Bisoños Usuarios de GNU/Linux de Mallorca y Alrededores | Bergantells Usuaris de GNU/Linux de Mallorca i Afegitons

Trabajando con proyectos de software libre (37454 lectures)

Per **Jesús Roncero Franco**, ***golan*** (http://www.roncero.org)

Creado el 10/06/2003 21:17 modificado el 10/06/2003 21:17

¿Alguna vez has querido hacer tu propio proyecto de software libre y no sabías por dónde empezar?
¿Has querido colaborar con algún proyecto y no sabes qué hacer?

En este artículo, traducido del original en inglés[1] de Havoc Pennington[2], trabajador de red hat, se da una serie de consejos útiles para aquellos que no saben por dónde empezar tanto si lo que se quiere es aprender como si se quiere programar, documentar, etc. en un proyecto de software libre.

Muchos programadores quieren empezar a trabajar en un proyecto de software libre, pero no están muy seguros de cómo funcionan las cosas. Esta página es una colección informal de "reglas no escritas y consejos" para gente que le gustaría participar de voluntario. He aprendido alguna de estas reglas comentiendo errores, y algunas veces incluso incumplo flagrantemente mis consejos; Es sólo una guía :) Seguro que otra gente ofrece otra panorámica sobre el asunto.

No empieces creando tu propio proyecto Mucha gente quiere escribir software libre y lo primero que hacen es escribir algo de código, adosarle la GPL y liberar la versión 0.0.1 alpha. Aunque sea divertido y posiblemente educativo, esto es totalmente improductivo. Veamos por qué:

- Es mas instructivo leer y aprender del código de otra gente añadiendo pequeñas funcionalidades aquí y allá o arreglando algunos bugs. La mayoría de los proyectos tienen un sistema de bugs; Por ejemplo, en el proyecto Gnome tenemos bugs.gnome.org[3] , Debian tiene el mismo sistema, bugs.debian.org[4], etc. Encuentra un bug en el sistema y corrigelo. O añade la funcionalidad que estabas deseando.
- Obviamente, es mucho más útil limpiar el código existente que empezar un proyecto en solitario que nunca va a ser finalizado.
- Casi seguro, algún otro ya está trabajando en lo que quieras escribir; es mejor terminar un proyecto que tener dos proyectos sin terminar. Te prometo que el 95% de los proyectos de software libre caen en el olvido antes de que se conviertan en útiles. Tanto desde el punto de vista de tu propio aprendizaje como desde el punto de ganar fama y mejorar tu ego, necesitas gente que te ayude a entrar en el 5% ganador.
- Si no has estado curioseando y participando en un proyecto de software libre, no sabrás como se hacen las cosas y pasarás unos malos ratos intentando llevar el tuyo adelante.

Una vez dicho todo esto, si tienes una idea genial y piensas que sería divertido programar para ella, hazlo por todo los medios. Todos lo hemos hecho. :-) Algunos de estos proyectos llegan a algún lado. Y si tienes alguna experiencia codificando y hay una aplicación interesante en la que nadie está trabajando, definitivamente, ve a por ella.

Programa, Programa, Programa. Si empiezas un proyecto, la cosa más importante es escribir código. Tienes que escribir suficiente para hacer que la aplicación sea útil; esto puede llevar meses o años de trabajo solitario, a menos que algún alma caritativa decida ayudarte con tu aplicación en vez de empezar la suya. :-) Tienes que liberar versiones a menudo, arreglar los bugs rápidamente y, generalmente, mantener las cosas excitantes. Cuando te das cuenta, escribir una aplicación libre supone una cantidad enorme de trabajo. Si estás programando por tu cuenta, dedica unas 10-20 horas cada semana, previendo el futuro. Si no puedes dedicar todo este tiempo, no te molestes siquiera en empezar el proyecto. Si ni siquiera sabes programar, lo dicho.

Comienza por tí mismo. Mucha gente manda un mensaje diciendo que va a escribir la aplicación X, o anuncia la versión 0.0.1 alpha y entonces lo abandona cuando no encuentra ninguna respuesta. Aceptalo: No habrá mucha respuesta hasta que tengas usuarios. Y no tendrás usuarios hasta que escribas código. Estarás tu y tu determinación por

escribir ese software.

