Contando Patatas: El tamaño de Debian 2.2

Jesús M. González-Barahona

Miguel A. Ortuño Pérez

Pedro de las Heras Quirós

José Centeno González

Vicente Matellán Olivera

Debian es la mayor distribución de software libre, en su última distribución estable supera ampliamente los 2.800 paquetes de código fuente. Es mayor que las demás, pero ¿cuánto exactamente?. En este artículo usamos el sistema “sloccount” de David Wheeler para determinar el número de lineas físicas de código fuente (physical SLOC, Physical Source Lines of Code) de Debian 2.2 (conocida como “Potato”). Veremos como Debian 2.2 incluye más de 56.000.000 líneas físicas de código, casi el doble que Red Hat 7.1, distribuida ocho meses después. Esto muestra que para distribuciones de este tamaño, el modelo de desarrollo de Debian, basado en el trabajo de un gran grupo de voluntarios repartidos por el mundo es, al menos, tan capaz como otros modelos, por ejemplo el usado por Red Hat o Microsoft, que es más centralizado y que está basado en el trabajo de empleados.

las Heras Quirós, José Centeno González y Vicente Matellán Olivera.

Si hay versions más actualizadas de este artículo estarán disponbles en http://people.debian.org/~jgb

Contacto con los autores: jgb@debian.org

1. Introducción

El 14 de agosto de 2000 el Proyecto Debian anuncia Debian GNU/Linux 2.2, la distribución “Joel ’Espy’ Klecker” [Debian22Ann] [Debian22Rel]. Su nombre familiar es “potato”, y es la última distribución (hasta la fecha) del Sistema Operativo Debian GNU/Linux. En este trabajo hemos medido la distribución, mostrando su tamaño y comparándolo con otras distribuciones.

Debian no es sólo la mayor distribución GNU/Linux, es también una de las más fiables, con varios premios basados en las preferencias de los usuarios. Aunque es difícil de estimar el número de usuarios (El Proyecto Debian no vende CDs ni ningún otro soporte con el software), es sin duda importante dentro del mercado Linux. Se preocupa especialmente de beneficiar a los usuarios con una de las ventajas fundamentales del software libre: La disponibilidad del código fuente. Así, los paquetes fuente se preparan cuidadosamente para permitir la reconstrucción de los fuentes originales (conocidos como “upstream”). Estas precauciones resultan muy convenientes para hacer mediciones y, en general, para obtener estadísticas.

La idea de este artículo surgió del interesante trabajo de David Wheeler [Wheeler2001]. Animamos al lector a, al menos, ojearlo, y comparar los datos que ofrece con los aportados aquí.

La estructura de este artículo es la siguiente: La próxima sección proporciona algunas ideas de contexto sobre el proyecto Debian y la distribución Debian 2.2 GNU/Linux. Posteriormente, examinaremos el método empleado para recopilar los datos mostrados, para después ofrecer los resultados de la medición de Debian 2.2 (incluyendo totales,

tamaños máximos, mediciones por lenguaje, etc). En la siguiente sección se comentan algunas de las cifras y cómo deben ser consideradas, así como algunas comparaciones con Red Hat Linux y otros sistema operativos. Finalizaremos con conclusiones y trabajo relacionado.

2. Algunas ideas de contexto sobre Debian

La distribución Debian 2.2 GNU/Linux está recopilada por el Proyecto Debian, quien también se encarga de su mantenimiento. En esta sección aportaremos alguna información básica sobre Debian como proyecto, y sobre la distribución 2.2.

2.1. El Proyecto Debian

Debian es un sistema operativo libre, que actualmente usa el kernel Linux como núcleo de todo el software de la distribución Debian GNU/Linux (si bien se esperan en un futuro próximo distribuciones basadas en otros núcleos como Hurd). La distribución está disponible para varias arquitecturas: Intel x86, ARM, Motorola 680x0, PowerPC, Alpha y SPARC.

El núcleo de la distribución Debian es la sección principal (“main”). Constituye con mucho el grueso de los paquetes, y está compuesto sólo por software libre, conforme con lo que la DFSG (Debian Free Software Guidelines) [DFSG] entiende por software libre. Esta distribución se puede descargar de la red, y muchos re-distribuidores la venden en CD u otros formatos. La distribución Debian es recopilada por el Proyecto Debian, un grupo de 900 desarrolladores voluntarios repartidos por todo el mundo y colaborando a través de Internet. No sólo se ocupan de adaptar y empaquetar los programas de la distribución, también de la infraestructura web, el sistema de control de errores, la “internacionalización”(adaptación a países e idiomas), las listas de correo de Debian de desarrollo y mantenimiento, y en un sentido amplio, a toda la infraestructura que hace la distribución Debian posible.

Los desarrolladores de Debian empaquetan el software que obtienen de los autores originales (“upstream”), asegurándose de que funciona correctamente con el resto de

los programas Debian. Para ello, hay un conjunto de reglas que todo paquete debe cumplir, el Manual de Política Debian (Debian Policy Manual) [DebianPol]. La mayor parte del esfuerzo de empaquetar un determinado programa generalmente consiste en hacerlo compatible con estas normas. Los desarrolladores también gestionan los errores en los programas, intentan solucionarlos (informando de problemas y soluciones a los autores originales), siguen el desarrollo de nuevos programas y construye_ todo el software intermedio necesario para que el sistema Debian funcione. Los fallos y los problemas de seguridad se discuten abiertamente, y diariamente se ponen a disposición de los usuarios actualizaciones para las distribuciones estables, para solucionar problemas importantes de forma que los sistemas permanezcan tan seguros y libres de errores como sea posible.

