# Historia del juguete: pasado, presente y futuro de Debian

# Table of Contents

# Historia del juguete: pasado, presente y futuro de Debian


## Gregorio Robles

Grupo de Sistemas y Comunicaciones − Universidad Rey Juan Carlos

<grex@gsyc.escet.urjc.es>

## Jesús M. González Barahona

Grupo de Sistemas y Comunicaciones − Universidad Rey Juan Carlos

<jgb@debian.org> <jgb@gsyc.escet.urjc.es>

## Luis Rodero Merino

Grupo de Sistemas y Comunicaciones − Universidad Rey Juan Carlos

<lrodero@gsyc.escet.urjc.es>

## Miguel A. Ortuño Pérez

Grupo de Sistemas y Comunicaciones − Universidad Rey Juan Carlos

<mortuno@gsyc.escet.urjc.es>





**Resumen**

El sistema operativo Debian es una de las distribuciones de GNU/Linux más populares a día de hoy. Desde su nacimiento, hace una década, ha sufrido muchos cambios técnicos, estructurales y organizativos. Este artículo pretende estudiar la evolución de Debian en los últimos cinco años, comparando las últimas cuatro versiones estables de esta distribución en tamaño, lenguajes de programación y paquetes. También se analiza la evolución del número de desarrolladores voluntarios en el proyecto Debian y se realiza una estimación del esfuerzo en términos humanos y económicos que sería necesario para producir un software de ese tamaño. Las evidencias principales que hemos encontrado es que Debian dobla el tamaño en líneas de código y número de paquetes aproximadamente cada dos años, mientras que el tamaño medio de los paquetes permanece constante. Una gran mayoría de los paquetes de la primera versión considerada en este estudio han "sobrevivido" al paso del tiempo y se encuentran presentes en versiones más modernas de Debian, muchos de ellos incluso con el mismo número de versión de paquete. El lenguaje de programación más utilizado es C, aunque su importancia vaya decreciendo con el tiempo. Para concluir, se utilizan los resultados de la evolución de Debian durante estos últimos cinco años para realizar una predicción de cómo podría ser la próxima versión estable de Debian.


---

**Tabla de contenidos**

# 1. Introducción

A principios de los noventa, las primeras distribuciones vinieron de la mano de la unión de las herramientas GNU con el núcleo Linux. Su finalidad era facilitar en la medida de lo posible la instalación de herramientas libres, una tarea ardua y que requería una gran paciencia, llegando a veces a ser incluso artesanal. El segundo hito de las distribuciones −ya avanzada la década de los noventa− se debe a los sistemas de gestión de paquetes que permitían no sólo instalar una distribución en el disco duro del usuario de manera sencilla, sino además gestionar los paquetes una vez instalados.

Las distribuciones ocuparon, en consecuencia, un espacio que en el mundo del software propietario rara vez alcanza proporciones importantes: los integradores. Su trabajo consiste en tomar las fuentes −generalmente de sus autores originales−, agruparlo con otras herramientas y aplicaciones que pudieran ser interesantes y empaquetarlo de tal manera que el usuario final vea facilitada la tarea de instalar o actualizar cantidades ingentes de paquetes sin que el sistema en su conjunto se resienta.

Las organizaciones y empresas que crean distribuciones son asimismo responsables de la calidad del producto final, una tarea muy importante si tenemos en cuenta que la mayoría de los proyectos de software libre están gestionado por voluntarios [Michlmayr2003]. En este sentido, son responsables ante sus usuarios de la estabilidad y seguridad de la distribución resultante. A raíz de todas estas situaciones, no es difícil imaginarse por qué las distribuciones pronto ocuparon un lugar importante en cuanto a la popularización del software libre en general y de los sistemas GNU/Linux en particular.

Existen multitud de distribuciones diferentes, cada una con sus propias peculiaridades. Entre las diferencias más notables podemos nombrar su carácter comercial (algunas tienen empresas detrás), su tamaño en cuanto al número de paquetes que incorporan, su estrategia de publicación de nuevas versiones, etc. De entre todas ellas, este estudio se va a centrar en una distribución particular, aunque bastante extendida y muy popular: Debian.

Este artículo muestra los resultados más interesantes de manera general y, en muchas ocasiones, sin entrar en detalle. Al lector interesado le sugerimos que visite la página web donde encontrará más información, gráficas y más datos estadísticos [DebianCounting]. Asimismo, en [Libresoft] encontrará más artículos e información sobre la ingeniería del software libre, la rama de la ingeniería del software en la que se clasifican este tipo de estudios.

# 2. Sobre Debian

Debian es un sistema operativo libre que en la actualidad utiliza el núcleo de Linux para llevar a cabo su distribución (aunque se espera que existan distribuciones Debian basadas en otros núcleos, como es el caso con The HURD, en el futuro). Actualmente está disponible para varias arquitecturas diferentes, incluyendo Intel x86, ARM, Motorola, 680x0, PowerPC, Alpha y SPARC.

Debian no es sólo la distribución GNU/Linux más grande en la actualidad, también es una de las más estables y disfruta de varios premios en cuanto a la preferencia de los usuarios. Aunque su base de usuarios sea difícil de estimar, ya que el proyecto Debian no vende CDs u otros medios con su software y el software que contiene puede ser redistribuido por cualquier que así lo desea, podemos suponer sin faltar mucho a la verdad que se trata de una distribución importante dentro del mercado de GNU/Linux.

En Debian existe una categorización según la licencia y los requisitos de distribución de los paquetes. El núcleo de la distribución Debian (la sección llamada "main" que aglutina una gran variedad de paquetes) está compuesto sólo por software libre de acuerdo con las [DFSG] (Debian Free Software Guidelines). Está

disponible en Internet para ser descargado y muchos redistribuidores lo venden en CDs u otros medios.

Las distribuciones de Debian son creadas por cerca de un millar de voluntarios (generalmente profesionales de la informática). La labor de estos voluntarios radica en tomar los programas fuente −en la mayoría de los casos de sus autores originales−, configurarlos, compilarlos y empaquetarlos, de manera que un usuario típico de una distribución Debian sólo tenga que seleccionar el paquete para que el sistema lo añada sin mayores problemas. Esto que a simple vista puede parecer simple, se torna complejo en cuanto se introducen factores como las dependencias entre los diferentes paquetes (el paquete A necesita, para poder funcionar, del paquete B) y las diferentes versiones de todos estos paquetes.

La labor de los integrantes del proyecto Debian es la misma que la que se realiza en cualquier otra distribución: la integración de software para su correcto funcionamiento conjunto. Además del trabajo de adaptación y empaquetamiento, los desarrolladores Debian se encargan de mantener una infraestructura de servicios basados en Internet (sitio web, archivos en línea, sistema de gestión de errores, listas de correo de ayuda, soporte y desarrollo, etc.), de varios proyectos de traducción e internacionalización, del desarrollo de varias herramientas específicas de Debian y, en general, de cualquier elemento que hace la distribución Debian posible.

Aparte de su naturaleza voluntaria, el proyecto Debian tiene una característica que lo hace especialmente singular: el contrato social de Debian [DebianSocialContract]. Este documento contiene no sólo los objetivos principales del proyecto Debian, sino también los medios que se utilizarán para llevarlos a cabo.

Debian también es conocida por tener una política de paquetes y de versionado muy estricta con el fin de conseguir una mayor calidad del producto [DebianPol]. Así, en todo momento existen tres "sabores" diferentes de Debian: una versión estable, una inestable y otra en pruebas. Como su propio nombre indica, la versión estable es la versión indicada para sistemas y personas no aptas a sobresaltos. Su software ha de pasar un periodo de congelación en el que sólo se corrigen erratas. La norma es que en la versión estable de Debian no ha de haber ningún error crítico conocido. Por contra, la versión estable de Debian no suele tener las últimas versiones del software (lo más novedoso).

Para los que deseen tener una versión con el software más actual existen otras dos versiones de Debian coetáneas con la estable. La versión en pruebas incluye paquetes en vía de estabilización, mientras que la versión inestable, como su propio nombre indica, es la más proclive a fallar y contiene lo último de lo último en lo que a novedades de software se refiere.

En el momento de este estudio, la versión estable de Debian es Debian 3.0 (también conocida como "Woody"), la inestable recibe el sobrenombre de "Sid" y la que se encuentra en pruebas es "Sarge". Pero en el pasado, Woody pasó también por una etapa inestable y, antes de eso, otra en pruebas. Esto es importante, porque lo que vamos a considerar en este artículo son las diferentes versiones estables de Debian, desde que se publicara la versión 2.0 allá por 1998. Así, tenemos a Debian 2.0 (alias "Hamm"), Debian 2.1 ("Slink"), Debian 2.2 ("Potato") y, por último, Debian 3.0 ("Woody").

Los apodos de las versiones de Debian corresponden a los protagonistas de la película de dibujos animados "Toy Story", una tradición que se implantó medio en serio, medio en broma cuando se publicó la versión 2.0 y Bruce Perens, entonces líder del proyecto y después fundador de la Open Source Initiative y del término Open Source, trabajaba para la empresa que se encargaba de realizar esta película. Se pueden encontrar más detalles sobre la historia de Debian y la distribución Debian en general en [DebianHistory].

## 3. Metodología del estudio

La metodología que hemos utilizado para el análisis de las versiones estables de Debian es muy simple. Primero se descargan todos los paquetes que las componen. Para cada paquete se cuenta el número de líneas de código fuente que contiene y el lenguaje de programación en el que está escrito ese código.

La cuenta se realiza mediante una herramienta llamada SLOCCount [SLOCCount]. SLOCCount toma como entrada un directorio donde se encuentran las fuentes, identifica mediante una serie de heurísticos los ficheros que contienen código fuente, mediante otros heurísticos identifica el lenguaje de programación en el que están

escritos y finalmente se dedica a contar el número de líneas de código fuente que contienen. Como veremos más adelante en la definición formal de las líneas de código fuente, éstas no incluyen ni comentarios ni líneas en blanco, por lo que la identificación del lenguaje de programación en el que está escrito un fichero se hace imprescindible habida cuenta de que la sintaxis de los comentarios difiere entre lenguajes.

