

Software libre

Juan José Amor Iglesias
Israel Herraiz Tabernero
Gregorio Robles Martínez

XP06/M2120/02157



Desarrollo de proyectos de software libre

David Megías Jiménez

Coordinador

Ingeniero en Informática por la UAB.
Magíster en Técnicas Avanzadas de Automatización de Procesos por la UAB.
Doctor en Informática por la UAB.
Profesor de los Estudios de Informática y Multimedia de la UOC.

Jordi Mas Hernández

Coordinador

Ingeniero de software en la empresa de código abierto Ximian, donde trabaja en la implementación del proyecto libre Mono. Como voluntario, colabora en el desarrollo del procesador de textos Abiword y en la ingeniería de las versiones en catalán del proyecto Mozilla y Gnome. Es también coordinador general de Softcatalà. Como consultor, ha trabajado para empresas como Menta, Telépolis, Vodafone, Lotus, eresMas, Amena y Terra España.

Juan José Amor Iglesias

Autor

Licenciado en Informática por la Universidad Politécnica de Madrid. Fundador de LuCAS (actualmente TLDP-ES). Fundador de Hispalinux. En la actualidad cursa los estudios de Doctorado en la Universidad Rey Juan Carlos.

Israel Herraiz Tabernero

Autor

Ingeniero Industrial por la Universidad de Cádiz. En la actualidad, cursa los estudios de Doctorado en Informática y Modelización Matemática en la Universidad Rey Juan Carlos.

Gregorio Robles Martínez

Autor

Ingeniero de Telecomunicación por la Universidad Politécnica de Madrid. Proyecto fin de carrera en la TU Berlín. Profesor ayudante en la Universidad Rey Juan Carlos. Actualmente, está terminando el doctorado. Consultor del Máster Internacional de Software Libre en la UOC.

Segunda edición: febrero 2007

© Fundació per a la Universitat Oberta de Catalunya

Av. Tibidabo, 39-43, 08035 Barcelona

Material realizado por Eureka Media, SL

© Autores: Juan José Amor Iglesias, Israel Herraiz Tabernero, Gregorio Robles Martínez

Se garantiza permiso para copiar, distribuir y modificar este documento según los términos de la GNU Free Documentation License, Version 1.2 o cualquiera posterior publicada por la Free Software Foundation, sin secciones invariantes ni textos de cubierta delantera o trasera. Se dispone de una copia de la licencia en el Apéndice A, junto con una traducción no oficial en el Apéndice B.

Índice

Agradecimientos	7
Prólogo	9
Materiales previos	9
Introducción	11
Objetivos	12
Conocimientos previos	13
Requisitos técnicos	14
Contenidos de este material	14
1. Herramientas de gestión de proyectos	17
1.1. Servicios útiles para proyectos de software libre ...	18
1.2. Sitios de desarrollo	19
1.2.1. Software-libre.org	20
1.2.2. Savannah	20
1.2.3. Alioth	21
1.2.4. BerliOS	21
1.2.5. SourceForge	21
1.3. Registro del proyecto	22
1.4. La cuenta de línea de comandos	27
1.4.1. Acceso mediante SSH sin clave	28
1.4.2. Cómo se pueden copiar los ficheros de la web	29
1.5. Configuración del CVS	30
1.5.1. Acceso anónimo	30
1.5.2. Acceso como desarrollador	31
1.6. Descargas del proyecto	32
1.7. Listas de correo	36
1.8. Tracker y el sistema de seguimiento de fallos	39
2. Control de versiones	45
2.1. Instalación y configuración inicial de CVS	47
2.1.1. Creación del repositorio	47
2.1.2. Preparación del acceso anónimo	48
2.1.3. Apertura de cuentas para los desarrolladores	49

2.2. Operativa básica del CVS	49
2.2.1. Acceso anónimo a un repositorio	49
2.2.2. Acceso al CVS por el desarrollador	50
2.2.3. Creación de un proyecto en el repositorio	50
2.2.4. Obtención del proyecto	51
2.2.5. Creación de ficheros y directorios	51
2.2.6. Modificación de los ficheros.	
Fusión de cambios	52
2.2.7. Eliminación de ficheros	54
2.3. Operativa avanzada en el CVS	54
2.3.1. Versiones	55
2.3.2. Etiquetas	56
2.3.3. Ramas	57
2.3.4. Información sobre etiquetas y ramas	59
2.4. Subversion: la próxima generación	60
3. Sistemas de seguimiento de fallos	63
3.1. Seguimiento de fallos con Bugzilla	64
3.1.1. Los fallos en Bugzilla	64
3.2. Instalación y configuración de Bugzilla	65
3.2.1. Instalación de Bugzilla	65
3.2.2. Configuración de Bugzilla	66
3.3. Notificación de fallos	69
3.3.1. Creación de cuentas	69
3.3.2. Notificación de un fallo	70
3.4. Búsqueda y tratamiento de fallos	72
3.4.1. Búsqueda de fallos	72
3.4.2. Estados de un fallo	73
3.4.3. Edición de un fallo	75
4. Listas de correo electrónico	77
4.1. Qué es una lista de correo	77
4.2. Herramientas	78
4.3. Alternativas	79
4.4. Listas de correo con Mailman	80
4.4.1. Instalación	80
4.4.2. Configuración de la lista de gestión	83
4.4.3. Operativa de usuario de listas	84
4.4.4. Operativa de administración de listas	91
5. La gestión de un proyecto	97
5.1. Elección de la licencia	97
5.2. El sitio web del proyecto	980
5.3. Estructura de la web del proyecto	100

5.4. Instalación sencilla	101
5.5. Consejos para la distribución y difusión del software	102
5.6. En busca del efecto en red	104
5.7. Código e internacionalización	106
5.8. Esfuerzo dedicado a tareas de gestión	107
5.9. Conclusiones	110
Bibliografía	111
Appendix A. GNU Free Documentation License	113
Apéndice B. Licencia de Documentación Libre de GNU	125

Agradecimientos

Los autores agradecen a la Fundación para la Universitat Oberta de Catalunya (<http://www.uoc.edu>) la financiación de la primera edición de esta obra, enmarcada en el Máster Internacional de Software Libre ofrecido por la citada institución.

Prólogo

Materiales previos

Algunos textos de estos apuntes están basados en materiales previos (utilizados con permiso, cuando no han sido completamente reelaborados). Entre ellos podemos mencionar los siguientes (a riesgo de olvidar alguno importante):

- La parte dedicada a la liberación de proyectos de software libre ha sido elaborada a partir de las transparencias y presentación que hace Jesús M. González Barahona, profesor de la Universidad Rey Juan Carlos, en el curso de doctorado *Software Libre* que tiene lugar conjuntamente en las Universidades Politécnica de Madrid y Rey Juan Carlos.
- Las herramientas que se presentan han sido tomadas también de las transparencias y las clases que imparte Joaquín Seoane, profesor de la Universidad Politécnica de Madrid, en el citado curso de doctorado.
- Algunas gráficas del capítulo dedicado al sistema de control de versiones (CVS) han sido tomadas del libro *Open Source Development with CVS*, de Karl Fogel, que se publica bajo una licencia libre.
- La traducción de la Licencia de Documentación Libre de GNU es una actualización adaptada de la realizada por Igor Támara y Pablo Reyes para la versión 1.1, a los que agradecemos su confección y su permiso para modificarla.

Introducción

El software libre se ha convertido en los últimos años en un fenómeno imparable. Ciertamente, el desafío que supone va más allá de elementos meramente técnicos, ya que se diferencia del software “tradicional” propietario en aspectos más fundamentales, que involucran desde razones filosóficas hasta nuevas pautas económicas y de mercado. En este material abordaremos exclusivamente cuestiones técnicas, que abarcan desde el entorno en el que se crea el software libre hasta el modo de conseguir que nuestros proyectos de software libre se aprovechen de este fenómeno para tener éxito.

Y es que, a pesar de este incipiente éxito, todavía son poco frecuentes los documentos y materiales que expliquen de manera detallada qué hacer para que nuestro proyecto se beneficie de todos esos aspectos que parecen estar asociados al fenómeno del software libre. En cualquier caso, podemos afirmar, sin riesgo a equivocarnos, que este conocimiento aún no ha llegado ni a los cursos universitarios o de otros centros docentes, ni a los grandes libros de ingeniería del software.

Dos aspectos fundamentales caracterizan el desarrollo de software libre:

- a) Se trata de un desarrollo en red. Es decir, está orientado a unas prácticas que posibiliten la creación de software a personas que están dispersas, pero que pueden comunicarse mediante Internet.
- b) Está muy orientado a las herramientas que lo soportan. Esto quiere decir que suele haber una herramienta específica para cada proceso que se ha convertido en una especie de estándar *de facto*. Las herramientas utilizadas a día de hoy tienen su sentido histórico y es posible que con el paso del tiempo evolucionen con el propio software libre. Como el lector habrá podido imaginar, se trata de herramientas que tienen un componente muy arraigado en Internet, precisamente por ser ésta la primera característica del desarrollo de software libre.

Aprender cómo es el desarrollo software libre consiste en gran parte en asimilar y saber utilizar las herramientas que lo sustentan. Por este motivo, gran parte de este material se centra en estas herramientas y muestra su uso con detalle.

Finalmente, cabe señalar que el éxito de un proyecto de software – ya sea de software libre con los métodos que se explican en este material, o de software propietario utilizando cualquiera de los procesos de desarrollo actuales– depende de multitud de factores y que no está asegurado en ningún caso. Suponemos que este aspecto no es nuevo para el lector, pues, por desgracia, la ingeniería del software todavía está en una fase que permite indicar el camino por donde hay que ir, pero no asegurar que se llegue a la meta.

El lector encontrará aquí unos conocimientos que han sido adquiridos mediante la observación y la experiencia de muchos proyectos de software libre y de estudios elaborados hasta la fecha. Éstos y el día a día de los proyectos de software libre y de los problemas a los que el lector se enfrenta en su afán por sacar adelante un proyecto de software libre serán, sin duda, la mejor fuente de sabiduría de la que el programador podrá echar mano. Si, finalmente, el proyecto de software libre tiene éxito, siempre podrá decir que ha sido gracias a sus grandes dotes técnicas y de gestión de recursos (humanos y técnicos). Si, por el contrario, fracasa, podrá echar la culpa al escaso conocimiento que de este campo se tiene en la actualidad y, en especial, a este texto y a sus autores.

Objetivos

Este material pretende ser, en definitiva, una guía para quienes quieran crear un proyecto de software libre y pretendan beneficiarse de aquellos aspectos que se destacan en la literatura cuando se habla del fenómeno del software libre. Entre dichos aspectos, se podrían mencionar la incorporación de otros programadores que ayuden en el desarrollo de la aplicación, una alta realimentación por parte de los usuarios o la creación de una comunidad en torno a la aplicación que no sólo la utilice, sino que lleve a cabo tareas de promoción, documentación o traducción a otros idiomas.

Una vez terminado el curso, el lector estará completamente familiarizado con las herramientas que se utilizan tanto para el desarrollo colaborativo en el mundo del software libre como para la intercomunicación de los diferentes agentes que participan en el mismo. Con estos conocimientos, estará en disposición de crear y lanzar un software propio o entrar a formar parte de alguno de los muchos proyectos de software libre que existen.

Conocimientos previos

Para el completo aprovechamiento de los contenidos que se exponen en estos materiales, el lector deberá contar con unos conocimientos previos en lo que se refiere a la filosofía del software libre, algunos conocimientos básicos del proceso de desarrollo del software libre, las diferentes licencias de software libre que existen y los proyectos de software libre más importantes a día de hoy. Gran parte de estos conocimientos, si no todos, pueden encontrarse en *Introducción al software libre*, el libro de texto que la Fundación Universitat Oberta de Catalunya (Fundación UOC) ha editado para el Máster de Software Libre que ofrece y que está a disposición pública en Internet.

En el ámbito técnico, el lector ha de estar familiarizado con un entorno GNU/Linux y con algunas herramientas básicas como son el navegador web, el cliente de correo y la línea de comandos (*shell*). En este sentido, se recomienda el material *Introducción a GNU/Linux* del Máster de Software Libre ofrecido por la Fundación UOC.

Es importante destacar que en este curso no se va a enseñar a programar. Se da por supuesto que el lector cuenta con las nociones de programación necesarias para crear software. Los aspectos en los que se va a incidir tienen más que ver con cómo conseguir que ese software entre en una dinámica de llegar a nuevos desarrolladores y usuarios, y cree a su alrededor una comunidad que finalmente desarrolle el software de manera colaborativa.

El lector apreciará, en este momento como muy tarde, que ser un buen programador es solamente uno de los requisitos necesarios, pero no es ni mucho menos suficiente para que el proyecto de soft-

ware libre tenga éxito. Son otros aspectos, más relacionados con la gestión o la comunicación, los que diferencian los proyectos de éxito de los que no consiguen despegar. Y son estos aspectos los que vamos a estudiar aquí, los cuales, aun teniendo poco que ver con el arte de la programación, sin duda se pueden considerar conocimientos técnicos, como se podrá observar en breve.

Por otro lado, también será de gran ayuda un buen conocimiento de la lengua inglesa, ya que muchos de los entornos web y herramientas que se van a utilizar están en ese idioma.

Requisitos técnicos

Para el seguimiento de este material, el lector debe contar con un sistema GNU/Linux instalado y configurado. Se recomienda el uso de la distribución Debian (en especial, la última versión estable) por ser la que se ha utilizado en la confección de estos materiales, pero cualquier distribución medianamente actual que se utilice no debería dar problemas. Si así se desea, también se podrán utilizar otros sistemas UNIX (por ejemplo, de la familia BSD o Solaris) e incluso Windows, aunque es probable que las instrucciones y las herramientas cambien con el entorno y no se correspondan con las que se presentan aquí (sobre todo en el caso de Windows).

En cualquier caso, deberá contar con la posibilidad de instalar nuevos paquetes en la máquina y con una (buena) conexión a Internet.

Contenidos de este material

A continuación se repasan brevemente los contenidos que se van a tratar en este material.

En el capítulo 1 se presentarán las herramientas que se utilizan de manera casi universal en el desarrollo de proyectos de software libre. Dado que hay sitios que las ofrecen gratuitamente para proyectos con licencias libres, se dará una breve descripción del aspecto que

presentan estos sitios y de cómo se utilizan. En este capítulo se presentarán brevemente las herramientas más típicas, aunque se dejará su estudio en detalle para capítulos siguientes. En particular, el sitio que hemos elegido para tal menester es el conocido sitio de desarrollo SourceForge, aunque se indicarán otros sitios de características semejantes para que el lector pueda decidir cuál es el que más le conviene.

El capítulo 2 muestra en profundidad el sistema de control de versiones más utilizado en el mundo del software libre, el CVS (*concurrent versions system*). Se enfocará desde dos puntos de vista: como administrador de un repositorio CVS o como simple usuario. Desde el punto de vista de administrador, se enseñará cómo se instalan, configuran y gestionan las cuentas de usuarios. En cuanto al punto de vista de los usuarios, se incidirá en su modo de funcionamiento básico y nos adentraremos en algunos elementos avanzados.

El capítulo 3 trata los sistemas de gestión de fallos, en particular el sistema más difundido y utilizado en grandes proyectos de software libre: Bugzilla. Al igual que con el sistema de control de versiones, también mostraremos cómo instalar y configurar Bugzilla, por si el lector pretende hacer de administrador. Para los que tengan como objetivo simplemente el uso efectivo de Bugzilla, bastará con que sepan cómo se notifican, buscan y gestionan los errores en este sistema de control de errores.

El capítulo 4 está dedicado a los gestores de listas de correo, en especial al gestor de listas de correo GNU Mailman. Este sistema se examinará también desde dos puntos de vista: el del usuario de las listas (por ejemplo, cómo puede darse de alta y de baja o configurar sus propias opciones de entrega de mensajes) y el del administrador, que cubre en este caso la instalación y configuración de Mailman y la gestión de listas de correo.

Finalmente, llegamos a un apartado que es independiente de las herramientas que utilicemos para nuestro proyecto de software libre. El capítulo 5 pretende ser una guía de los pasos que hay que seguir para conseguir crear una comunidad en torno a nuestro proyecto. Las prácticas y las buenas maneras que se proponen en este capítulo son lecciones aprendidas de los proyectos de software libre que han

tenido éxito. Es probable que este apartado sea el más provechoso para el lector a largo plazo, ya que el aprendizaje de las herramientas suele ser mucho más sencillo. Pero, sin lugar a dudas, en la temática que se trata en este último capítulo está la verdadera esencia del desarrollo de software libre.

1. Herramientas de gestión de proyectos

El desarrollo de software libre está fuertemente arraigado al de Internet y a una serie de herramientas de apoyo. Así como la tarea de implementación suele ser concebida como una actividad individual, salvo en excepciones como la programación extrema, muchas otras tareas requieren la interacción de diversas personas y grupos de trabajo.

Ejemplo

Así como la generación de documentación ha sido uno de los caballos de batalla en el ciclo de desarrollo de software tradicional a la búsqueda de una mejora en la calidad del software, la práctica de la documentación en el mundo del software libre nunca ha estado muy extendida, en parte por la propia naturaleza voluntaria de los desarrolladores, que evitan las tareas tediosas y menos divertidas. Sin embargo, existen mecanismos de intercambio de información, necesario para el desarrollo de software con otros desarrolladores y con los usuarios a diferentes niveles que van desde el propio código fuente hasta las listas de correo electrónico.

Para cada tipo de comunicación suele existir una herramienta específica que con el tiempo se ha convertido en el estándar *de facto*. Una vez conocidas las tareas más comunes, empezaron a aparecer portales que las integraban todas dentro de una misma interfaz y que permitían a los desarrolladores apoyarse en las herramientas que se les ofrecían y olvidarse así de su gestión.

Evidentemente, a la hora de elegir la infraestructura necesaria para el proyecto, tenemos dos alternativas claras:

1. Elegir nosotros mismos todas las herramientas necesarias para cada sistema del proyecto (listas de correo, CVS, sistema de se-

Nota

En este material se expondrá cómo llevar a cabo el desarrollo de software con la infraestructura necesaria en los dos casos mencionados, empezando por el más fácil, que es cuando ya disponemos de una plataforma lista para ser utilizada.

guimiento de fallos, etc.), y llevar a cabo toda la configuración y administración.

2. Emplear algún servicio en el cual todas estas herramientas ya estén montadas, como los que se van a presentar en el siguiente apartado.

La decisión debe tomarse teniendo en cuenta el tamaño del proyecto. Salvo en proyectos realmente grandes, no está justificado llevar a cabo el montaje, la configuración y la administración de toda la infraestructura. Sin embargo, es difícil establecer una frontera clara entre los proyectos que requieren una infraestructura propia y los que no la requieren. Como norma, podríamos decir que si se trata de un proyecto personal o de un grupo pequeño de participantes, es más indicado utilizar una plataforma externa para no tener que completar las tareas de instalación, configuración y mantenimiento relacionadas. Si, por el contrario, un proyecto o proyectos que se llevan a cabo por un grupo numeroso de miembros (una asociación, una empresa mediana o grande, etc.), entonces puede resultar conveniente tener una infraestructura propia.

1.1. Servicios útiles para proyectos de software libre

En el desarrollo de software libre se suelen utilizar las siguientes herramientas:

- Alojamiento (*hosting*) de webs, con posibilidad de usar guiones (*scripts*) PHP y guiones CGI para hacer las webs dinámicas.
- Archivo permanente de ficheros, donde se pueda descargar el software.
- Tablón y foro de mensajes, para la intercomunicación con desarrolladores y usuarios.
- Listas de correo, para la intercomunicación con desarrolladores y usuarios.
- Organizador de tareas, para gestionar los recursos humanos del proyecto.

- Sistema de seguimiento de fallos y de solicitud de nuevas características, para facilitar la notificación de errores y la realimentación a la comunidad.
- Bases de datos para la administración del proyecto (por ejemplo, para tener un sitio web basado en un gestor de contenidos, donde cada usuario tiene su cuenta).
- Sistema de control versiones, para poder trabajar de manera simultánea en el código; la mayoría ofrecen CVS, aunque parece que próximamente migrarán a Subversion (SVN).
- Cuenta de línea de comandos (*shell*) en el servidor, para manipular los ficheros relacionados con la web.
- Administración total basada en una interfaz web, que simplifica sobremanera la administración de todos los elementos anteriores.

1.2. Sitios de desarrollo

Muchos sitios ofrecen alojamiento gratuito para proyectos de software libre que incluyen los servicios que acabamos de comentar. En general, estos sitios están contruidos sobre la base de una aplicación web llamada Alexandria, que inicialmente se desarrolló para el sitio SourceForge. Sin embargo, desde hace un par de años, la empresa que mantiene SourceForge ha decidido que las nuevas versiones de Alexandria no se publicarían bajo una licencia de software libre, por lo que apareció el proyecto Gforge, que no es más que una ramificación (*fork*) a partir de la última versión libre de Alexandria.

Evidentemente, al estar Gforge disponible como software libre, cualquiera puede instalarlo y configurarlo y ofrecer un nuevo sitio para alojar y desarrollar proyectos de software libre. Así lo hacen muchos, como podrá observar el lector si visita la página principal del proyecto, aunque no es una tarea fácil, si se quiere ofrecer un servicio impecable. Sin embargo, ésta no es una labor necesaria para nuestros propósitos y no se va a tratar en este material. Los sitios que se encargan de esta tarea tienen equipos dedicados exclusivamente a ofrecer este servicio, y cuentan con los conocimientos necesarios y con la infraestructura (servidores, ancho de banda, espacio en disco) requeridos para tal fin.

Los sitios que se mencionan a continuación están basados en plataformas similares entre sí, aunque tanto sus funcionalidades como su apariencia pueden variar ligeramente según la versión de Alexandria/GForge que utilicen.

1.2.1. [Software-libre.org](http://www.software-libre.org)

Software-libre.org nació como una iniciativa de la asociación Hispalinux con el propósito de tener una forja de proyectos propia. En principio nació como “un sitio en el que poder desarrollar conocimiento libre en general y software libre en particular en idioma español, aunque no se descarta el uso de otros idiomas”. Sin embargo, a día de hoy su política restrictiva de usuarios (no permite tener cuenta a usuarios que no pertenezcan a una organización que tenga un acuerdo con Hispalinux) ha dificultado su popularización.

Además, en este momento, sólo se encuentra en castellano, por lo que tampoco es recomendable para proyectos de software libre que busquen una gran difusión.

1.2.2. [Savannah](http://savannah.gnu.org)

Savannah es una iniciativa de la Free Software Foundation. Cualquier persona puede abrir una cuenta en este sitio, aunque en el caso de registro de proyectos, se imponen una serie de restricciones que se detallan a continuación:

- Los recursos puestos a disposición de los usuarios no pueden usarse para promocionar o desarrollar software no libre, entendido como *software libre* aquel software que cumple la definición dada por la Free Software Foundation.
- El software alojado no puede depender de software no libre.
- No puede ponerse publicidad en el sitio web, excepto anuncios comerciales de apoyo al proyecto.
- No pueden emplearse ficheros de imagen en formato GIF porque este tipo de formato está patentado (aunque recientemente expiró la patente en todo el mundo).

Nota

Software-libre.org:

<http://www.software-libre.org>

Nota

Savannah:

<http://savannah.gnu.org>

- La licencia del proyecto ha de ser libre; es decir, cualquiera puede utilizar, copiar, modificar y distribuir el software sin restricciones, y deben respetarse estas libertades básicas en el caso de que se distribuya.
- Si se emplea el término GNU dentro del nombre del proyecto, deben cumplirse una serie de requisitos adicionales.

Savannah tiene como característica curiosa que distingue entre proyectos que pertenecen al proyecto GNU y proyectos que no.

1.2.3. Alioth

Alioth es el sitio de Debian para ofrecer alojamiento a proyectos de software libre. Su política de usuarios no es restrictiva, esto es, cualquier usuario puede obtener una cuenta en este sitio. Tampoco tiene restricciones especiales, excepto que el proyecto ha de ser de software libre o estar relacionado con el software libre. Su interfaz está disponible en diversos idiomas, entre ellos el español y el inglés.

1.2.4. BerliOS

BerliOS es similar a los sitios anteriores. Ofrece todos los servicios ya mencionados y su política de usuarios y proyectos no es restrictiva. Está disponible en diversos idiomas, entre ellos el español y el inglés.