Debian es único por muchos motivos. Es destacable su dedicación al software libre, su naturaleza sin ánimo de lucro y su modelo abierto de desarrollo (donde la mayor parte de las discusiones se hacen en listas de correo públicas). El Proyecto Debian está comprometido con el software libre, como refleja el Contrato Social Debian. La definición de lo que Debian considera software libre se encuentra en las Directrices del Software Libre de Debian (DFSG, Debian Free Software Guidelines), que esencialmente es el mismo software que entra en la categoría “open source” . (Lo que no resulta extraño, ya que la definición “open source” deriva de la DFSG).

2.2. Debian potato

Debian 2.2 (potato) es la última distribución oficial, la que actualmente es considerada “estable”. Vio la luz en agosto de 2000, e incluye todo el software libre de cierta relevancia disponible en ese momento. Tan solo la distribución principal, compuesta únicamente por software libre (según la DFSG) consta de más de 2.600 paquetes de código fuente. La distribución completa incluye cerca de 4.000 paquetes binarios, que el usuario puede instalar fácilmente desde diversos soportes o desde servidores en Internet.

Debian 2.2 está divido en dos archivos, el “normal” y el **non-US**. Este último incluye paquetes que tienen algún impedimento legal para ser exportados desde Estados Unidos (generalmente la legislación estadounidense sobre criptografía)

Cada archivo está compuesto de varias “distribuciones”: Las llamadas **main**, **contrib** y **non-free**.

En el trabajo al que hacemos referencia en este artículo hemos considerado sólo la distribución **main** del archivo “normal”. Es con mucho la mayor parte del archivo, está compuesto sólo por software libre y no tiene restricciones de exportación. En muchos aspectos, es la mayor colección coordinada de software libre disponible en Internet.

3. Captura de Datos

En resumen, el enfoque empleado para la captura de los datos presentados en este artículo es el siguiente:

1. Código Fuente de una Distribución Debian

   Por fortuna, el código fuente de las distribuciones Debian presentes y pasadas está archivado, disponible en Internet para cualquiera. El único problema es determinar la lista de paquetes de código fuente para una determinada distribución, y dónde acceder a ellos.

2. Descarga y captura datos

   Una vez que conocemos los ficheros que nos interesan, debemos descargarlos para obtener los datos. Pero no los descargamos todos al tiempo (lo que ocuparía mucho espacio en disco), si no que, secuencialmente, descargamos un paquete, lo desempaquetamos, analizamos y borramos antes de pasar al siguiente.

3. Análisis final

   Análisis de los datos recogidos y obtención de estadísticas, atendiendo al número total de SLOC de la distribución, las SLOC para cada uno de los diversos lenguajes de programación considerados, etc.

En las siguientes secciones estos tres pasos están descritos con mayor detalle.

3.1. Código fuente de una distribución Debian

En el sistema de paquetes de Debian hay dos tipo de paquete: fuente y binario. De cada paquete fuente, de forma automática, se pueden construir uno o más paquetes binarios. Para este trabajo, sólo los paquetes fuente son relevantes, con lo que en lo sucesivo no volveremos a hacer referencia a los binarios.

Al construir un paquete fuente, un desarrollador de Debian comienza por el directorio con el código fuente "original" del programa en cuestión. En terminología Debian, este es el fuente "upstream" ("corriente-arriba"). El desarrollador Debian crea modificaciones en forma de parche de estos fuentes originales si es necesario, y crea un subdirectorio **debian** con todos los ficheros de configuración Debian (incluyendo los datos necesarios para construir los paquetes binarios). Se tiene entonces el paquete fuente, que generalmente (pero no siempre) consta de tres ficheros: Los fuentes originales (un fichero **tar.gz**), los parches para obtener el directorio Debian fuente (un fichero **diff.gz** con los parches y el directorio **debian**), y un fichero de descripción con extensión **dsc**. (aunque sólo las últimas distribuciones incluyen este fichero) Los paquetes Debian "nativos" (desarrollados por Debian, en los que no hay fuentes "upstream") no incluyen parches.

Los paquetes fuente de las distribuciones Debian actuales forman parte del archivo de Debian. En cada distribución están en el directorio **source**. En Internet hay servidores con los paquetes fuente de cada distribución Debian actual (generalmente "mirrors" de archive.debian.org (ftp://archive.debian.org)). Desde Debian 2.0, en cada distribución hay un fichero **Sources.gz** en el directorio **source**, con información sobre los paquetes fuente de la distribución, incluyendo los ficheros que componen cada paquete. Esta es la información que usamos para determinar qué paquetes fuente, y qué ficheros deben ser tenidos en cuenta para Debian 2.2.