Otra de las tareas que realiza SLOCCount, aunque de manera bastante primitiva, es la identificación de ficheros idénticos y de código generado automáticamente. Para lo primero cuenta con una base de datos de hashes de los ficheros, que se comparan dos a dos para ver si son idénticos, mientras que para lo segundo establece otra serie de heurísticos mediante los cuales pretende encontrar ficheros generados de manera automática. Sin duda, estos mecanismos tienen notables carencias: encontrar ficheros idénticos con ligeras modificaciones (p.ej. el identificador automático incluido para el CVS) mediante el uso de hashes se demuestra poco eficaz, mientras que los heurísticos sólo atienden a casos conocidos y comunes, pero no por ello ni a todos los existentes ni a otros que se puedan dar en el futuro.

Los resultados del análisis de SLOCCount se transforman posteriormente a un formato XML que permite su fácil visualización, manipulación y transformaciones a otros formatos. Entre las transformaciones más interesantes encontramos la de pasar los datos a SQL e introducirlos en una base de datos. Entonces, mediante un simple interfaz web cualquiera puede tener acceso a los datos en crudo e incluso a otros ya más elaborados que faciliten un primer análisis. El grupo de investigación que ha llevado a cabo este estudio ofrece, en consecuencia, un portal web donde se podrán encontrar pormenorizadamente todos los datos, estadísticas y gráficas mostradas en este estudio ([DebianCounting]).

Se puede encontrar una descripción más detallada de la metodología utilizada, así como sus principales causas de error en [GBarahona2001] y [GBarahona2003].

# 4. Evolución de los desarrolladores de Debian

Desde junio de 1999, Debian lleva una base de datos [DBDebian] con datos relacionados con los integrantes del proyecto, de manera que se facilite la comunicación con y entre los mismos. Los datos que contiene son datos tales como el nombre, el nombre de usuario, la dirección de correo y la clave PGP/GPG. Además, incluye datos sobre el país de residencia −interesante para conocer desarrolladores Debian cercanos− y la fecha de ingreso en el proyecto (si ésta es posterior a la fecha de creación de la base de datos). Para este artículo, hemos tomado algunos de estos datos y los hemos procesado convenientemente para preservar por una parte el anonimato de los desarrolladores y por otra conseguir información sobre la evolución del número de desarrolladores y de los países en los que residen. En [Robles2001] ya se hizo uso de estos datos para realizar un estudio similar al que se va a presentar a continuación.

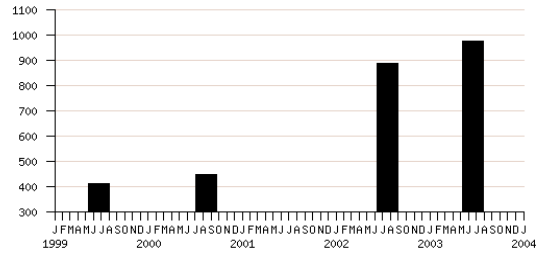
En la figura Figura 1. Número de desarrolladores Debian en los momentos de publicación de versiones estable podemos ver el número de desarrolladores de Debian en los momentos de publicación de una nueva versión estable. Se puede observar que entre las versiones 2.1 y 2.2 (para la 2.0 no tenemos datos) existe un ligero crecimiento, que se acentúa notablemente en el espacio de tiempo existente entre Debian 2.2 y Debian 3.0. En esos dos años, el número de desarrolladores de Debian se dobla. La última columna corresponde al número de desarrolladores contabilizados actualmente en la base datos de Debian. Podemos ver cómo el proyecto Debian sigue creciendo a buen ritmo, aunque no tan firmemente como en el tiempo entre Debian 2.2 y Debian 3.0.

También hemos incluido una figura en la que podemos ver la adscripción de nuevos miembros por semana al proyecto Debian. Para ello, como se ha comentado anteriormente, sólo contamos con datos a partir del 21 de junio de 1999. Lo primero y más sorprendente, es observar un periodo de congelación en cuanto al número de desarrolladores que se extiende desde junio de 1999 (o quizás antes, ya que no contamos con datos anteriores) hasta marzo del año 2000. Este parón se puede explicar por un cambio de política en cuanto a los requisitos de entrada en el proyecto. Parece ser que había gente que había entrado sin conocer, entender o estar de acuerdo con las líneas filosóficas de Debian recogidas en [DebianSocialContract], por lo que las discusiones se tornaron insoportables. Los integrantes del proyecto decidieron entonces que debían poner en práctica un mecanismo para evitar estos casos en el futuro y mientras tanto no se admitieron más desarrolladores.

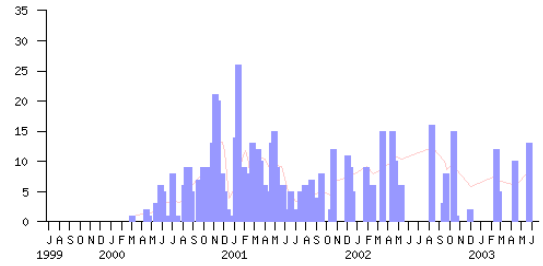
Una vez que el proceso de admisión se volvió a abrir, el número de desarrolladores Debian creció sin parar y a buen ritmo durante el resto del año 2000 y 2001, hasta que a mediados de 2002 las incorporaciones parecen

ralentizarse de manera ostensible. En enero de 2001 encontramos el pico de incorporaciones con 26 incorporaciones en una semana. Por otra parte, el hecho de que parezca que los nuevos desarrolladores entren en tandas a partir de mediados de 2002 puede ser debido a que la base de datos se actualice de manera periódica y no continuamente.

**Figura 1. Número de desarrolladores Debian en los momentos de publicación de versiones estable**




Número de desarrolladores Debian en los momentos de publicación




Nuevos desarrolladores que se integran en el proyecto Debian

```
La figura de la izquierda muestra el número de desarrolladores en las
fechas de publicación de las distribuciones, mientras que en la
de la derecha se puede ver el número de desarrolladores nuevos que se
integran en el proyecto a lo largo del tiempo.
```

En la tabla Tabla 1. Países con mayor número de desarrolladores de Debian se puede ver la distribución de los desarrolladores Debian según los países de residencia y a lo largo del tiempo para los 11 países que cuentan con más desarrolladores. Se caen de la tabla los 36 países restantes que también cuentan a día de hoy con al menos un voluntario en Debian. Se puede observar una tendencia a la descentralización del proyecto, algo que se constata en el hecho de que el crecimiento de los desarrolladores en Estados Unidos −el país que más aporta− es inferior a la media. Y es que, por lo general, los países han conseguido doblar el número de voluntarios en los últimos cuatro años, siendo Francia el ejemplo más claro en este sentido, ya que ha conseguido multiplicar por cinco su presencia. Considerando que los primeros pasos de Debian tuvieron lugar en el continente americano (en particular en Estados Unidos y Canadá), podemos ver que en los últimos cuatro años ha sufrido una "europeización " del proyecto. Suponemos que el siguiente paso será la ansiada mundialización con la incorporación de países sudamericanos, africanos y asiáticos (exceptuando Corea y Japón, ya bien representadas), aunque los datos que manejamos (2 desarrolladores en Egipto, China e India, 1 en México, Turquía y Colombia en junio de 2003) no son muy halagüeños en este sentido.

**Tabla 1. Países con mayor número de desarrolladores de Debian**

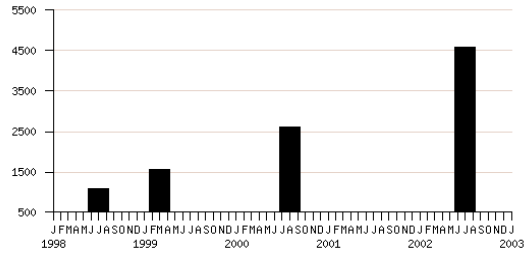
| **País** | **1.7.1999** | **1.7.2000** | **1.7.2001** | **1.7.2002** | **20.6.2003** |
|---|---|---|---|---|---|
| Estados Unidos | 162 | 169 | 256 | 278 | 297 |
| Alemania | 54 | 58 | 101 | 121 | 136 |
| Reino Unido | 34 | 34 | 55 | 63 | 75 |

| | | | | | |
|---|---|---|---|---|---|
| Australia | 23 | 26 | 41 | 49 | 52 |
| Francia | 11 | 11 | 24 | 44 | 51 |
| Canadá | 20 | 22 | 41 | 47 | 49 |
| España | 10 | 11 | 25 | 31 | 34 |
| Japón | 15 | 15 | 27 | 33 | 33 |
| Italia | 9 | 9 | 22 | 26 | 31 |
| Países Bajos | 14 | 14 | 27 | 29 | 29 |
| Suecia | 13 | 13 | 20 | 24 | 27 |

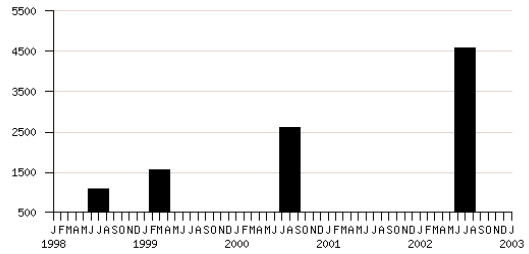
# 5. Líneas de código fuente (SLOC) físicas

El número de líneas de código fuente físicas es una de las técnicas usadas comúnmente para comparar software. A partir de las mismas se pueden usar métodos establecidos para la estimación de esfuerzo y temporización óptima (como es el caso de COCOMO). La definición de una línea de código física en este contexto se define como "una línea que termina en una marca de línea nueva o una marca de final de fichero, y que contiene al menos un carácter que no es un espacio en blanco ni comentario" El acrónimo de la unidad de línea de código fuente física es SLOC (del inglés "Source Line of Code"), aunque el uso en KSLOC es más común. Debido al tamaño del software que consideramos en este artículo, se hace necesaria en ocasiones la utilización de la medida en millones de líneas de código, MSLOC. En la tabla Figura 2. Tamaño, en SLOC, y número de paquetes para las versiones en estudio. se puede ver el número de MSLOC y de paquetes fuente de las versiones estables de Debian.