Este sitio es uno de los más populares entre los desarrolladores de software libre y siempre se ha visto como la competencia de SourceForge que nunca ha terminado de arrancar.

1.2.5. SourceForge

Sin duda, SourceForge es el sitio más famoso entre los desarrolladores de software libre. Cuenta con casi un millón de usuarios registrados y casi cien mil proyectos dados de alta, aunque muchos de ellos nunca han presentado ninguna actividad. Ofrece todos los servicios ya mencionados. Su interfaz está sólo disponible en inglés.

Presenta el inconveniente de la masificación, que en ocasiones perjudica su rendimiento. Aunque, a su vez, tanta actividad repercute sin

Nota

Alioth:

<http://alioth.debian.org>

Nota

BerliOS:

<http://developer.berlios.de>

Nota

SourceForge:

<http://sourceforge.net>

Nota

La única herramienta que vamos a necesitar por ahora para seguir estas instrucciones es un navegador web.

duda positivamente en el fomento del software, tanto desde el punto de vista del conocimiento de su existencia como por la posibilidad de atraer desarrolladores.

1.3. Registro del proyecto

De entre todos los sitios comentados, emplearemos SourceForge para mostrar cómo comenzar un proyecto de software libre.

El primer paso consiste en obtener una cuenta en el sitio. En el caso de SourceForge no existe ninguna restricción para que podamos crearnos una cuenta. Para ello tenemos que pulsar sobre el enlace 'New User Via SSL' en la página principal de SourceForge. En el formulario de inscripción debemos escribir una contraseña y una dirección de correo electrónico. Tras enviar estos datos, se nos va a pedir que revisemos la dirección de correo electrónico, y si es correcta, que continuemos con el proceso de registro. Si lo hacemos, recibiremos un mensaje a esa dirección de correo electrónico para confirmar el alta del usuario. El sistema nos responderá que recibiremos un mensaje en el plazo de 24 horas. Normalmente, el mensaje suele ser instantáneo, aunque debido a los problemas de masificación de SourceForge, puede que tardemos algún tiempo en recibirlo.

El contenido del mensaje es el siguiente:

```
From: SourceForge.net <noreply@sourceforge.net>
To: alguien@algunsitio.com
Date: Wed, 12 Oct 2023 06:50:29 -0800
Subject: SourceForge.net Account Registration: Email Verification
```

This email has been generated automatically by SourceForge.net as part of your request to register a SourceForge.net user account.

The purpose of this email is to verify that the email address you provided to SourceForge.net exists and that you may read the mail sent to this address. It is important that you use a private email address which will remain in your control (not a disposable email address). SourceForge.net will use this email address in the future to notify you of account problems, or if you need to recover a lost account password.

To proceed with account registration, please access the following URL:

https://sourceforge.net/account/newuser_register.php?confirm_hash=xxxxxxx

This URL should be entered into your browser on a single line with no spaces.

By accessing this URL, you will confirm that we are able to send email to this email address (alguien@algunsitio.com).

```
After email address validation is completed, you will be permitted to select a username and provide account details. Questions or concerns about this process may be directed to http://SourceForge.net.
```

```
Questions or concerns about this process may be directed to SourceForge.net staff at:  
https://sourceforge.net/docs/C01/
```

```
We hope you enjoy SourceForge.net.
```

```
-- the SourceForge.net staff
```

Si seguimos el enlace incluido en el mensaje, deberemos confirmar de nuevo la dirección de correo electrónico y la contraseña elegida, tras lo cual podremos por fin especificar los detalles de nuestra cuenta, como el nombre del usuario o el idioma elegido (entre los que se cuentan la lengua inglesa y la castellana). Al pulsar el botón para verificar los datos, el sistema nos mostrará de nuevo el mismo formulario para comprobar que los datos son correctos. Una vez revisados, habremos obtenido por fin nuestra cuenta en SourceForge. Recibiremos un mensaje de bienvenida en la dirección que proporcionamos.

Una vez obtenida la cuenta, podemos entrar mediante el enlace 'Login Vía SSL' situado en la parte superior izquierda de la página. La primera página que vemos tras nuestra identificación (*login*) es la página personal del usuario. Desde esta página podremos acceder en un futuro a los diferentes proyectos que tengamos registrados.

El primer paso que deberemos hacer es completar nuestros datos, tal y como nos pide el mensaje que aparece en nuestra página personal:

```
You have not yet set the True Identity details for your account. Please do so at this time.
```

También es interesante completar nuestro perfil de desarrollador. Para ello, usamos la opción 'Account Options' de nuestra página personal. Después elegimos 'Skills Profile' y definimos cuáles son nuestras habilidades como desarrollador.

Ejemplo

En la figura 1-1 hemos definido que nuestro perfil sea público y hemos añadido algunas habilidades ficticias. Como se puede observar, se puede elegir la antigüedad en cada habilidad y el nivel técnico. Este perfil puede resultar útil a la hora de participar en otros proyectos en SourceForge, a modo de *curriculum vitae*.

Figura 1-1. Datos de perfil de desarrollador de SouceForge

The following option determines if others can see your resume online. If they can't, you can still enter your skills, and search for matching jobs.

Publicly Viewable:
 No
 Yes

Give us some information, either a resume, or an explanation of your experience.

Resume / Description of Experience:

[Update Profile](#)

Skill	Level	Experience	Action
Programming Language :: Python	Wrote The Book	< 6 Months	Update
Operating System :: Modern (Vendor-Supported) Desktop Operating Systems :: Linux	Competent	< 6 Months	Update
Operating System :: Modern (Vendor-Supported) Desktop Operating Systems :: FreeBSD	Want to Learn	< 6 Months	Update
Programming Language :: C++	Wizard	2 yr - 5 yr	Update

Add A New Skill

Database Environment :: Database API :: ADOdb	Want to Learn	< 6 Months	Add Skill
---	---------------	------------	-----------

Una vez que hayamos escrito nuestro perfil como desarrollador, podemos elegir entre unirnos a un proyecto ya existente, o crear un proyecto propio. El registro de proyectos no es automático, sino que requiere la validación del personal de SourceForge. Debemos explicar en qué consiste nuestro proyecto, cuáles son sus objetivos, y por qué queremos registrarlo en SourceForge.

Veamos un ejemplo. En la página personal del usuario hay un enlace para comenzar el proceso de registro de un nuevo proyecto (abajo a la derecha).

La primera página nos pide que aclaremos si sabemos en qué consiste el software libre y qué clase de proyecto vamos a registrar.

Una vez hecho esto, una nueva página nos explica qué pasos vamos a seguir en el proceso de registro. Los pasos son (entre corchetes su traducción):

1. Hosting information [Información del alojamiento]

2. Registering a project (current step) [Registro de proyecto (paso actual)]
3. Terms of use agreement [Acuerdo de términos de uso]
4. Hosting requirements [Requisitos de alojamiento]
5. Project license details [Detalles de licencia del proyecto]
6. Project description details [Detalles de la descripción del proyecto]
7. Project name details [Detalles del nombre del proyecto]
8. Final review [Revisión final]
9. Submission completed [Finalización del proceso]

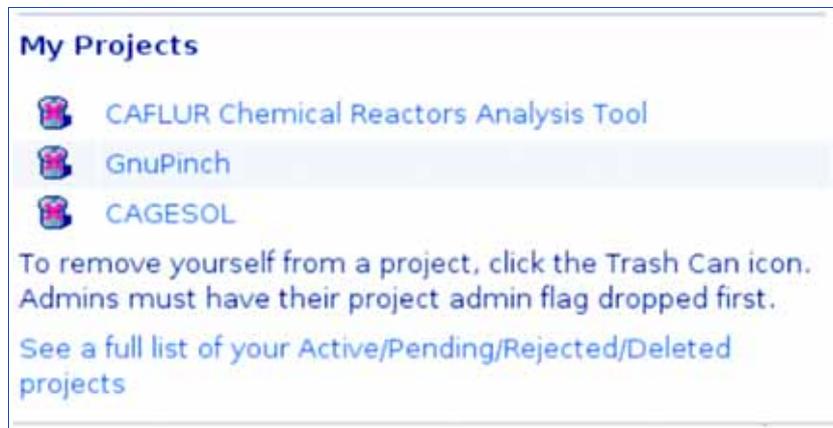
Debemos leer detenidamente la información mostrada en cada paso. Un requisito imprescindible para registrar un proyecto es que su licencia sea de software libre. Podemos elegir la licencia de entre una lista o bien especificar una licencia propia, cuyos términos estén de acuerdo con el concepto de software libre.

En la descripción del proyecto, debemos escribir en qué consiste y cuáles son sus objetivos. Una de las descripciones será la que aparezca en la página del proyecto y la otra es la que se dirige a los administradores de SourceForge para que decidan acerca del registro del proyecto. Todas las descripciones deben estar escritas en inglés.

Una vez cumplimentados todos los formularios, deberemos esperar a que los administradores de SourceForge aprueben nuestro proyecto. Esta aprobación puede tardar varios días y en general no suelen denegar proyectos a menos que se escapen totalmente de la temática del sitio.

Cuando nuestro proyecto sea aprobado, en nuestra página personal aparecerá un enlace a la página del proyecto (abajo a la derecha). Por ejemplo, en la figura siguiente vemos que el usuario tiene registrados tres proyectos.

Figura 1-2. Proyectos registrados en la página personal de un usuario de SourceForge



Al pulsar sobre el enlace de algún proyecto, nos dirigiremos a la página del proyecto seleccionado, donde podemos controlar todas las herramientas.

En la figura 1-3 vemos las diferentes opciones disponibles en la página del proyecto. Las iremos viendo todas una por una, y pondremos un énfasis especial en las herramientas de administración y desarrollo.

Figura 1-3. Página principal de un proyecto SourceForge



En la parte superior derecha, debajo de 'Developer Info', podemos ver la lista de administradores del proyecto y el número de desarrolladores que forman parte del proyecto. Inicialmente, sólo hay un administrador y un desarrollador.

En la barra de herramientas podemos observar toda la infraestructura que SourceForge pone a disposición de nuestro proyecto. Pulsando sobre cualquier herramienta, obtenemos una vista como usuario de esa herramienta. Si queremos configurar alguna herramienta, tenemos que hacerlo desde la interfaz de administración.

También podemos ver la descripción que escribimos durante el proceso de registro del proyecto. En cuanto a las donaciones y las características del proyecto, inicialmente no veremos nada. Para editar la categorización de nuestro proyecto, debemos usar el enlace que señalamos con el texto “EDITAR CATEGORÍAS”.

A continuación, vamos a examinar las herramientas más importantes con mayor detenimiento.

1.4. La cuenta de línea de comandos

SourceForge proporciona a cada usuario una cuenta de línea de comandos (*shell*) para poder efectuar cambios en la web y acceder como desarrollador al CVS. Podemos acceder a nuestra cuenta mediante SSH. Para ello usamos el cliente SSH, presente en la mayoría de los sistemas UNIX (entre ellos Linux). Dentro de un terminal de línea de comandos, tecleamos la siguiente orden:

```
ssh username@projname.sf.net
```

donde *username* es el nombre de usuario, y *projname* es el nombre del proyecto. Se nos pedirá la contraseña del usuario para poder entrar en el sistema.

Los ficheros de la web de nuestro proyecto se encuentran en el directorio `/home/groups/p/pr/projname/htdocs/`. Del mismo modo, nuestro directorio de usuario se encuentra en `/home/users/u/us/username/`, donde *projname* es el nombre del proyecto (y *p* y *pr* son la primera y las dos primeras letras del nombre del proyecto), y *username* es el nombre del usuario (y *u* y *us* son la primera y las dos primeras letras del nombre del usuario).

La misma contraseña que acabamos de escribir se nos pedirá cada vez que accedamos al CVS como desarrollador. Para evitar tener que introducir cada vez la clave, en el apartado siguiente vamos a explicar cómo instalar Keychain en nuestro ordenador.

1.4.1. Acceso mediante SSH sin clave

Keychain es una herramienta que podemos emplear para evitar teclear nuestra contraseña cada vez que accedamos al CVS o que copiemos algún fichero mediante el mandato `scp`.

En sistemas basados en Debian, Keychain se instala simplemente con la orden:

```
apt-get install keychain
```

Una vez instalado, debemos añadir las siguientes líneas al fichero `.bash_profile` presente en nuestro directorio de usuario. La primera línea ejecuta Keychain usando el par de claves generado y la segunda línea lanza el guión que crearemos cuando configuremos Keychain.

```
keychain ~/.ssh/id_rsa ~/.ssh/id_dsa
. ~/.keychain/${HOSTNAME}-sh
```

El paso siguiente es generar un par de claves, pública y privada, que se emplearán para autenticar al usuario. Para ello usamos la orden:

```
ssh-keygen -t rsa
```

Al ejecutar esta orden, se nos pide un fichero, donde se escribirá la clave privada, y una contraseña para usar como clave privada. Para el fichero dejamos la opción por defecto. En cuanto a la contraseña, muchos eligen no poner ninguna contraseña, para no tener que escribirla cuando acceden por SSH. Esto es peligroso, porque si en algún momento nuestra cuenta de usuario se ve comprometida, también lo estará la cuenta en SourceForge. Además, usando Keychain, sólo será necesario introducir nuestra contraseña una única vez por sesión.

Nota

Para otras distribuciones, pueden encontrarse paquetes en:

<http://dev.gentoo.org/~agriffis/keychain/>

La clave pública que hayamos generado estará en el fichero `.ssh/id_rsa.pub` en nuestro directorio de usuario. Copiamos su contenido, y nos dirigimos a <https://sourceforge.net/account/editsshkeys.php> (este enlace está en 'Account Options', dentro de nuestra página personal de SourceForge, al final, bajo el epígrafe 'Number of SSH Shared Keys on file'). Pegamos el contenido en el cuadro de texto y le damos al botón 'Update'.

Cuando hayamos acabado, deberemos esperar unas horas a que los cambios tomen efecto en el servidor, y ya podremos acceder a las cuentas de nuestra línea de comandos y al CVS tecleando la contraseña que usamos al generar el par de claves pública y privada.

La primera vez que abramos un terminal virtual, o una consola virtual, Keychain nos pedirá la clave privada. A partir de entonces, Keychain la introducirá automáticamente cada vez que esta clave nos sea requerida. En otras palabras, introducimos la contraseña una vez al principio y luego será reutilizada cuando Keychain la necesite.

1.4.2. Cómo se pueden copiar los ficheros de la web

Podemos usar el comando `scp` para copiar los ficheros de la web. Para copiar un solo fichero:

```
scp filename username@projname.sf.net:/home/groups/p/pr/projname/htdocs/
```

donde *filename* es el nombre del fichero.

Si queremos copiar todo un árbol de directorios y reproducirlo exactamente en el servidor, podemos usar la herramienta `rsync`:

```
rsync -v -rsh = "ssh -l username" source_path  
username@projname.sf.net:/home/groups/p/pr/projname/htdocs/
```

donde *source_path* es el nombre del directorio local. Si hemos usado Keychain, no debería pedirnos la contraseña esta segunda vez. Este mandato copia el directorio local *source_path* en el ser-

vidor, respetando toda la estructura del directorio; esto es, la copia de ficheros al servidor es recursiva.

1.5. Configuración del CVS

Si pulsamos sobre el enlace CVS, en la barra de herramientas, obtendremos la interfaz de usuario del CVS.

Nota

Como se menciona en la misma página, cualquier usuario anónimo puede acceder al código fuente de un proyecto y utilizar el CVS. En cambio, para poder modificar el código fuente almacenado en el CVS, es necesario ser un desarrollador del proyecto.

El CVS (*concurrent versions system*) es una herramienta muy popular entre los proyectos de software libre. Se emplea para gestionar los cambios al código fuente del proyecto, de modo que varios desarrolladores puedan trabajar de manera coordinada sobre el mismo código fuente.

En los apartados siguientes se mostrará simplemente cómo se puede acceder al código alojado en un repositorio CVS, de manera anónima o como desarrollador de un proyecto. Más adelante, en un apartado íntegramente dedicado a la gestión y uso del CVS, veremos con más detalle cómo se utiliza esta herramienta.⁴

1.5.1. Acceso anónimo

Para acceder de manera anónima al servidor CVS, vamos a usar el cliente CVS, presente en la mayoría de sistemas UNIX (incluido Linux). Para ello, tecleamos lo siguiente en un terminal de comandos:

```
cvs -d:pserver:anonymous@cvs.sf.net:/cvsroot/projname login
Login como usuario anónimo
cvs -z3 -d:pserver:anonymous@cvs.sf.net:/cvsroot/projname co modulename
Descarga del módulo
```

donde *projname* es el nombre del proyecto, y *modulename* es el nombre del módulo.

Podemos averiguar fácilmente qué módulos hay en el CVS pulsando sobre el enlace 'Browse CVS Repository' de la interfaz web. En la figura 1-4 se muestra un ejemplo de la interfaz web de un proyecto.

Figura 1-4. Interfaz web de un proyecto



En este caso hay dos módulos: PinchPython y gnupinch.

1.5.2. Acceso como desarrollador

El acceso como desarrollador nos permite realizar cambios en el repositorio de código fuente. Lo primero de todo, tenemos que decirle al cliente CVS que emplee SSH para comunicarse con el servidor. Esto lo hacemos mediante la orden:

```
export CVS_RSH = ssh
```

Otra variable de entorno útil que podemos definir es CVSROOT, para no tener que indicar el repositorio mediante la opción `-d` del comando `cvs`.

```
export CVSROOT = :ext:username@cvs.sf.net:/cvsroot/projname
```

También podemos añadir las dos líneas anteriores al `.bash_profile` para evitar tener que escribirlo cada vez que vayamos a acceder al servidor CVS. Tras esto, ejecutamos la orden:

```
cvs -z3 co modulename
```

1.6. Descargas del proyecto

Otra opción de SourceForge que utilizaremos con frecuencia es la descarga. Desde luego, siempre podremos optar por poner los ficheros en algún directorio de nuestra web y que se puedan descargar por HTTP. Sin embargo, si empleamos la herramienta de descargas de SourceForge, podremos obtener estadísticas de descargas, un historial de publicaciones (*releases*), y además, los usuarios podrán hacer uso de todas las réplicas (*mirrors*) de SourceForge para descargar los ficheros (las webs y el CVS no están replicados, sólo los ficheros de las publicaciones) lo que generalmente agiliza la descarga.

La herramienta de descargas se administra desde la interfaz de administración (véase la figura 1-3). Dentro de la interfaz de administración, tenemos acceso a una barra con las diferentes herramientas de administración (figura 1-5).

Figura 1-5. Barra de herramientas de administración de un proyecto SourceForge

Admin | Members | Recruitment | Public Info | Backups | Audit | Publicity | Registration | Removal | Donations | **File Release** | Tracker | Tasks | DocManager | Screenshots | Lists/Forums | Shell/DB/Web | CVS | Stats

En este momento, nos centraremos en la herramienta 'File Releases'. Si seguimos este enlace, llegamos a una página que nos informa acerca de esta herramienta. Al final de la página, tenemos un formulario para crear paquetes nuevos (figura 1-6).

Figura 1-6. Interfaz de creación de paquetes nuevos

File Release Packages

Releases	Package Name	Status	Update
[Add Release] [Edit Releases]	<input type="text"/>	Active ▾	<input type="button" value="Update"/>
[Add Release] [Edit Releases]	<input type="text"/>	Active ▾	<input type="button" value="Update"/>
[Add Release] [Edit Releases]	<input type="text"/>	Hidden ▾	<input type="button" value="Update"/>
[Add Release] [Edit Releases]	<input type="text"/>	Active ▾	<input type="button" value="Update"/>

New Package Name:

En un primer momento, no veremos ningún paquete ya creado.

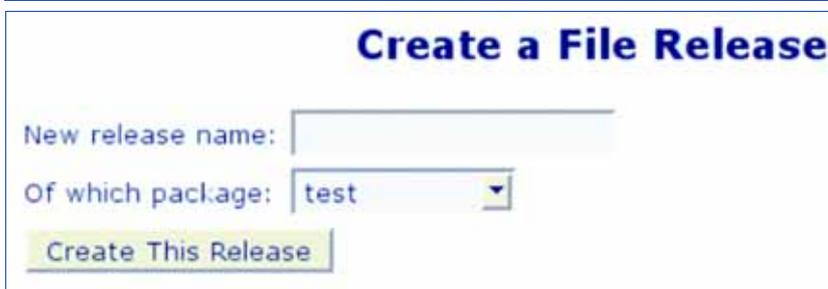
Las descargas se organizan en paquetes y versiones. Los **paquetes** pueden entenderse como subproyectos dentro del proyecto y cada paquete puede contener varias versiones.

Lo primero que hay que hacer para añadir ficheros para descargar es crear un paquete. Escribimos el nombre del paquete y le damos al botón 'Create This Package'.

Una vez creado el paquete, podemos ocultarlo o activarlo. No se pueden borrar paquetes una vez creados, así que en lugar de borrar un paquete, lo ocultaremos.

Para añadir versiones a un paquete, usamos el enlace 'Add Release'. Elegimos el nombre (o el número de la versión), el paquete del que forma parte, y creamos esa versión.

Figura 1-7. Interfaz de creación de versiones



Tras crear la versión, llegaremos a un formulario que consta de tres pasos:

1. En el primer paso debemos rellenar las notas de la versión y el registro de cambios. Las notas de la versión consisten en un resumen de las funcionalidades nuevas y errores corregidos, mientras que el registro de cambios es la bitácora detallada de cambios que se han efectuado. Por otra parte, en los proyectos de software libre existe la sana costumbre de contar con un fichero denominado *ChangeLog* donde se introduce, tras cada cambio efectuado, la fecha del cambio, el nombre del desarrollador (y su correo electrónico), el fichero modificado y una breve descripción de lo

Ejemplo

En nuestro proyecto podemos tener dos paquetes: *programa* y *documentación*. Y en el paquete *programa*, la versión 1.0, la 2.0, etc.

que se ha modificado. En ese caso, el contenido de ese fichero se puede copiar en el campo destinado a ese propósito.

Figura 1-8. Interfaz de registro de cambios de una versión

Step 1: Edit Existing Release

Release Date:

Release Name:

Status:

Of Package:

Edit the Release Notes or Change Log for this release of this package. These changes will apply to all files attached to this release. You can either upload the release notes and change log individually, or paste them in together below. Release Notes and Change Log content must be between 20 and 256000 bytes in length.

Upload Release Notes:

Upload Change Log:

Paste The Notes In:

Paste The Change Log In:

Preserve my pre-formatted text.

2. El segundo paso consiste en elegir los ficheros que forman parte de la versión. Debemos elegir los ficheros de nuestra versión de entre los que aparecen en la lista (figura 1-9). Los ficheros que aparecen en esta lista son los que están en el directorio `incoming` de la máquina `upload.sf.net`. Podemos poner cualquier fichero en ese directorio accediendo por FTP anónimo. Los nombres de los ficheros no pueden contener espacios, paréntesis ni el carácter `'~'`.

Es importante elegir sólo nuestros ficheros, puesto que una vez que se seleccionan para formar parte de una versión, ya no aparecen más en la lista (y por tanto, su dueño no podrá usarlo para su versión, y tendrá que volver a subirlo). Dentro de una versión, podemos incluir tantos ficheros como queramos.

Figura 1-9. Interfaz de selección de ficheros de una versión

Step 2: Add Files To This Release

Next, choose your files from the list below. Choose **ONLY YOUR** files. If you choose someone else's files, they will not be able to access them and they will be rightfully upset.

You can upload new files using **anonymous** FTP to **upload.sourceforge.net** in the **incoming** directory. When you are done uploading, just hit the refresh button to see the new files. Further information regarding this process may be found in our [Guide to the File Release System](#); please refer to this document if you encounter any difficulties in performing this file release.

PLEASE NOTE: filenames may not contain a space, parenthesis or the "~" character..

- FreeDolo_0.6.7b.hpux.tgz
- GroundTexture-128x128.tar.gz
- cc_v1_3.tgz
- cc_v1_4.tgz
- epc-4.4.0.tar.gz
- imc-1.2-11.rhel3.i386.rpm
- libcli-1.0.4.tar.gz
- motioncleanup.tar.gz
- motionwatchdog.tar.gz
- mygosuclan-patch-sql.zip
- scim-1.1.0.tar.gz
- scm-tables-0.5.0.tar.gz
- webinject-1.30.src.tar.gz
- webinject-1.30.win32.zip

[Add Files and/or Refresh View](#)

- El tercer paso consiste en clasificar cada fichero de la versión (figura 1-10). Debemos indicar para qué arquitectura es el fichero (o si es independiente de arquitectura; sólo los ficheros binarios deberían ser dependientes de la arquitectura) y de qué tipo de fichero se trata (código fuente, documentación, etc). También podemos borrar un fichero de una versión.