De todas formas, hay que tener en cuenta que no todos los paquetes en **Sources.gz** deben ser analizados al contar líneas de código. La razón principal para no hacerlo es la existencia, en ocasiones, de varias versiones del mismo programa. Por ejemplo, en Debian 2.2 tenemos paquetes fuente **emacs19** (para **emacs-19.34**), y paquetes fuente **emacs20** (para **emacs-20.7**). Contar ambos paquetes implicaría contar Emacs dos veces, que no es lo que se pretende. Por tanto para cada distribución es necesaria una inspección manual, detectando aquellos que son esencialmente distintas versiones de

un mismo programa y eligiendo un “rep_esentante” para cada familia de versiones.

Estos casos pueden hacer infra-estimar el número de líneas de la distribución, ya que las diferentes versiones de un paquete puedan compartir una buena parte del código pero no todo. (Pensemos por ejemplo en PHP4 y PHP3, donde el primero está re-escrito casi por completo). De todas formas consideramos este defecto asumible, y compensado por otras sobre-estimaciones (como veremos después).

En otras ocasiones, hemos decidido analizar paquetes que pueden tener cantidades significativas de código en común. Este es el caso, por ejemplo de **emacs** y **xemacs**. Si bien el segundo es una ramificación del segundo y ambos comparten una buena parte del código, sin ser iguales son evoluciones del mismo “antepasado”. Otros casos similares son **gcc** y **gnat**. El segundo, un compilador Ada, está construido sobre el primero (un compilador C), añadiendo muchos parches y mucho código nuevo. En ambos casos consideramos que el código es lo bastante distinto como para considerarlo paquetes diferentes. Esto probablemente lleva a sobre-estimar el número de líneas de código de la distribución.

El resultado de este paso es la lista de paquetes (y los ficheros que la componen) que consideramos para analizar el tamaño de una distribución Debian. Esta lista está hecha a mano para cada distribución (con ayuda de unos scripts muy simples).

3.2. Descarga y captura de datos

Una vez que se establecen los paquetes y ficheros de Debian 2.2 a analizar, se descargan de alguno de los servidores de la red de “mirrors” Debian. Para este paso usamos algunos sencillos scripts Perl. El proceso consta de las siguientes fases:

- Descarga de los ficheros que componen el paquete.
- Extracción del directorio fuente correspondiente al paquete original. (Descomprimiendo el fichero **tar.gz**). Tras esto se consiguen los datos sobre los fuentes originales.
- Aplicación del parche al directorio original con el fichero **diff.gz** file, para obtener el directorio fuente Debian. Tras la extracción, se obtienen los datos de este directorio.

- Borrado del directorio **debian** para evitar contar los scripts que contiene, que son los hechos por el desarrollador de Debian.

No todos los paquetes tienen versión original (“upstream”), por lo que en en este proceso hay que prestar atención a estas situaciones.

La captura de los datos se hace usando los scripts de **sloccount**, tres veces por paquete (una vez en cada fase, ver el párrafo anterior), que almacenan número de líneas de código de cada paquete en un directorio diferente, preparado para un análisis e informe posterior.

La razón de recoger datos tres veces por cada paquete es el analizar el impacto del desarrollo Debian en el paquete original. Este impacto puede ser en forma de parches al original (con frecuencia para hacerlo más estable y seguro, para hacerlo consistente con la política de instalación Debian o para añadir alguna funcionalidad) o de scripts de instalación (que pueden ser identificados al medir los fuentes sin el directorio debian).

El resultado final de este paso es la recopilación de todos los datos obtenidos de los paquetes descargado, organizados por paquete, y listos para ser analizados. Estos datos consisten fundamentalmente en listas de ficheros, con el número de lineas de código que corresponden a cada uno, detallando subtotales por lenguaje de programación.

3.3. Análisis final

El último paso es la generación de informes, usando **sloccount** y algunos scripts, para estudiar los datos obtenidos. Ya que en este punto todos los datos obtenidos están disponibles localmente, y resulta sencillo procesarlos, el análisis puede ser hecho rápidamente y reiterado automáticamente de forma sencilla, buscando diferentes tipos de información.

El resultado final de este paso es un conjunto de informes y análisis estadísticos, usando los datos obtenidos en el paso anterior, y considerándolos desde diferentes puntos de vista. Estos resultados se presentan en la siguiente sección.

4. Resultados de la medición de Debian

Los resultados principales de nuestro análisis de la distribución Debian 2.2 GNU/Linux se pueden organizar en las siguientes categorías.

- Tamaño de Debian potato.
- Relevancia de los lenguajes de programación más usados.
- Análisis de la evolución en el tamaño de los paquetes más importantes.
- Estimaciones de esfuerzo.