**Figura 2. Tamaño, en SLOC, y número de paquetes para las versiones en estudio.**




Número de SLOC para cada versión




Número de paquetes en cada versión

```
En ambas gráficas de esta figura, las versiones estudiadas se encuentran
esparcidas en el tiempo a lo largo del eje X según su fecha de
publicación. En la de la izquierda podemos ver el número de MSLOC
que incluye cada versión, mientras que la de la derecha muestra el
número de paquetes.
```

Debian 2.0 incluía 1.096 paquetes fuente que tenían más de 25 MSLOC. La siguiente versión estable de Debian, la 2.1 (publicada alrededor de nueve meses más tarde) tenía más de 37 MSLOC distribuidos en 1.551 paquetes fuente. Debian 2.2 (que salió a la luz 15 meses después de Debian 2.1) estaba compuesta por su parte por 59 MSLOC en 2.611 paquetes, mientras que la última versión estable hasta el momento, Debian 3.0

(publicada dos años después que Debian 2.2), agrupaba 4.579 paquetes de código fuente con casi 105 MSLOC.

**Tabla 2. Tamaño de las distribuciones Debian estudiadas**

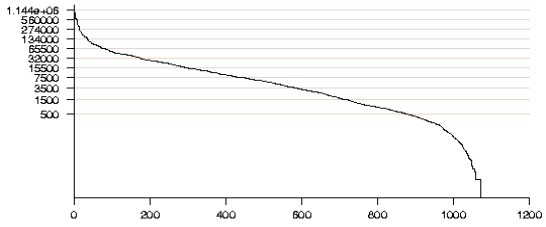
| Versión | Fecha de publicación | Paquetes fuente | Tamaño (MSLOC) | Tamaño medio de los paquetes (SLOC) |
|---|---|---|---|---|
| Debian 2.0 | Julio 1998 | 1.096 | 25 | 23.050 |
| Debian 2.1 | Marzo 1999 | 1.551 | 37 | 23.910 |
| Debian 2.2 | Agosto 2000 | 2.611 | 59 | 22.650 |
| Debian 3.0 | Julio 2002 | 4.579 | 105 | 22.860 |

# 6. Paquetes

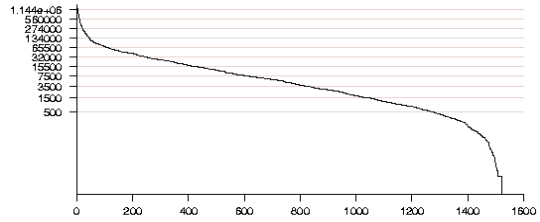
Las distribuciones están organizadas internamente en paquetes. Los paquetes suelen corresponderse casi unívocamente a aplicaciones o bibliotecas, aunque comúnmente en Debian se intenta modularizar los paquetes al máximo, por lo que se suelen dividir las fuentes de la documentación y de los datos, por ejemplo. Esto no afecta mucho a nuestros resultados, ya que las cuentas que realizamos consideran únicamente las líneas de código fuente y los paquetes de documentación en general contienen poco o nada de código. Por otro lado, debemos diferenciar dos tipos de paquetes: los paquetes fuente y los paquetes binarios. Los primeros contienen las fuentes de las aplicaciones y bibliotecas que una vez compiladas y enlazadas pueden producir varios paquetes binarios, que son los que generalmente se instalan los usuarios en sus ordenadores Por ejemplo, Debian 3.0 consta de unos 4.500 paquetes fuente, pero tiene alrededor de 10.000 paquetes binarios.

En las siguiente figura se muestran las gráficas de la distribución del tamaño de los paquetes incluido en las diferentes versiones de Debian. Se puede observar que hay un número pequeño de paquetes grandes (por encima de las cien mil líneas de código) y que el tamaño de estos paquetes tiende, como es lógico a aumentar con el tiempo. Sin embargo, parece sorprendente que a pesar del crecimiento que ha sufrido Debian a lo largo del tiempo, la gráfica no muestre grandes variaciones. Pero ciertamente lo que resulta todavía más llamativo es comprobar que el tamaño medio de los paquetes incluidos en Debian sea sorprendentemente regulares (alrededor de 23.000 SLOC para Debian 2.0, 2.1, 2.2 y 3.0). Con los datos que tenemos en la actualidad es difícil dar una explicación contundente a este hecho, pero nos podemos aventurar a lanzar alguna teoría: quizás el "ecosistema" de Debian es tan rico que mientras muchos paquetes crecen en tamaño, otros más pequeños son incluidos haciendo que la media se mantenga aproximadamente constante.

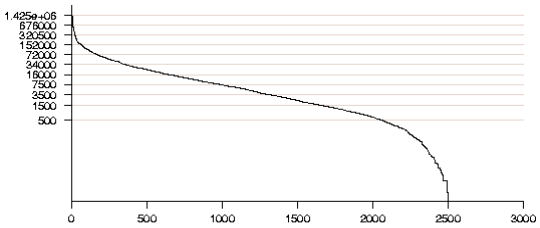
**Figura 3. Tamaño de los paquetes en las distribuciones Debian. Los paquetes están ordenados por tamaño según el eje X, mientras las cuentas en SLOC se representan en el eje Y (en escala logarítmica)**



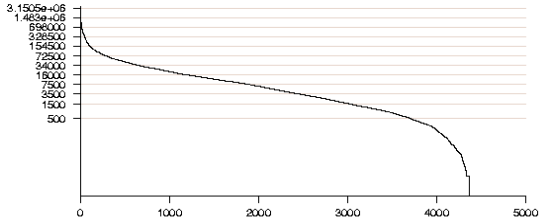

Debian 2.0



Debian 2.1



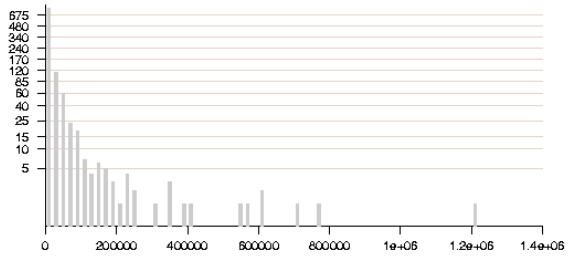

Debian 2.2



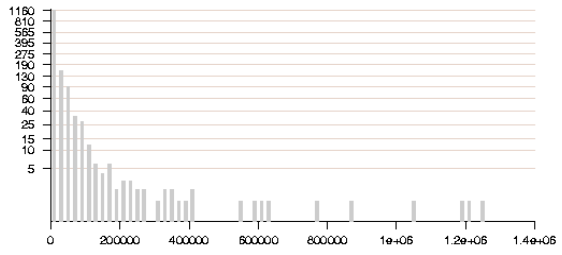

Debian 3.0

El histograma con el tamaño de los paquetes muestra los mismos datos desde otra perspectiva. Se puede observar nítidamente cómo los paquetes grandes aumentan con el tiempo en tamaño, a la vez que cada vez existen más y más paquetes cerca del origen, hecho que se puede constatar especialmente para el caso de paquetes muy pequeños (menos de mil líneas de código), pequeños (menos de diez mil) y medianos (entre diez mil y cincuenta mil líneas de código).

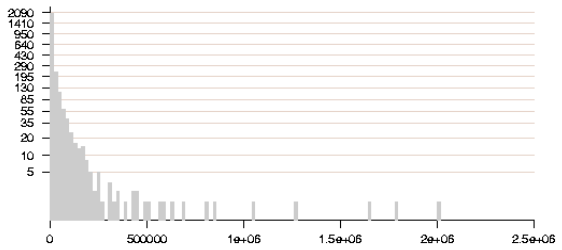
**Figura 4. Histograma con la distribución de SLOC para paquetes en Debian**



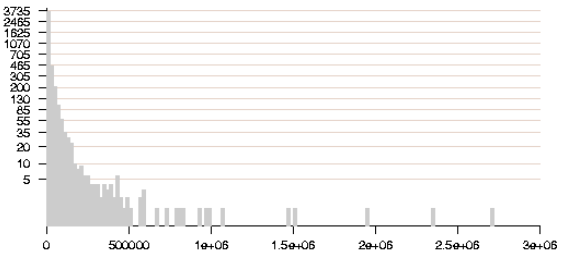

Debian 2.0




Debian 2.1




Debian 2.2




Debian 3.0

También es interesante ver la evolución de los paquetes más grandes incluidos en cada una de las versiones estables de Debian. Muchos de estos paquetes corresponden a aplicaciones significativas, muy conocidas y populares y que han sido documentadas en detalle en varios artículos científicos. El estudio de cómo evolucionan estos paquetes en tamaño, así como ver la composición de los 10 más grandes en el tiempo puede ofrecer una perspectiva interesante de las distribuciones de Debian.

Hay mucho movimiento entre el selecto grupo de paquetes más grandes. El hecho de que sólo tres de ellos prevalezcan en Debian 3.0 desde la primera versión considerada en este estudio, Debian 2.0, después de casi cuatro años es indicativo en este sentido. Algunos de los "nuevos" en el club de los paquetes más grandes han sido incluidos tardíamente (como es el caso del navegador Mozilla), mientras que en el caso de otros se trata de composiciones realizadas a partir de otros paquetes (tal es el caso para mingw32, un compilador cruzado C/C++ orientado a ejecutables Win32).

Por otro lado, se puede observar que existe una clara tendencia a que el límite inferior de los diez paquetes más grandes crezca a medida que pasa el tiempo: Mientras que en Debian 2.0 podemos ver cómo gcc con unas 460.000 SLOC se situaba en la décima posición, el décimo paquete más grande para Debian 3.0, ncbi−tools (una serie de bibliotecas para aplicaciones del ámbito de la biología) constaba de más de 700.000 líneas de código. Es más, sólo el paquete más grande de Debian 2.0 entraría entre los diez más grandes de Debian 3.0.

Pero los paquetes más grandes no sólo tienden a tener más código fuente, sino que también muestran una tendencia a tener ficheros de código fuente más grandes. Mientras la media de SLOC por fichero se encuentra en el rango 352−359 para paquetes entre los diez más grandes, la media para todos los paquetes en esas versiones se sitúa entre las 228 y las 243 líneas de código fuente por fichero. Existe, sin embargo, una gran varianza en este sentido, que va desde los 138 SLOC por fichero en la versión 1.1.2 de egcs (un derivado del compilador GNU gcc) a los 806 SLOC por fichero de bigloo (un sistema de compilación para Scheme) en su versión 2.4b.