Figura 1-10. Interfaz de clasificación de los ficheros de una versión

Step 3: Edit Files In This Release

Once you have added files to this release you **must** update each of these files with the correct information or they will not appear on your download summary page.

Filename Release	Processor Release Date	File Type Update
Chronus.jar test : 1.0	Must Choose One 2005-01-05	Must Choose One Update/Refresh Delete File <input type="checkbox"/> I'm Sure

Con esto ya habremos definido una versión dentro de un paquete. Ahora, el paquete debería aparecer en la lista de descargas (figura 1-11), y al pulsar sobre el paquete, la versión que acabamos de crear.

Figura 1-11. Lista de descargas

Summary | Admin | Home Page | Forums | Tracker | Bugs | Support | Patches | RFE | Lists | Tasks | Docs | Screenshots | News | CVS | **Files** | Donations |

Below is a list of the files for the selected file package. Other views: [all files released by this project] Before downloading, you may want to read Release Notes and ChangeLog (accessible by clicking on release version).

Package	Release & Notes	Filename	Size	Date D/L	Arch.	Type
test						
 1.0 [show only this release]				2005-01-05 19:16		
Chronus.jar			1384804	0 None	None	
Project Totals:	<i>1</i>	<i>1</i>	<i>1384804</i>	<i>0</i> 		

Generalmente, una vez que hayamos publicado una nueva versión de nuestro software, deberíamos anunciarlo convenientemente para que su difusión sea la máxima. Para tal fin, es una buena idea hacer uso de las listas de correo electrónico que ofrece SourceForge y cuya administración se presenta en el siguiente apartado.

1.7. Listas de correo

SourceForge pone a disposición de los proyectos la posibilidad de crear y gestionar listas de correo. Para este propósito emplea gestor de listas de correo GNU Mailman, aunque la creación y gestión de las listas de correo se efectúa desde la interfaz de SourceForge.

Las listas de correo se pueden administrar con la barra de herramientas que muestra la figura 1-12.

Figura 1-12. Herramienta de administración de las listas de correo

Admin | Members | Recruitment | Public Info | Backups | Audit | Publicity | Registration | Removal | Donations | File Releases | Tracker | Tasks | DocManager | Screenshots | **Lists/Forum** | Shell/DB/Web | CVS | Stats

Una vez en la página de administración, podemos elegir la opción de habilitar o deshabilitar las listas de correo y los foros en la página del proyecto (figura 1-13).

Figura 1-13. Herramienta de administración de las listas de correo y los foros de discusión

Mailing Lists and Discussion Forums

When developing software, communication is vital. To aid in your efforts to communicate within your development team and for your developers to effectively communicate with your end-users, SourceForge.net provides projects the ability to establish mailing lists (email-based) and discussion forums (web-based). For management of support issues, bugs, patches and feature requests, we also provide the Tracker system.

Enable Mailing Lists? The following box should be checked to enable Mailing Lists:

Enable Discussion Forums? The following box should be checked to enable Discussion Forums:

[Update](#)

Mailing List Admin:

Mailing Lists are managed in two places. The creation of mailing lists is handled from the Mailing List Admin page on the SourceForge.net site. Mailing lists may be created by Project Admins. Once created, [mailing lists cannot be removed](#) (flagging the list as deleted merely removes it from the listing of available mailing lists, but does not shut down the mailing list -- contact the SourceForge.net team by submitting a Support Request for further assistance).

Once a list has been created, it is configured using the Mailman mailing list management software (which SourceForge.net uses to provide its mailing list service). Access to manage a mailing list via Mailman is provided using a separate admin password for each list -- this password does not match your SourceForge.net user password; there is only one admin password for each list, regardless of how many people administrate that list.

At time of list creation, a default password is sent to the list creator via email. If this password is lost, it may be reset via the [Administer/Update Lists](#) page for your project.

- [Manage Mailing Lists](#)
- [Links to the Mailman list administration interface](#) for each list may be found

Para la administración de las listas, nos dirigiremos al enlace 'Manage Mailing Lists'. Una vez allí, podemos añadir nuevas listas, o gestionar las ya existentes (figura 1-14).

Figura 1-14. Creación y gestión de listas de correo

Admin

[Add Mailing List](#)

[Administer/Update Lists](#)

Si elegimos añadir una nueva lista, debemos cumplimentar el formulario que se muestra en la figura 1-15.

Figura 1-15. Interfaz de creación de una nueva lista de correo

Admin

Lists are named in this manner:
projectname-listname@lists.sourceforge.net

It will take **6-24 Hours** for your list to be created.

Existing Mailing Lists

list_name
 project-privat@lists.sourceforge.net
 project-privat@lists.sourceforge.net

Mailing List Name:
 project-privat@lists.sourceforge.net

Is Public?
 Yes
 No

Description:

Once created, this list will ALWAYS be attached to your project and cannot be deleted!

Add This List

Regarding the "Is Public?" flag: Public mailing lists are shown when a user goes to the Mailing Lists page for your project, and have mailing list archives that are accessible to the public. The ability to subscribe and post to a list is controlled in the Mailman interface for your list, not via this flag. Lists flagged as deleted will not appear in the listing of mailing lists for your project, but will still be present within Mailman.

Nota

Pueden pasar algunas horas hasta que la lista se cree definitivamente.

Las listas no se crean inmediatamente, sino que la creación de listas es una tarea de `cron`. Las tareas de `cron` se ejecutan periódicamente según cómo las haya configurado el administrador del sitio.

Debemos tener en cuenta que las listas no se pueden borrar. Tan sólo podremos marcar una lista como en desuso y no se podrán consultar sus archivos (si éstos son públicos, como se explica a continuación).



Las listas pueden ser públicas o privadas. Las **listas privadas** sólo permiten acceder al archivo de la lista a los usuarios suscritos a esa lista.

Posteriormente, desde la interfaz de Mailman podremos configurar las opciones de suscripción para que cualquier usuario pueda registrarse o que se requiera la aprobación del administrador de la lista.

Cuando la lista de correo ya ha sido creada, podemos cambiar algunas opciones de configuración (figura 1-16).

Figura 1-16. Opciones de configuración de una lista de correo

Admin

You can administrate lists from here. Please note that private lists can still be viewed by members of your project, but are not listed on SourceForge.net.

List	Status	Update	List Admin
example@project	Is Public? <input type="radio"/> Yes <input checked="" type="radio"/> No <input type="radio"/> Deleted	<input type="button" value="Update"/>	[Administer this list in GNU Mailman] [Search/Display subscriber list] [Change list admin password]
example@project	Is Public? <input checked="" type="radio"/> Yes <input type="radio"/> No <input type="radio"/> Deleted	<input type="button" value="Update"/>	[Administer this list in GNU Mailman] [Search/Display subscriber list] [Change list admin password]

Description:

Lista de correo para los miembros institucionales

Description:

Lista privada de correo, sin suscripción

Regarding the "Is Public?" flag: Public mailing lists are shown when a user goes to the Mailing Lists page for your project, and have mailing list archives that are accessible to the public. The ability to subscribe and post to a list is controlled in the Mailman interface for your list, not via this flag. Lists flagged as deleted will not appear in the listing of mailing lists for your project, but will still be present within Mailman.

Desde esta página también podemos acceder a la interfaz de Mailman para configurar al detalle las opciones de la lista de correo.

Nota

En el capítulo 4 hay una sección dedicada a la gestión avanzada de Mailman.

1.8. Tracker y el sistema de seguimiento de fallos

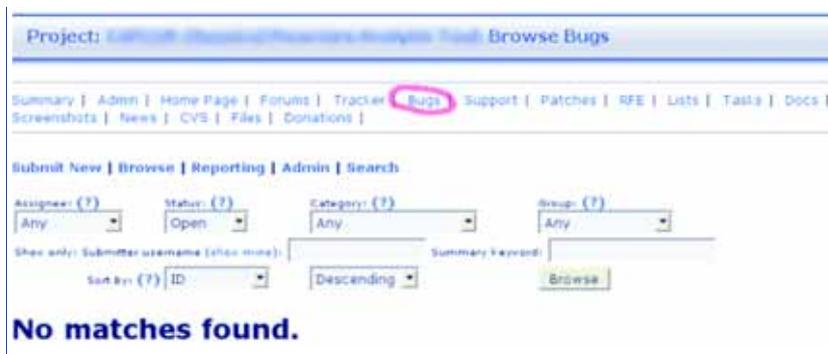
El sistema de seguimiento de fallos de SourceForge se denomina Tracker, y no está ideado sólo para la gestión de fallos. Las posibilidades que contempla son (traducción entre corchetes):

- Support requests [Peticiónes de soporte]
- Bug reports [Informes de fallos]

- Feature requests [Peticións de funcionalidad]
- Patches [Parches]

El funcionamiento para los tres tipos de informes ('support requests', 'Bug reports' y 'Feature requests') es similar. Nosotros nos centraremos en el caso de los fallos. A la interfaz para el usuario se accede desde el enlace 'Bugs' de la lista de herramientas de la página principal del proyecto (figura 1-17).

Figura 1-17. Acceso a la opción 'Bug reports'



Project: [unreadable] Browse Bugs

Summary | Admin | Home Page | Forums | Tracker | **Bugs** | Support | Patches | RFE | Lists | Tasks | Docs | Screenshots | News | CVS | Files | Donations |

Submit New | Browse | Reporting | Admin | Search

Assignee: (?) Any | Status: (?) Open | Category: (?) Any | Group: (?) Any

Show only: Submitter username (click more): | Summary Keyword: |

Sort by: (?) ID | Descending | Browse

No matches found.

Si hemos entrado con nuestra cuenta de usuario, veremos el enlace 'Admin' en la barra de enlaces de la herramienta. Si pulsamos en ese enlace, podemos gestionar el Tracker para todas las categorías. Todas las interfaces son iguales, ya que sólo cambia el propósito de cada Tracker. Por ejemplo, en el caso de fallos, podemos gestionar las opciones que aparecen en la figura 1-18.

Figura 1-18. Interfaz 'Bugs'



Submit New | Browse | Reporting | Admin | Search

Admin functions: Add/Browse Artifact Types | Edit/Update Options in: Bugs

Add/Update Categories
Add categories like, 'mail module', 'gant chart module', 'cvs', etc

Add/Update Groups
Add groups like, 'v1.2', 'unsupported', 'unverified', etc

Add/Update Canned Responses
Create/Change generic response messages for the tracker.

Add/Update Users & Permissions
Add/remove users to/from this tracker.

Mass Re-assign Tracker Items
Re-assign Tracker items from a past/present technician to a present technician.

Update preferences
Set up prefs like expiration times, email addresses, etc.

Las opciones son:

- Añadir y actualizar categorías. Por ejemplo, dentro de un sistema operativo, los fallos del kernel y los fallos de la línea de comandos.
- Añadir y actualizar grupos de fallos. Por ejemplo, grupos de fallos por versiones del software).
- Añadir y actualizar respuestas automáticas a los informes de fallos.
- Añadir y actualizar usuarios y permisos (por ejemplo, dotar a algunos usuarios de permisos para modificar los informes de fallos)
- Asignar un grupo de fallos de un desarrollador a otro.
- Actualizar otras preferencias (visibilidad del Tracker, periodos de tiempo para cambiar un fallo a obsoleto, etc.).

Una vez configurado el Tracker, los usuarios y desarrolladores pueden mandar fallos al Tracker usando el enlace 'Submit New' de la interfaz de la herramienta (figura 1-19).

Figura 1-19. Interfaz 'Submit New'

Submit New | Browse | Reporting | Admin | Search

For Project:
Category: (?)

Category: (?)
None (admin)

Group: (?)
None (admin)

Assigned To: (?)
None (admin)

Priority: (?)
5 - Medium

Summary: (?)

Detailed Description:

DO NOT enter passwords or other confidential information!

Check to Upload and Attach a File: (?)

Browse...

File Description:

SUBMIT

El usuario puede asignar un fallo a un desarrollador. Cuando creamos un fallo, aparece automáticamente en el Tracker (figura 1-20).

Figura 1-20. Asignación de un fallo a un desarrollador en el Tracker

Submit New | Browse | Reporting | Admin | Search

Assignee: (?) Any | Status: (?) Open | Category: (?) Any | Group: (?) Any

Show only: Submitter username (show mine): | Summary keyword: |

Sort By: (?) ID | Descending | Browse

Request ID	Summary	Open Date	Priority	Assigned To	Submitted By
<input type="checkbox"/> 1096834	Test	2005-01-05 23:46	5	nobody	Remove

Check All - Clear All

Posteriormente, un administrador del proyecto o un desarrollador que tenga permisos suficientes puede cambiar el desarrollador asignado a un fallo (figura 1-21).

Figura 1-21. Cambio del desarrollador asignado a un fallo

[1096834] Test

Monitor (?)

Submitted By: [David Martínez - Normal](#)

Last Updated By: Item Submitter - Tracker Item Submitted

Number of Comments: 0

Data Type: (?) Bugs

Category: (?) None (admin)

Assigned To: (?) None (admin)

Status: (?) Open

Summary: (?) Test

test

Use Canned Response: (?) None (admin)

Date Submitted: 2005-01-05 23:46

Date Last Updated: No updates since submission

Number of Attachments: 0

Submit Changes

Group: (?) None (admin)

Priority: (?) 5 - Medium

Resolution: (?) None

Cuando se ha asignado un fallo a un desarrollador, de manera automática recibirá un correo electrónico y el fallo aparecerá en su página principal (figura 1-22).

Figura 1-22. Anuncio de fallo en la página principal del desarrollador del proyecto SourceForce



Del mismo modo, el usuario que notificó el fallo también recibe un correo electrónico con cada novedad que se va produciendo y puede ver el fallo en su página personal de SourceForge (figura 1-23).

Figura 1-23. Anuncio de fallo en la página principal del usuario



Cuando un desarrollador resuelve un fallo, deberá modificar el estado del fallo a *Cerrado*, y el fallo quedará almacenado como tal en SourceForge para consultas posteriores. Tanto el desarrollador como el usuario que notificó el fallo son informados mediante correos electrónicos del cambio de estado en el fallo.

2. Control de versiones

Cuando uno o varios programadores desarrollan un mismo software, normalmente desean mantener las diferentes versiones de los ficheros que desarrollan. Además, con frecuencia, llegados a un punto, se suelen producir bifurcaciones en el desarrollo, y diferentes programadores comienzan a desarrollar funcionalidades distintas de forma independiente sobre un mismo fichero.

En un momento dado, el responsable de ese fichero puede decidir mezclar los cambios hechos en las diferentes ramas y crear ficheros nuevos que reúnan el trabajo de todos los desarrolladores.



Las herramientas de control de versiones facilitan este trabajo y proporcionan, como mínimo, mecanismos para:

- a) almacenar versiones distintas de un mismo fichero, para que cualquiera de ellas pueda recuperarse en cualquier momento.
- b) definir diferentes ramas de desarrollo, para que los programadores puedan generar versiones nuevas independientes en cada rama.
- c) automatizar la “fusión” (en inglés, *merge*) de ficheros de versiones o ramas diferentes. De este modo, la integración del trabajo de cada programador sobre el mismo fichero será, en la medida de lo posible, automática.

Por otro lado, unos sistemas de control de versiones serán más interesantes que otros en función de sus posibilidades.

Ejemplo

En CVS [Fogel00, Cederqvist04, Vesperman03], el sistema de control de versiones más empleado, no es posible mantener diferentes versiones de los directorios. En cambio, en Subversion (cuya abreviatura es SVN, y que se presenta como el sistema de control de versiones de siguiente generación), sí que se puede.

La herramienta de control de versiones también será muy útil a efectos de la publicación de proyectos libres: con frecuencia se crea el usuario anónimo, sin contraseña, que permite acceder a los repositorios de software, aunque no permite escribir en ellos, tal y como ya hemos señalado en el capítulo dedicado a las herramientas que SourceForge provee.

De este modo, los usuarios interesados pueden obtener las últimas versiones del software en el mismo momento en que los programadores las envían al control de versiones y probar así las últimas características implementadas. Este hecho hará posible que muchos interesados puedan probar la última versión de nuestro software e incluso mandarnos un parche que corrija algún error o añada alguna funcionalidad. En cualquier caso, ofrecer la posibilidad de descarga anónima del CVS es una práctica recomendable a menos que contemos con contenidos privados.

En los apartados siguientes vamos a ver cómo instalar y configurar un repositorio CVS propio y a continuación vamos a enseñar los usos básicos y avanzados del CVS. Tener un repositorio CVS propio puede resultar interesante para muchas cosas: desde efectuar copias de respaldo de nuestros archivos hasta, por qué no, alojar nuestros proyectos de software libre.

Para hacer un uso acertado del CVS, es preciso tener en cuenta un detalle: el versionado sólo funciona con ficheros de texto, ya que se basa en herramientas que comprueban la diferencia entre dos archivos de texto como `diff`. Los ficheros binarios, por tanto, no disfrutarán de las ventajas que ofrecen estas herramientas y se guardarán siempre íntegramente.

2.1. Instalación y configuración inicial de CVS

En muchas distribuciones de GNU/Linux encontraremos CVS empaquetado y con instalación y configuración casi automática, que nos permite despreocuparnos de los detalles. A continuación mostraremos cómo configurar a mano un repositorio CVS que reúna las siguientes condiciones, de uso recomendable:

1. Todos los desarrolladores dispondrán de una cuenta de acceso al servidor CVS.
2. El acceso se llevará a cabo mediante conexiones cifradas, para evitar el envío en claro de las contraseñas.
3. Se habilitará un mecanismo de acceso anónimo para que los usuarios puedan acceder a los repositorios y obtener las versiones más recientes, sin que puedan modificar nada.

Cada desarrollador tendrá que tener una cuenta UNIX en el servidor CVS. Definiremos un grupo UNIX al que van a pertenecer todos los desarrolladores y otorgaremos a ese grupo permiso de escritura en el repositorio. Asumiremos que este grupo existe y se llama `src`.

Por otro lado, para habilitar acceso anónimo, debemos habilitar el protocolo `pserver` de CVS, sin contraseña y con acceso de lectura únicamente.

2.1.1. Creación del repositorio

Veamos paso a paso las instrucciones para realizar las tareas mencionadas. Para crear el repositorio, se han de ejecutar las siguientes órdenes:

```
umask 002

mkdir /var/lib/cvs

chgrp src /var/lib/cvs

chmod 3775 /var/lib/cvs

cvs -d /var/lib/cvs init
```

Estos mandatos han servido para crear el repositorio y obligan mediante el `sticky-bit` de ficheros de UNIX a que todos los nuevos objetos creados dentro del directorio principal también pertenezcan al grupo `src`. Por otro lado, también se garantizan los permisos precisos para que los usuarios de ese grupo puedan escribir siempre. Por último, el mandato `init` de CVS creará una serie de ficheros que conformarán el estado inicial del repositorio. Estos ficheros se pueden modificar para un uso avanzado de CVS que excede los objetivos de este documento.

2.1.2. Preparación del acceso anónimo

El acceso anónimo al repositorio se asigna por el protocolo `pserver`, que permite a cualquier usuario con un cliente CVS obtener el repositorio desde cualquier lugar.

En primer lugar, crearemos el fichero `/var/lib/cvs/passwd` con el siguiente contenido:

```
anoncvs::cvsanonimo
```

Asimismo, crearemos el fichero `/var/lib/cvs/readers` con el contenido:

```
anoncvs
```

Ambos ficheros no deberán permitir la escritura a nadie, por razones de seguridad. Para ello, ejecutaremos la siguiente orden:

```
chmod 0444 /var/lib/cvs/passwd /var/lib/cvs/readers
```

El primer fichero, `passwd`, establece una cuenta para acceso `pserver`, llamado `anoncvs`, sin contraseña. Además, cuando ese usuario acceda, se utilizará el usuario UNIX `cvsanonimo`, que habremos de crear, pero dejando bloqueado su acceso al sistema, pues no será necesario para ninguna otra función:

```
useradd -g src -d /tmp -s /bin/true cvsanonimo
```

```
passwd -l cvsanonimo
```

La función del fichero `readers` es indicar a CVS que ese usuario sólo tendrá acceso de lectura al repositorio.

Por último, es necesario activar el protocolo `pserver`. Para ello, debemos arrancarlo como un servicio más del superservidor `inetd`. Editaremos el fichero `/etc/inetd.conf` y añadiremos la línea:

```
cvspserver stream tcp nowait root /usr/sbin/tcpd /usr/sbin/cvs-pserver
```

2.1.3. Apertura de cuentas para los desarrolladores

Todos los desarrolladores que quieran escribir en nuestro CVS deberán tener una cuenta UNIX normal, y pertenecer, al menos, al grupo `src`. Por ejemplo, podríamos crear las cuentas con una orden similar a la siguiente:

```
useradd -g developer -G src -d /home/programador1 -m programador1
```

Dado que el método de acceso será por SSH, es necesario que el usuario tenga cuenta de línea de comandos completa. Una vez creada, le asignaremos una contraseña inicial, que el desarrollador puede cambiar más tarde al entrar en el sistema:

```
passwd programador1
```

2.2. Operativa básica del CVS

Tras instalar y configurar nuestro repositorio CVS, vamos a ver cómo se utiliza el CVS como cliente. Veremos que el repositorio permite un conjunto muy amplio de interacciones, aunque el que se usa con frecuencia es ciertamente limitado, ya que se reduce a acceder al repositorio, obtener el código de un proyecto, sincronizar el repositorio con nuestras modificaciones y sincronizar nuestra copia local con el repositorio.

2.2.1. Acceso anónimo a un repositorio

Para acceder a un repositorio CVS de manera anónima, se utilizarán los mandatos CVS correspondientes, aunque previamente habrá que entrar

como anónimo. Un usuario que, por ejemplo, desee obtener de nuestro sistema el repositorio `proyecto1`, escribiría la orden siguiente:

```
export CVSROOT=:pserver:anoncvs@nombre-maquina.dominio.com:/var/lib/cvs
cvs login
Password: (pulsará INTRO dejándola en blanco)
cvs co proyecto1
```

Podrá hacer cualquiera de las demás operaciones CVS, siempre que no supongan modificar el repositorio.

2.2.2. Acceso al CVS por el desarrollador

Los desarrolladores acceden al CVS mediante un túnel cifrado SSH, pero no por ello resulta complicado. Se hace de forma similar al acceso que hemos visto para SourceForge; es decir, primero se lanza la orden:

```
export CVS_RSH=ssh
```

También podemos añadirlo al `.bash_profile` para no tener que escribirlo cada vez que vayamos a acceder al servidor CVS. Tras esto, ejecutamos, para acceder al `proyecto1` de nuestro repositorio, la sentencia siguiente:

```
cvs -z3 -d :ext:usuario@nombre-maquina.dominio.com:/var/lib/cvs co proyecto1
```

2.2.3. Creación de un proyecto en el repositorio

Cada proyecto que quiera mantenerse en el repositorio debe enviarse por primera vez al mismo. Para ello, un desarrollador cualquiera con derechos de escritura se situará en el directorio que contenga los ficheros que quiera enviar inicialmente (o un directorio vacío) y lanzará la orden:

```
cvs import proyecto1 company inicio
```

Esto creará el `proyecto1` en nuestro repositorio. Además, se anota el nombre de nuestra empresa (`company`) y una etiqueta que se añadirá a los ficheros que hayamos enviado en esta operación (si el directorio está vacío, la etiqueta no se usará).

Todas las operaciones de escritura en un repositorio requieren incluir un comentario que detalle cualquier cosa relevante. Para ello, se nos abrirá el editor estándar (normalmente, el `vi` de UNIX). Al salvar ese fichero de detalle se producirá la escritura efectiva en el repositorio.

2.2.4. Obtención del proyecto

Una vez creado el proyecto, cada desarrollador –incluidos los usuarios anónimos– puede obtener su copia local de trabajo mediante la orden:

```
cvs co proyecto1
```

Esta operación de obtención de ficheros del repositorio se denomina *check-out*. Tras el *check-out*, aparecerá un directorio especial junto a los ficheros de trabajo llamado CVS, que la aplicación va a utilizar y cuyo contenido podemos ignorar por ahora. Obsérvese que no basta con crear un proyecto en el repositorio, sino que además, tal y como hemos hecho en el punto anterior, habrá que obtenerlo del mismo. Lo que nos descargamos del repositorio se llama *copia local*.