Ocurre el mismo fenómeno cuando se pide ayuda. En última instancia, si un bug, una característica que falta o una falta en la documentación se cruza en tu camino, probablemente tendrás que arreglarla por ti mismo. Los hackers son suficientemente amables con los novatos que no saben dónde empezar, pero tarde o temprano esperan que seas capaz de solventar tus propios problemas.

Usa las listas de correo. Si tienes una pregunta, hazla en la lista de correo. Es un poco maleducado enviar mensajes privados a los desarrolladores, a menos que tengas una razón para creer que es la única persona que pueda responder a la pregunta. (Si usas el correo privado, lo más probable es que se lo mandes al desarrollador equivocado y termines sin tener una respuesta porque quizá la persona a la que escribiste no sepa del asunto.)

Las listas de correo y la documentación las ponen los programadores para hacer posible ayudar a un gran número de usuarios. Recuerda que todo el mundo es un voluntario, y que si necesitas consultoría pagada, probablemente esté disponible.

Nadie está al cargo. La gente suele esperar que alguien esté al cargo de un proyecto de software libre, o esperan que alguien les asigne algún trabajo que hacer y una fecha límite. No funciona de esa manera. No puedes controlar sobre lo que otra gente trabaja, y nadie va a decirte sobre qué tienes que trabajar, aunque habrá muchos consejos. Tendrás que *pringarte* un poco y hacer que algo ocurra.

Coordinate con otros. Si pasas 3 meses escribiendo alguna funcionalidad guapa, y luego descubres que el mantenedor odia esa idea y que no aceptará el parche, o encuentras que tu parche no se puede aplicar a la última versión en desarrollo, o ves que alguien ya hizo el mismo trabajo, estarás fustrado. Si tienes intención de trabajar sobre algo, manda una pequeña nota al mantenedor para que lo sepa. La mayoría serán escepticos (reciben muchas notas y nunca ven los resultados) pero harán una nota mental y quizá te den algún consejo.

Si acabas fuertemente involucrado en un proyecto, se puede tener una coordinación más fina usando una combinación de email, CVS e IRC. El CVS hace un gran trabajo uniendo los cambios cuando la comunicación se rompe.

No hay respuesta a la pregunta: ¿Cuándo estará X terminado?" ó "¿Se implementará la funcionalidad X?" La respuesta a ambas preguntas es "cuando alguien haga el trabajo". Si ese alguien eres tu, entonces sabes la respuesta. Si ese alguien no eres tu, entonces tendrás que esperar y ver qué pasa. :-) No hay ningún coordinador ni nadie que se asegure de que las cosas ocurren.

Los programadores de sillón son malos. Un programador de sillón parece tener un flujo incesante de ideas birllantes cuando escribe en las listas de correo, pero o bien no programa o no sabe cómo programar. Si no sabes programar, no sabes como diseñar software tampoco. Punto. Solo causarás problemas.

Hay un montón de cosas que los no programadores pueden hacer de todas formas: Mandar informes de errores, petición de características (siempre que no sean muy especulativas y no estén relacionadas con el código existente), escribir documentación, ayudar a los usuarios con sus preguntas y la instalación, grupos de usuarios, mantenimiento de páginas web, administracion de servidores, hacer paquetes para distribuciones, etc. Los hackers aprecian el interés por su trabajo y tu ayuda.

Curiosear[*] es bueno. La tentación de apuntarte a una lista de correo y empezar a comentar sobre todo suele ser fuerte. Pero no te conviertas en un programador de sillón. Escribe si tienes cosas relevantes que contar, si descubres cómo reproducir un bug, si sabes como coger experiencia en en el tema, si conoces las respuestas a las preguntas. Si no es así, no mandes mensajes. Además, siempre es buena idea curiosear durante un tiempo antes de que empieces a escribir, simplemente para ver de qué va el tema.
[*] Lurk es la palabra utilizada en inglés (Nota del Traductor)

Comprende los copyrights, las patentes, licencias, trademarks y demás. Tienes que ser un poco abogado para involucrarte en software libre. Esto significa que tienes una reponsabilidad para contigo para aprender sobre ello. Tradicionalmente, una manera de aprender es observar las mismas peleas de siempre que ocurren semana tras semana en el grupo de noticias gnu.misc.discuss. Una manera más rápida es leer el web de GNU[5], en particular su análisis de la terminología. No envies correos sobre temas legales si no los comprendes. Pero deberías comprenderlos si vas a escribir software y aplicarle una licencia al mismo.