4.1. Tamaño de Debian Potato

Hemos contado el número de líneas físicas de código fuente de Debian GNU/Linux 2.2 de tres formas, con los siguientes resultados (todas las cifras son aproximadas, consultar el apéndice para más detalles):

- Número de líneas en paquetes originales “upstream” “tal cual”: *52.810.000* SLOC
- Número de líneas en paquetes fuente Debian: *56.180.000* SLOC
- Número líneas en paquetes fuente Debian sin directorio **debian**: *55.920.000* SLOC

Para más detalles sobre el significado de cada categoría, el lector puede volver a la subsección “ Descarga y Captura de datos”. En resumen, la medida de los paquetes “upstream” originales puede ser considerado como el tamaño del software original usado en Debian. La medida de los paquete fuente Debian representa la cantidad de código presente actualmente en la distribución Debian 2.2, incluyendo tanto el trabajo de los autores originales como el de los desarrolladores Debian. Este último incluye los scripts relativos a Debian y los parches. Los parches pueden estar hechos por los desarrolladores Debian (por ejemplo aquellos que adaptan un paquete a la _ política Debian) o pueden ser obtenidos de algún otro sitio. La medida de los paquetes Debian sin el directorio **debian** excluye los scripts relacionados con Debian, por lo que son una

buena medida del tamaño de los paquetes tal y como aparecen en Debian, excluyendo los scripts referentes a Debian.

Es importante tener en cuenta que los paquete desarrollados específicamente para Debian en general no tienen paquete fuente “upstream” original. Este es por ejemplo el caso de **apt**, presente sólo como paquete fuente Debian.

4.2. Lenguajes de Programación

El número de SLOC físico para los paquetes fuente originales, clasificado por lenguajes de programación es, redondeando :

- C: *39.960.000* SLOC (71,12%)
- C++: *5.500.000* SLOC (9,79%)
- LISP: *2.800.000* SLOC (4,98%)
- Shell: *2.640.000* SLOC (4,70%)
- Perl: *1.330.000* SLOC (2,36%)
- FORTRAN: *1.150.000* SLOC (2.04%)
- Tcl: *550.000* SLOC (0.99%)
- Objective C: *425.000* SLOC (0.76%)
- Ensamblador: *425.000* SLOC (0.75%)
- Ada: *405.000* SLOC (0.73%)
- Python: *360.000* SLOC (0.65%)

Por debajo del 0,5% encontramos otros lenguajes: Yacc (0,46%), Java (0,20%), Expect (0,20%), Lex (0,13%), y otros con porcentaje inferior al 0,1%.

Cuando contamos las líneas de los paquetes fuente Debian sin el directorio **debian** (que contiene ficheros de configuración del paquete y scripts hechos por el desarrollador de Debian) las cifras son similares, lo que significa que estos scripts no representan una parte significativa de la distribución. La principal diferencia está en las lineas de Shell

(unas 150.000 menos) y en la líneas Perl (unas 80.000 menos), lo que revela los lenguajes preferidos para estos scripts.

A pesa de esto, al contar los paquetes “upstream” originales hay algunas diferencias destacables: Unas 2.000.000 líneas de código C, 300.000 líneas de LISP, 200.000 líneas FORTRAN, y pequeñas variaciones en otros lenguajes. Estas diferencias se deben generalmente a los parches a los fuentes originales hechos por los desarrolladores Debian. Por tanto, consultado estos resultados, podemos saber en qué lenguajes están escritos la mayoría de los paquetes a los que se han aplicado parches.

4.3. Los mayores paquetes

Los mayores paquetes en la distribución Debian potato son:

- Mozilla (M18): 2.010.000 SLOC (2.010.000). Un total de 1.260.000 SLOC en C++, 702.000 en C. Mozilla es un navegador WWW muy conocido.
- Linux kernel (2.2.19): 1.780.000 SLOC (1.780.000). 1.700.000 SLOC en C, 65.000 en ensamblador. Los núcleos Linux 2.x eran la serie estable en la época de Debian 2.2.
- XFree86 (3.3.6): 1.270.000 SLOC (1.265.000). Fundamentalmente 1.222.000 SLOC de C. Es una implementación de X Window que incluye un servidor de gráficos y programas básicos.
- PM3 (1.1.13): 1.115.000 SLOC (1.114.000). 983.000 SLOC de C, 57.000 de C++. PM3 es la distribución de Modula-3 de la Escuela Politécnica de Montreal, incluye compilador y librerías.
- OSKit (0.97): 859.000 SLOC (859.000). Un total de 842.000 SLOC de C. OSKit es el “Flux Operating System Toolkit” un entorno de trabajo para el diseño de sistemas operativos.
- GDB (4.18): 801.000 SLOC (800.000). Incluye 727.000 lineas de C y 38.000 de Expect. GDB es el depurador GNU a nivel de fuente.

- GNAT (3.12p): 688.000 SLOC (687.000). Unas 410.000 SLOC de C y 248.000 SLOC de Ada. GNAT es el compilador GNU de Ada 95, incluyendo una serie de librerías.
- Emacs (20.7): 630.000 SLOC (629.000). 454.000 SLOC de LISP, 171.000 SLOC de C. Emacs es, entre otras muchas cosas, un editor de texto muy conocido .
- NCBI Librerías (6.0.2): 591.000 SLOC (591.000). La mayoría en C, 590.000 SLOC. Este paquete incluye librerías para aplicaciones de Biología.
- EGCS (1.1.2): 578.000 SLOC (562.000). Incluye 470.000 SLOC de C y 55.000 SLOC de C++. Este paquete incluye las librerías GNU de extensión de C++.
- XEmacs, base support (21): 513.000 SLOC (513.000). Casi todo en LISP , 510.000 SLOC. Includes the base extra Emacs LISP files needed to have a working XEmacs. Incluye una serie de ficheros en Emacs LISP necesarios para la parte básica de XEmacs

Los números entre paréntesis representan aproximadamente el SLOC de los paquetes fuente originales, el resto de las cifras es el SLOC aproximado de los paquetes fuente Debian. Sólo se muestran los datos de los lenguajes más relevantes en cada paquete. El lector puede advertir que en la mayor parte de los casos, las cifras en ambos casos son aproximadamente iguales, lo que evidencia que, en esos casos, lo añadido por los desarrolladores Debian es mínimo (aunque estas modificaciones puedan ser importantes).