2.2.5. Creación de ficheros y directorios

Desde dentro del directorio de trabajo del proyecto, podemos añadir cualquier fichero nuevo al repositorio del proyecto mediante la orden:

```
cvs add fichero(s)
```

Esto también puede hacerse con los directorios que queramos crear.

Cuando se quieren guardar ficheros binarios en el repositorio, es necesario añadir la siguiente opción al mandato:

```
cvs add -kb fichero(s)
```

En este caso, no disfrutaremos de algunas características de CVS (como la mezcla automática de versiones), pues sólo son posibles con ficheros de texto.

La operación `add` solo marca los ficheros para ser enviados, pero no los envía. El envío se lleva a cabo con una operación de *check-in*. Sin parámetros, se efectuará un *check-in* recursivo de todos los ficheros y directorios marcados para ser enviados al repositorio:

```
cvs ci
```

Como todas las operaciones de escritura, nos solicitará que expliquemos los detalles de los ficheros que enviamos.

2.2.6. Modificación de los ficheros. Fusión de cambios

Al terminar de preparar nuestras modificaciones a un fichero de nuestra área de trabajo, podemos enviarlas al repositorio mediante la operación de *check-in*:

```
cvs ci fichero(s)
```

En otros sistemas de control de versiones, la obtención de ficheros en el área de trabajo (*check-out*) bloquea las modificaciones a otros usuarios. En cambio, en CVS no es así. Por lo tanto, es probable que al efectuar nuestro *check-in*, el sistema detecte que otro desarrollador también ha modificado el fichero. En este caso nos avisará de la situación:

```
$ cvs ci main.c
cvs commit: Examining
cvs commit: Up-to-date check failed for 'main.c'
cvs [commit aborted]: correct above errors first!
```

Aquí, será necesario hacer una fusión de nuestro trabajo con las modificaciones efectuadas por otros desarrolladores. Para ello utilizaremos el mandato:

```
cvs update
```

Al actualizar (*update*), pueden ocurrir dos cosas:

1. Que CVS detecte que las modificaciones son en secciones distintas del fichero. En ese caso, CVS es capaz de "fusionar" automáticamente nuestros cambios con los de los demás desarrolladores.
2. Que CVS no sea capaz de fusionar los cambios automáticamente. En este caso, se produce un conflicto.

CVS nos avisará de la situación de conflicto al efectuar la actualización:

```
$ cvs update
cvs update: Updating
RCS file: /home/devell1/cvsrepos/proyecto1/main.c,v
retrieving revision 1.1
retrieving revision 1.2
Merging differences between 1.1 and 1.2 into main.c
rcsmerge: warning: conflicts during merge
cvs update: conflicts found in main.c
C main.c
```

Los conflictos figurarán en nuestra copia del fichero, de modo que podamos editarlo y resolverlos a mano.

Ejemplo

Nuestro fichero podría tener el siguiente conflicto:

```
#include <stdlib.h>

void main()
{
<<<<<< main.c
  puts("Hello world!"); puts("Good bye!");
=====
  puts("Hello world!");
>>>>>> 1.2
}
```

Nuestras modificaciones son el trozo entre '`<<<<<< main.c`' y '`=====`', que entran en conflicto con las que vienen en el CVS, en nuestro caso, concretamente de la revisión 1.2.

Corregiremos el conflicto sobre el fichero anterior. Una vez conformes con el resultado, mediante una operación *check-in* normal sincronizaremos nuestra versión local y la del repositorio.

Aunque las operaciones de fusión automáticas no suelen fallar, cuando el sistema nos avise de ello, conviene revisar el fichero por si se hubieran introducido mal.



En general, es buena práctica efectuar una operación de actualización antes de editar el fichero por si otro desarrollador ha introducido los cambios. Pero cuando trabajen varios desarrolladores en paralelo, es prácticamente imposible evitar la necesidad de un *merge*.

2.2.7. Eliminación de ficheros

Es posible eliminar ficheros del repositorio. Para ello utilizaremos la orden:

```
cv$ rm fichero(s)
```

Seguida del correspondiente *check-in*.

Si los ficheros no se han borrado aún de nuestro espacio de trabajo, será necesario añadir el modificador `-f` a la orden.

Es preciso señalar que los ficheros no se borran realmente del repositorio, sino que son movidos a un directorio especial, llamado `Attic`. Ello nos permitirá recuperar cualesquiera de las versiones de los ficheros borrados en el futuro, si así lo deseamos.

2.3. Operativa avanzada en el CVS

Con las operaciones indicadas en la sección anterior, se puede trabajar con un control de versiones básico, sin preocuparnos de temas

avanzados como etiquetas o ramas. Aunque no es objetivo de este documento profundizar en estos temas, plantearemos una introducción a la gestión de etiquetas y ramas en CVS.

2.3.1. Versiones

Cada vez que efectuamos un *check-in* o un *commit* en el CVS, se asigna un número de versión al fichero. Podemos ver el número de versión en el código fuente del fichero si incluimos en el mismo la macro: `Id`. Al incluir esta macro como comentario del lenguaje en que se encuentre el fichero de código fuente y hacer el *check-in*, la macro se expande y nos da varias informaciones, entre ellas el número de versión asignado.

Ejemplo

Podríamos encontrar, por ejemplo, la siguiente línea de comentario:

```
/* $Id: main.c,v 1.4 2004/12/30 22:09:57 devel1 Exp $ */
```

Quizás lo más interesante de aquí sea saber la versión que estamos manejando (1.4), la fecha del envío y el usuario que lo hizo (devel1).

Es posible recuperar una versión concreta de un fichero. Para ello escribiremos la orden:

```
cv s update -r versión fichero
```

También podemos elegir el número de versión que queremos enviar, siempre y cuando sea mayor que los que ya se han utilizado (por tanto, no podemos sobrescribir una versión, sino que siempre habrá una versión nueva en el repositorio):

```
cv s ci -r versión fichero
```

Nota

Para más información, remitimos a la siguiente bibliografía que se puede consultar íntegramente en la Red:

- *Manual de GNU*:
http://www.gnu.org/software/cvs/manual/html_chapter/cvs_toc.html
- *Open source Development with CVS*:
<http://cvsbook.red-bean.com/>

En otro caso, el *check-in* siempre buscará un número de versión automáticamente.

Ejemplo

La historia de un fichero típico puede ser, por ejemplo, la siguiente:

1.1 → 1.2 → 1.3 → 3.0 → 3.1 → 3.2 ...

En este caso, la versión 3.0 la hemos especificado nosotros, y las demás se han generado automáticamente.

2.3.2. Etiquetas

Los números de versión no son suficientes para identificar los ficheros. Supongamos que queremos publicar una versión *beta* del proyecto que estamos desarrollando, para que el público la pruebe. En general, el proyecto consta de decenas de ficheros, unos con más versiones y otros con menos. La tarea de empaquetar una versión de cada fichero para conformar la versión *beta* que deseamos publicar va a requerir un tedioso trabajo de localización y extracción de cada versión de los ficheros.

En cualquier sistema de control de versiones es posible etiquetar versiones de ficheros con un identificador común.

Lo más sencillo es etiquetar las versiones de cada fichero que tenemos en el espacio de trabajo en cada momento. Para ello, en el directorio raíz del proyecto lanzaremos el mandato siguiente:

```
cvs tag etiqueta
```

Esta orden pone la etiqueta elegida en cada fichero del espacio de trabajo. En la figura 2-1 vemos las versiones que podrían haber sido etiquetadas.

Figura 2-1. Etiquetado de ficheros por versiones

File A	File B	File C	File D	File E
			1.1	
			1.2	
			1.3	
			1.4	
			1.5	
			1.6	
			1.7	
	1.1		1.8	
	1.2		1.9	
	1.3		1.10	1.1
	1.4		1.11	1.2
	1.5		1.12	1.3
	1.6		1.13	1.4
	1.7	1.1	1.14	1.5
	1.8	1.2	1.15	1.6
1.1	1.9	1.3	1.16	1.7
1.2	1.10	1.4	1.17	1.8
1.3	1.11	1.5		1.9
		1.6		1.10

La operación de etiquetado de ficheros es una de las pocas de escritura en el CVS que no nos pide explicar la causa y la edita en un fichero de registro de sesión (*log*). Sin embargo, por esa misma razón conviene que la etiqueta sea lo más descriptiva posible. Además, es una buena práctica incluir la fecha.

Una buena etiqueta puede ser: *Release-Beta1-20041110*, que nos indicaría que ésta es la liberación de la versión Beta1, e incluye además la fecha de la publicación.

Una vez etiquetado, el acceso a las versiones se hace como si se solicitase una revisión concreta, es decir, mediante el modificador `-r`. Así pues, si un usuario anónimo quiere obtener esta versión Beta1 del CVS, lanzará la orden siguiente:

```
cvs -r Release-Beta1-20041110 co proyecto1
```

2.3.3. Ramas

Por último, vamos a introducirnos en el manejo de las **ramas** de CVS. Veamos en primer lugar un ejemplo práctico.

Supongamos que hemos liberado una versión de nuestro proyecto. A partir de ahora, solamente nos interesa corregir fallos de seguridad y otros errores graves, y dejamos aparcadas las nuevas funcionalidades para otra liberación. Esto se puede gestionar fácilmente en el CVS mediante las ramas.



La idea de **rama** es que en un momento determinado se puede marcar un inicio de rama en los ficheros del proyecto. A partir de ese momento, el desarrollador puede elegir entre modificar los ficheros de la rama principal o bien los de la otra rama. Es decir, aparecen dos desarrollos independientes.

Siguiendo con nuestro ejemplo, en algún momento vamos a querer integrar las funcionalidades nuevas añadidas al proyecto con las correcciones de seguridad que se han ido implementando en la otra rama. Es el momento de volver a unir las dos ramas por medio de una operación de fusión.

La definición y el acceso a las ramas con el CVS se hace de forma muy similar al etiquetado. Lo más sencillo es crear una rama a partir de la última versión de trabajo de cada fichero, con la siguiente orden:

```
cv$ tag -b nombre-rama
```

Tras esta operación, nuestro espacio de trabajo continuará en la rama principal. En cambio, otro desarrollador puede acceder a cualquiera de las dos ramas para trabajar sobre ella. Por ejemplo, para acceder a la rama en la que hemos estado trabajando nosotros, escribirá un simple:

```
cv$ co proyecto
```

o puede acceder a la rama nueva con:

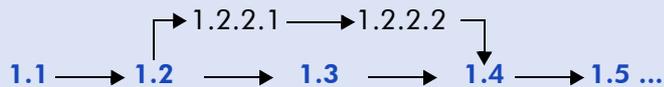
```
cv$ co -r nombre-rama proyecto
```

A partir de aquí, cuando haga *check-in*, sus cambios se enviarán a una rama o a otra, según la que se haya elegido.

Obsérvese que los números de versión en las ramas nuevas tendrán cifras nuevas, para identificar así en qué rama estamos.

Ejemplo

Podríamos tener, por ejemplo, las revisiones y las ramas siguientes:



En este ejemplo hemos creado una rama a partir de la versión 1.2 de uno de los ficheros, se han llevado a cabo dos modificaciones independientes y finalmente se ha fusionado sobre la rama principal el trabajo efectuado sobre la otra rama para generar la versión 1.5 de nuestro fichero.

La fusión de las dos ramas es sencilla. Por ejemplo, para fusionar nuestro espacio de trabajo, perteneciente a la rama principal, con el trabajo efectuado en otra rama *parches-seguridad-1-0*, se ordenará la operación siguiente:

```
cvs update -j parches-seguridad-1-0
```

Esta fusión es frecuente en versiones con muchas diferencias, de modo que es bastante probable que surjan conflictos que tengamos que resolver a mano, tal y como ya hemos visto.

Una vez editado el fichero para resolver el conflicto, enviaremos los cambios (*check-in*) y se generará así la nueva revisión en la rama principal, que mezcla los cambios de la rama de parches (sería la versión 1.5 del ejemplo anterior).

2.3.4. Información sobre etiquetas y ramas

El uso del etiquetado y las ramas en un proyecto puede complicar bastante la elección de las versiones de trabajo. Con CVS es posible

obtener información sobre las ramas y etiquetas de cada fichero mediante la orden `status`.

Por ejemplo, si nuestro fichero `main.c` ha tenido la historia de revisiones y ramas del ejemplo anterior y acabamos de introducir la versión 1.2.2.2, esta orden nos mostraría lo siguiente:

```
$ cvs status -v main.c
=====
File: main.c Status: Up-to-date
Working revision: 1.2.2.2 Fri Dec 31 00:17:04 2004
Repository revision: 1.2.2.2/home/devell1/cvsrepos/proyecto1/main.c,v
Sticky Tag: parches-seguridad-1-0 (branch: 1.2.2)
Sticky Date: (none)
Sticky Options: (none)
Existing Tags:
    parches-seguridad-1-0 (branch: 1.2.2)
```

De este modo sabemos que trabajamos con una versión de la rama *parches-seguridad-1-0* (por el número de revisión 1.2.2.x). La orden `status` vista desde la rama principal nos mostrará también esta etiqueta, de manera que sabremos con qué etiquetas tenemos que solicitar a CVS la fusión del código.

2.4. Subversion: la próxima generación

Subversion es la generación siguiente de control de versiones que previsiblemente va a conquistar próximamente el mundo del software libre [Collins04]. El objetivo principal de sus creadores era tener un sistema de control de versiones que solucionara las principales deficiencias de CVS, pero que a su vez se basase en la filosofía y forma de trabajar del CVS para facilitar la migración de un sistema a otro.

Subversion ofrece las características adicionales siguientes respecto a CVS:

- **Versionado de directorios.** Con CVS sólo los ficheros tienen versiones.

- **Verdadero control de versiones.** Así, por ejemplo, se pueden mover los ficheros de directorio en directorio, mientras que en CVS esta tarea sólo se puede hacer en el repositorio.
- **Commits atómicos.** De este modo se eliminan problemas de concurrencia, entre otros).
- **Metadatos.** Subversion admite datos adicionales tanto sobre los ficheros como sobre su versionado.
- **Diferentes modos de acceso**
- **Consistencia de datos.** Subversion no diferencia entre ficheros de texto y ficheros binarios.

3. Sistemas de seguimiento de fallos

Los sistemas de seguimiento de fallos (*Bug Tracking Systems* o *BTS*, en inglés) son herramientas destinadas a la gestión automática de notificación de errores y su corrección.

Cuando un usuario autorizado detecta un fallo en un software, acudirá al *BTS* para informar del mismo y el *BTS* lo notificará de forma automática al responsable del software. Los programadores pueden tener acceso al sistema, lo cual les permite conocer los partes de trabajo que tienen abiertos, así como notificar su resolución.



El sistema de seguimiento de fallos (*BTS*) notifica de forma automática (por correo electrónico) los cambios de estado de un parte a las personas que puedan estar interesadas: el usuario que notificó el error, los programadores que han de corregirlo o un tercero (por ejemplo, el equipo responsable del control de calidad de la empresa de desarrollo).

En resumen, un sistema de seguimiento de fallos debe contar con las siguientes características:

- Capacidad de almacenar un histórico de fallos, y dentro de cada uno, sus diferentes estados.
- Capacidad de asignar diversos atributos de interés a cada fallo, como la plataforma sobre la que corre; que admita la clasificación por severidad o prioridad.
- Capacidad de envío automático de notificaciones de cambio a las partes afectadas.

Nota

Hemos visto las aplicaciones de tipo SourceForge en el capítulo 1.

- Sistema de permisos que autorice y desautorice determinadas operaciones a ciertos tipos de usuario.
- Posibilidad (deseable) de generar algún tipo de estadísticas sobre los fallos almacenados.

Hay en el mercado bastantes variantes de BTS, algunas como software libre y otras como software propietario. Otras veces forman parte de paquetes más grandes, como el que se incluye en las aplicaciones de tipo SourceForge y que ya hemos visto brevemente. En ocasiones, el BTS tiene una interfaz de gestión basada en web o en una aplicación cliente, lo que facilita su manejo.

En este capítulo veremos el sistema de seguimiento de fallos más utilizado entre los grandes proyectos de software libre: Bugzilla.

3.1. Seguimiento de fallos con Bugzilla

Bugzilla es un sistema de seguimiento de fallos muy popular, probablemente por su facilidad de instalación y gestión, su cierta madurez y su flexibilidad [Bugzilla04]. Aunque quizás también lo sea porque es el utilizado por proyectos como Mozilla, GNOME o KDE.

Bugzilla se administra casi por completo mediante su interfaz web, y en el caso de que no tengamos que instalarlo a mano, es raro que tengamos que ejecutar tareas administrativas desde la línea de comandos (tareas que se centrarían en modificaciones directas de su base de datos, o de los propios guiones de Bugzilla).

Bugzilla está escrito en lenguaje Perl y ha sido diseñado de modo que separa los guiones y las plantillas de la interfaz web, y así puede ser fácilmente adaptado a un entorno concreto.

3.1.1. Los fallos en Bugzilla

Antes de proseguir, nos detendremos brevemente en cómo organiza Bugzilla los fallos. En este BTS, los fallos forman parte de un componente, y cada componente forma parte de un producto.

Así pues, a la hora de organizar nuestro BTS, debemos crear un producto por cada sistema de software que esté bajo nuestro control, y dentro de cada producto, vamos a dividir los diferentes subsistemas para definir los componentes.

Cada componente tendrá un propietario (a quien inicialmente se asignan todos los fallos de ese componente) y las versiones son comunes a todos los componentes de un producto.

Normalmente, una aplicación se asimilará, en el sentido de su supervisión, a un producto, de modo que se creará un componente por cada subsistema, que estará coordinado por un responsable.

Otros aspectos de la gestión de fallos, como los distintos niveles de severidad, los distintos estados de error posibles o las plataformas, deben establecerse por edición directa de los guiones y la base de datos, lo cual excede del propósito de este material.

3.2. Instalación y configuración de Bugzilla

En los siguientes apartados se explica cómo efectuar una instalación completa de Bugzilla, y cómo crear las cuentas necesarias para poder empezar a trabajar con este BTS.

3.2.1. Instalación de Bugzilla

Lo mejor es dejar que el paquete se instale automáticamente, ya que la configuración íntegramente manual es muy laboriosa.

Asumimos que tenemos instalado Debian (Sorge o posterior). Antes de instalar Bugzilla nos aseguraremos de que está instalado y configurado MySQL, y que conocemos la contraseña del usuario *root* de MySQL (que es el usuario con capacidad administrativa en todo el sistema de base de datos). Por defecto, la contraseña de este usuario está en blanco, por lo que es recomendable haberla cambiado antes.

Ejemplo

Posiblemente, en casi cualquier aplicación podamos tener un subsistema de interfaz de usuario, otro de base de datos, etc.

Si se cumplen las condiciones anteriores, instalaremos Bugzilla con la siguiente orden:

```
apt-get install bugzilla
```

El programa de instalación nos preguntará si queremos hacer una instalación asumiendo usuarios y valores por defecto o si, por el contrario, queremos proporcionarlos nosotros. Esta última opción es mejor, sobre todo si hemos cambiado algunos valores predeterminados de la instalación de MySQL, o simplemente si hemos puesto contraseña al usuario administrativo de la base de datos.

El sistema nos preguntará por el usuario administrativo (que será *root*), su contraseña, el servidor de base de datos (que será *localhost*), el puerto (aceptaremos el valor predeterminado, 3306), el usuario deseado en MySQL para gestionar la base de datos del sistema Bugzilla (que será *bugs*), el nombre de la base de datos (también *bugs*), el correo electrónico del administrador de Bugzilla (que recibirá notificaciones y por tanto debe existir; además, también es el identificador de administrador del sistema Bugzilla), el nombre del administrador y la contraseña que queramos dar al administrador de Bugzilla.

Bugzilla se instalará, creará la base de datos y quedará instalado y listo para su configuración.

3.2.2. Configuración de Bugzilla

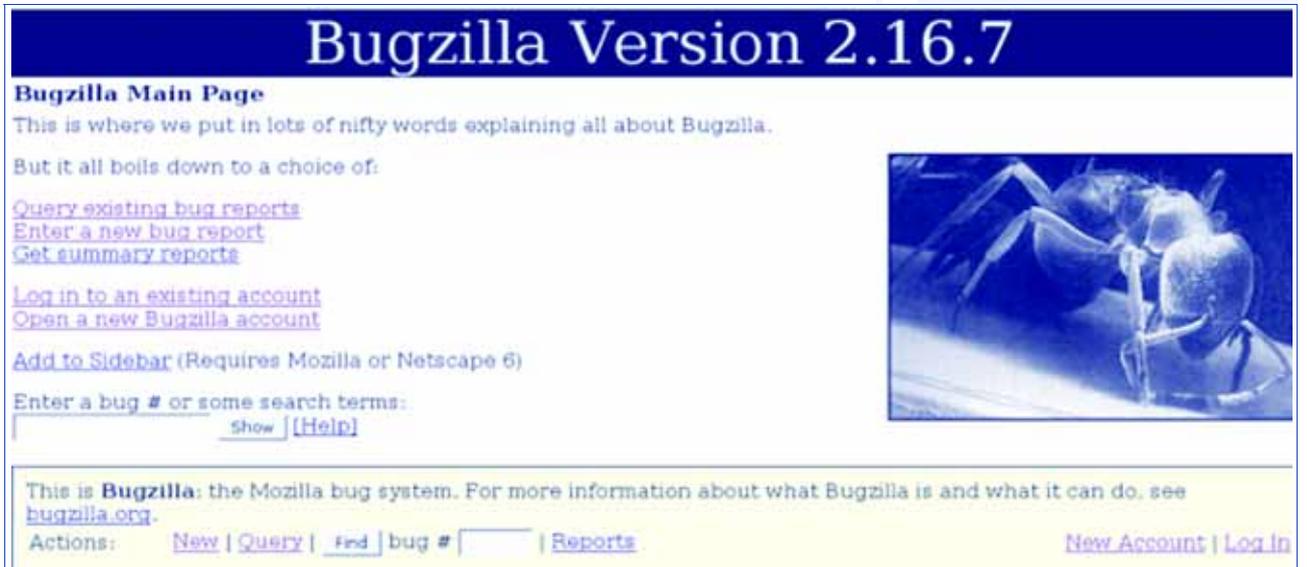
Si hemos seguido los pasos anteriores, tendremos un sistema Bugzilla instalado, funcional pero con algunos aspectos pendientes de configuración.

El resto de la configuración se lleva a cabo desde la interfaz principal de Bugzilla, a la que se accede por un navegador. Abriremos un navegador por la dirección:

```
http://nombre-servidor/bugzilla
```

Nos aparecerá la pantalla principal de Bugzilla (figura 3-1).

Figura 3-1. Interfaz principal de Bugzilla



Pulsaremos 'Log In' en el enlace inferior e introduciremos como usuario el correo electrónico de administrador y como contraseña, la que hemos proporcionado durante la instalación.

A continuación, se nos mostrará una pantalla de consulta de errores, como un formulario lleno de opciones. Por ahora nos desplazaremos hacia la parte inferior de la página hasta un cuadro como el de la figura 3-2.

Figura 3-2. Cuadro de acceso a las acciones de Bugzilla



Pulsaremos el enlace 'parameters' y rellenaremos el formulario que aparecerá a continuación. Debemos introducir todos los valores, pero específicamente el del nombre del responsable del BTS ('Bugzilla Maintainer').

Antes de crear las cuentas de usuario, es necesario configurar los grupos. Los **grupos** no son más que patrones que se utilizan para asignar permisos a los nuevos usuarios. No profundizaremos mucho

en este tema y simplemente configuraremos nuestro BTS para que los nuevos usuarios puedan editar solamente las notificaciones que sean propias. Para ello, seleccionaremos el enlace 'groups' y en la página que nos mostrará, dejaremos en blanco todos los patrones de usuario 'User RegExp' que aparecen para los campos antes de darle al botón de enviar (figura 3-3).

Figura 3-3. Interfaz de configuración de grupos

BIT Name Description User RegExp Active Action

[Submit changes](#) [Add Group](#)

Name is what is used with the UserInGroup() function in any customized cgi files you write th by email to limit a bug to a certain groupset.

Description is what will be shown in the bug reports to members of the group where they ca group.