Respeta los deseos del mantenedor del paquete. Es siempre bueno usar la misma licencia, el estilo de código, etc. que el que utiliza el mantenedor del paquete al que le estás mandando un parche. Si usas CVS, no envies tus parches sin pedir primero permiso al mantenedor del paquete.

Usa la opción -u de diff cuando mandes un parche. Un punto de menor importancia, pero casi siempre todo el mundo prefiere este formato.

Recuerda que todo el mundo es voluntario. Trátalos con el respeto que merecen. Sólo están trabajando en ello porque disfrutan. Es de desagradecidos maltratar a alguien que te ha dado algo gratis.

Intenta estar concentrado. Muchos tenemos problemas con esto, pero cuanto más te centres en una tarea y la termines, tus resultados serán más útiles. Yo encuentro que tengo que coger pequeñas tareas para llevar esto adelante. Otra gente es mejor en los proyectos de más largo plazo. Organiza tus esfuerzos en concordancia con tu personalidad. Intenta terminar proyectos antes de empezar 100 proyectos superambiciosos.

Aprende sobre la comunidad. Es una buena idea mantenerte informado en los sitios de noticias como LinuxToday[6], LWN[7] ó Slashdot[8] y también los sitios relacionados con los proyectos con los que estés trabajando. (Los tres que he mencionado son tres que yo leo). Un buen libro sobre la historia de la comunidad es *Hackers*, escrito por Steven Levy. www.gnu.org[5] tiene mucha información también y curioseando en las listas de correo aprenderás un montón.

Participarás en peleas. De cualquier manera, hagas lo que hagas, alguién te atacará y te verás envuelto en una pelea o *flame*. Internet es así. Algunas personas no han aprendido esta lección todavía y se cabrean y responden virulentamente cuando pasa; No lo hagas. Si lees las listas de correo, pronto aprenderás qué gente es más propensa a tener puntos de vista válidos y cuáles son *flamers* habituales. (Date cuenta que ambos no son mútuamente excluyentes; algunos de los hackers más productivos son también flamers). Desarrolla una piel resistente, la vas a necesitar.

Diviertete. Programar es, al fin y al cabo, trabajar. Es sentarse y dedicar tiempo a escribir código o documentación. Pero hay muchas oportunidades de socializar, en conferencias, en IRC, a través del email. Escribir código es divertido durante la mayoría del tiempo. Así que disfrutalo. Eso es parte del motivo. No tomes demasiado en serio este documento o alguna de sus reglas.

Enlaces: Las ideas de Eric Raymond sobre este tema están en aquí[9] [traducción Aquí[10]]; Su HOWTO describe como unirte a la "cultura hacker". La cultura no es realmente necesaria para participar en proyectos de software libre, en mi opinión. Mientras sigas la manera de trabajar que sigue la comunidad no tienes que meterte en los temas sociales de la misma (a menos que quieras). Advogato (el alter ego de Raph Levien) también tiene un ensayo aquí[11]. Después de leer todos estos consejos, en las palabras de RMS, *happy hacking!*.

Traducido con permiso del autor del original en http://www106.pair.com/rhp/hacking.html[1], por Robert Sanford Havoc Pennington.

Lista de enlaces de este artículo:

1. http://www106.pair.com/rhp/hacking.html
2. http://www106.pair.com/rhp/
3. http://bugs.gnome.org
4. http://bugs.debian.org
5. http://www.gnu.org
6. http://www.linuxtoday.com
7. http://lwn.net
8. http://slashdot.org
9. http://catb.org/~esr/faqs/hacker-howto.html
10. http://sindominio.net/biblioweb/telematica/hacker-como.html
11. http://www.advogato.org/article/22.html

E-mail del autor: jesus _ARROBA_ roncero.org
Podrás encontrar este artículo e información adicional en: http://bulma.net/body.phtml?nIdNoticia=1785