**Tabla 3. 10 paquetes más grandes en Debian 2.0**

| Puesto | Nombre del paquete | Versión | SLOC | Ficheros | SLOC/fichero |
|---|---|---|---|---|---|
| 1. | xfree86 | 3.3.2.3 | 1.189.621 | 4.100 | 290,15 |
| 2. | xemacs20 | 20.4 | 777.350 | 1.794 | 433,31 |
| 3. | egcs | 1.0.3a | 705.802 | 4.437 | 159,07 |
| 4. | gnat | 3.10p | 599.311 | 1.939 | 309,08 |
| 5. | kernel−source | 2.0.34 | 572.855 | 1.827 | 313,55 |

| | | | | | |
|---|---|---|---|---|---|
| 6. | gdb | 4.17 | 569.865 | 1.845 | 308,87 |
| 7. | emacs20 | 20.2 | 557.285 | 1.061 | 525,25 |
| 8. | lapack | 2.0.1 | 395.011 | 2.387 | 165,48 |
| 9. | binutils | 2.9.1 | 392.538 | 1.105 | 355,24 |
| 10. | gcc | 2.7.2.3 | 351.580 | 753 | 466,91 |

**Tabla 4. 10 paquetes más grandes en Debian 2.1**

| Puesto | Nombre del paquete | Versión | SLOC | Ficheros | SLOC/fichero |
|---|---|---|---|---|---|
| 1. | mozilla | M18 | 1.269.186 | 4.981 | 254,81 |
| 2. | xfree86 | 3.3.2.3a | 1.196.989 | 4.153 | 288,22 |
| 3. | kernel−source | 2.2.1 | 1.137.796 | 3.927 | 289,74 |
| 4. | prc−tools | 0.5.0r | 1.035.230 | 3.025 | 342,22 |
| 5. | egcs | 1.1.2 | 846.610 | 6.106 | 138,65 |
| 6. | xemacs20 | 20.4 | 777.976 | 1.796 | 433,17 |
| 7. | emacs20 | 20.5a | 630.052 | 1.116 | 564,56 |
| 8. | gnat | 3.10p | 599.311 | 1.939 | 309,08 |
| 9. | gdb | 4.17 | 582.834 | 1.862 | 313,02 |
| 10. | ncbi−tools6 | 6.0 | 554.949 | 951 | 583,54 |

**Tabla 5. 10 paquetes más grandes en Debian 2.2**

| Puesto | Nombre del paquete | Versión | SLOC | Ficheros | SLOC/fichero |
|---|---|---|---|---|---|
| 1. | mozilla | M18 | 1.940.167 | 9.315 | 208,28 |
| 2. | kernel−source | 2.2.19.1 | 1.731.335 | 5.082 | 340,68 |
| 3. | pm3 | 1.1.13 | 1.649.480 | 10.260 | 160,77 |
| 4. | xfree86 | 3.3.6 | 1.256.423 | 4.351 | 288,77 |
| 5. | prc−tools | 0.5.0r | 1.035.125 | 3.023 | 342,42 |
| 6. | oskit | 0.97.20000202 | 851.659 | 5.043 | 168,88 |
| 7. | gdb | 4.18.19990928 | 797.735 | 2.428 | 328,56 |
| 8. | gnat | 3.12p | 678.700 | 2.036 | 333,35 |
| 9. | emacs20 | 20.7 | 630.424 | 1.115 | 565,4 |
| 10. | ncbi−tools6 | 6.0.2 | 591.987 | 988 | 599,18 |

**Tabla 6. 10 paquetes más grandes en Debian 3.0**

| Puesto | Nombre del paquete | Versión | SLOC | Ficheros | SLOC/fichero |
|---|---|---|---|---|---|
| 1. | kernel−source | 2.4.18 | 2.574.266 | 8.527 | 301,9 |
| 2. | mozilla | 1.0.0 | 2.362.285 | 11.095 | 212,91 |
| 3. | xfree86 | 4.1.0 | 1.927.810 | 6.493 | 296,91 |
| 4. | pm3 | 1.1.15 | 1.501.446 | 7.382 | 203,39 |
| 5. | mingw32 | 2.95.3.7 | 1.291.194 | 6.840 | 188,77 |
| 6. | bigloo | 2.4b | 1.064.509 | 1.320 | 806,45 |
| 7. | gdb | 5.2.cvs20020401 | 986.101 | 2.767 | 356,38 |
| 8. | crash | 3.3 | 969.036 | 2.740 | 353,66 |
| 9. | oskit | 0.97.20020317 | 921.194 | 5.584 | 164,97 |
| 10. | ncbi−tools6 | 6.1.20011220a | 830.659 | 1.178 | 705,14 |

Desde el punto de vista del dominio de las aplicaciones, no se ven diferencias significativas en la lista de los paquetes más grandes. Copan esta clasificación herramientas del sistema (compiladores, depuradores...), sistemas gráficos, bibliotecas de propósito específico y navegadores (Mozilla). El núcleo del sistema

operativo, Linux empaquetado como kernel−source, es un fijo en este apartado.

Hasta ahora hemos podido comprobar cómo a lo largo de las últimas versiones estables, Debian ha ido creciendo en cuanto a número de paquetes y número de SLOC. En los siguientes párrafos, sin embargo, nos gustaría centrarnos en lo contrario: lo que no ha cambiado. Hemos visto con anterioridad en la lista de paquetes más grande que hay paquetes que han sido añadidos en versiones estables de Debian más recientes. Otros paquetes, sin embargo, se pueden haber "caído".

Aunque pueda parecer sorprendente, de los 1096 paquetes que incluía Debian 2.0, sólo 754 aparecen en la última versión de Debian considerada en este estudio. Esto quiere decir que un poco más del 25% de los paquetes han ido desapareciendo de Debian en los últimos cuatro años. Pero, esto que podría explicarse porque ha pasado un espacio de tiempo bastante largo en cuanto al mundo del software se refiere, se puede volver a constatar si miramos que el número de paquetes de Debian 2.2 incluidos también en Debian 3.0 es de 1920 sobre un total de 2610, que viene a resultar en un porcentaje parecido de paquetes que "desaparecen" entre estas dos versiones.

Las tablas Tabla 7. Paquetes y versiones en común para Debian 2.0, Tabla 8. Paquetes y versiones en común para Debian 2.1, Tabla 9. Paquetes y versiones en común para Debian 2.2 y Tabla 10. Paquetes y versiones en común para Debian 3.0 muestran los paquetes en común entre las diferentes versiones estables de Debian. Suponemos que dos versiones de Debian tienen un paquete en común, si ese paquete está incluido en ambas, independientemente de la versión del paquete. Cada tabla presenta en su segunda columna el número de paquetes en común que tiene una versión de Debian con las demás versiones. Se incluye, para facilitar la comparación en términos relativos y absolutos, la propia versión de Debian que se compara. De esta forma y como es lógico, Debian 2.0 tiene en común consigo misma los 1096 paquetes fuente de los que consta.

Por otra parte, también se ha de considerar que las distribuciones contienen aplicaciones y bibliotecas que van evolucionando con el tiempo. Esto se traduce en que el número de versión propio de los paquetes incluidos también evoluciona. Por ejemplo, las fuentes de Linux vienen generalmente empaquetadas en un paquete llamado kernel−source, tal y como pudimos ver en las tablas de los paquetes fuente más grandes. En cada versión de Debian, el número de versión de kernel−source va cambiando, por lo que vemos que Linux evoluciona con el tiempo y que se van introduciendo los cambios y mejoras en Debian. Esto no tiene por qué ser así para todos los paquetes. Si con anterioridad estábamos interesados en paquetes en común sin importarnos si su número de versión ha cambiado o no, ahora vamos a considerar aquéllos cuyo número de versión no varía entre distribuciones. Consideramos, por tanto, como paquetes comunes con la misma versión a aquellos paquetes fuente que están incluidos en dos versiones diferentes de Debian con el mismo número de versión de paquete. Al incluir la propia versión de Debian en la comparación, otra vez nos encontramos con el hecho de que en la versión 2.0 de Debian coincidan los 1096 comunes en número de versión.

Resulta muy llamativo el hecho de que Debian 3.0 tenga 221 paquetes que no han evolucionado en su versión desde Debian 2.0 (cuatro años antes), lo que viene a decir que un 20% de los paquetes fuente incluidos en Debian 2.0 se han mantenido casi inalterados desde que fueron publicados en Debian 2.0 hasta que lo hicieron en Debian 3.0. Como es lógico, por otra parte, el número de paquetes con versiones en común aumenta cuando las distribuciones son más cercanas en el tiempo.

**Tabla 7. Paquetes y versiones en común para Debian 2.0**

| Versión Debian | Paquetes en común | Versiones en común | SLOC versiones en común | Ficheros versiones en común | SLOC paquetes en común |
|---|---|---|---|---|---|
| Debian 2.0 | 1.096 | 1.096 | 25.267.766 | 110.587 | 2.5267.766 |
| Debian 2.1 | 1.066 | 666 | 11.518.285 | 11.5126 | 26.515.690 |
| Debian 2.2 | 973 | 367 | 3.538.329 | 86.810 | 19.388.048 |
| Debian 3.0 | 754 | 221 | 1.863.799 | 70.326 | 15.888.347 |

**Tabla 8. Paquetes y versiones en común para Debian 2.1**

| Versión Debian | | | | | |
|---|---|---|---|---|---|

| | Paquetes en común | Versiones en común | SLOC de las versiones en común | Ficheros de las versiones en común | SLOC de los paquetes en común |
|---|---|---|---|---|---|
| Debian 2.0 | 1.066 | 666 | 11.518.285 | 115.126 | 26.515.690 |
| Debian 2.1 | 1.551 | 1.551 | 37.086.828 | 161.303 | 37.086.828 |
| Debian 2.2 | 1.384 | 602 | 8.460.239 | 133.140 | 30.052.890 |
| Debian 3.0 | 1.076 | 322 | 3.152.790 | 108.071 | 24.743.063 |

**Tabla 9. Paquetes y versiones en común para Debian 2.2**

| Versión Debian | Paquetes en común | Versiones en común | SLOC de las versiones en común | Ficheros de las versiones en común | SLOC de los paquetes en común |
|---|---|---|---|---|---|
| Debian 2.0 | 973 | 367 | 3.538.329 | 86.810 | 19.388.048 |
| Debian 2.1 | 1.384 | 602 | 8.460.239 | 133.140 | 30.052.890 |
| Debian 2.2 | 2.610 | 2.610 | 59.138.348 | 257.724 | 59.138.348 |
| Debian 3.0 | 1.921 | 771 | 8.356.302 | 186.508 | 42.938.562 |

**Tabla 10. Paquetes y versiones en común para Debian 3.0**

| Versión Debian | Paquetes en común | Versiones en común | SLOC de las versiones en común | Ficheros de las versiones en común | SLOC de los paquetes en común |
|---|---|---|---|---|---|
| Debian 2.0 | 754 | 221 | 1.863.799 | 70.326 | 15.888.347 |
| Debian 2.1 | 1.076 | 322 | 3.152.790 | 108.071 | 24.743.063 |
| Debian 2.2 | 1.921 | 771 | 8.356.302 | 186.508 | 42.938.562 |
| Debian 3.0 | 4.578 | 4.578 | 104.305.557 | 403.285 | 104.702.397 |

# 7. Lenguajes

Como ya comentamos en la metodología de este estudio, antes de contar el número de SLOC se identifica el lenguaje en el que está escrito un fichero. Gracias a esto, podemos conocer la implantación y utilización de los diferentes lenguajes en Debian. El lenguaje más utilizado en todas las versiones es C con porcentajes que se sitúan entre el 60% y el 85% y con una amplia ventaja sobre su más inmediato perseguidor, C++. Se puede observar, sin embargo, cómo la importancia de C va disminuyendo paulatinamente, mientras que otros lenguajes crecen a buen ritmo.

