberían fomentar en las primeras fases de los proyectos de investigación financiados la adopción de licencias de estilo BSD para software, datos, resultados y hardware abierto. También deberían fomentar la formación de estándares basados en sistemas de código abierto implementados y proyectos de código abierto en curso.
- La política gubernamental debe minimizar los costos y las dificultades para pasar de la investigación a la implementación. Cuando sea posible, las subvenciones deben exigir que los resultados estén disponibles bajo una licencia de estilo BSD compatible con la comercialización.
- En muchos casos, los resultados a largo plazo de una licencia de estilo BSD reflejan con mayor precisión los objetivos proclamados en la carta de investigación de las universidades que lo que ocurre cuando los resultados están protegidos por derechos de autor o patentados y sujetos a una licencia universitaria de propiedad. Existen pruebas anecdóticas de que las universidades son mejor recompensadas financieramente a largo plazo al publicar los resultados de la investigación y luego apelar a las donaciones de los ex alumnos comercialmente exitosos.
- Las empresas han reconocido desde hace mucho tiempo que la creación de estándares de facto es una técnica de marketing clave. La licencia BSD cumple bien este papel, si una empresa realmente tiene una ventaja única en la evolución del sistema. La licencia es legalmente atractiva para la audiencia más amplia, mientras que la experiencia de la empresa asegura su control. Hay ocasiones en las que la GPL puede ser el vehículo adecuado para intentar crear tal estándar, especialmente cuando se intenta socavar o cooptar a otros. La GPL, sin embargo, penaliza la evolución de ese estándar, porque promueve una suite en lugar de un estándar comercialmente aplicable. El uso de una suite de este tipo plantea constantemente problemas legales y de comercialización. Puede que no sea posible mezclar estándares cuando algunos están bajo la GPL y otros no. Una verdadera norma técnica no debería exigir la exclusión de otras normas por razones no técnicas.
- Las empresas interesadas en promover un estándar en evolución, que puede convertirse en el núcleo de los productos comerciales de otras empresas, deben tener cuidado con la GPL. Independientemente de la licencia utilizada, el software resultante generalmente se transferirá a quien realmente realice la mayoría de los cambios de ingeniería y entienda el estado del sistema. La GPL simplemente agrega más fricción legal al resultado.
- Las grandes empresas, en las que se desarrolla código Open Source, deben ser conscientes de que los programadores aprecian Open Source porque deja el software disponible para el empleado cuando cambia de empleador. Algunas empresas fomentan este comportamiento como una ventaja laboral, especialmente cuando el software involucrado no es directamente estratégico. En efecto, es un beneficio de jubilación anticipado con posibles costos de oportunidad perdida pero sin costos directos. Alentar a los empleados a trabajar para obtener el reconocimiento de sus compañeros fuera de la empresa es un beneficio portátil económico que una empresa a veces puede ofrecer con un inconveniente cercano a cero.
- Las empresas pequeñas con proyectos de software que sean vulnerables a quedar huérfanos

deben intentar usar la licencia BSD cuando sea posible. Las empresas, sean del tamaño que sean, deben considerar la creación de proyectos de Código Abierto cuando les convenga mantener los gastos legales y de organización asociados con un verdadero proyecto de código abierto de estilo BSD.

- Las organizaciones sin ánimo de lucro deben participar en proyectos de código abierto cuando sea posible. Para reducir los problemas de ingeniería del software, tales como mezclar código bajo diferentes licencias, se deben fomentar las licencias de estilo BSD. Desconfiar de la licencia GPL debería ser particularmente el caso de las organizaciones sin ánimo de lucro que interactúan con el mundo del desarrollo. En algunos lugares donde la aplicación de la ley se convierte en un ejercicio costoso, la simplicidad de la nueva licencia BSD, en comparación con la GPL, puede ser una ventaja considerable.

12. Conclusión

A diferencia de la GPL, que está diseñada para evitar la comercialización patentada de código fuente abierto, la licencia BSD impone restricciones mínimas sobre el comportamiento futuro. Esto permite que el código BSD siga siendo de código abierto o se integre en soluciones comerciales, a medida que cambian las necesidades de un proyecto o empresa. En otras palabras, la licencia BSD no se convierte en una bomba de tiempo legal en ningún momento del proceso de desarrollo.

Además, dado que la licencia BSD no viene con la complejidad legal de las licencias GPL o LGPL, permite a los desarrolladores y empresas dedicar su tiempo a crear y promover un buen código en lugar de preocuparse si ese código viola la licencia.

Referencias bibliográficas

- [1] http://www.gnu.org/licenses/gpl.html
- [2] http://archives.cnn.com/2000/TECH/computing/03/28/cyberpatrol.mirrors/
- [3] Open Source: the Unauthorized White Papers, Donald K. Rosenberg, IDG Books, 2000. Las citas son de la página 114, "Effects of the GNU GPL".
- [4] En la sección "What License to Use?" de http://www.oreilly.com/catalog/opensources/book/brian.html

Este informe es un compendio del trabajo original disponible en http://alumni.cse.ucsc.edu/~brucem/open_source_license.htm