Las versiones de las distribuciones no son, obviamente, las actuales, pero eran las disponibles cuando Debian 2.2 fue congelado (primavera de 2000). La clasificación podría ser diferente si los desarrolladores Debian tuvieran que empaquetar las cosas de otra forma. Por ejemplo, si todas las extensiones de Emacs estuvieran en el paquete Emacs, hubiera sido mucho mayor. En todo caso, los paquetes fuente de Debian generalmente encajan bien con la idea de paquete que se tiene en general, que es la que suelen tener los autores originales.

Los siguientes paquetes según su tamaño (entre 350.000 y 500.000 _ SLOC) son Binutils (ensamblador GNU, linker y utilidades binarias), TenDRA (compiladores y comprobadores C y C++), LAPACK (un conjunto de librerías para álgebra lineal) y el Gimp (el paquete GNU de manipulación de imágenes). La mayoría de estos paquetes

están escritos en C, excepto LAPACK, escrito principalmente en FORTRAN.

4.4. Esfuerzo y estimaciones del coste

Usando el modelo COCOMO básico [Boehm1981] , se puede estimar el esfuerzo necesario para construir un sistema con el tamaño de Debian 2.2. Esta estimación supone un modelo de desarrollo de software propietario “clásico”, con lo que no es válido para estimar el esfuerzo aplicado a la construcción de este software. Pero al menos puede darnos un orden de magnitud del esfuerzo que sería necesario si se hubiera empleado un modelo de desarrollo de software propietario.

Aplicando el contador de SLOC para los paquetes fuente Debian los datos que proporciona el modelo COCOMO básico son los siguientes:

- SLOC físicas totales: 56.184.171
- Esfuerzo estimado: 171.141 personas-mes (14.261 personas-año)

  Formula: *2.4 * (KSLOC**1,05)*
- Tiempo estimado: 72.53 meses (6.04 años)

  Formula: 2,5 * (Esfuerzo**0,38)
- Coste estimado del desarrollo: 1.848.225.000 dólares

Para calcular la estimación de costes, hemos usado el salario medio para un programador de sistemas a tiempo completo en el año 2000, que según Computer World [ComWorld2000] es de 54.000 dólares USA al año, con un factor de conversión de 2,4.

5. Algunos comentarios y comparaciones

Los números ofrecidos en las secciones anteriores no son más que estimaciones. Pueden darnos al menos órdenes de magnitud, y permitir las comparaciones, pero no deben ser tomados como datos exactos, hay demasiadas fuentes de error, así como margen para diversas interpretaciones. En esta sección discutiremos algunas de las presunciones más importantes que se han hecho, con el objetivo de aportar contexto al lector para interpretar los números.

5.1. Qué es una línea de código fuente

Ya que dependemos de la herramienta **sloccount** de David Wheeler para medir el SLOC, también dependemos de su definición de "líneas físicas de código fuente". Por tanto, podemos decir que contabilizamos una SLOC cuando **sloccount** lo hace, si bien **sloccount** ha sido cuidadosamente diseñado para responder a la definición habitual de SLOC físicas: *" una línea física de código fuente es una linea que acaba en un marcador de nueva línea o de fin de fichero, y que contiene al menos un carácter que no es un espacio en blanco ni un comentario.;"* .

Hay otra medida similar que se prefiere en ocasiones, el SLOC "lógico". Por ejemplo, una línea escrita en C con dos punto y coma sería considerado como dos SLOC lógico, mientras que sería un solo SLOC físico. En todo caso, para los propósitos de este artículo (como para casi cualquier otro), las diferencias entre ambos SLOC son despreciables, especialmente comparadas con otras fuentes de error e interpretación.

5.2. Fuentes de imprecisión en la medición de SLOC

Las mediciones de líneas de código presentadas en este artículo no son más que estimaciones. En ningún caso pretendemos afirmar que sean exactas, especialmente cuando se refieren a agregación de paquetes. Hay varios factores que causan imprecisiones, algunas debidas a las herramientas empleadas para contar, otras, debidas a la selección de los paquetes.

- Algunos ficheros pueden no haber sido medidos de forma precisa.

Si bien **sloccount** incluye heurísticos cuidadosamente diseñados para detectar ficheros de código fuente, y para distinguir las líneas de código de los comentarios, estos heurísticos no siempre funcionan de la manera prevista. Además, con frecuencia es difícil distinguir los ficheros generados automáticamente (que no deberían ser contados), aunque **sloccount** hace un buen esfuerzo para reconocerlos.

- No todos los lenguajes de programación son reconocidos.