Como ejemplo, en la tabla Tabla 11. Lenguajes más utilizados en Debian se muestra la evolución de los lenguajes más significativos −los que superan el 1% de código− en Debian 3.0. Por debajo de la frontera del 1% se sitúan en Debian 3.0, en este orden, PHP, Ada, Modula3, Objective C, Java, Yacc y ML (todos con porcentajes entre el 0.30% y el 0.60%).

**Tabla 11. Lenguajes más utilizados en Debian**

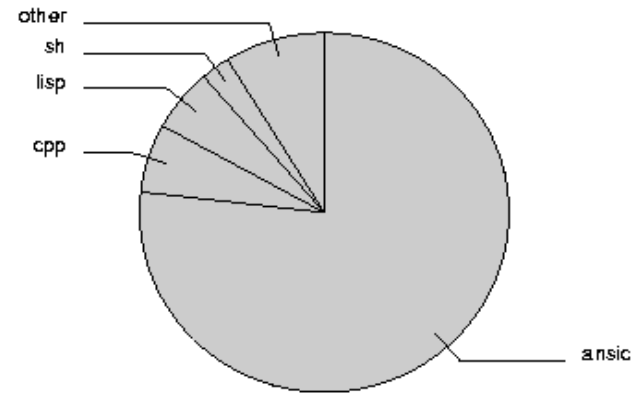
| Lenguaje | KSLOC Debian 2.0 | Porcentaje Debian 2.0 | KSLOC Debian 2.1 | Porcentaje Debian 2.1 | KSLOC Debian 2.2 | Porcentaje 2.2 |
|---|---|---|---|---|---|---|
| C | 19.371 | 76,67% | 27.773 | 74,89% | 40.878 | 69,12% |
| C++ | 1.557 | 6,16% | 2.809 | 7,57% | 5.978 | 10,11% |
| Shell | 645 | 2,55% | 1.151 | 3,10% | 2.712 | 4,59% |
| Lisp | 1.425 | 5,64% | 1.892 | 5,10% | 3.197 | 5,41% |
| Perl | 425 | 1,68% | 774 | 2.09% | 1,395 | 2,36% |
| Fortran | 494 | 1,96% | 735 | 1,98% | 1.182 | 1,99% |

| | | | | | | |
|---|---|---|---|---|---|---|
| Python | 122 | 0,48% | 211 | 0,57% | 349 | 0,59% |
| Tcl | 311 | 1,23% | 458 | 1,24% | 557 | 0,94% |

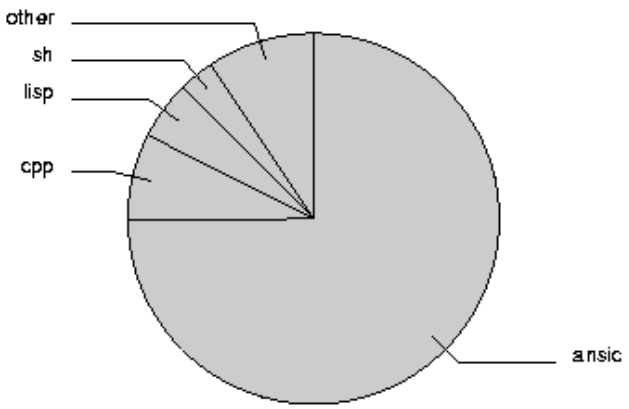
Existen lenguajes que podríamos considerar minoritarios que alcanzan un puesto bastante alto en la clasificación. Esto se debe a que aún encontrándose presentes en un número reducido de paquetes, éstos son bastante grandes. Tal es el caso de Ada, que en tres paquetes (gnat, un compilador de Ada, libgtkada, un enlace a la biblioteca GTK, y Asis, un sistema para gestionar fuentes en Ada) aglutina 430.000 SLOC de un total de 576.000 SLOC que se han contabilizado en Debian 3.0 para Ada. Otro caso parecido es el de Lisp, que cuenta sólo con GNU Emacs y con XEmacs con más de 1.200.000 SLOC de los alrededor de 4 MSLOC en toda la distribución.

Las tartas de distribución de lenguajes muestran la clara tendencia que existe en cuanto a la aportación de C al sistema global. Algo parecido parece ocurrirle a Lisp, que pasa de ser el tercer lenguaje más utilizado en Debian 2.0 a ser el cuarto en Debian 3.0, y que previsiblemente seguirá retrocediendo en el futuro. Por contra, tanto la parte de la tarta correspondiente a C++, a shell y a otros lenguajes de programación aumenta.

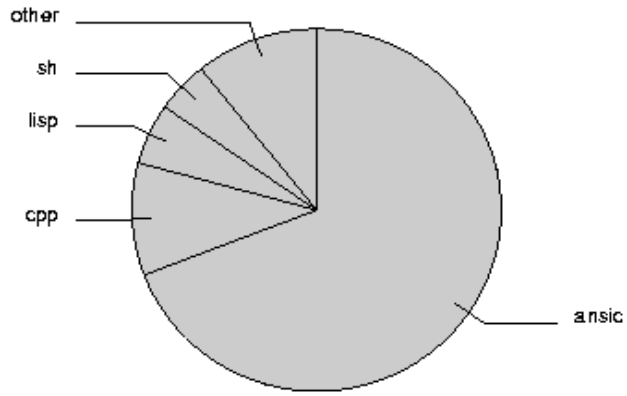
**Figura 5. Tarta con la distribución de líneas de código fuente para los lenguajes mayoritarios en las versiones de Debian**



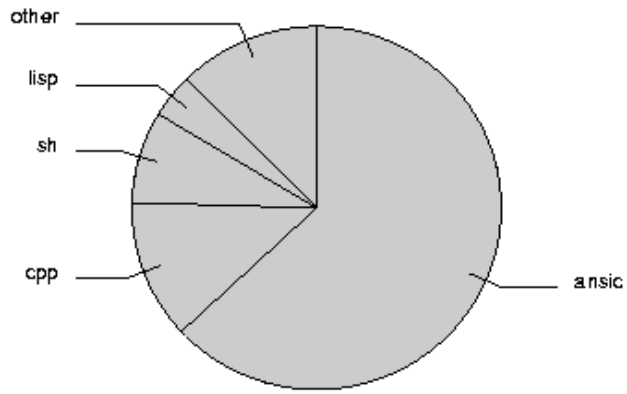

Debian 2.0




Debian 2.1




Debian 2.2




Debian 3.0

La gráfica de la evolución relativa de los lenguajes nos da una nueva perspectiva del crecimiento de los lenguajes a través de la historia de las últimas cuatro versiones estables de Debian. Para ello, tomamos como referencia la distribución Debian 2.0 y suponemos que la presencia de cada lenguaje en ella es del 100%.

Los gráficos de tartas mostraban que C esta retrocediendo en cuanto a presencia relativa se trataba. En esta podemos ver que aún así, C ha crecido más de un 300% a lo largo de las cuatro versiones, un hecho nada despreciable.Sin embargo, se puede ver que son los lenguajes de script (shell, Python y Perl) los que han sufrido un extraordinario crecimiento, todos ellos multiplicando su presencia por factores superiores a siete, acompañados por C++. Lenguajes que crecen en menor cuantía son los lenguajes compilados (Fortran y Ada). Esto nos puede dar una idea de la importancia que los lenguajes interpretados han empezado a tener para el software libre.

La gráfica incluye los lenguajes más representativos que hay en Debian, excluyendo a Java y a PHP, ya que el crecimiento de estos dos es enorme, debido más bien a que su presencia en Debian 2.0 era testimonial.