User RegExp is optional, and if filled in, will automatically grant membership to this group to matches this regular expression.

The **Active** flag determines whether or not the group is active. If you deactivate a group it will although bugs already in the group will remain in the group. Deactivating a group is a much l group would be.

In addition, the following groups that determine user privileges exist. You can only edit the U duplicate the Names of any of them in your user groups.

Also please note that both of the Submit Changes buttons on this page will submit the chang convenience.

BIT	Name	Description	User RegExp
1	tweakparams	Can tweak operating parameters	
2	editusers	Can edit or disable users	
4	creategroups	Can create and destroy groups.	
8	editcomponents	Can create, destroy, and edit components.	
16	editkeywords	Can create, destroy, and edit keywords.	
32	editbugs	Can edit all aspects of any bug.	
64	canconfirm	Can confirm a bug.	

[Submit changes](#)

This is **Buzzilla**, the Mozilla bug system. For more information about what Buzzilla is and for

A continuación, puede ser una buena idea crear cuentas de usuario, al menos con los responsables del software que vamos a supervisar. Seleccionaremos el enlace 'users' de la página anterior, y dentro de éste, el enlace 'Add a new user'. Nos aparecerá la siguiente pantalla (figura 3-4).

Figura 3-4. Interfaz de alta de nuevos usuarios

Add user

Login name:

Real name:

Password: (enter new password to change)

Disable text:

If non-empty, then the account will be disabled, and this text should explain why.

Groups and Privileges: The new user will be inserted into groups based on their userregexps. To change the group permissions for this user, you must edit the account after creating it.

[Add](#)

El nombre de usuario ('Login name') será el correo electrónico del desarrollador, por el cual recibirá las notificaciones. Debemos introducir este valor, el campo 'Real Name' y una contraseña inicial, y dejar vacío el cuadro de texto 'Disable text'.

Con esta interfaz tendremos que crear, al menos, las cuentas de los desarrolladores que vayan a ser responsables iniciales de cada componente de software que se supervise. Más adelante, los propios interesados ya los podrán crear directamente desde la interfaz web.

A continuación, seleccionaremos 'products' y crearemos un producto por cada sistema de software que vayamos a supervisar. Dentro de cada producto, podemos editar sus componentes (subsistemas) y sus versiones. Recordemos que es necesario asignar un desarrollador inicial para cada componente, a quien se asignarán y recibirá todos los fallos que se envíen para ese componente.

Hecho esto, podemos dar a nuestro BTS por configurado. A partir de este momento, es necesario que los desarrolladores y usuarios lo conozcan y nos comiencen a enviar fallos por el sistema.

3.3. Notificación de fallos

Un BTS para software libre ha de aceptar fallos enviados por cualquier usuario, aunque inicialmente no sepamos nada de él. En Bugzilla no crearemos nosotros las cuentas de usuario, sino que los mismos usuarios se las podrán crear la primera vez que quieran enviarnos un fallo. El administrador de Bugzilla creará directamente sólo las cuentas de los desarrolladores necesarios para adjudicar los fallos.

3.3.1. Creación de cuentas

Probablemente, un usuario que acceda a nuestro BTS desde la dirección web de su interfaz quiera empezar por notificar un fallo. Sin embargo, al intentarlo, se le solicitará darse de alta en una cuenta. Así pues, pulsará el enlace correspondiente y rellenará el formulario (como dato imprescindible debe dar una dirección de correo electróni-

co, que se usará para enviarle una contraseña de acceso, así como las notificaciones posteriores que le correspondan).

Figura 3-5. Interfaz de creación de una cuenta de usuario

Create a new Bugzilla account

To create a Bugzilla account, all that you need to do is to enter a legitimate e-mail address. The account will be created, and its password will be mailed to you. Optionally you may enter your real name as well.

E-mail address:

Real name:

[Create Account](#)

3.3.2. Notificación de un fallo

El usuario utilizará su recién adquirida contraseña para acceder al sistema desde el enlace 'Login' que aparece en el cuadro inferior de la página (figura 3-2). Una vez dentro, accederá de nuevo al cuadro inferior y buscará el enlace 'New'. Se le mostrará el formulario para crear el informe de fallo (figura 3-6).

Figura 3-6. Interfaz de notificación de fallos

Enter Bug This page lets you enter a new bug into Bugzilla.

Before reporting a bug, please read the [bug writing guidelines](#), please look at the list of [most frequently reported bugs](#), and please [search](#) for the bug.

Reporter: Products: TrainControl

Version: Component:

Platform: OS:

Priority: Severity:

Assigned: (Leave blank to assign to default component owner)

CC:

URLs:

Summary:

Description:

A la hora de crear el informe de fallo, el usuario debe seguir unas normas con el objetivo de describirlo de la forma más concisa y útil posible. Por ello, deberá señalar el producto, el componente y la versión exactos que le han producido el error, así como (si es aplicable y posible) la plataforma y el sistema operativo. Rellenará los campos de prioridad y severidad a su criterio, pero preferiblemente consultará otros fallos previos para elegir los valores adecuados para este caso. Dejará en blanco los campos de 'asignación a' y 'copia a' para que el sistema les dé un valor por defecto.

El campo de la URL no suele ser aplicable. En caso de serlo (por ejemplo, URL de la aplicación que origina el fallo), debe proporcionarse. El resumen deberá ser lo más conciso posible (una frase descriptiva). La descripción es un texto libre, pero es deseable que contenga la siguiente información:

- **Resumen del fallo.** Es un texto que explica del modo más detallado posible el fallo detectado (cuatro o cinco líneas).
- **Pasos para reproducir el fallo.** Si el fallo es reproducible, debemos indicar los pasos que hemos seguido para provocarlo.
- **Resultados esperados y obtenidos.** Debemos incluir los resultados que hemos observado en contraste con los que esperábamos.
- **Información adicional.** Probablemente deseemos introducir información adicional sobre cómo reproducir el fallo (por ejemplo, versiones de algún software que deba estar instalado) o que facilite el trabajo del desarrollador que lo vaya a analizar y corregir.



Debemos notificar el fallo de la forma más breve y precisa posible, lo que contribuirá sin duda a que el fallo sea corregido en un plazo de tiempo más corto.

Cuando pulsemos el botón de envío inferior, se registrará el fallo y se enviará una notificación al responsable del componente elegido.

Antes de decidimos a enviar un fallo, es recomendable revisar, dentro de lo posible, que no haya sido ya enviado o corregido. Para ello podemos hacer una búsqueda (tal como se mostrará en la siguiente sección) y tal vez procurar tener instalada la última versión del software que nos ha dado el fallo. Sin embargo, cuando el número de fallos almacenados en el sistema es muy elevado, resulta muy difícil no repetir los mismos fallos que otras personas, si bien el desarrollador lo detectará pronto y marcará el fallo como 'Duplicado'. Los grandes proyectos de software libre que utilizan Bugzilla suelen tener tal cantidad de informes de fallos que a menudo cuentan con una persona dedicada exclusivamente a gestionar los fallos en Bugzilla [Villa03]. Se trata de una tarea muy ardua, pero muy agradecida por los desarrolladores, ya que, si se hace bien, les permite tener los fallos clasificados y dedicarse directamente a la corrección del error.

3.4. Búsqueda y tratamiento de fallos

Bugzilla no sólo permite informar de los fallos, sino que es una excelente herramienta para su seguimiento. Uno de los objetivos que marcaron su creación fue facilitar en lo posible la tarea de los desarrolladores y que éstos pudieran conocer de una manera sencilla y rápida los fallos que hay en su software y su grado de prioridad.

Además, cada fallo puede ser asignado a otro desarrollador o su estado puede ser cambiado. A continuación veremos cómo se llevan a cabo estas acciones con Bugzilla.

3.4.1. Búsqueda de fallos

La página de búsquedas de Bugzilla puede consultarse con o sin cuenta de usuario (aunque recordemos que sólo podremos modificar su contenido con una cuenta). Si en el cuadro inferior de cualquier página de Bugzilla seleccionamos el enlace 'Query', se nos mostrará un formulario con muchos campos, que rellenaremos para afinar la búsqueda en la medida de lo posible. En la figura 3-7 mostramos parte de la página de búsquedas.

Figura 3-7. Interfaz de búsquedas de Bugzilla

The screenshot shows the 'Search for bugs' interface. It includes several sections for filtering search results:

- Summary:** A dropdown menu set to 'contains all of the words/strings' and a search button.
- Product:** A dropdown menu set to 'TrainCentral'.
- Component:** A dropdown menu set to 'CommunicationSubsystem'.
- Version:** A dropdown menu set to '1.0', with other options '1.1pre' and 'unspecified' visible.
- A comment:** A dropdown menu set to 'contains all of the words/strings'.
- The URL:** A dropdown menu set to 'contains all of the words/strings'.
- Status:** A dropdown menu with options: UNCONFIRMED, ASSIGNED, REOPENED, RESOLVED, VERIFIED, CLOSED.
- Resolution:** A dropdown menu with options: FIXED, INVALID, WON'TFIX, LATER, REMIND, DUPLICATE, WORKSFORME.
- Severity:** A dropdown menu with options: blocker, critical, major, normal, minor, trivial, enhancement.
- Priority:** A dropdown menu with options: P1, P2, P3, P4, P5.
- Hardware:** A dropdown menu with options: All, DEC, HP, Macintosh, PC, SGI, Sun.
- OS:** A dropdown menu with options: All, Windows 3.1, Windows 95, Windows 98, Windows ME, Windows 2000, Windows NT.
- Email and Numbering:**
 - Any of: bug owner, reporter, CC list member, commenter.
 - Any of: bug owner, reporter, CC list member, commenter.
 - contains: [dropdown menu]
 - contains: [dropdown menu]
 - Only include: [dropdown menu] bugs numbered: [input field]
- Bug Changes:**
 - Only bugs changed in the last days: [input field]
 - Only bugs where any of the fields: [dropdown menu]
 - were changed between [input field] and [input field]

Al hacer la búsqueda, el sistema nos contestará con una lista de errores, posiblemente dividida en varias páginas. Cuando hayamos encontrado lo que buscamos, podemos seleccionar su enlace y llegamos a la página de visualización y edición del fallo (figura 3-8).

Figura 3-8. Interfaz de visualización y edición de fallos

The screenshot displays a web-based bug tracking interface. At the top, the bug number '216181' is shown with an 'alias' field. Below this, several dropdown menus are visible for 'Product' (Mozilla Application Suite), 'Component' (Component), and 'Status' (VERIFIED). A 'Resolution' field indicates 'DUPLICATE of bug 46845'. The 'Assigned To' field contains a note: 'This bug is awaiting triage to an appropriate owner or component <general@browser.bugs>'. Other fields include 'QA Contact' (general@browser.bugs), 'URL', 'Summary' (selected item in Combo / Drop-down list not refreshed correctly after reload), 'Status Whiteboard', and 'Keywords'. On the right side, there are filters for 'Hardware' (PC), 'OS' (All), 'Version' (1.4 Branch), 'Priority', 'Severity' (major), and 'Target Milestone'. A 'Reporter' field shows 'Jerome Lacoste <mlfle>'. Below these are 'Add CC' and 'CC' fields. A 'Flags (Help)' section lists several flags: 'blocking-avary1.0-1.10N', 'blocking1.7.5', 'blocking1.7.5', and 'blocking1.8a8'. A table with columns 'Attachment', 'Type', 'Created', 'Size', 'Flags', and 'Actions' is present, with a link to 'Create a New Attachment'. Below the table, there are sections for 'Bug 216181 depends on' and 'Bug 216181 blocks', both with links to 'Show dependency tree' and 'Show dependency graph'. There are also 'Votes: 0' and 'Additional Comments' sections. At the bottom, there are radio buttons for 'Login as VERIFIED DUPLICATE' and 'Reopen bug', along with 'View Bug Activity' and 'Format For Printing' links. A 'Description' section at the very bottom contains a small text block with a 'Copy' link and an 'Open' button.

3.4.2. Estados de un fallo

Una de las características que más nos interesan al consultar un fallo es su estado. El **estado de un fallo** es el campo que nos resume cómo ha quedado el fallo, si ha sido corregido o no, etc.

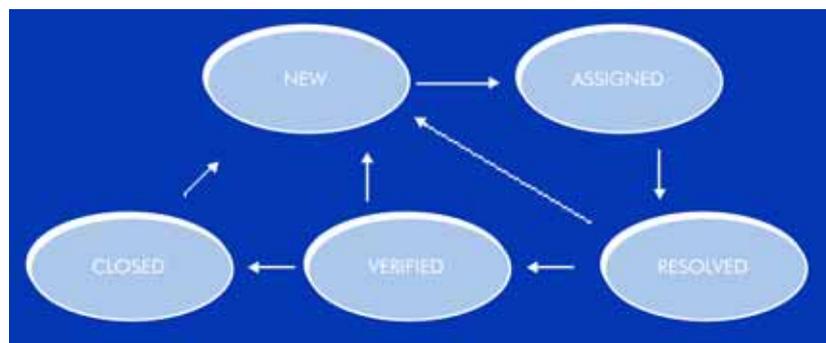
Aunque los estados se pueden personalizar, los más habituales son:

- NEW (nuevo). Es el estado que tiene un fallo cuando acaba de ser enviado por un usuario.
- ASSIGNED (asignado). Este estado se asigna cuando un desarrollador ha aceptado y se responsabiliza de su corrección.

- c) **RESOLVED** (resuelto). Indica que el fallo ha sido resuelto. Asociado a este estado se encuentran las causas. Las más habituales son:
- **FIXED** (corregido). El fallo ha sido corregido.
 - **DUPLICATE** (duplicado). Se ha detectado que existe un informe de este fallo realizado en otro momento, con lo que se anota que este fallo es idéntico a otro (se acompaña del número del otro fallo).
 - **INVALID** (inválido). El fallo no es tal. Generalmente si se selecciona este estado, el desarrollador debe explicar claramente por qué no lo es.
 - **WONTFIX** (no se corregirá). Es un fallo que por alguna causa no será corregido, al menos en la versión actual.
- d) **VERIFIED, CLOSED** (verificado, cerrado). Son estados que indican que el fallo fue resuelto, y posteriormente verificada su resolución o bien definitivamente cerrado.
- e) **REOPENED/NEW** (reabierto). Un fallo se reabre cuando se comprueba que el error no se ha corregido realmente, pese a haber sido marcado como *corregido*.

En la figura 3-9 podemos ver el ciclo de vida de un fallo.

Figura 3-9.



Como se puede observar, cuando un fallo es resuelto, sea cual sea su estado de resolución, puede ser reabierto y pasar de nuevo al estado **NEW** (sin asignación). Sin embargo, lo deseable es no reabrir los fallos que han sido verificados o cerrados. Normalmente, serán los miembros del equipo de control de calidad los que cerrarán los

fallos definitivamente y dejarán la verificación al usuario que envió el error originalmente. Sin embargo, a menudo los usuarios no se ocupan de marcar los fallos verificados, lo cual deja cientos de éstos en estado de *resuelto*, sin que se cierren en mucho tiempo.

3.4.3. Edición de un fallo

Cuando estamos en la página correspondiente a un fallo, podremos efectuar, según el tipo de permisos que tengamos, diversos tipos de cambios:

- **Añadir un comentario.** Cualquier usuario, por lo general, puede hacer comentarios sobre cualquier fallo, por ejemplo, aclarar los pasos para reproducirlo o dar posibles pistas sobre su resolución.
- **Cambio de estado.** Esto solamente debe hacerlo el desarrollador que tiene asignado el fallo (para marcar el fallo como resuelto, por ejemplo) o el propio usuario (en casos como reabrir un fallo erróneamente cerrado).
- **Reasignar.** Es posible cambiar al desarrollador que tiene asignada la resolución de un fallo. Normalmente lo hará el responsable del proyecto o bien alguno de los desarrolladores involucrados en el mismo.
- **Cambio de datos básicos.** Esto se hace con menor frecuencia: cambiar el resumen o recolocar el fallo en otro producto o componente. Por supuesto, solamente en caso de error evidente.

Todos los usuarios pueden, de forma predeterminada, editar cualquier atributo de un fallo, incluso marcarlo como corregido. Es buena práctica que el administrador modifique los permisos de los usuarios para impedirlo, tal como ya indicamos al editar los grupos de usuarios. Así, solamente los usuarios que notifican un error y los desarrolladores que tienen el fallo asignado podrán modificarlo, aunque todos podrán introducir comentarios en cualquier momento.

4. Listas de correo electrónico

Las listas de correo electrónico son herramientas muy utilizadas en cualquier comunidad en Internet, no solamente en la de software libre. Cualquier afición en la vida real puede llevar a crear una comunidad virtual en Internet sustentada, casi siempre, en una lista de correo o en un foro en el que se hallan suscritos los miembros de esa comunidad.

La necesidad de las listas de correo en el mundo del software libre es similar, y en muchos casos, resulta estratégica para lograr la divulgación de un proyecto: por un lado, al comenzar el desarrollo casi siempre necesitaremos una lista para comunicar y coordinar a los programadores. Por otro, cuando publiquemos nuestra aplicación, seguramente se creará una lista para alentar la comunicación y el intercambio de experiencias entre los usuarios.

A poco que crezca un proyecto, surgirán iniciativas nuevas, tales como traducirlo a otros idiomas o crear documentación. En torno a estas actividades surgirán comunidades de personas que trabajan en ellas



Las comunidades que surjan en torno a un proyecto de desarrollo de software libre se sustentarán casi con seguridad en listas de correo.

4.1. Qué es una lista de correo

Podemos definir una lista de correo como una aplicación basada en correo electrónico que tiene las siguientes características:

- a) Almacena un conjunto de direcciones de correo electrónico llamadas *suscriptores*.

- b) Cualquier mensaje enviado a la lista es remitido automáticamente a todos los suscriptores.

Además, según la herramienta elegida, se podrá disponer de algunas características más, entre las que destacamos las siguientes:

- Mecanismos automatizados para la gestión de la lista (es decir, alta y baja de suscriptores) basados en correo electrónico.
- Interfaces web para la gestión de la lista.
- Archivo automático de los mensajes de la lista. Un sistema avanzado es capaz de archivar todo lo que se envía a la lista y proporcionar algún método de acceso a esos archivos (casi siempre, mediante una interfaz web).
- Funciones avanzadas: permisos de envío, envío periódico (*digest*), filtrado, etc.

4.2. Herramientas

En cualquier servidor de correo electrónico contamos normalmente con una herramienta para poner en marcha inmediatamente una lista de correo: el fichero de alias (que acostumbra a encontrarse en `/etc/aliases`).

Un **fichero de alias** maneja directamente el agente de correo (Sendmail, Postfix, etc.), y contiene una lista de pares (alias, direcciones). Cuando este fichero existe, todo mensaje enviado a uno de sus alias será remitido automáticamente a cada una de las direcciones del alias.

Ejemplo

Supongamos, por ejemplo, que nuestro fichero `/etc/aliases` fuese el siguiente:

```
# Fichero /etc/aliases

proyecto1-devel:usuario1@dominio1,
                usuario2@dominio2, usuario3@dominio3
```

```
proyecto1-users:usuario4@dominio4,  
    usuario5@dominio5, usuario6@dominio6,  
usuario7@dominio7, usuario8@dominio8
```

Cada vez que alguien escriba a: `proyecto1-devel@nuestrodominio.com` se remitirá una copia del mensaje a `usuario1@dominio1`, `usuario2@dominio2` y `usuario3@dominio3`.

De la misma manera, cuando alguien escriba a `proyecto1-users@nuestrodominio.com`, se remitirá una copia del mensaje a cada uno de los miembros que aparecen listados en el fichero.

Las posibilidades de `/etc/aliases` acaban aquí: para añadir o quitar usuarios, el administrador del sistema deberá editar el fichero: no hay gestión automática, por lo que su gestión se volverá muy tediosa para el administrador a poco que crezca la comunidad. Además, los mensajes no se almacenan en ningún sitio (si bien existen trucos para hacerlo, son poco flexibles), y no se pueden establecer permisos ni cualquier otra función medianamente avanzada. No obstante, los ficheros de alias pueden ser la solución para comunidades pequeñas.

Hay soluciones más avanzadas que ésta para gestionar listas de correo electrónico. Nosotros nos centraremos en la más potente que existe dentro del repertorio del software libre, el gestor Mailman de listas de correo GNU. Aunque hay otros no menos populares como Sympa, Ecartis, FML, Majordomo o Listserv (estos dos últimos, por cierto, con licencias que no son libres).

4.3. Alternativas

Antes de mostrar la configuración y gestión de Mailman, mencionaremos otras alternativas utilizadas para montar comunidades de usuarios o desarrolladores:

- USENET. Los mensajes a los grupos de USENET son similares a los de correo electrónico, al menos en lo que a formato se refiere. Sin embargo, en lugar de entregarse a los buzones de los usuarios,

rios, son los usuarios los que deben conectarse a su servidor local de USENET para obtenerlos. En otro tiempo USENET fue muy popular, pese a la proliferación de sistemas similares basadas en interfaz web. Probablemente, la invasión del *spam* que han sufrido estas soluciones ha hecho decaer enormemente tanto su uso como su popularidad.

- Foros web. Son similares a USENET, en el sentido de que los mensajes no se envían a los buzones de los usuarios, sino que son éstos los que se conectan al foro para obtenerlos. Los foros web son bastante populares en algunas comunidades de usuarios poco técnicos, porque evitan al usuario tener que suscribirse y que efectuar otras gestiones propias en la lista.
- Sistemas mixtos. En ocasiones, se unen las ventajas del foro web con las de la lista de correo electrónico. Son sistemas que permiten enviar los mensajes del foro a los buzones de correo de los usuarios, pero al mismo tiempo permiten las consultas por web. Y del mismo modo, los mensajes a la lista pueden ser enviados tanto por correo electrónico, como desde la interfaz web. Uno de los servicios más populares de este tipo es el que ofrece Yahoo Groups.

4.4. Listas de correo con Mailman

Tener una lista de correo para nuestro proyecto es, sin duda, lo más conveniente, dada su flexibilidad y facilidad de gestión. En los siguientes apartados instalaremos un sistema Mailman y presentaremos sus opciones de configuración más interesantes.

4.4.1. Instalación

La instalación de Mailman puede ser completamente manual, es decir, es posible obtener las fuentes de la página principal, compilarlas y configurar a mano todo el sistema [Warsaw03].

Sin embargo, dado que la configuración de Mailman con todos sus servicios requiere bastante trabajo, recurriremos de nuevo a las faci-

lidades de instalación casi automática que nos ofrece el paquete Debian correspondiente.

Así, en una distribución Debian (Sarge o posterior), procederemos con la siguiente orden:

```
apt-get install mailman
```

Este mandato iniciará la instalación de paquetes. Durante este proceso se nos formularán algunas preguntas: en primer lugar, en qué idiomas queremos la implementación (elegiremos normalmente la lengua inglesa y nuestro idioma local, como mínimo). Después, elegiremos el idioma predeterminado (según el uso que vayamos a darle al sistema o al carácter más "internacional" del proyecto, quizás nos interese decantarnos por el inglés como idioma predeterminado, aunque luego los usuarios podrán escoger cualquiera de los idiomas instalados desde la interfaz web).

Finalizada la instalación, crearemos la primera lista: la lista de gestión, que es la que se utiliza como remitente de los mensajes de gestión del sistema [Kollar00]. Para ello, ejecutaremos:

```
newlist mailman
```

Tendremos que especificar algunos aspectos que detallamos a continuación (marcamos en negrita la parte que deberemos escribir como administradores):

```
$ newlist mailman
```

```
Enter the email of the person running the list: correo-electrónico@dominio
```

```
Initial mailman password: XXXXXX
```

```
To finish creating your mailing list, you must edit your /etc/aliases (or equivalent) file by adding the following lines, and possibly running the 'newaliases' program:
```

```
## mailman mailing list
```

```
mailman: "|/var/lib/mailman/mail/mailman post mailman"
```

```
mailman-admin: "|/var/lib/mailman/mail/mailman admin mailman"
```

```
mailman-bounces: "|/var/lib/mailman/mail/mailman bounces mailman"
```

```
mailman-confirm: "|/var/lib/mailman/mail/mailman confirm mailman"
```

Nota

Previamente, asumimos que se tiene instalado y configurado correctamente un agente de correo (como Postfix o Sendmail) y un servidor web (como Apache) en los que Mailman se apoyará.

```
mailman-join: "|/var/lib/mailman/mail/mailman join mailman"
mailman-leave: "|/var/lib/mailman/mail/mailman leave mailman"
mailman-owner: "|/var/lib/mailman/mail/mailman owner mailman"
mailman-request: "|/var/lib/mailman/mail/mailman request mailman"
mailman-subscribe: "|/var/lib/mailman/mail/mailman subscribe mailman"
mailman-unsubscribe: "|/var/lib/mailman/mail/mailman unsubscribe mailman"
```

Hit enter to notify mailman owner...