  Para capturar los datos usamos la versión 1.9 de **sloccount**, que reconoce unos 20 lenguajes distintos. De todas formas, algunos de los lenguajes empleados en Debian (como Modula-3 o Erlang) no están soportados. Obviamente esto lleva a una infra-estimación en los paquetes con ficheros escritos en estos lenguajes.

- Distintas percepciones en la agregación de familias de paquetes y en la selección de los representativos.

  Como comentamos en la subsección donde tratamos sobre la selección de la lista de paquetes a medir, las razones para incluir o excluir un paquete no son incuestionables. ¿Deberíamos contar distintas versiones del mismo paquete? ¿Deberíamos contar sólo una vez el código presente en varios paquetes o no? El criterio habitual para medir SLOC es "líneas de código fuente distribuido". Desde este punto de vista, todos los paquetes debieran ser considerados tal y como aparecen en la distribución Debian. En todo caso esto es difícil de aplicar cuando algunos paquetes son claramente evoluciones de otros. En vez de considerarlos todos como "distribuidos", parece más productivo considerar los más antiguos como "versiones beta". De todas formas en el ámbito del software libre es bastante frecuente el distribuir versiones estables cada 6 o 12 meses. Estas versiones estables representan mucho trabajo sólo para asegurar su estabilidad, aunque sólo sean la base para posteriores versiones.

  En la mayoría de los casos, hemos adoptado una decisión intermedia: Contar sólo familias de paquetes que sean una línea de evolución (como en el caso de **emacs19** y **emacs20**, pero contar por separado familias de paquetes que comparten algo de código pero que son en sí mismas desarrollos destinos (como en el caso de **gcc** y **gnat**).

5.3. Estimación de esfuerzo y _oste

Los modelos de estimación actual, y específicamente COCOMO, sólo consideran modelos clásicos de desarrollo de software propietario. Pero los modelos de desarrollo de software libre son bastante distintos. De esta forma sólo podemos estimar el coste del sistema si hubiera sido desarrollado por métodos convencionales, pero no los costes reales (en esfuerzo o dinero) del desarrollo del software incluido en Debian 2.2

Algunas de las diferencias que hacen imposibles el uso de estos modelos de estimación son:

- Proceso continuo de distribuciones. El modelo COCOMO está basado en el concepto de “SLOC de versiones distribuidas” , lo que implica un punto en el ciclo de vida del proyecto en que el producto es distribuido. A partir de ese momento, el principal esfuerzo de desarrollo está dedicado al mantenimiento. Por el contrario, la mayoría del software libre libera versiones con tanta frecuencia que podría ser considerado como un proceso de distribuciones continuas. Esto conlleva una casi continua estabilización del código, al mismo tiempo que evoluciona. Los proyectos de software libre suelen mejorar y modificar los programas al mismo tiempo que los preparan para los usuarios finales.
- Control y solución de error. Mientras que todos los sistemas de software propietario precisan de costosos ciclos de depuración, el software libre cuenta con la ayuda de voluntarios ajenos al proyecto, en forma de valiosos informes de errores e incluso soluciones para ellos.
- Reutilización, evolución e intercambio del código. En los proyectos de software libre es común la reutilización de código de otros proyectos de software libre como una parte central del sistema en desarrollo. También es frecuente que varios proyectos evolucionen del mismo sistema base, e incluso todos usen código de todos al mismo tiempo. A veces esto mismo puede suceder en desarrollos propietarios, pero incluso en grandes empresas con muchos proyectos abiertos, no es común, es mucho más frecuente en proyectos de software libre.
- Modelo de desarrollo distribuido. Aunque algunos sistemas propietarios son desarrollados por equipos distribuidos geográficamente, el grado de distribución que

suelen presentar los proyectos de software libre es varios órdenes de magnitud superior. Hay excepciones, pero en general los proyectos de software libre corren a cargo de personas de diferentes países, que no trabajan en la misma empresa, que dedican distinta cantidad de esfuerzo al proyecto, que cooperan principalmente a través de Internet, y que, la mayor parte de las veces (especialmente en proyectos grandes), el equipo de desarrollo nunca ha coincidido físicamente en un mismo sitio.

Algunos de estos factores incrementan el esfuerzo necesario para construir el software, mientras que otros lo decrementan. Sin analizar detalladamente el impacto de estos (y otros) factores, los modelos de estimación en general, y COCOMO en particular no son directamente aplicables al desarrollo de software libre.

5.4. Comparación con los tamaños estimados de otros sistemas

Para poner en contexto las cifras mostradas anteriormente, aportamos estimaciones del tamaño de algunos Sistemas Operativos, y una comparación más detallada con las estimaciones de la distribución Red Hat Linux.

Como se indica en [Lucovsky2000] (para Windows 2000) [Wheeler2001] (para Red Hat Linux) [Schneier2000] (para el resto de los sistemas) este es el tamaño estimado para varios Sistemas Operativos, en líneas de código. (Siempre en cifras aproximadas):

- Microsoft Windows 3.1: 3.000.000
- Sun Solaris: 7.500.000
- Microsoft Windows 95: 15.000.000
- Red Hat Linux 6.2: 17.000.000
- Microsoft Windows 2000: 29.000.000
- Red Hat Linux 7.1: 30.000.000
- Debian 2.2: 56.000.000

Casi todas las estimaciones (en realidad, todas, excepto las de Red Hat) no son detalladas, por lo que es difícil saber qué consideran una línea de código. De todas formas, las estimaciones deberían ser lo suficientemente próximas a las mediciones de SLOC como para que sean válidos para una comparación.