**Figura 6. Evolución de los cuatro lenguajes de programación más usados en Debian**

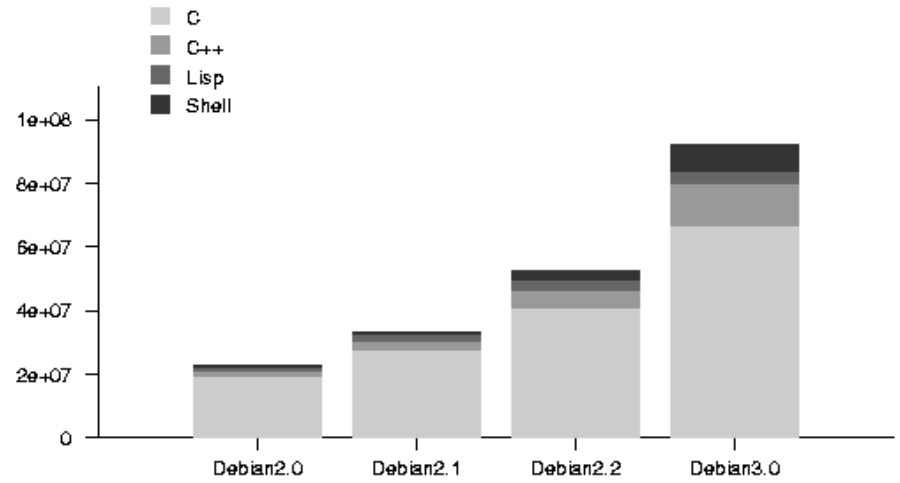
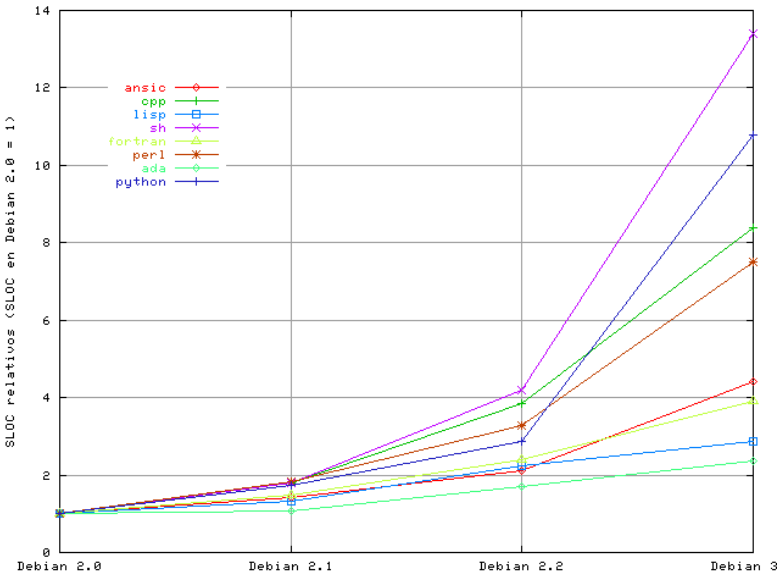



**Figura 7. Crecimiento relativo de algunos lenguajes de programación en Debian**




En cuanto al tamaño de fichero medio para los lenguajes de programación más importantes, resulta interesante comprobar cómo a pesar del espectacular aumento de algunos de ellos en cuanto a uso, su tamaño medio de fichero suelen ser más o menos constantes. Así, para C la longitud media ronda los 260 a 280 líneas de código fuente por fichero, mientras que en C++ se sitúa en una horquilla que va desde los 140 hasta los 185. La excepción a esta regla la podemos encontrar en el lenguaje de shell, que triplica su tamaño medio. Esto es debido a que el lenguaje shell es muy singular: casi todos los paquetes incluyen algo de shell para su instalación, configuración o como "pegamento". Es probable que este tipo de scripts vayan complicándose con el tiempo al complicarse estos procedimientos.

Resulta curioso ver cómo los lenguajes estructurados suelen tener longitudes de fichero medias más grandes que los lenguajes orientados a objetos. Así los ficheros de C (o Yacc) suelen ser bastante más grandes, en media, que los de C++. Esto nos hace pensar que la modularidad de los lenguajes se refleja también en el tamaño medio de los ficheros.

**Tabla 12. Tamaño de fichero medio para algunos lenguajes**

| Lenguaje | Debian 2.0 | Debian 2.1 | Debian 2.2 | Debian 3.0 |
|---|---|---|---|---|
| C | 262,88 | 268,42 | 268,64 | 283,33 |
| C++ | 142,5 | 158,62 | 169,22 | 184,22 |
| Lisp | 394,82 | 393,99 | 394,19 | 383,60 |

| | | | | |
|---|---|---|---|---|
| shell | 98,65 | 116,06 | 163,66 | 288,75 |
| Yacc | 789,43 | 743,79 | 762,24 | 619,30 |
| Media | 228,49 | 229,92 | 229,46 | 243,35 |

# 8. COCOMO

El modelo COCOMO [Boehm1981] nos da una estimación del esfuerzo humano y monetario necesario para generar software a partir del tamaño del mismo. Toma como medida de entrada el número de líneas de código fuente. COCOMO es un modelo pensado para procesos de generación de software "clásicos" (desarrollo en cascada o en V) y para proyectos de tamaño medio o grande, por lo que las cifras que nos ofrece en nuestro caso han de ser tomadas con mucho cuidado. En cualquier caso, los resultados nos pueden dar una idea del orden de magnitud en el que nos movemos, dándonos los esfuerzos óptimos necesarios si se hubiera utilizado un modelo de desarrollo propietario.

En general, lo que más asombra de los resultados de COCOMO es su estimación de costes. En dicha estimación se tienen en cuenta dos factores: el salario medio de un desarrollador y el factor de " overhead". En el cálculo de la estimación de costes, se ha tomado el salario medio para un programador de sistemas a tiempo completo de acuerdo con la encuesta del año 2000 de acuerdo con [CompWorld2000]. El "overhead" es el sobrecoste que toda empresa ha de asumir para que el producto salga a la calle con independencia del salario de los programadores. En este apartado se incluyen desde el salario para las secretarias y el equipo de márketing hasta los costes de las fotocopias, luz, equipos hardware, etc. En resumen, el coste calculado por COCOMO es el coste total que le supondría a una empresa crear un software del tamaño especificado y no se ha de ver simplemente como el dinero que percibirían los programadores por realizar el software. Una vez incidido en esto, los cálculos de costes dejan de parecer tan abultados.

En la tabla Tabla 13. Estimaciones de esfuerzo, tiempo y coste de desarrollo para cada versión de Debian podemos observar los resultados de aplicar el modelo de COCOMO básico a las diferentes versiones estables de Debian. Ha sido obtenido mediante el cálculo por separado del coste de cada paquete y su posterior suma. Nótese que al ser COCOMO un modelo no lineal, la suma de los costes de los diferentes paquetes por separado no es igual al coste de la suma del tamaño de todos los paquetes. El primero lo que nos da, siendo estrictos, es el límite inferior del esfuerzo, ya que no se consideran las tareas de integración, mientras que en el segundo caso tendríamos un límite superior, ya que no se tienen en cuenta ahorros de tener proyectos independientes. En [ DebianCounting] se pueden obtener las dos cifras para su comparación. Para nuestros objetivos en este artículo nos vale con una estimación del orden de magnitud y, por tanto, se presenta sólo una de ellas.

**Tabla 13. Estimaciones de esfuerzo, tiempo y coste de desarrollo para cada versión de Debian**

| Versión | MSLOC | Esfuerzo (personas−año) | Tiempo (años) | Coste (USD) |
|---|---|---|---|---|
| Debian 2.0 | 25 | 6.360 | 4,93 | 860.000.000 |
| Debian 2.1 | 37 | 9.425 | 4,99 | 1.275.000.000 |
| Debian 2.2 | 59 | 14.950 | 6,04 | 2.020.000.000 |
| Debian 3.0 | 105 | 26.835 | 6,81 | 3.625.000.000 |

# 9. Comparación con otras distribuciones

Existe un estudio similar al mostrado en este artículo, aunque en su caso la distribución objeto de estudio es Red Hat. Red Hat se puede considerar la distribución canónica entre las comerciales, que tienen una estrategia y una filosofía diferente a la que hemos presentado para Debian. Como comparación, Red Hat sirve perfectamente a nuestros propósitos. No olvidemos que el sistema de paquetes de Red Hat (RPM) es el que utilizan como sistema de gestión de paquetes una gran mayoría de distribuciones, seguido −de lejos− por el sistema utilizado en Debian (los conocidos .deb) tal y como se muestra en [DistroWatch]. Nos encontramos, por tanto, probablemente ante la comparación de las dos distribuciones más significativos de GNU/Linux en particular y del mundo del software libre en general. Los resultados de Red Hat han sido extraídos en parte de [Wheeler2000] y [Wheeler2001].

Las principales diferencias con Debian las encontramos en el hecho de que detrás de Red Hat hay una empresa. Esto quiere decir que esta empresa tendrá un número determinado de empleados dedicados a integrar todo el software de manera homogénea para facilitar tanto su instalación, como su configuración y actualización. En otras palabras, mientras en Debian los paquetes incluidos dependen de si hay colaboradores voluntarios que gestionen lo que haya que hacer para que un software específico sea empaquetado, en Red Hat entran en juego ciertos cálculos económicos para ver el esfuerzo que supone hacer una distribución nueva.

Fruto de estas divergencias en su concepción nacen una serie de diferencias entre Red Hat y Debian que podemos analizar y comparar. Una de las principales diferencias es el hecho de que el número de paquetes en Red Hat sea notoriamente inferior al de versiones coetáneas de Debian. Así, Debian 2.2 dobla en tamaño a Red Hat 7.1, cuando en realidad su publicación fue unos meses más tarde.

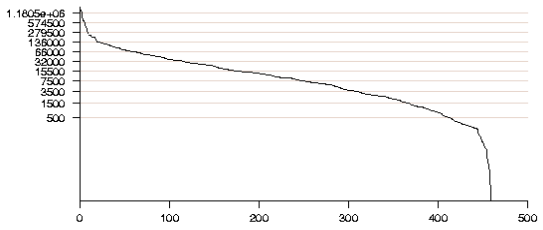
**Tabla 14. Comparación con otras distribuciones de GNU/Linux**

| Nombre | Publicación | MSLOC | Esfuerzo (pers.−año) | Tiempo (años) | Coste (USD) |
|---|---|---|---|---|---|
| Red Hat 5.2 | Abril 1998 | 12 | 3.216 | 4,93 | 434.500.000 |
| Red Hat 6.0 | Abril 1999 | 15 | 3.951 | 5,08 | 534.000.000 |
| Red Hat 6.2 | Marzo 2000 | 17 | 4.550 | 5,45 | 615.000.000 |
| Debian 2.0 | Julio 1998 | 25 | 6.360 | 4,93 | 860.000.000 |
| Red Hat 7.1 | Abril 2001 | 30 | 7.950 | 6,53 | 1.075.000.000 |
| Debian 2.1 | Marzo 1999 | 37 | 9.425 | 4,99 | 1.275.000.000 |
| Red Hat 8.0 | Septiembre 2002 | 50 | 13.315 | 7,35 | 1.800.000.000 |
| Debian 2.2 | Agosto 2000 | 59 | 14.950 | 6,04 | 2.020.000.000 |
| Debian 3.0 | Julio 2002 | 105 | 26.835 | 6,81 | 3.625.000.000 |

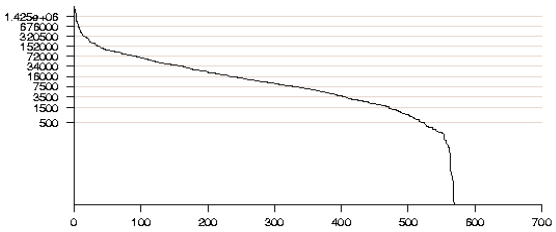
Por otro lado, las distribuciones de Red Hat suelen incluir las versiones más actuales del software, mientras que en Debian hay unos plazos de congelación que harán que lo que se obtenga en una versión estable nunca sea lo último. Esto se demuestra fácilmente echando un ojo a las versiones de los paquetes incluidos en Red Hat y en Debian. Por ejemplo, podemos ver que muchos paquetes de Red Hat 6.2 y de Debian 2.2 coinciden aún cuando Debian 2.2 fue publicada cinco meses más tarde. Es más, en algunos casos, Debian 2.2 incluso incluye versiones más antiguas que las que se obtenían en Red Hat 6.2.