Vemos que nos deja una tarea para hacer a mano: editar nuestro fichero `/etc/aliases` (o similar, según el agente de correo que tengamos instalado).

Si el fichero `/etc/aliases` cambia, es importante incluir toda la información necesaria sobre la lista: propósito, administrador o administradores y todo lo que creamos necesario.

Por ejemplo, nuestro fichero `/etc/aliases` podría quedar así:

```
# Listas de Correo Mailman

## mailman mailing list

## Administrador: usuario@nuestrodominio.com

## Proposito: Gestion interna de Mailman

mailman: "|/var/lib/mailman/mail/mailman post mailman"
mailman-admin: "|/var/lib/mailman/mail/mailman admin mailman"
mailman-bounces: "|/var/lib/mailman/mail/mailman bounces mailman"
mailman-confirm: "|/var/lib/mailman/mail/mailman confirm mailman"
mailman-join: "|/var/lib/mailman/mail/mailman join mailman"
mailman-leave: "|/var/lib/mailman/mail/mailman leave mailman"
mailman-owner: "|/var/lib/mailman/mail/mailman owner mailman"
mailman-request: "|/var/lib/mailman/mail/mailman request mailman"
mailman-subscribe: "|/var/lib/mailman/mail/mailman subscribe mailman"
mailman-unsubscribe: "|/var/lib/mailman/mail/mailman unsubscribe mailman"
```

No olvidemos que tras editar este fichero debemos actualizar la base de datos del agente de correo, con la orden:

```
newaliases
```

Y hasta aquí llega la fase de la configuración inicial. Si todo ha ido bien, podemos iniciar nuestro sistema Mailman y no nos debería dar ningún error. Ejecutaremos:

```
/etc/init.d/mailman start
```

Una vez arrancado, podremos acceder a la interfaz web de Mailman, en la siguiente dirección:

```
http://servidor.nuestrodominio.com/cgi-bin/mailman/listinfo
```

4.4.2. Configuración de la lista de gestión

Antes de pasar nuestro sistema a producción, tenemos que configurar mínimamente la lista de correo de gestión, llamada mailman, que hemos creado durante la instalación. Lo más importante será seguramente ocultarla y hacer que el acceso a su archivo sea de carácter privado. Para ello, debemos leer el correo electrónico del administrador inicial, donde habrá llegado durante la instalación un mensaje de bienvenida con la contraseña de administrador. El mensaje contendrá un texto similar al siguiente:

```
From: mailman-admin@servidor.nuestrodominio.com
```

```
To: usuario@nuestrodominio.com
```

```
The mailing list 'mailman' has just been created for you. The following is some basic information about your mailing list.
```

```
Your mailing list password is:
```

```
mailman
```

```
You need this password to configure your mailing list. You also need it to handle administrative requests, such as approving mail if you choose to run a moderated list.
```

```
You can configure your mailing list at the following web page:
```

```
http://servidor.nuestrodominio.com/cgi-bin/mailman/admin/mailman
```

```
[...]
```

Entraremos en la interfaz de administración de la lista con la contraseña indicada en el mensaje. Para tal fin, introduciremos en el navegador web la dirección indicada en el mensaje y se nos pedirá la contraseña. De este modo entraremos directamente en la interfaz de administración. La página web para administrar cualquier lista contiene un menú superior como el que se muestra en la figura 4-1.

Figura 4-1. Interfaz de administración de la lista de correo de Mailman



Empezaremos por seleccionar la opción 'Passwords' para establecer una nueva contraseña de administración (no es necesario establecer la de moderador, ya que no se utilizará en esta lista inicial).

Después seleccionaremos en el menú superior, 'Privacy options...' para marcar afirmativamente la opción 'Require approval', y que aquellos que quieran apuntarse a la lista tengan que contar con nuestro visto bueno; y negativamente la opción 'Advertise this list', para que no aparezca anunciada públicamente en el listado de las listas de correo electrónico del servidor. Finalmente, pulsaremos el botón 'Submit your changes'.

A continuación, seleccionaremos 'Archiving options' del menú superior y marcaremos para "Is archive file source for public or private archival?", en la parte inferior, la opción 'Private'. De este modo, los mensajes de esta lista serán de acceso exclusivo para sus suscriptores (no suscribiremos a nadie de momento; si fuese necesario, lo haremos más adelante administrando esta lista como otra cualquiera).

4.4.3. Operativa de usuario de listas

Las listas de correo están diseñadas para que los interesados puedan apuntarse y consultar de manera sencilla los mensajes enviados has-

ta el momento. En los siguientes apartados se describe cómo se llevan a cabo estos procesos en Mailman.

Suscripción como invitado

Lo primero que necesitamos como usuarios de una lista de correo es suscribirnos [Oda03]. Hay varias formas de hacerlo. La más sencilla es que nos “inviten”, es decir, que el administrador proponga nuestra entrada en la lista y nosotros solamente tengamos que confirmarla.

En el caso de ser invitados, recibiríamos un mensaje parecido al siguiente:

```
From: proyecto1-users-request@servidor.midominio.com
```

```
To: usuario@sudominio.com
```

```
Subject: confirm cbf28fadlee8290848ab
```

Como usuario reconocido de Proyecto1, te invitamos a pertenecer a la lista de soporte a usuarios, esperando que te sea útil y puedas hacerla más útil a los demás.

Your address "usuario@sudominio.com" has been invited to join the Proyecto1-users mailing list at *servidor.midominio.com* by the Proyecto1-users mailing list owner. You may accept the invitation by simply replying to this message, keeping the Subject: header intact. You can also visit this web page:

```
http://servidor.midominio.com/cgi-bin/mailman/confirm/proyecto1-users/cbf28fadlee8290848ab
```

Or you should include the following line -- and only the following line -in a message to *proyecto1-users-request@servidor.midominio.com* confirm cbf28fadlee8290848ab. Note that simply sending a 'reply' to this message should work from most mail readers. If you want to decline this invitation, please simply disregard this message. If you have any questions, please send them to *proyecto1-users-owner@servidor.midominio.com*.

Este mensaje nos notifica que hemos sido invitados a entrar en la lista *proyecto1-users*. Si aceptamos la invitación, sólo tenemos que contestar al mensaje con nuestro programa de correo preferido, procurando no modificar el asunto, que contiene un *hash* que permite identificar la invitación.

Otra opción es, según las instrucciones del mensaje, pulsar sobre el enlace web proporcionado para confirmarla. Y por último, siempre podemos desechar la invitación, simplemente haciendo caso omiso del mensaje.

También es posible que nos suscriban directamente, sin pedirnos permiso, en cuyo caso normalmente recibiremos una notificación del hecho, en la que se nos indicarán las instrucciones para dejar la lista, si es nuestro deseo.

Suscripción por web

La forma más sencilla de entrar en la lista de correo como usuarios es utilizar la dirección web de la lista e interactuar con ésta.

Ejemplo

Al navegar por la página de un proyecto de nuestro interés, es probable que en algún lugar se nos indique una dirección web donde suscribirnos, por ejemplo:

```
http://servidor.dominio.com/cgi-bin/mailman/listinfo/proyecto1-users
```

Esta dirección introducida en nuestro navegador nos llevará a una página web en la que encontraremos, entre otras cosas, un formulario de suscripción como el de la figura 4-2.

Rellenaremos el formulario con nuestra dirección de correo, nuestro nombre (aunque no es obligatorio) y la contraseña que queremos usar para gestionar nuestra suscripción (habrá que introducirla dos veces). Al dar al botón 'Suscribir' recibiremos un mensaje de correo similar al de la invitación, con el que podremos confirmar la suscripción, al igual que hicimos antes, desde la página web o simplemente respondiendo al mensaje.

Este correo de confirmación se envía para evitar que nadie nos pueda suscribir sin nuestro permiso. Esta política es una de las ventajas de Mailman frente a otros gestores de listas de correo.

Figura 4-2. Interfaz de suscripción a un proyecto de software libre

Subscribing to Project01-users

Subscribe to Project01-users by filling out the following form.

Se le mandará un mensaje de correo electrónico pidiéndole una confirmación, para prevenir que otras personas le suscriban sin que usted lo sepa. Esta lista es privada, lo que significa que los suscriptores de la lista no están disponibles a los que no estén suscritos.

Your email address:

Your name (optional):

You may enter a privacy password below. This provides only mild security, but should prevent others from messing with your subscription. **Do not use a valuable password** as it will occasionally be emailed back to you in cleartext.

If you choose not to enter a password, one will be automatically generated for you, and it will be sent to you once you've confirmed your subscription. You can always request a mail-back of your password when you edit your personal options.

Pick a password:

Reenter password to confirm:

Which language do you prefer to display your messages?

Would you like to receive list mail batched in a daily digest? No Yes

Suscripción por correo electrónico

También es posible suscribirse voluntariamente al proyecto sin utilizar la interfaz web en ningún momento. Para ello, nos proporcionarán una dirección de correo, parecida a:

`nombre-de-lista-request@servidor.midominio.com`

Para solicitar la suscripción, bastará enviar a este sitio un mensaje con la palabra "subscribe".

Una vez más recibiremos un mensaje de confirmación como respuesta parecido al que hemos visto para las invitaciones y las suscripciones por web, para evitar suscripciones involuntarias.

La interfaz de correo electrónico permite hacer muchas más cosas. Para conocerlas, podemos remitir a esa misma dirección un mensaje con el texto "help".

Envío de mensajes a la lista

Para hacer llegar un mensaje a la lista, la dirección de envío será parecida a:

`nombre-de-lista@servidor.midominio.com`

En principio, cualquier mensaje que enviemos a la lista deberá ser remitido a todos los suscriptores. Cuando esto no sucede, puede deberse a alguna de las causas siguientes:

- Que la lista sea moderada. En este caso, un mensaje nos informará de la situación, y deberemos esperar a que un moderador autorice el mensaje.
- Que no estemos autorizados. En algunas listas, es preciso ser suscriptor de la misma para poder escribir en ella. Esto evita la mayoría del spam o correo no solicitado.

Además, si estamos suscritos a la lista, normalmente recibiremos una copia del mensaje que hemos enviado. Aunque no siempre, ya que a veces las listas se configuran para que no nos remitan copia de nuestros mensajes.

Consulta del archivo

Es posible consultar el almacén de mensajes enviados a la lista, incluso aunque no estemos suscritos a ella (normalmente, las listas de software libre son de consulta pública). Así pues, algunas páginas web de desarrollo de proyectos de software libre nos pueden informar de la dirección donde se halla el archivo, que suele ser una similar a esta:

`http://servidor.midominio.com/pipermail/nombre-de-lista/`

El archivo permite consultar la lista por fechas (meses o semanas), nombre o autores, según esté configurada y deseemos nosotros.

Ejemplo

En la figura 4-3 mostramos un archivo de lista de correo en un navegador.

Figura 4-3. Archivo de lista de correo de un proyecto



de correo electrónico, para cambiar nuestras opciones de suscripción desde la interfaz web.

Para ello, nos dirigiremos a la página de gestión de cuentas de la lista, la que tiene una dirección similar a:

`http://servidor.midominio.com/cgi-bin/mailman/options/nombre-de-lista/`

Como en el apartado anterior, buscaremos en la parte inferior una casilla donde introducir nuestra dirección de correo y pulsaremos el botón 'Anular suscripción o editar preferencias'. Accederemos a la página de preferencias desde la que podemos efectuar algunas operaciones interesantes:

- Borrarnos de la lista.
- Cambiar nuestra contraseña.
- Cambiar nuestras opciones de configuración de la lista. Para ver las opciones, buscaremos una tabla al final de la página, parecida a la que se muestra en la figura 4-5.

Figura 4-5. Interfaz de configuración de una lista de correo

Mail delivery Set this option to <i>Enabled</i> to receive messages posted to this mailing list. Set it to <i>Disabled</i> if you want to stay subscribed, but don't want mail delivered to you for a while (e.g. you're going on vacation). If you disable mail delivery, don't forget to re-enable it when you come back; it will not be automatically re-enabled.	<input checked="" type="radio"/> Enabled <input type="radio"/> Disabled <input type="checkbox"/> Set globally
Set Digest Mode If you turn digest mode on, you'll get posts bundled together (usually one per day but possibly more on busy lists), instead of singly when they're sent. If digest mode is changed from on to off, you may receive one last digest.	<input type="radio"/> Off <input checked="" type="radio"/> On
Get MIME or Plain Text Digests? Your mail reader may or may not support MIME digests. In general MIME digests are preferred, but if you have a problem reading them, select plain text digests.	<input type="checkbox"/> MIME <input checked="" type="checkbox"/> Plain Text <input type="checkbox"/> Set globally
Receive your own posts to the list? <small>Otherwise, you will not see a copy of every message you post to the list. If you don't want to</small>	<input type="checkbox"/> No <input checked="" type="checkbox"/> Yes

Entre estas opciones, destacaremos las de uso más habitual:

- Mail delivery: permite desactivar o reactivar la recepción de mensajes de la lista. De esta forma podemos estar suscritos sin que nos lleguen mensajes. Esto es interesante si queremos tener dere-

cho de publicación en una lista privada sin que nos lleguen los mensajes de la misma.

- **Digest mode:** si activamos esta opción, recibiremos un mensaje diario con el resumen (*digest*) de todos los mensajes enviados ese día, en lugar de uno nuevo cada vez. Si se superase cierto umbral de mensajes al día, podríamos recibir más de un resumen.

4.4.4. Operativa de administración de listas

Para terminar con este capítulo dedicado a las listas de correo electrónico, y en particular a Mailman, vamos a adentrarnos en algunas de las tareas a las que un administrador se enfrenta con mayor frecuencia.

Alta y baja de listas de correo

Durante la instalación ya adelantamos cómo se crean las listas de correo con la lista de gestión de Mailman. Para crear cualquier lista, ejecutaremos la orden:

```
newlist -l es nombre-de-lista administrador@dominio.com nombre-de-lista
```

que nos mostrará por pantalla los alias que debemos poner en el fichero `/etc/aliases` de nuestro agente de correo. Esta forma de crear la lista incluye en la línea de comandos la contraseña inicial, igual que el nombre de la lista. El administrador de ésta deberá cambiarla cuanto antes.

Es poco habitual dar de baja una lista de correo, pero si fuera necesario, habrá que decidir si quedarnos con una copia de los archivos de la lista o bien borrarlos. En el primer caso, borraremos los alias de `/etc/aliases` y lanzaremos la orden:

```
rmlist nombre-de-lista
```

Si además queremos eliminar los archivos, el mandato es:

```
rmlist -a nombre-de-lista
```

Configuración de la lista

Una vez creada la lista, su administrador tomará el control y procederá a entrar en la interfaz web de administración con la contraseña que se le suministrará por correo electrónico.

Para ello, utilizará la dirección web que se le proporcionará en el mismo mensaje, normalmente de la forma siguiente:

```
http://servidor.nuestrodominio.com/cgi-bin/mailman/admin/nombre-de-lista
```

En la figura 4-6 se muestran algunas opciones de configuración interesantes, aunque es seguro que el usuario interesado querrá explorarlas todas.

Figura 4-6. Opciones de configuración de una lista de correo

Categorías de configuración		Otras actividades administrativas
<ul style="list-style-type: none"> • [Opciones Generales] • Claves: • Opciones de idiomas • Administración de los suscriptores... • Opciones de entrega regular • Opciones de recopilaciones 	<ul style="list-style-type: none"> • Opciones de Privacidad... • Gestión de rebotes • Opciones de Almacenaje • Pasarela de Correo-e<->Noticias • Contestador automático • Filtrado de contenido • Temas 	<ul style="list-style-type: none"> • Ocuparse de las peticiones pendientes de moderar • Ir a la página de información general sobre la lista. • Editar el código HTML de las páginas de acceso público • Ir al archivo de la lista
		<ul style="list-style-type: none"> • Desconexión

a) **Opciones generales.** En esta categoría tenemos las opciones generales de configuración de una lista de correo (*anonymous_list*). Las de uso más habitual son:

- El texto descriptivo de la lista (*info*).
- Los textos anexos a los mensajes de alta y baja de usuarios en la lista (*welcome_msg*, *goodbye_msg*).
- La posibilidad de ocultar las direcciones de los usuarios para evitar el *spam* (*anonymous_list*).
- El filtro administrativo, que intercepta los mensajes de gestión de listas enviados a la lista en lugar de al administrador (*administrivia*).

- La longitud máxima aceptada de los mensajes, para evitar grandes ficheros anexos que acaban molestando a otros usuarios (*max_message_size*).
 - Desde hace poco tiempo, Mailman separa el rol de administrador del de moderador (*moderator*). Por tanto, podemos incluir en esta opción la lista de moderadores que recibirán las peticiones de moderación.
- b) **Claves.** En esta opción podemos cambiar nuestra clave y también la clave usada por los moderadores.
- c) **Administración de los suscriptores.** En esta opción podemos llevar un control de las suscripciones existentes y crear otras nuevas:
- En la lista de suscriptores: podemos cambiar cualquier opción de cualquiera de los suscriptores. Puede ser útil para volver a activar usuarios que hayan quedado desactivados por fallos de la cuenta de correo (¡atención!, porque también pueden haberse desactivado voluntariamente).
 - En suscripciones masivas: podemos pegar muchas direcciones de correo y ordenar su suscripción, o solamente su invitación. Se recomiendan solamente las invitaciones masivas, para que los usuarios decidan por sí mismos si quieren estar en la lista o no.
 - En bajas masivas: se pueden dar de baja varias direcciones a la vez.
- d) **Opciones de entrega regular.** Aquí ajustaremos opciones que afectan solamente a los mensajes que se entregan directamente a los usuarios, no por lotes (*digest*). Lo habitual en este apartado es personalizar el pie de los mensajes (*msg_footer*).
- e) **Opciones de recopilaciones.** Afectan solamente a los suscriptores que reciben los mensajes por lotes (*digest*). Podemos decidir el periodo de envío mínimo (diario, mensual, etc.) y el tamaño máximo (*digest_size_threshold*) del mensaje de recopilación (superado éste, se recibirá más de un mensaje).

f) **Opciones de privacidad.** Las más interesantes son:

- La política de suscripción (*subscribe_policy*), en la que estableceremos si se requiere la aprobación del administrador para aceptar suscriptores y si es necesaria la confirmación del solicitante. Para evitar suscripciones no deseadas, siempre debe activarse, como mínimo, la opción de confirmación.
- La visibilidad de la lista de suscriptores (*private_roster*), con la que podremos ocultar a los usuarios las direcciones del resto de suscriptores.
- Podemos gestionar los filtros de remitente, lo cual dará lugar a que la lista sea moderada. Como mínimo, debería activarse la opción de retener en la opción '*generic_nonmember_action*', para evitar que lleguen a la lista los mensajes de terceros (que en ocasiones, y si la lista se publicita mucho, serán *spam*). Por supuesto, no se descartan, sino que un moderador deberá revisar el mensaje para autorizarlo o no. Otro filtro interesante es activar la opción *default_member_moderation* para hacer la lista totalmente moderada.

Moderación

Cuando activemos la opción de moderación en alguna de las opciones de privacidad de una lista mostradas, o cuando algún usuario no autorizado envíe un mensaje a la lista, se enviará un aviso a los moderadores de la lista. Ya hemos indicado que los roles de administrador y moderador se han separado en versiones recientes de Mailman. Pero como, en realidad, cualquier administrador puede moderar, los administradores también recibirán los avisos de moderación.

Un aviso de moderación tendrá el aspecto siguiente:

From: nombre-de-lista-owner@servidor.midominio.com

To: nombre-de-lista-owner@servidor.midominio.com

Subject: El envío a nombre-de-lista de usuario@otrodominio.net precisa de aprobacion

Date: Sun, 02 Jan 2005 21:12:03 +0100

As list administrator, your authorization is requested for the following mailing list posting:

List: *nombre-de-lista@servidor.midominio.com*

From: *usuario@otrodominio.net*

Subject: Prueba de correo

Reason: Mensaje dirigido a una lista privada procedente de una dirección que no pertenece a la lista

At your convenience, visit:

<http://servidor.midominio.com/cgi-bin/mailman/admindb/nombre-de-lista>

to approve or deny the request.

En versiones más recientes de Mailman, se anexa además el mensaje completo recibido y unas instrucciones para moderar directamente, sin necesidad de utilizar la interfaz web. En este caso, basta con responder al mensaje para que Mailman entienda que se quiere descartar.

Para aceptar el mensaje, utilizaremos la interfaz web. Seleccionaremos la dirección indicada, se nos solicitará la contraseña de administrador o de moderador y se nos mostrará las solicitudes pendientes de moderación (figura 4-7).

Figura 4-7. Interfaz con la lista de solicitudes pendientes de moderación

Categorías de configuración		Otras actividades administrativas
<ul style="list-style-type: none"> • [Opciones Generales] • Claves: • Opciones de idiomas • Administración de los suscriptores... • Opciones de entrega regular • Opciones de recopilaciones 	<ul style="list-style-type: none"> • Opciones de Privacidad... • Gestión de rebotes • Opciones de Almacenaje • Pasarela de Correo-e<->Noticias • Contestador automático • Filtrado de contenido • Temas 	<ul style="list-style-type: none"> • Ocuparse de las peticiones pendientes de moderar • Ir a la página de información general sobre la lista. • Editar el código HTML de las páginas de acceso público • Ir al archivo de la lista <p>• Desconexión</p>

Una de las opciones más interesantes es la que permite descartar todos los mensajes pendientes, sobre todo cuando vemos que todo lo que hay pendiente es *spam*. Para ello, marcaremos la opción 'Discard all messages marked Defer' y pulsaremos el botón 'Enviar todos los datos'.

En caso contrario, marcaremos la casilla 'Aceptar' de los mensajes que deban ser publicados. Incluso aunque no esté suscrito, si es un remitente habitual de mensajes interesantes, puede interesarnos marcar 'Aceptar' en la opción 'Añadir *usuario@dominio* a uno de estos filtros de remitentes'.

Órdenes interesantes desde la línea de comandos

El administrador de listas de correo no necesita tener acceso de línea de comandos a la máquina que alberga las listas. Sin embargo, en ocasiones tiene acceso como administrador de la máquina, y en estos casos puede lanzar directamente algunas instrucciones interesantes:

- `add_members -r fichero nombre-de-lista`
Con este mandato puede añadir a la lista seleccionada las direcciones contenidas en el fichero.
- `remove_members -f fichero nombre-de-lista`
Este mandato permite, por el contrario, borrar de la lista indicada todos los suscriptores incluidos en el fichero.
- `genaliases`
Mandato que permite obtener los alias de todas las listas definidas que hay que introducir en el fichero `/etc/aliases`.
- `newlist` y `rmlist`, mandatos que ya habíamos visto para crear y eliminar listas de correo.

Para conocer el resto de órdenes a su disposición, el administrador puede echar un vistazo a los archivos de:

```
/usr/lib/mailman/bin
```

5. La gestión de un proyecto

Hasta el momento, hemos visto las herramientas que se utilizan generalmente en el desarrollo de software libre y nos hemos centrado en las más populares, que se han descrito más exhaustivamente. Sin embargo, al igual que el hecho de contar con un buen programa no basta para seguir unas prácticas que nos aseguren el éxito (sea cual sea la definición de éxito que más se ajuste a nuestra situación), tampoco lo garantiza el uso de estas herramientas, aunque, evidentemente, ambos elementos ayudan en gran medida.

En este capítulo vamos a ver las buenas maneras de proceder para alcanzar ese ansiado éxito. Por supuesto, damos por hecho que contamos con un programa cuyas prestaciones pueden resultar interesantes a la comunidad y con una infraestructura (ya sea propia o en un portal que ofrezca estos servicios) que permita el desarrollo colaborativo. Estamos ante una de las etapas decisivas del proyecto: su **liberación**.



Nuestro objetivo es crear una comunidad en torno al proyecto para que en algún momento el nivel de actividad llegue a ser tal que el desarrollo del proyecto sea autocatalítico, es decir, que la propia comunidad resuelva las necesidades que se plantea por sí misma.