Nótese que mientras que Red Hat y Debian incluyen muchas aplicaciones, con frecuencia varias aplicaciones del mismo tipo de programa, tanto los Sistemas Operativos de Microsoft como los de Sun son mucho más limitados en este sentido. Si las aplicaciones más usadas en estos entornos fueran contabilizadas juntas, el tamaño sería mucho mayor. En todo caso, también es cierto que todas esas aplicaciones ni son desarrolladas ni integradas por el mismo equipo de desarrollo, como en el caso de las distribuciones basadas en Linux.

A partir de estas cifras, queda claro que las distribuciones basadas en Linux, en general, y Debian 2.2 en particular, son unas de las mayores colecciones de software nunca integradas por un equipo de desarrollo.

5.5. Comparación con Red Hat Linux

El único Sistema Operativo para el que hemos encontrado medidas detalladas de líneas de código fuente es Red Hat Linux (consultar“ Estimating Linux’s Size” y “More Than a Gigabuck: Estimating GNU/Linux’s Size;”). Siendo también una distribución basada en Linux, donde los paquetes software incluidos en Debian y en Red Hat son bastante similares, la comparación puede ser ilustrativa. Además, siendo Red Hat Linux una distribución muy común, probablemente la más conocida, la comparación con ellas puede aportar un contexto apropiado para el lector familiarizado con ella.

Lo primero que nos sorprendió cuando medimos Debian 2.2 fue su tamaño, comparado con Red Hat 6.2 (Distribuido en Marzo de 2000) y con Red Hat 7.1 (distribuido en Abril de 2001). Debian 2.2 aparece en agosto de 2000 , y su tamaño es casi el doble que el de Red Hat (Distribuido unos seis meses después) y más de tres veces el tamaño de Red Hay 6.2 (distribuido cinco meses antes)_ Algunas de estas diferencia pueden ser debidas a diferentes consideraciones sobre qué paquetes incluir al medir, pero con todo aportan una buena idea de los tamaños relativos.

El principal factor causante de las diferencias es el número de paquetes incluidos en cada distribución, en el caso de Debian hemos considerado 2630 paquetes fuente (con una media de unos 21.300 SLOC por paquete), mientras que Red Hat 7.1 incluye sólo 612 paquetes (de unos 49.000 SLOC por paquete)

Cuando se comparan los paquetes mayores en ambas distribuciones, encontramos en Debian todos los incluidos en Red Hat, lo que no es cierto a la inversa: Varios paquetes que suman una buena cantidad de SLOC en Debian no están presentes en Red Hat. Por ejemplo, entre los 11 mayores paquetes en Debian 2.2, los siguientes no están en Red Hat 7.1: PM3 (unos 1.115.000 SLOC), OSKit (unos 859.000 SLOC), GNAT (688.000), NCBI (591.000). Por el contrario, entre los 11 paquetes mayores en Red Hat 7.1, no echamos ninguno en falta en Debian 2.2.

De todas formas hay una gran colección de paquetes software presentes en Red Hat 7.1 y no en Debian 2.2: El escritorio KDE y sus utilidades. Debido a problemas de licencia, Debian decidió no incluir software KDE hasta después de Debian 2.2, cuando la licencia para Qt cambió a GPL. Por tanto, podemos decir que Debian 2.2 es mayor, incluso sin todo el código de KDE. Sólo para dar una idea, los mayores paquetes KDE en Red Hat 7.1 son kdebase, kdelibs, koffice, y kdemultimedia, que suman sobre 1.000.000 SLOC. Todos ellos están ausentes en Debian. Esto sugiere que si las medidas hubieran sido hechas en la distribución actual Debian (Aún no distribuida oficialmente), las diferencias hubieran sido mayores.

Las diferencias entre el mismo paquete en cada distribución son atribuibles a las diferentes distribuciones incluidas en ellos. Por ejemplo, el kernel Linux suma 1.780.000 SLOC (Versión 2.2.19) en Debian 2.2, mientras que el mismo paquete suma 2.437.000 SLOC (Versión 2.4.2) en Red Hat 7.1. XFree incluye 1.270.000 SLOC (Versión 3.3.6) en Debian 2.2, mientras que la versión incluida en Red Hat 7.1 (XFree 4.0.3) suma 1.838.000 SLOC. Estas diferencias hacen difícil el comparar directamente Red Hat y Debian.

El lector debe percatarse de que hay una diferencia metodológica entre el estudio de Red Hat y el nuestro sobre Debian. El primero extrae todos los paquetes fuente y usa “checksum” MD5 para evitar duplicados entre todo el código de la distribución. En el caso de Debian, hemos extraído los paquetes uno a uno, evitando paquetes duplicados pero no ficheros individuales repetidos. De todas formas, la suma total no debería verse

muy afectada por esta diferencia.