Otro aspecto interesante es que Red Hat parece mostrar un menor interés por paquetes pequeños, como se muestra en la figura Figura 8. Tamaño de cada paquete en las distribuciones Red Hat. Los paquetes están ordenados según tamaño en el eje de las X. El número de SLOC para cada paquete se representa en escala logarítmica en el eje vertical.. Como consecuencia de esto último, el número de SLOC por paquete va creciendo con el tiempo, mientras que recordemos que con Debian mantenía un valor aproximadamente constante.

**Figura 8. Tamaño de cada paquete en las distribuciones Red Hat. Los paquetes están ordenados según tamaño en el eje de las X. El número de SLOC para cada paquete se representa en escala logarítmica en el eje vertical.**




Red Hat 6.2




Red Hat 7.1

Para una comparación más detenida de las versiones de Debian y las de Red Hat remitimos al lector a [GBarahona2003b].

# 10. Comparación con otros sistemas operativos

Si dicen que todas las comparaciones son odiosas, las de software libre con software propietario lo son más. Todo el estudio de Debian ha sido posible por su condición de software libre. El acceso al código (y a otra información que ha sido expuesta en este artículo) es indispensable para estudiar a fondo las diferentes versiones en cuanto a número de líneas, paquetes, lenguajes de programación utilizados... Pero las ventajas del software libre (y, por tanto, de la ingeniería del software libre, véase [Robles2002]) van más allá, porque además facilitan la revisión de terceras personas, ya sean grupos de investigación o sencillamente personas interesadas.

En los sistemas propietarios, en general, realizar un estudio así es tarea imposible. De hecho, las cuentas que se ofrecen a continuación tienen sus fuentes en las propias compañías que están detrás del desarrollo de software, por lo que no podemos avalar su veracidad. Para más inri, en muchos casos no sabemos si se está hablando de líneas de código fuente físicas (SLOC) tal y como hemos venido haciendo a lo largo de este artículo o también incluyen en sus cuentas las líneas en blanco y las de comentarios. A esto hay que añadir que tampoco sabemos a ciencia cierta lo que consideran en su software, por lo que para algunas versiones de Microsoft Windows no sabemos si incluyen el paquete de Microsoft Office o no.

En cualquier caso, y teniendo en cuenta todo lo que se ha comentado al respecto en párrafos anteriores, pensamos que incluir esta comparativa es interesante, ya que nos ayuda a situar las diferentes versiones de Debian dentro de un panorama más amplio. Lo que parece estar fuera de toda duda es que tanto Debian como Red Hat, pero especialmente el primero, son las colecciones de software más grandes vistas jamás por la humanidad hasta el momento.

Los números que se citan a continuación proceden de [Lucovsky2000] para Windows 2000, [SunPressRelease] para StarOffice 5.2, [McGraw] para Windows XP y [Schneier2000] para el resto de sistemas. En la tabla Tabla 15. Comparación con sistemas propietarios se muestra la comparativa en orden creciente.

**Tabla 15. Comparación con sistemas propietarios**

| Sistema | Fecha de publicación | Líneas de código |
|---|---|---|
| Microsoft Windows 3.1 | Abril 1992 | 3.000.000 |
| SUN Solaris 7 | Octubre 1998 | 7.500.000 |
| SUN StarOffice 5.2 | Junio 2000 | 7.600.000 |
| Microsoft Windows 95 | Agosto 1995 | 15.000.000 |
| Debian 2.0 | Julio 1998 | 25.000.000 |
| Microsoft Windows 2000 | Febrero 2000 | 29.000.000 |
| Debian 2.1 | Marzo 1999 | 37.000.000 |
| Windows NT 4.0 | Julio 1996 | 40.000.000 |
| Debian 2.2 | Agosto 2000 | 55.000.000 |

| | | |
|---|---|---|
| Debian 3.0 | Julio 2002 | 105.000.000 |

# 11. Mirando en la bola de cristal: La próxima versión de Debian

A partir de los datos obtenidos de las cuatro últimas versiones estables de Debian, hemos podido ver cómo ha ido evolucionando esta distribución. Adentrándonos un poco la senda predictiva, podemos utilizar estos resultados para intentar adivinar cómo podría ser la siguiente versión estable de Debian. Nótese que si en algunos casos ya con las versiones pasadas de Debian hemos hablado de estimaciones (al aplicar el cálculo de costes y esfuerzo), ahora vamos a entrar en un mundo de especulaciones y como tal han de tomarse. El que nuestras predicciones se ajusten más o menos a la realidad no depende solamente de parámetros técnicos, sino también de otros organizativos y estructurales, lo que hace todavía más imprevisible la evolución de Debian, y por tanto su predicción.

Uno de los puntos clave para la predicción es saber la fecha de publicación de la siguiente versión. Esto es, sin duda, una pregunta a la que deberán dar respuesta los integrantes del proyecto Debian y que a buen seguro no dependerá sólo de parámetros técnicos, sino también organizativos y humanos. Hasta ahora el espacio de tiempo entre versiones estables se ido viendo incrementando paulatinamente (entre Debian 2.0 y la versión 2.1 sólo pasaron 8 meses, mientras que entre las siguientes versiones el tiempo que hubo que esperar fue de 17 y 23 meses respectivamente). Si partimos de estas cifras y suponemos que Debian va aumentar en tamaño, y por tanto su integración va a ser más dificultosa, nos vamos a aventurar a situar la siguiente versión estable en diciembre de 2004, 28 meses después de la última.

En cuanto a la estimación de tamaño, en número líneas de código fuente, tenemos dos suposiciones diferentes, debido en parte a que estamos considerando un número demasiado pequeño (cuatro) de elementos para poder predecir el siguiente con exactitud. Por una parte, si seguimos la idea de que el código se dobla aproximadamente cada dos años, la siguiente versión de Debian debería contar con alrededor de 220 MSLOC − partiendo de que tenga lugar después de unos 28 meses. Por otro lado, el factor de crecimiento entre versiones estables nunca ha sido tan grande. Podemos ver cómo Debian 2.1 supuso un crecimiento de un 50% frente a Debian 2.0, mientras que Debian 2.2 lo sitúa en un 60% y Debian 3.0 en casi un 80% frente a sus versiones anteriores. Esto puede ser debido a que la integración se vuelve más complicada con el número de paquetes, algo lógico por otra parte, y a que el periodo de congelación antes de lanzar la nueva versión estable ha de aumentar. La estimación, por tanto, siguiendo estos segundos parámetros, nos sitúa en una próxima versión estable que rondará los 185 MSLOC.

En las versiones estables de Debian estudiadas en este artículo, hemos podido ver, para nuestra sorpresa, cómo el tamaño medio de los paquetes se mantenía constante. Si suponemos que esto va a seguir siendo así, mediante una simple conversión de los cálculos anteriores realizados en líneas de código fuente, obtendremos el número de paquetes incluidos en la próxima versión de Debian. Para la primera aproximación (unos 220 MSLOC), contaríamos con la extraordinaria cifra de 9600 paquetes fuente, mientras que para la segunda aproximación (unos 185 MSLOC) el número de paquetes fuente incluidos será ligeramente superior a los 8000.

Tomando como entrada unos 200 MSLOC, las estimaciones que nos da COCOMO son astronómicas. Para generar un software de tales dimensiones, se necesitarían un millón de meses−hombre (algo más de 80 mil años−hombre) y el tiempo estimado óptimo corresponde a 450 meses (más de 37 años) en el que dos mil desarrolladores trabajaran en el proyecto. El coste total estimado ascendería a unos 10.000 millones de euros aproximadamente. Es importante hacer notar que hemos partido de la idea de que Debian es un solo proyecto (y no la suma de muchos proyectos más pequeños), ya que en nuestras previsiones previas no hemos hecho una estimación del tamaño de los paquetes que serán incluidos. Como siempre, estas cifras son orientativas y como tal han de entenderse.

A continuación nos vamos a aventurar a mirar en la bola de cristal para ver los paquetes más grandes incluidos en la próxima versión estable. Hasta ahora, hemos visto que a pesar de haber una gran movilidad en este apartado, suelen prevalecer herramientas del sistema, bibliotecas de propósito específico, compiladores y un navegador, Mozilla. Podemos aventurarnos a la vista de la polémica que está causando la tecnología .NET y el éxito de iniciativas como MONO o dotGNU, que podremos ver un compilador, o al menos una suite de clases, de C# entre los paquetes más grandes. El giro del software libre hacia el usuario final y el escritorio también podrá verse reflejado en esta categoría con la más que probable inclusión de la suite ofimática

OpenOffice.org. En cualquier caso, el peaje que se ha de pagar para entrar en este selecto club de los más grandes va a ser muy grande: casi con toda seguridad habrá que superar la barrera del millón de líneas de código fuente para poder presumir de ello.

En cuanto a la distribución de lenguajes, podemos asegurar que C seguirá siendo el lenguaje con mayor presencia dentro de Debian. Sin embargo, su supremacía seguirá menguando hasta el punto que podemos afirmar que casi la mitad del código de Debian no estará escrito en C. C++, por su parte, seguirá creciendo en cuanto a importancia relativa, alcanzando previsiblemente el 15% y llegando a los 30 MSLOC (ayudado por la más que probable inclusión de OpenOffice.org que está compuesto por unos 4 MSLOC escritos principalmente en este lenguaje). Sin embargo, serán los lenguajes de programación de guión, como PHP, Python y Perl (nos atrevemos a indicar que va a ser incluso en este orden) los que sufrirán un notable aumento en código y en importancia. Los lenguajes compilados seguirán con su tendencia relativa a la baja, como viene siendo el caso para Fortran, Ada o Pascal.