Si ello sucede y nuestro proyecto tiene éxito, es probable que nuestra labor cambie radicalmente y pasemos a dedicar mucho tiempo y esfuerzo a gestionar todos estos recursos externos.

5.1. Elección de la licencia

En este momento del desarrollo hay que tomar una serie de decisiones adicionales, que son tan importantes como las de carácter técni-

Nota

Las herramientas de creación de proyectos de software de GNU (autoconf/automake) pueden generar-nos estos ficheros automáticamente.

co para el futuro éxito del proyecto. La primera decisión que debemos considerar es la elección de la licencia bajo la que se registrarán su distribución, uso, modificaciones e incluso las redistribuciones de nuestro software (incluida la documentación). Según nuestros objetivos, nos podrá interesar más un tipo de licencia robusta (o *copyleft*), que asegure que la naturaleza del software que creemos siempre va a ser libre; o permisiva (estas licencias también son conocidas como tipo BSD), si no nos importa que aparezcan versiones de nuestro trabajo bajo otras condiciones (incluso con licencias propietarias) en el futuro. Siempre se puede elegir una licencia dual, que como autores del software nos ofrece cierta flexibilidad. Por último, también existe la posibilidad de crear una licencia propia, aunque esto no es una tarea fácil.

En cualquier caso, los tipos de licencias y sus características son un tema que se escapa del ámbito de este material, por lo que remitimos al lector a [Gbarahona04]. Lo importante para nosotros en este punto es que, una vez elegida la licencia, nuestro software debería incluir su texto para que cualquiera pueda conocer las condiciones que rigen para ese software. Generalmente este archivo recibe el nombre de 'COPYING' o 'LICENSE'.

Asimismo, la licencia deberá constar, junto con toda la información de *copyright*, en todos los archivos de código fuente de que consta nuestro software (incluyendo la documentación, si elegimos la misma licencia para la documentación).

Ahora ya estamos en condiciones de poner nuestro software a disposición del público, para que comience el proceso de distribución. A partir de ahora deberemos distribuir y modificar el programa como si no fuera nuestro. En otras palabras, aunque tenemos el reconocimiento como autores del software, gracias a la licencia de software libre que le hemos atribuido, nadie posee *de facto* la propiedad sobre el software.

5.2. El sitio web del proyecto

Un aspecto de la infraestructura de un proyecto de software libre sobre el cual, de momento, hemos pasado de puntillas es su sitio web.

Junto con la instalación y la interfaz de usuario, el sitio web es el tercero de los elementos que proporcionan la primera impresión sobre nuestro proyecto.

El sitio web de un proyecto de software libre es fundamental. Los objetivos perseguidos con el sitio web en el desarrollo de un proyecto de software libre son varios. En primer lugar, debe proporcionar visibilidad, es decir, debe dar a conocer el proyecto a los usuarios y los desarrolladores. Además, el sitio web es fundamental para alcanzar una base de usuarios, colaboradores ocasionales y desarrolladores suficiente para la sostenibilidad del proyecto.

Así como las empresas e instituciones invierten una cantidad considerable de dinero en imagen corporativa, un proyecto de software libre también debe cuidar su imagen. En primer lugar, la web será el primer contacto de cualquier persona con el proyecto, por lo que debe tener una estructura adecuada y clara y contener la información necesaria para una primera toma de contacto con el proyecto. Evidentemente, la infraestructura dependerá del público objetivo de nuestra web: no es lo mismo una web destinada a aumentar la visibilidad de nuestro proyecto que una orientada a que otra gente se involucre en el mismo como codesarrolladores.

Para mantener un nivel de actividad suficiente en el proyecto, hay que procurar motivar la implicación de los usuarios. Para ello, se dispone de diversos medios, como foros o listas de correo de usuarios. Algún integrante del proyecto debería encargarse de su mantenimiento, para arreglar los problemas que pudieran aparecer. Además, la comunidad que forma el proyecto debería resolver las preguntas y dudas que se planteen en el foro y en las listas de correo.

En cualquier caso, es importante mantener la información actualizada. También puede disponerse información en la web del proyecto acerca de herramientas útiles que se pueden emplear junto con nuestro software o de proyectos relacionados.

Es muy importante que en la web haya disponible documentación sobre el proyecto, destinada tanto a usuarios como a desarrolladores. Sin esa documentación, difícilmente se van a implicar un usuario o un desarrollador en su desarrollo.

Ejemplo

Un sitio web cuya última noticia es de hace un par de años dará la sensación de inactividad, coloquialmente, de "proyecto muerto".

5.3. Estructura de la web del proyecto

En esta sección se propone una estructura para la web que cumpla todos los requisitos expuestos hasta ahora.

En primer lugar, debe haber una sección de **información del proyecto**. En esta sección debe incluirse una descripción del proyecto: en qué consiste, para qué sistemas está disponible, de qué versiones se dispone (por ejemplo, estable y en desarrollo), etc. Además, debe incluirse información acerca de las últimas versiones que han sido publicadas, con enlaces que lleven directamente a las páginas de esas versiones y, a ser posible, una lista de las últimas noticias del proyecto.

También es necesario que haya una sección de **descarga del software**, en la que estén enlazadas las diferentes versiones (por ejemplo, estable y en desarrollo), tanto en forma de paquetes binarios como en paquetes de código fuente. También puede ser interesante tener preparados paquetes para las distribuciones más comunes como Debian, RedHat o SUSE. Muchos proyectos ofrecen además la posibilidad de descargarse un paquete con la última versión del código fuente que hay en el CVS. Generalmente, esto se hace de manera automática mediante un guión que cada noche recopila lo que hay en el CVS y lo empaqueta debidamente.

Otra sección que no hay que descuidar es la de **últimas noticias y comunicados**, en la que deben incluirse todas las noticias relacionadas con el proyecto. Esta sección debe actualizarse regularmente para evitar la sensación de falta de actividad. Un proyecto activo siempre cuenta con un efecto de llamada para usuarios y desarrolladores, ya que da la impresión de que podrán resolver sus dudas o, si no está la funcionalidad específica que necesitan, esperar a que la contenga próximamente.

Una sección fundamental es la de **documentación**. La documentación puede estar dividida en dos grupos: la dirigida a usuarios y la orientada a desarrolladores. Además, es conveniente que esté disponible tanto en forma de paquetes para descargar, como para consultar en línea. En el caso de la documentación que se puede descargar, debe estar disponible en diversos formatos (por ejemplo, en HTML y PDF).

Entre la documentación también pueden incluirse ejemplos de uso, tutoriales, etc., que muestren el uso del software. También es interesante incluir una sección de consejos y trucos. Si nuestra aplicación tiene una interfaz gráfica, podemos añadir unas cuantas capturas de pantalla. Esto ayuda a que muchos usuarios se decidan a probarlo.

Por último, también puede resultar útil una sección que incluya herramientas relacionadas. Se puede añadir información sobre la infraestructura de apoyo de nuestro proyecto, indicar dónde están los foros de usuarios, qué listas de correo existen, dónde está el sistema de seguimiento de fallos o las instrucciones acerca de cómo el usuario debe emplear estos medios.

5.4. Instalación sencilla

Una de las reglas básicas que aprenderemos a apreciar con el tiempo es que debemos cuidar la instalación. Si el software es complejo de instalar, tiene montones de dependencias, es necesario compilar un par de bibliotecas antes que nuestro software, hay que modificar a mano el `makefile` en algunos casos, etc., la mayoría de los usuarios no llegarán a instalárselo. Es necesario eliminar todas estas barreras de entrada para los usuarios inexpertos. Por otra parte, a aquellos usuarios que se decidan a instalarlo les surgirán preguntas, y posiblemente errores, y deberemos dedicarnos a resolver dudas de instalación en vez de dedicar ese precioso tiempo a otros menesteres probablemente más productivos.

No hay que escatimar esfuerzos en este sentido, ya que cuando un usuario busca un software, su proceso de evaluación se suele limitar a la instalación de un producto y a un primer vistazo de su funcionalidad (si es en modo gráfico, a su interfaz). Un fallo en la instalación suele bastar para que el usuario deseche nuestro programa por muy bueno que éste sea. Y si ello ocurre con frecuencia, limitamos nuestras posibilidades, ya no sólo desde el punto de vista de número de usuarios, sino también desde la perspectiva de la realimentación en cuanto a informes de errores, documentación externa, o que puedan incluso llegar a ser codesarrolladores. Es por este motivo por lo que resulta casi imprescindible una documentación exhaustiva del pro-

Ejemplo

Las *demos* pueden consistir, por ejemplo, en tutoriales con una serie de ficheros de nuestro software para seguir los ejemplos que se desarrollan en el tutorial.

yecto, al menos por lo que se refiere a las instrucciones de compilación, instalación y empaquetamiento.

También es una buena idea proporcionar una serie de *demos*, es decir, archivos de demostración sobre lo que puede hacer nuestra aplicación. Sólo si se evitan estas barreras de entrada, podremos lograr una masa crítica de usuarios que posibilite que el desarrollo del proyecto sea sostenible.

Puesto que empezamos a intuir que estas tareas de distribución resultan más bien arduas, probablemente lo mejor es que lo hagan otros por nosotros. En el mundo del software libre existen canales que se encargan de ello de manera efectiva: las distribuciones. Un buen modo de que incluyan nuestro software es simplificar el proceso de compilación, instalación e incluso empaquetar nuestro software para que lo tomen tal cual. Existen herramientas como `autoconf` o `automake` —ésta muy extendidas en el mundo del software libre que facilitarán estas tareas.

5.5. Consejos para la distribución y difusión del software

En el mundo del software libre, los programas suelen seguir unas pautas de distribución y difusión comunes. Desde luego, no es obligatorio seguir estas pautas, pero si se hace, facilita mucho la integración de nuestro proyecto dentro de la comunidad de software libre. Por ejemplo, será más probable que llegue a formar parte de una distribución de GNU/Linux o que alguno de los codesarrolladores se dedique a crear los paquetes para su distribución.

En primer lugar, deberemos distribuir nuestro software de los modos más diversos posibles para que el usuario pueda elegir el que más le convenga. Entre ellos:

- a) Un paquete comprimido (generalmente `tar.gz`, aunque si es muy grande, nos podemos decantar por `tar.bz2`) con el código fuente, incluidos guiones y `makefiles` para compilar el programa fácilmente. Si nuestra aplicación se puede ejecutar sobre Windows,

también es una buena idea incluir el paquete comprimido en formato .zip, ya que su difusión es mayor en esa plataforma.

- b) Un paquete comprimido (`tar.gz`, `tar.bz2` o `zip`) con los binarios ya compilados.
- c) Un paquete Debian, para las distribuciones Debian y basadas en Debian como KNOPPIX, Ubuntu, etc.
- d) Un paquete RPM (*red hat package manager*), para las distribuciones que emplean este formato de paquetes; incluso se pueden ofrecer paquetes RPM especialmente contruidos para las distribuciones más populares (Fedora, SUSE, Mandrake).

La sección de descargas debe ser clara para que un usuario pueda encontrar fácilmente el software en el formato que le interesa, y descargárselo sólo en ese formato.

Los paquetes, tanto si contienen el código fuente como los binarios, deben incluir unos ficheros muy comunes en toda la distribución de software libre:

- El fichero 'README', con información general acerca del programa e información de contacto: URL del proyecto, dónde encontrar ayuda y documentación, etc. Algunos ficheros de este tipo incluyen también la secuencia de instalación.
- Un fichero con la licencia (generalmente denominado 'COPYING'), tal y como hemos visto en este capítulo.
- Otro fichero con las instrucciones de instalación ('INSTALL'). Si nuestro programa requiere otros para su correcto funcionamiento, dicha información se suele incluir en este fichero o en uno llamado 'DEPENDS'.
- Un archivo con la bitácora de cambios efectuados en el software ('CHANGELOG'). La información contenida en este fichero permitirá a los desarrolladores conocer los últimos cambios, así como dónde han sido efectuados.
- Existe la costumbre de tener un archivo llamado 'AUTHORS' (o similar) donde se encuentran listados los autores del programa. Se suele distinguir en estos casos entre aquellos que son los que lle-

van el proyecto y los desarrolladores que han participado de manera esporádica o puntual.

- Documentación de la aplicación, generalmente en un subdirectorio llamado 'doc'.
- Algunos binarios para diferentes arquitecturas; para asegurar una mayor compatibilidad con todos los sistemas posibles, es mejor que estén enlazados de manera estática con las bibliotecas que el proyecto utilice.
- En cualquier caso, el nombre del paquete y algunos de los ficheros deben dar información sobre la versión del programa.

Además, el paquete de código fuente debe incluir también un guión *configure* que genere el *makefile*, para poder compilar e instalar el programa fácilmente. Este método estándar de compilación también facilitará la creación de paquetes para las diferentes distribuciones y su posible integración dentro de éstas.

5.6. En busca del efecto en red

Pudiera parecer que hemos terminado con el proceso de liberación de nuestro proyecto y, sin embargo, no hemos hecho más que empezar. A partir de ahora hay que emplear un esfuerzo en absoluto despreciable para llevar a cabo una serie de acciones fundamentales para su éxito.



En resumen, en este punto se trata de maximizar la difusión de nuestro software entre la comunidad para incrementar la probabilidad de captar la atención de usuarios y posibles desarrolladores.

Conseguir usuarios y desarrolladores suele ser el primer paso para conseguir más usuarios y más desarrolladores, un fenómeno que ha venido a denominarse *efecto en red*. A continuación se presentan algunas de las prácticas comunes para lograr dicho objetivo.

Debemos dejar el software en algún sitio público donde pueda ser descargado, preferentemente mediante HTTP y/o FTP. Evidentemente, el código fuente también debería estar disponible en los mismos lugares y, como se describió en el capítulo dedicado al CVS, nunca está de más ofrecer la posibilidad de que la última versión se pueda obtener de manera anónima.

Para facilitar su máxima difusión, deberemos lanzar una especie de campaña publicitaria para dar a conocer nuestro programa. Nos valdremos tanto de la infraestructura propia del proyecto, como de los procesos de difusión de información de los que dispone la comunidad del software libre. Así, lo anunciaremos en nuestras listas de correo electrónico y en los foros de usuarios, dentro de nuestro propio proyecto. En general, es una buena idea dar de alta el proyecto en el portal Freshmeat, dedicado exclusivamente a informar de las últimas versiones de programas libres. Asimismo, si se consigue llegar a sitios de noticias populares, como Barrapunto, Libertonía o Slashdot, nuestro proyecto ganará en audiencia considerablemente.

También existe la posibilidad de promocionar nuestro software en jornadas y congresos de software libre que se celebran regularmente por todo el planeta. En estas ocasiones, es importante aprovechar el efecto de aparecer personalmente para convencer de que nuestro software funciona, por lo que no sólo se debería hacer una presentación con transparencias, sino que no se debería escatimar en enseñar, al menos durante algunos minutos, lo que éste puede hacer. Por otra parte, muchas de estas jornadas exigen que las ponencias y los textos se manden por escrito, por lo que al final de las mismas la presentación y la documentación generadas se podrán colgar en la web del proyecto.

Últimamente, también son muy populares entre los desarrolladores de software libre los *blogs*, donde se van detallando los pormenores del proceso de desarrollo del proyecto. Estos *blogs* suelen unirse en “planetas”, que aglutinan en una sola página web las entradas en diferentes bitácoras. Por tanto, hay que procurar que nuestro *blog* forme parte de algún planeta cuya temática esté relacionada con nuestro proyecto.

Nota

Freshmeat

<http://www.freshmeat.net>

Ejemplo

Así, a modo de ejemplo, si nuestro proyecto está escrito en el lenguaje de programación C# y hace uso de la plataforma Mono, nuestro *blog* puede unirse al Planeta Mono-Hispano (<http://planeta.monohispano.org/>), que es un sitio donde se concentra la comunidad de Mono-Hispano y que probablemente nos suponga beneficios en cuanto a realimentación, usuarios y publicidad.

En cualquier caso, cuando se generan estas noticias, hay que tener en cuenta dos principios básicos: nunca precipitarse, es decir, generar noticias sólo cuando el proyecto haya realmente alcanzado un hito; y evitar la publicidad excesiva, que puede llegar a identificar nuestros anuncios casi como *spam*.

5.7. Código e internacionalización

Una vez hayamos conseguido llamar la atención de algunos desarrolladores, nos será de gran interés tener las puertas abiertas a su colaboración. Es, por tanto, una buena idea mantener nuestro código limpio y estructurado, con todos los comentarios y los nombres de variables y funciones comprensibles. En muchos casos, sino en todos, ello significa utilizar el inglés.

Aun teniendo la infraestructura básica y el código en inglés, el software ha de estar preparado para que sea fácilmente adaptable a otros idiomas, en particular si el público objetivo de nuestro proyecto es el usuario de a pie. Es fundamental que las bibliotecas que empleemos en el desarrollo se puedan integrar fácilmente con las herramientas de traducción, como por ejemplo GNU gettext. Nuestra tarea no consiste sólo en traducir nuestra aplicación, lo que técnicamente se denomina *localización* (o **l10n**), sino que, además, es necesario modificar otros parámetros de carácter cultural como la moneda, los formatos de fecha, etc. Debemos proveer la denominada *internacionalización* (**i18n**) de la aplicación, que consiste precisamente en utilizar los medios técnicos para que la localización sea posible.

Nota

Las abreviaturas de *internacionalización* (**i18n**) y *localización* (**l10n**) corresponden a la primera y última letras de cada palabra y los caracteres de en medio se sustituyen por el número de letras desechadas.

En este punto también cuenta disponer de una infraestructura que favorezca la colaboración y que ya se ha descrito con detalle en los capítulos anteriores. Recordemos que es fundamental contar con listas de correo para la comunicación entre usuarios y desarrolladores, sistema de control de versiones para el trabajo coordinado de varios desarrolladores y sistema de seguimiento de fallos para la gestión de nuestros errores. También se pueden montar foros para los usuarios más inexpertos y una página de noticias relacionadas con el proyecto. Todo esto con el fin de que la información acerca del proyecto fluya lo máximo posible.

Cabe tener en cuenta que tener la infraestructura es solamente el primer paso, ya que la finalidad en sí es mimar a nuestros colaboradores, incluso aunque éstos sean esporádicos.



La parte más importante va a consistir en gestionar las contribuciones, los parches, las modificaciones y las sugerencias recibidas.

5.8. Esfuerzo dedicado a tareas de gestión

Durante todo este material, hemos indicado repetidamente que el modelo de desarrollo que seguimos no se limita solamente a programar el software y a tareas meramente técnicas. Tras describir exhaustivamente las diferentes herramientas que se suelen utilizar y las prácticas más recomendables para que nuestro proyecto gane en visibilidad dentro de la comunidad, estamos seguros de que el lector se habrá percatado de que las tareas de gestión se llevan una gran parte del tiempo dedicado a la generación de software libre.

En este apartado, nos gustaría mostrar cuánto tiempo suponen todas estas tareas. Para ello, nos referiremos a unas estimaciones proporcionadas por Brian Behlendorf [Behlendorf99], uno de los creadores de Apache. En su documento, Brian divide el esfuerzo en dos: la parte dedicada al lanzamiento del proyecto por un lado y el esfuerzo continuado que necesitamos dedicar a su gestión. El primer aspecto

viene dado en número de horas que habrá que dedicarle al comienzo del proyecto, mientras que las unidades del segundo corresponden a número de horas a la semana.

El primer punto que se analiza es el de la gestión de la infraestructura. Se necesitarán recursos humanos para que alguien ponga en pie las listas de correo electrónico, de los foros, el servidor web, el gestor de informes de fallos, el repositorio CVS, etc. No está muy claro si en esta tarea se incluye el diseño de la web y la creación de sus contenidos. El tiempo estimado para tener todo en pie es de unas 100 horas y el esfuerzo continuado se sitúa alrededor de 20 horas a la semana, lo que vendría a ser un trabajo a tiempo parcial. Cuando se escribió este artículo, no existían sitios de alojamiento especialmente diseñados para proyectos de software libre como los que hemos visto en el ejemplo de SourceForge, por lo que es probable que hoy en día –al menos para proyectos de tamaño pequeño y mediano– éste sea un esfuerzo que nos podamos ahorrar. Esto no ocurrirá en caso de tener un proyecto grande, ya que la flexibilidad que proporciona contar con una infraestructura propia hace rentable este esfuerzo.

La tarea de coordinación del desarrollo, incluido el control de calidad, es la que se requiere para limpiar el código y hacer que todo esté disponible para consumo público. Behlendorf estima que el tiempo que se ha de dedicar a esta tarea es de unas 40 a 200 horas en el momento del lanzamiento, mientras que la aplicación y supervisión de parches y de los *commits* en el CVS supone unas 20 horas semanales.

Otra tarea muy ligada a la anterior, pero que no toca el código fuente de la aplicación propiamente, es la gestión de los fallos que llegan al sistema de control de errores. Como se ha mostrado en el capítulo dedicado a los sistemas de control de errores, en particular a Bugzilla, la gestión de los errores es una tarea compleja que necesita de inteligencia humana para ser llevada a cabo convenientemente. El encargado de la atención de los informes de fallos ha de facilitar, en la medida de lo posible, la comunicación entre usuarios y desarrolladores, de modo que los primeros vean la resolución de los fallos de que informan, y los segundos tengan acceso rápido y ordenado a los fallos que deben corregir. Behlendorf no especifica el tiempo que hay que invertir en esta tarea al comienzo del proyecto –simple-

mente indica que el desarrollador responsable de dicha tarea debería familiarizarse con el código y las herramientas de control de errores–, pero añade: de 10 a 15 horas semanales durante la vida del proyecto.

Contar con buena documentación, tanto del código para los desarrolladores como del programa para los usuarios, se estima en unas 60 horas para la primera publicación, basándose, eso sí, en que no exista ninguna. A partir de ahí, hay que dedicar 10 horas semanales a actualizar esta información.

Finalmente, hay que añadir una componente de relaciones públicas que publicite el proyecto al máximo, como se ha visto en varios apartados de este mismo capítulo. En este caso tampoco se da una estimación para el comienzo del proyecto, pero sí que se dice que serían necesarias unas 20 horas a la semana para llevarlo a cabo.

Evidentemente, todas las estimaciones deben tomarse con cautela, ya que no se especifica el tamaño del proyecto. A partir del currículum del autor, podemos imaginarnos que se refiere a proyectos del tamaño de Apache, de tamaño medio en líneas de código (entre 50 mil y 100 mil líneas), pero con una amplia comunidad tanto de desarrolladores como de usuarios.

Si sumamos todos los tiempos estimados, obtenemos un valor de entre 200 y 360 horas para lanzar un proyecto de software libre y una dedicación semanal de 80 a 85 horas semanales. Como se ha indicado en su momento, en el caso del lanzamiento y gracias a los servicios ofrecidos por algunos portales, este tiempo puede minimizarse hasta el punto de que una gran parte del esfuerzo se resume en limpiar el código y añadir documentación. Por su parte, según las estimaciones con las que estamos trabajando, el mantenimiento del proyecto supondría tener a dos personas a tiempo completo dedicadas exclusivamente a la gestión de la infraestructura y el intercambio de información en el proyecto. Evidentemente, este tiempo no es de desarrollo propiamente dicho, sino de apoyo a nuestra “comunidad”, por lo que se presupone que lo que ganamos con este apoyo debería ser mayor que lo que pudieran ofrecer estas dos personas trabajando a tiempo completo específicamente en mejorar el programa.

5.9. Conclusiones

En este capítulo hemos dado varios consejos para el éxito de nuestro proyecto partiendo de la base de que ya contamos con una infraestructura de desarrollo que se ha ido presentando a lo largo de este texto. Vamos a repetir aquí los principios fundamentales que no podemos olvidar:

- Hay que atraer a los usuarios al sitio web de nuestro proyecto.
- Hay que mantener a los usuarios en el proyecto; para ello, es necesario sostener un nivel mínimo de actividad y de difusión de las novedades del proyecto.
- Si podemos crear una comunidad con usuarios y desarrolladores, el mantenimiento del proyecto será autocatalítico, y no será necesario que nosotros nos responsabilicemos del nivel de actividad del mismo, al menos de todo el proyecto en su conjunto.
- Los usuarios deben tener pocas dificultades para usar e instalar el software y participar en la comunidad del proyecto.
- Hay que lograr automatizar todo lo que podamos, porque esto facilitará la delegación de tareas en el futuro, y disminuirá la carga de trabajo de los administradores del proyecto.