6. Conclusiones y Trabajo Relacionado

Es importante darse cuenta de que estas mediciones pueden representar aproximadamente toda la colección de paquetes de software libre estable disponibles para GNU/Linux en el momento de la aparición de Debian 2.2 (Agosto de 2000). Naturalmente, hay software libre no incluido en Debian, pero si hablamos de programas bien conocidos, usables y estables, la mayoría han sido empaquetados por un desarrollador Debian e incluidos en la distribución Debian. Por tanto, con ciertas precauciones puede decirse que el software del que estamos hablando suponía unos 60.000.000 SLOC en agosto de 2000. Usando el modelo COCOMO, esto implicaría un coste (usando las técnicas convencionales de desarrollo de software propietario) de cerca de 2.000 millones de dólares y un esfuerzo de más de 170.000 personas/mes.

También podemos comparar esta medición con otras distribuciones Linux, especialmente Red Hat. En números redondos, el tamaño de Debian 2.2 es el doble que el de Red Hat 7.1, que es unos ocho meses posterior. También es mayor que el último Sistema Operativo de Microsoft (Aunque como vimos en la sección correspondiente, esta comparación puede ser engañosa).

Entrando en los detalles, pueden verse algunos datos interesantes. Por ejemplo, el lenguaje más empleado en la distribución es C (más del 70%), seguido de C++ (cerca del 10%), LISP y Shell (sobre el 5%), Perl y FORTRAN (un 2% aproximadamente). Los mayores paquetes en Debian 2.2 son Mozilla (unas 2.000.000 SLOC), el núcleo Linux (unas 1.800.000 SLOC), XFree86 (1.250.000) y PM3 (Más de 1.100.000).

No hay muchos estudios detallados sobre el tamaño de Sistemas Operativos modernos y completos. Entre ellos, el trabajo de David Wheeler midiendo Red Hat 6.2 y Red Hat 7.1 es la referencia más parecida al presente articulo. Otros trabajos interesantes, con puntos en común con este es [GodfreyTu2000] , un estudio de la evolución a través del tiempo del núcleo Linux. Algunos otros artículos, ya citados, proporcionan mediciones globales de algunos Sistemas Operativos de Sun y Microsoft, pero no son lo bastante

detallados, excepto para indicar estimaciones.

Por último, es importante repetir una vez más que estamos dando sólo estimaciones, no cifras reales. Estas dependerían demasiado en la elección del software a medir y de algunos otros factores que ya hemos tratado aquí. Pero creemos que tienen la suficiente precisión como para extraer algunas conclusiones y para compararlos con otros sistemas.

7. Agradecimientos

Este artículo esta evidentemente inspirado en el de David A. Wheeler [Wheeler2001] del que hemos usado su herramienta sloccount. Sin su trabajo, el nuestro hubiera sido completamente impensable. También queremos agradecer los comentarios y sugerencias de muchos desarrolladores de Debian, que nos han ayudado a mejorar este artículo.

8. Sobre los autores

- Jesús M. González Barahona es profesor en la Universidad Rey Juan Carlos (Madrid), y colaborador de BarraPunto.Com. Comenzó a trabajar en la promoción del software libre en 1991, en el grupo PDSOFT (más adelante grupo Sobre). Desde entonces, ha realizado diversas actividades en este área, como la organización de seminarios, la realización de cursos y la participación en grupos de trabajo sobre software libre. Actualmente colabora con varios proyectos de software libre (entre ellos Debian y La Espiral), colabora con asociaciones como Hispalinux y EuroLinux, escribe en varios medios sobre temas relacionados con software libre, y asesora a empresas en sus estrategias relacionadas con estos temas.
- Miguel A. Ortuño Pérez es Ingeniero en Informática, profesor de la Universidad Rey Juan Carlos (Madrid), donde su área de interés es la computación móvil y el software

libre. Previamente trabajó en la Universidad de Oviedo en varios proyectos relacionados con sistemas de formación a distancia.

- José Centeno González es profesor en la Universidad Rey Juan Carlos. Se incorporó al Departamento de Informática de la Universidad Carlos III de Madrid en 1993, donde trabajó hasta 1999. Sus intereses en investigación incluyen la programación de sistemas distribuidos, los protocolos de comunicaciones y la movilidad. También está interesado por el impacto del software libre en la docencia en Ingeniería Informática y en la industria. Es Ingeniero de Telecomunicación por la Universidad Poliécnica de Madrid, en la que espera finalizar el doctorado este año.
- Pedro de las Heras Quirós es profesor en la Universidad Rey Juan Carlos, y colaborador de BarraPunto.Com. Desde principios de los 90 ha sido usuario de software libre, habiendo colaborado en el grupo Sobre dedicado al software libre desde su creación. Ha participado en la organización de congresos, expos y cursos relacionados con software libre, y ha sido editor y autor de varias publicaciones relacionadas con el software libre.
- Vicente Matellán Olivera es profesor en la Universidad Rey Juan Carlos (Madrid, España). Ha realizado varias actividades relacionadas con el software libre, entre ellas la creación de OpenResources.com y BarraPunto.com.

Bibliografía

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

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

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

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

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

[DFSG] Debian Project, *Debian Free Software Guidelines*, http://www.debian.org/social_contract#guidelines .

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

[Lucovsky2000] Mark, Lucovsky, August 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/ .

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

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