En cuanto a Java, pensamos se librará de esta tendencia y auguramos un notable ascenso, debido principalmente a dos causas: la primera es la inclusión en Debian de varios paquetes del proyecto Apache (Jakarta, etc.) basados en Java que ya a día de hoy son bastante amplios y la segunda se basa en que Debian no puede ser ajeno al gran número de proyectos en Java que se han iniciado en los últimos tres años − debido probablemente a la nueva generación de desarrolladores que han aprendido Java en sus cursos universitarios. El lenguaje C# entrará por primera vez en los resultados para los lenguajes de programación, pero su presencia será puntual. Y es que, aunque apostamos por que la siguiente versión de Debian tenga un compilador y una jerarquía de clases para este lenguaje, creemos que todavía las aplicaciones que se creen con ellos no estarán los suficientemente maduros. En todo caso, estamos seguros de que C# será un lenguaje importante para la versión posterior a la siguiente.

En cuanto al número de desarrolladores voluntarios que participan en el proyecto Debian, suponemos que se seguirá manteniendo el crecimiento que ha habido en el último año y medio, por lo que rondará los 1.100 desarrolladores. Este hecho despierta serias dudas acerca del crecimiento futuro de Debian, ya que si se mantienen las previsiones el ratio de paquetes por desarrollador en la siguiente versión será de nueve, mientras que en las últimas versiones se ha situado entre los 4 de la versión 2.1 y los 6 de la 2.2 (en Debian 3.0 fue de 5 aproximadamente). Puede que sea por aquí donde nos encontremos el factor limitador en el crecimiento de Debian, ya que como se ha comentado esta distribución depende básicamente de que alguien quiera empaquetar un programa para que estuviera disponible.

# 12. Conclusiones

En este artículo se han presentado los resultados de estudiar en profundidad las versiones estables de Debian 2.0 en adelante. Hemos podido ver la evolución del número de líneas de código físicas, el número y tamaño de los paquetes y los lenguajes más utilizados. Estos datos han sido apoyados por el número de desarrolladores voluntarios con los que cuenta Debian para crear sus distribuciones, así como las estimaciones de esfuerzo, tiempo y coste utilizando el conocido método de COCOMO. También se han comparado las versiones de Debian con versiones de otras distribuciones, en nuestro caso Red Hat, y con sistemas propietarios grandes. Finalmente se ha realizado una predicción de cómo debería ser la siguiente versión estable de Debian, ateniéndonos a cómo han sido las versiones más recientes.

Entre las evidencias más importantes que se han presentado cabe destacar el hecho de que las versiones estables de Debian parecen doblar el número de líneas de código y paquetes cada dos años aproximadamente, una evolución que pensamos que sólo se podrá mantener si se unen más desarrolladores voluntarios al proyecto Debian. Al menos eso es lo que se concluye del hecho de que hasta ahora el tamaño medio de los paquetes ha sido aproximadamente constante, por lo que el número de paquetes crece linealmente con el número de líneas de código. Si el número de desarrolladores de Debian no crece en esas proporciones, el número de paquetes que habrá de mantener un desarrollador se disparará.

El tamaño de la última versión de Debian (la 3.0) nos hace pensar que estamos ante una de las colecciones de software más grande de la historia de la humanidad, sino la que más. Para crear sus 105 MSLOC, según el modelo de COCOMO, serían necesarios 27.000 personas−año y el coste rondaría los 3.600 millones de dólares. Ninguno de los otros sistemas con los que hemos comparado Debian (Red Hat, Solaris, Windows,

etc.) puede competir en la actualidad en tamaño con Debian.

En las aplicaciones más grandes de Debian predominan aplicaciones de bajo nivel (núcleo, software para desarrollo, bibliotecas de propósito específico...), aunque en los últimos tiempos con la inclusión de Mozilla ha habido un vuelco hacia las aplicaciones de usuario final. Suponemos que en futuras versiones, la suite ofimática OpenOffice.org haga más patente esta evolución.

En cuanto a los lenguajes de programación, C es el lenguaje más utilizado, aunque se vea cómo gradualmente va perdiendo peso. Los lenguajes de guión, C++ y Java son los que se perfilan como los que más van a seguir creciendo en las siguientes versiones, mientras que los lenguajes compilados tradicionales tienen tasas de crecimiento inferiores incluso a C.

Para finalizar, nos gustaría hacer hincapié en que estamos ofreciendo solamente estimaciones, aunque consideramos que son suficientes para sacar algunas conclusiones, comparar con otros sistemas y realizar algunas predicciones sobre el futuro de Debian.

## Bibliografía


[Boehm1981] Barry W. Boehm. 1981. *Software Engineering Economics*. Prentice Hall.

[ComWorld2000] Computer World. *Salary Survey 2000*. http://www.computerworld.com/cwi/careers/surveysandreports .

[DBDebian] Debian Project. *Debian Developers Database*. http://db.debian.org .

[DebianHistory] Debian Documentation Team. *A Brief History of Debian*. http://www.debian.org/doc/manuals/project−history/ .

[Debian22Ann] Debian Project. *Debian GNU/Linux 2.2, the Joel 'Espy' Klecker release, is officially released*. http://www.debian.org/News/2000/20000815 .

[DebianCounting] Jesús M. González Barahona y Gregorio Robles. *Debian Counting*. http://libresoft.dat.escet.urjc.es/debian−counting/ .

[DebianPol] Debian Project. *Debian Policy Manual*. http://www.debian.org/doc/debian−policy/ .

[DebianSocialContract] Debian Project. *Debian Social Contract*. http://www.debian.org/social_contract .

[Debian22Rel] Debian Project. *Debian GNU/Linux 2.2 release information*. http://www.debian.org/releases/2.2/ .

[DFSG] Debian Project. *Debian Free Software Guidelines (part of the Debian Social Contract)*. http://www.debian.org/social_contract .

[DistroWatch] Ladislav Bodnar. *Linux Distributions − Facts and Figures*. Mayo 2003. http://www.distrowatch.com/stats.php?section=packagemanagement .

[GBarahona2001] Jesús M. González Barahona, Miguel A. Ortuño Pérez, Pedro de las Heras Quirós, José Centeno González, y Vicente Matellán Olivera. *Counting potatoes: The size of Debian 2.2*. *Upgrade Magazine*. vol. 2. issue 6. Diciembre 2001. http://upgrade−cepis.org/issues/2001/6/up2−6Gonzalez.pdf . También disponible en http://people.debian.org/~jgb/debian−counting/ .

[GBarahona2003] Jesús M. González Barahona, Gregorio Robles, Miguel Ortuño−Pérez, Luis Rodero−Merino, José Centeno−González, Vicente Matellán−Olivera, Eva Castro−Barbero, y Pedro de−las−Heras−Quirós. *Measuring Woody: The size of Debian 3.0*. 2003. Pendiente de ser publicado. Estará accesible en http://people.debian.org/~jgb/debian−counting/ .

[GBarahona2003b] Jesús M. González Barahona, Gregorio Robles, Miguel Ortuño Pérez, Luis Rodero Merino, José Centeno González, Vicente Matellán Olivera, Eva Castro Barbero, y Pedro de las Heras Quirós. *Anatomy of two GNU/Linux distributions*. 2003. Pendiente de publicación en el libro "Free/Open Source Software Development" editado por Stefan Koch y publicado por Idea Group, Inc. .

[GodfreyTu2000] Michael W. Godfrey y Qiang Tu. Software Architecture Group (SWAG), Department of Computer Science, University of Waterloo. Agosto 3−4, 2000. *Evolution in Open Source Software: A Case Study*. 2000 International Conference on Software Maintenance http://plg.uwaterloo.ca/~migod/papers/icsm00.pdf .

[Libresoft] Jesús M. González Barahona y Gregorio Robles Martínez. *Libre Software Engineering*. http://libresoft.dat.escet.urjc.es/ .

[Lucovsky2000] Mark Lucovsky. Agosto 3−4, 2000. *From NT OS/2 to Windows 2000 and Beyond − A Software−Engineering Odyssey*. 4th USENIX Windows Systems Symposium, http://www.usenix.org/events/usenix−win2000/invitedtalks/lucovsky_html/ .

[McGraw] Gary McGraw. *Building Secure Software: How to avoid security problems the right way*. Citado por David A. Wheeler en http://www.dwheeler.com/sloc/ .

[Michlmayr2003] Martin Michlmayr y Benjamin Mako Hill. *Quality and the Reliance on Individuals in Free Software Projects*. Mayo 2003. http://opensource.ucc.ie/icse2003/3rd−WS−on−OSS−Engineering.pdf .

[Robles2001] Gregorio Robles, Henrik Scheider, Ingo Tretkowski, y Niels Weber. *WIDI − Who Is Doing It? A research on Libre Software developers*. Agosto 2001. http://widi.berlios.de/paper/study.pdf .

[Robles2002] Gregorio Robles. *Ingeniería del Software Libre − Una visión alternativa a la ingeniería del software tradicional*. Noviembre 2002. http://es.tldp.org/Presentaciones/200211hispalinux/robles/robles−ponencia−hispalinux−2002.pdf .

[Schneier2000] Bruce Schneier. 15 de marzo de 2000. *Software Complexity and Security*. Crypto−Gram Newsletter, http://www.counterpane.com/crypto−gram−0003.html .

[SLOCCount] David Wheeler. *SLOCCount*. http://www.dwheeler.com/sloccount/ .

[SunPressRelease] SUN Microsystems. 16 de October de 2000. *Sun Microsystems Announces Availability of StarOffice(TM) Source Code on OpenOffice.org*. http://www.collab.net/news/press/2000/openoffice_live.html .

[Wheeler2000] David A. Wheeler. *Estimating Linux's Size*. http://www.dwheeler.com/sloc .

[Wheeler2001] David A. Wheeler. *More Than a Gigabuck: Estimating GNU/Linux's Size*. http://www.dwheeler.com/sloc .