Bibliografía

Behlendorf, Brian [Behlendorf99] "Open Source as a Bussiness Strategy", (en "Open Sources, Voices from the Open Source Revolution").
<http://www.oreilly.com/catalog/opensources>

Bugzilla Team, The [Bugzilla04] "The Bugzilla Guide - 2.18 Release".
<http://www.bugzilla.org/docs/2.18/html/>

Cederqvist, Per y otros [Cederqvist04] "Version Management with CVS".
<https://www.cvshome.org/docs/manual/>

Collins-Sussman, Ben; Fitzpatrick Brian W.; Pilato, Michael [Collins04] "Version Control with Subversion".
<http://svnbook.red-bean.com/>

Fogel, Karl; Bar, Moshe [Fogel00] "Open Source Development with CVS, 3rd Ed."
<http://cvsbook.red-bean.com/>

González Barahona, Jesús M. [Gbarahona04] "¿Y cómo hago para que mi código sea libre?" (en "Sobre Software Libre. Compilación de ensayos sobre software libre").
<http://gsync.escet.urjc.es/~grex/sobre-libre/>

Collar, Chistopher [Kollar00] "GNU Mailman List Manager Documentation v.2.0".
<http://staff.imsa.edu/~ckolar/mailman/>

Oda, T. [Oda03] "GNU Mailman-List Member Manual".
<http://www.gnu.org/software/mailman/mailman-member/index.html>

Vesperman, Jennifer [Vesperman03] "Essential CVS". (1.a edición, junio de 2003). ISBN: 0-596-00459-1. Editorial O'Reilly.

Villa, Luis [Villa03] "Large Free Software Projects and Bugzilla. Lessons from GNOME Project QA" (Proceedings of the Linux Symposium 2003)
<http://archive.linuxsymposium.org/ols2003/Proceedings/All-Reprints/Reprint-Villa-OLS2003.pdf>.

Warsaw, Barry [Warsaw04] "GNU Mailman - Installation Manual".
<http://www.gnu.org/software/mailman/mailman-install/index.html>

Appendix A. GNU Free Documentation License

Copyright (C) 2000,2001,2002 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

A.1. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of *copyleft* which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

A.2. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The *Document* below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as *you*. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A *Modified Version* of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A *Secondary Section* is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a *Secondary Section* may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The *Invariant Sections* are certain *Secondary Sections* whose titles are designated, as being those of *Invariant Sections*, in the notice that says that the Document is released under this License. If a section does not fit the above definition of *Secondary* then it is not allowed to be designated as *Invariant*. The Document may contain zero *Invariant Sections*. If the Document does not identify any *Invariant Sections* then there are none.

The *Cover Texts* are certain short passages of text that are listed, as *Front-Cover Texts* or *Back-Cover Texts*, in the notice that says that the Document is released under this License. A *Front-Cover Text* may be at most 5 words, and a *Back-Cover Text* may be at most 25 words.

A *Transparent* copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not *Transparent* is called *Opaque*.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The *Title Page* means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, *Title Page* means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section *Entitled XYZ* means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as *Acknowledgements*, *Dedications*, *Endorsements*, or *History*. To *Preserve the Title* of such a section when you modify the Document means that it remains a section *Entitled XYZ* according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

A.3. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

A.4. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

A.5. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified

Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled *History*. Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled *History* in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the *History* section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled *Acknowledgements* or *Dedications* Preserve the Title of the section, and preserve in the section all the subs-

tance and tone of each of the contributor acknowledgements and/or dedications given therein.

- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled *Endorsements* Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled *Endorsements* or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled *Endorsements*, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

A.6. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled *History* in the various original documents, forming one section Entitled *History*; likewise combine any sections Entitled *Acknowledgements*, and any sections Entitled *Dedications*. You must delete all sections Entitled *Endorsements*.

A.7. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

A.8. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an *aggregate* if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

A.9. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled *Acknowledgements*, *Dedications*, or *History*, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

A.10. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

A.11. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation 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. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License *or any later version* applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

A.12. ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the

GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled *GNU Free Documentation License*

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the *with...Texts.* line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Apéndice B. Licencia de Documentación Libre de GNU

Ésta es una traducción no oficial de la GNU Free Document License (GFDL), versión 1.2 a Español. No ha sido publicada por la Free Software Foundation y no establece legalmente los términos de distribución para trabajos que usen la GFDL (sólo el texto de la versión original en Inglés de la GFDL lo hace). Sin embargo, esperamos que esta traducción ayude los hispanohablantes a entender mejor la GFDL. La versión original de la GFDL esta disponible en la Free Software Foundation (<http://www.gnu.org/copyleft/fdl.html>).

Esta traducción está basada en una de la versión 1.1 de Igor Támara y Pablo Reyes. Sin embargo la responsabilidad de su interpretación es de Joaquín Seoane.

Copyright (C) 2000, 2001, 2002 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA. Se permite la copia y distribución de copias literales de este documento de licencia, pero no se permiten cambios.

B.1. PREÁMBULO

El propósito de esta Licencia es permitir que un manual, libro de texto, u otro documento escrito sea *libre* en el sentido de libertad: asegurar a todo el mundo la libertad efectiva de copiarlo y redistribuirlo, con o sin modificaciones, de manera comercial o no. En segundo término, esta Licencia proporciona al autor y al editor una manera de obtener reconocimiento por su trabajo, sin que se le considere responsable de las modificaciones realizadas por otros.

Esta Licencia es de tipo *copyleft*, lo que significa que los trabajos derivados del documento deben a su vez ser libres en el mismo sentido. Complementa la Licencia Pública General de GNU, que es una licencia tipo *copyleft* diseñada para el software libre.

Hemos diseñado esta Licencia para usarla en manuales de software libre, ya que el software libre necesita documentación libre: un programa libre debe venir con manuales que ofrezcan la mismas libertades que el software. Pero esta licencia no se limita a manuales de software; puede usarse para cualquier texto, sin tener en cuenta su temática o si se publica como libro impreso o no. Recomendamos esta licencia principalmente para trabajos cuyo fin sea instructivo o de referencia.

B.2. APLICABILIDAD Y DEFINICIONES

Esta Licencia se aplica a cualquier manual u otro trabajo, en cualquier soporte, que contenga una nota del propietario de los derechos de autor que indique que puede ser distribuido bajo los términos de esta Licencia. Tal nota garantiza en cualquier lugar del mundo, sin pago de derechos y sin límite de tiempo, el uso de dicho trabajo según las condiciones aquí estipuladas. En adelante la palabra *Documento* se referirá a cualquiera de dichos manuales o trabajos. Cualquier persona es un licenciataria y será referido como *Usted*. Usted acepta la licencia si copia, modifica o distribuye el trabajo de cualquier modo que requiera permiso según la ley de propiedad intelectual.

Una *Versión Modificada* del Documento significa cualquier trabajo que contenga el Documento o una porción del mismo, ya sea una copia literal o con modificaciones y/o traducciones a otro idioma.

Una *Sección Secundaria* es un apéndice con título o una sección preliminar del Documento que trata exclusivamente de la relación entre los autores o editores y el tema general del Documento (o temas relacionados) pero que no contiene nada que entre directamente en dicho tema general (por ejemplo, si el Documento es en parte un texto de matemáticas, una Sección Secundaria puede no explicar nada de matemáticas). La relación puede ser una conexión histórica con el tema o temas relacionados, o una opinión legal, comercial, filosófica, ética o política acerca de ellos.

Las *Secciones Invariantes* son ciertas Secciones Secundarias cuyos títulos son designados como Secciones Invariantes en la nota que

indica que el documento es liberado bajo esta Licencia. Si una sección no entra en la definición de Secundaria, no puede designarse como Invariante. El documento puede no tener Secciones Invariantes. Si el Documento no identifica las Secciones Invariantes, es que no las tiene.

Los *Textos de Cubierta* son ciertos pasajes cortos de texto que se listan como Textos de Cubierta Delantera o Textos de Cubierta Trasera en la nota que indica que el documento es liberado bajo esta Licencia. Un Texto de Cubierta Delantera puede tener como mucho 5 palabras, y uno de Cubierta Trasera puede tener hasta 25 palabras.

Una copia *Transparente* del Documento, significa una copia para lectura en máquina, representada en un formato cuya especificación está disponible al público en general, apto para que los contenidos puedan ser vistos y editados directamente con editores de texto genéricos o (para imágenes compuestas por puntos) con programas genéricos de manipulación de imágenes o (para dibujos) con algún editor de dibujos ampliamente disponible, y que sea adecuado como entrada para formateadores de texto o para su traducción automática a formatos adecuados para formateadores de texto. Una copia hecha en un formato definido como Transparente, pero cuyo marcaje o ausencia de él haya sido diseñado para impedir o dificultar modificaciones posteriores por parte de los lectores no es Transparente. Un formato de imagen no es Transparente si se usa para una cantidad de texto sustancial. Una copia que no es *Transparente* se denomina *Opaca*.

Como ejemplos de formatos adecuados para copias Transparentes están ASCII puro sin marcaje, formato de entrada de Texinfo, formato de entrada de LaTeX, SGML o XML usando una DTD disponible públicamente, y HTML, PostScript o PDF simples, que sigan los estándares y diseñados para que los modifiquen personas. Ejemplos de formatos de imagen transparentes son PNG, XCF y JPG. Los formatos Opacos incluyen formatos propietarios que pueden ser leídos y editados únicamente en procesadores de palabras propietarios, SGML o XML para los cuáles las DTD y/o herramientas de procesamiento no estén ampliamente disponibles, y HTML, PostScript o PDF generados por algunos procesadores de palabras sólo como salida.

La *Portada* significa, en un libro impreso, la página de título, más las páginas siguientes que sean necesarias para mantener legiblemente el material que esta Licencia requiere en la portada. Para trabajos en formatos que no tienen página de portada como tal, *Portada* significa el texto cercano a la aparición más prominente del título del trabajo, precediendo el comienzo del cuerpo del texto.

Una sección *Titulada XYZ* significa una parte del Documento cuyo título es precisamente XYZ o contiene XYZ entre paréntesis, a continuación de texto que traduce XYZ a otro idioma (aquí XYZ se refiere a nombres de sección específicos mencionados más abajo, como *Agradecimientos*, *Dedicatorias*, *Aprobaciones* o *Historia*. Conservar el *Título* de tal sección cuando se modifica el Documento significa que permanece una sección *Titulada XYZ* según esta definición.

El Documento puede incluir Limitaciones de Garantía cercanas a la nota donde se declara que al Documento se le aplica esta Licencia. Se considera que estas Limitaciones de Garantía están incluidas, por referencia, en la Licencia, pero sólo en cuanto a limitaciones de garantía: cualquier otra implicación que estas Limitaciones de Garantía puedan tener es nula y no tiene efecto en el significado de esta Licencia.

B.3. COPIA LITERAL

Usted puede copiar y distribuir el Documento en cualquier soporte, sea en forma comercial o no, siempre y cuando esta Licencia, las notas de copyright y la nota que indica que esta Licencia se aplica al Documento se reproduzcan en todas las copias y que usted no añada ninguna otra condición a las expuestas en esta Licencia. Usted no puede usar medidas técnicas para obstruir o controlar la lectura o copia posterior de las copias que usted haga o distribuya. Sin embargo, usted puede aceptar compensación a cambio de las copias. Si distribuye un número suficientemente grande de copias también deberá seguir las condiciones de la sección 3.

Usted también puede prestar copias, bajo las mismas condiciones establecidas anteriormente, y puede exhibir copias públicamente.

B.4. COPIADO EN CANTIDAD

Si publica copias impresas del Documento (o copias en soportes que tengan normalmente cubiertas impresas) que sobrepasen las 100, y la nota de licencia del Documento exige Textos de Cubierta, debe incluir las copias con cubiertas que lleven en forma clara y legible todos esos Textos de Cubierta: Textos de Cubierta Delantera en la cubierta delantera y Textos de Cubierta Trasera en la cubierta trasera. Ambas cubiertas deben identificarlo a Usted clara y legiblemente como editor de tales copias. La cubierta debe mostrar el título completo con todas las palabras igualmente prominentes y visibles. Además puede añadir otro material en las cubiertas. Las copias con cambios limitados a las cubiertas, siempre que conserven el título del Documento y satisfagan estas condiciones, pueden considerarse como copias literales.

Si los textos requeridos para la cubierta son muy voluminosos para que ajusten legiblemente, debe colocar los primeros (tantos como sea razonable colocar) en la verdadera cubierta y situar el resto en páginas adyacentes.

Si Usted publica o distribuye copias Opacas del Documento cuya cantidad exceda las 100, debe incluir una copia Transparente, que pueda ser leída por una máquina, con cada copia Opaca, o bien mostrar, en cada copia Opaca, una dirección de red donde cualquier usuario de la misma tenga acceso por medio de protocolos públicos y estandarizados a una copia Transparente del Documento completa, sin material adicional. Si usted hace uso de la última opción, deberá tomar las medidas necesarias, cuando comience la distribución de las copias Opacas en cantidad, para asegurar que esta copia Transparente permanecerá accesible en el sitio establecido por lo menos un año después de la última vez que distribuya una copia Opaca de esa edición al público (directamente o a través de sus agentes o distribuidores).

Se solicita, aunque no es requisito, que se ponga en contacto con los autores del Documento antes de redistribuir gran número de copias, para darles la oportunidad de que le proporcionen una versión actualizada del Documento.

B.5. MODIFICACIONES

Puede copiar y distribuir una Versión Modificada del Documento bajo las condiciones de las secciones 2 y 3 anteriores, siempre que usted libere la Versión Modificada bajo esta misma Licencia, con la Versión Modificada haciendo el rol del Documento, por lo tanto dando licencia de distribución y modificación de la Versión Modificada a quienquiera posea una copia de la misma. Además, debe hacer lo siguiente en la Versión Modificada:

- A. Usar en la Portada (y en las cubiertas, si hay alguna) un título distinto al del Documento y de sus versiones anteriores (que deberían, si hay alguna, estar listadas en la sección de Historia del Documento). Puede usar el mismo título de versiones anteriores al original siempre y cuando quien las publicó originalmente otorgue permiso.
- B. Listar en la Portada, como autores, una o más personas o entidades responsables de la autoría de las modificaciones de la Versión Modificada, junto con por lo menos cinco de los autores principales del Documento (todos sus autores principales, si hay menos de cinco), a menos que le eximan de tal requisito.
- C. Mostrar en la Portada como editor el nombre del editor de la Versión Modificada.
- D. Conservar todas las notas de copyright del Documento.
- E. Añadir una nota de copyright apropiada a sus modificaciones, adyacente a las otras notas de copyright.
- F. Incluir, inmediatamente después de las notas de copyright, una nota de licencia dando el permiso para usar la Versión Modificada bajo los términos de esta Licencia, como se muestra en la Adenda al final de este documento.
- G. Conservar en esa nota de licencia el listado completo de las Secciones Invariantes y de los Textos de Cubierta que sean requeridos en la nota de Licencia del Documento original.

- H. Incluir una copia sin modificación de esta Licencia.
- I. Conservar la sección Titulada *Historia*, conservar su Título y añadirle un elemento que declare al menos el título, el año, los nuevos autores y el editor de la Versión Modificada, tal como figuran en la Portada. Si no hay una sección Titulada *Historia* en el Documento, crear una estableciendo el título, el año, los autores y el editor del Documento, tal como figuran en su Portada, añadiendo además un elemento describiendo la Versión Modificada, como se estableció en la oración anterior.
- J. Conservar la dirección en red, si la hay, dada en el Documento para el acceso público a una copia Transparente del mismo, así como las otras direcciones de red dadas en el Documento para versiones anteriores en las que estuviese basado. Pueden ubicarse en la sección *Historia*. Se puede omitir la ubicación en red de un trabajo que haya sido publicado por lo menos cuatro años antes que el Documento mismo, o si el editor original de dicha versión da permiso.
- K. En cualquier sección Titulada *Agradecimientos* o *Dedicatorias*, Conservar el Título de la sección y conservar en ella toda la sustancia y el tono de los agradecimientos y/o dedicatorias incluidas por cada contribuyente.
- L. Conservar todas las Secciones Invariantes del Documento, sin alterar su texto ni sus títulos. Números de sección o el equivalente no son considerados parte de los títulos de la sección.
- M. Borrar cualquier sección titulada *Aprobaciones*. Tales secciones no pueden estar incluidas en las Versiones Modificadas.
- N. No cambiar el título de ninguna sección existente a *Aprobaciones* ni a uno que entre en conflicto con el de alguna Sección Invariante.
- O. Conservar todas las Limitaciones de Garantía.

Si la Versión Modificada incluye secciones o apéndices nuevos que califiquen como Secciones Secundarias y contienen material no co-

piado del Documento, puede opcionalmente designar algunas o todas esas secciones como invariantes. Para hacerlo, añada sus títulos a la lista de Secciones Invariantes en la nota de licencia de la Versión Modificada. Tales títulos deben ser distintos de cualquier otro título de sección.

Puede añadir una sección titulada *Aprobaciones*, siempre que contenga únicamente aprobaciones de su Versión Modificada por otras fuentes –por ejemplo, observaciones de peritos o que el texto ha sido aprobado por una organización como la definición oficial de un estándar.

Puede añadir un pasaje de hasta cinco palabras como Texto de Cubierta Delantera y un pasaje de hasta 25 palabras como Texto de Cubierta Trasera en la Versión Modificada. Una entidad sólo puede añadir (o hacer que se añada) un pasaje al Texto de Cubierta Delantera y uno al de Cubierta Trasera. Si el Documento ya incluye un texto de cubiertas añadidos previamente por usted o por la misma entidad que usted representa, usted no puede añadir otro; pero puede reemplazar el anterior, con permiso explícito del editor que agregó el texto anterior.

Con esta Licencia ni los autores ni los editores del Documento dan permiso para usar sus nombres para publicidad ni para asegurar o implicar aprobación de cualquier Versión Modificada.

B.6. COMBINACIÓN DE DOCUMENTOS

Usted puede combinar el Documento con otros documentos liberados bajo esta Licencia, bajo los términos definidos en la sección 4 anterior para versiones modificadas, siempre que incluya en la combinación todas las Secciones Invariantes de todos los documentos originales, sin modificar, listadas todas como Secciones Invariantes del trabajo combinado en su nota de licencia. Así mismo debe incluir la Limitación de Garantía.

El trabajo combinado necesita contener solamente una copia de esta Licencia, y puede reemplazar varias Secciones Invariantes idénticas

por una sola copia. Si hay varias Secciones Invariantes con el mismo nombre pero con contenidos diferentes, haga el título único de cada una de estas secciones añadiéndole al final del mismo, entre paréntesis, el nombre del autor o editor original de esa sección, si es conocido, o si no, un número único. Haga el mismo ajuste a los títulos de sección en la lista de Secciones Invariantes de la nota de licencia del trabajo combinado.

En la combinación, debe combinar cualquier sección Titulada *Historia* de los documentos originales, formando una sección Titulada *Historia*; de la misma forma combine cualquier sección Titulada *Agradecimientos*, y cualquier sección Titulada *Dedicatorias*. Debe borrar todas las secciones tituladas *Aprobaciones*.

B.7. COLECCIONES DE DOCUMENTOS

Puede hacer una colección que conste del Documento y de otros documentos liberados bajo esta Licencia, y reemplazar las copias individuales de esta Licencia en todos los documentos por una sola copia que esté incluida en la colección, siempre que siga las reglas de esta Licencia para cada copia literal de cada uno de los documentos en cualquiera de los demás aspectos.

Puede extraer un solo documento de una de tales colecciones y distribuirlo individualmente bajo esta Licencia, siempre que inserte una copia de esta Licencia en el documento extraído, y siga esta Licencia en todos los demás aspectos relativos a la copia literal de dicho documento.

B.8. AGREGACIÓN CON TRABAJOS INDEPENDIENTES

Una recopilación que conste del Documento o sus derivados y de otros documentos o trabajos separados e independientes, en cualquier soporte de almacenamiento o distribución, se denomina un *agregado* si el copyright resultante de la compilación no se usa para limitar los derechos de los usuarios de la misma más allá de lo que

los de los trabajos individuales permiten. Cuando el Documento se incluye en un agregado, esta Licencia no se aplica a otros trabajos del agregado que no sean en sí mismos derivados del Documento.

Si el requisito de la sección 3 sobre el Texto de Cubierta es aplicable a estas copias del Documento y el Documento es menor que la mitad del agregado entero, los Textos de Cubierta del Documento pueden colocarse en cubiertas que enmarquen solamente el Documento dentro del agregado, o el equivalente electrónico de las cubiertas si el documento está en forma electrónica. En caso contrario deben aparecer en cubiertas impresas enmarcando todo el agregado.

B.9. TRADUCCIÓN

La Traducción es considerada como un tipo de modificación, por lo que usted puede distribuir traducciones del Documento bajo los términos de la sección 4. El reemplazo las Secciones Invariantes con traducciones requiere permiso especial de los dueños de derecho de autor, pero usted puede añadir traducciones de algunas o todas las Secciones Invariantes a las versiones originales de las mismas. Puede incluir una traducción de esta Licencia, de todas las notas de licencia del documento, así como de las Limitaciones de Garantía, siempre que incluya también la versión en Inglés de esta Licencia y las versiones originales de las notas de licencia y Limitaciones de Garantía. En caso de desacuerdo entre la traducción y la versión original en Inglés de esta Licencia, la nota de licencia o la limitación de garantía, la versión original en Inglés prevalecerá.

Si una sección del Documento está Titulada *Agradecimientos*, *Dedicatorias* o *Historia* el requisito (sección 4) de Conservar su Título (Sección 1) requerirá, típicamente, cambiar su título.

B.10. TERMINACIÓN

Usted no puede copiar, modificar, sublicenciar o distribuir el Documento salvo por lo permitido expresamente por esta Licencia. Cualquier otro intento de copia, modificación, sublicenciamiento o

distribución del Documento es nulo, y dará por terminados automáticamente sus derechos bajo esa Licencia. Sin embargo, los terceros que hayan recibido copias, o derechos, de usted bajo esta Licencia no verán terminadas sus licencias, siempre que permanezcan en total conformidad con ella.

B.11. REVISIONES FUTURAS DE ESTA LICENCIA

De vez en cuando la Free Software Foundation puede publicar versiones nuevas y revisadas de la Licencia de Documentación Libre GNU. Tales versiones nuevas serán similares en espíritu a la presente versión, pero pueden diferir en detalles para solucionar nuevos problemas o intereses. Vea <http://www.gnu.org/copyleft/>.

Cada versión de la Licencia tiene un número de versión que la distingue. Si el Documento especifica que se aplica una versión numerada en particular de esta licencia o *cualquier versión posterior*, usted tiene la opción de seguir los términos y condiciones de la versión especificada o cualquiera posterior que haya sido publicada (no como borrador) por la Free Software Foundation. Si el Documento no especifica un número de versión de esta Licencia, puede escoger cualquier versión que haya sido publicada (no como borrador) por la Free Software Foundation.

B.12. ADENDA: Cómo usar esta Licencia en sus documentos

Para usar esta licencia en un documento que usted haya escrito, incluya una copia de la Licencia en el documento y ponga el siguiente copyright y nota de licencia justo después de la página de título:

Copyright (c) AÑO SU NOMBRE. Se otorga permiso para copiar, distribuir y/o modificar este documento bajo los términos de la Licencia de Documentación Libre de GNU, Versión 1.2 o cualquier otra versión posterior publicada por la Free Software Foundation; sin Secciones Invariantes ni Textos de Cubierta Delantera ni Textos de Cubierta

Trasera. Una copia de la licencia está incluida en la sección titulada *GNU Free Documentation License*.

Si tiene Secciones Invariantes, Textos de Cubierta Delantera y Textos de Cubierta Trasera, reemplace la frase *sin ... Trasera* por esto:

siendo las Secciones Invariantes LISTE SUS TÍTULOS, siendo los Textos de Cubierta Delantera LISTAR, y siendo sus Textos de Cubierta Trasera LISTAR.

Si tiene Secciones Invariantes sin Textos de Cubierta o cualquier otra combinación de los tres, mezcle ambas alternativas para adaptarse a la situación.

Si su documento contiene ejemplos de código de programa no triviales, recomendamos liberar estos ejemplos en paralelo bajo la licencia de software libre que usted elija, como la Licencia Pública General de GNU (*GNU General Public License*), para permitir su uso en software libre.

