UNIVERSIDAD
REY JUAN CARLOS

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

Curso Académico 2009/2010

Proyecto Fin de Carrera

CarpeDM:
SISTEMA DE ANÁLISIS DE SEMEJANZAS ENTRE FICHEROS DE CÓDIGO FUENTE

Autor: Alberto Bermudo Delgado

Tutor: Jeśus M. González Barahona

A mi familia.

Agradecimientos

Agradezco a mi tutor Jesús Gonzalez Barahona y a todo el grupo *GSyC/Libresoft* por el apoyo prestado.

También agradezco a mi familia por toda su ayuda.

Muchas gracias.

Resumen

El software libre garantiza el acceso al código fuente de los programas. Debido a este acceso público al código fuente, en el software libre es más habitual un porcentaje destacable de código fuente reutilizado o semejante.

Esta poco estudiado cuanto código fuente ha sido reutilizado o semejante. Por este motivo, los esfuerzos de este proyecto están orientados en obtener respuestas en torno a esta incógnita. Para ello el proyecto provee herramientas que obtienen información del código fuente del software libre, analizan esa información en busca código fuente semejante y devuelve respuestas sobre ese código, utilizando una aproximación novedosa: el uso de algoritmos de resumen tipo NILSIMSA.

Se ha estimado el estudio de una distribución de software libre, Debian 3.1, como un punto de referencia válido. El desarrollo del proyecto ha seguido una línea en espiral hacia la herramienta final.

Se ha probado las distintas aproximaciones hacia la herramienta final, y se ha podido estimar la cantidad de código similar o repetido , a nivel de ficheros.

Algunos de los resultados obtenidos son que para Debian 3.1 se ha obtenido 8.716 paquetes, compuestos a su vez por 2.403.211 ficheros.
Para los 2.403.211 ficheros, se ha obtenido un total de 194.172 códigos MD5 repetidos y 202.177 códigos NILSIMSA repetidos.
Estas cifras nos indican que en este sistema de software hay código repetido y durante los próximos capítulos se estudian los detalles de estos datos.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

El presente proyecto busca respuestas a la incógnita de la semejanza dentro del código libre, así como medir el código repetido a nivel de ficheros en Debian. Para ello se generan herramientas software que sean capaces de extraer información relevante para el estudio del software libre, almacenar y recuperar la información extraída, y finalmente analizar dicha información para obtener respuestas.

Durante este capítulo se detallan los conceptos de algoritmos HASH, MD5, SHA-1 y NILSIMSA, usados actualmente para el análisis de clones y semejanzas de ficheros. También se describen que mecanismos usa el software libre para ofrecer públicamente su código fuente. Seguidamente se introducirá el lenguaje programación usado en este proyecto, Python, y los motivos que propiciaron su elección.

Este proyecto se ha centrado en el análisis de una distribución de GNU/Linux de código libre como es Debian, cuyas características han facilitado el análisis de la misma.

Las herramientas desarrolladas obtienen respuestas enfocadas a contabilizar cuanto software es igual, muy parecido o poco parecido dentro de una mismo conjunto de software libre como es la distribución Debian analizada.

Este primer capítulo es una descripción de problema, introducción de objetivos y descripción general de la herramienta. El segundo capítulo describe los antecedentes de estudio anteriores a este proyecto, y conceptos básicos como son los algoritmos MD5, NILSIMSA o el lenguaje de programación Python. En el tercer capítulo analiza el diseño de la herramienta desarrollada. El cuarto capítulo se incluyen los resultados extraídos a partir de dicha herramienta. Las conclusiones de este proyecto son expuestas en el capítulo final.

1.1. Descripción del problema

La situación inicial es la disposición del código fuente de una distribución de software libre GNU/Linux, en este caso en concreto Debian. A cualquier sistema de software libre se le presupone una cantidad de código igual o semejante.

La incógnita de la cantidad de código semejante existente en una distribución de software libre es el principal problema a solucionar.

Para la búsqueda de soluciones a la incógnita principal, existen varias aproximaciones o granularidades. Las granularidades pueden ir desde una granularidad fina, como es una búsqueda a nivel de palabras, funciones, comentarios, etc, a una granularidad mas elevada como son las búsquedas a nivel de fichero, paquetes, subsistemas, etc.

Este proyecto en concreto se centra en la búsqueda a nivel de ficheros, esto es, se trata de encontrar ficheros semejantes.

Existen varias herramientas para comparar ficheros entre sí. Una de las más extendidas es MD5, la cual resuelve la incógnita de si dos ficheros son exactamente iguales. Para muchos estudios esta herramienta sería suficiente, pero en este caso necesitamos una herramienta que además sea capaz de establecer un grado de similitud entre dos ficheros, que este en el entorno intermedio entre que un fichero sea igual a otro y no lo sea.

Se ha estudiado la posibilidad que ofrece una herramienta como NILSIMSA, para poder ejecutar la tarea de establecer similitudes. Nilsimsa se usa a menudo como herramienta para combatir el *spam* en el correo electrónico. En este entorno, NILSIMSA ha demostrado ser una herramienta eficiente y a priori parece ser un candidato que se ajusta a las necesidades de este proyecto.

Por lo tanto se usara NILSIMSA como herramienta de apoyo en la búsqueda de soluciones entorno a la incógnita principal del proyecto.

1.2. Objetivos

Este proyecto tiene como principal objetivo construir una herramienta con las siguientes funciones:

- Obtener información del código fuente de una distribución GNU/Linux.
- Analizar esa información en busca de ficheros similares.
- Obtener respuestas sobre los ficheros similares.
- Desarrollar una herramienta que facilita la realición del analisis.

Como se vera en la sección 1.4, el software libre tiene a disposición pública su código fuente. Dicho código es conocido por ser reutilizado y semejante en muchos casos, pero por ahora no son conocidos los índices de reutilización.

El sistema desarrollado en este proyecto trata de responder como de semejante es el código fuente de una distribución de software libre.

1.3. Sistema CarpeDM

El sistema desarrollado, CarpeDM, es un conjunto de herramientas que responde a los objetivos descritos anteriormente en la sección 1.2.

- Analiza fuentes de información realmente extensas, como es el código fuente de la distribución Debian GNU/Linux, de mas de 200 millones de líneas de código.
- Crea automáticamente la base de datos que necesita para realizar consultas necesarias para obtener respuestas.
- Obtiene respuestas concretas a la incógnita de las semejanzas en el código fuente del software libre.

Aun así el sistema esta limitado en algunos aspectos tales como:

- Se analiza un único repositorio de código fuente a la vez, en este caso una única distribución Debian GNU/Linux.

- El sistema necesita que el repositorio este correctamente generado y no tenga incoherencias en su escructura.
- El tiempo de procesamiento puede ser elevado dependiendo del repositorio de código fuente.

1.4. Software libre

El software libre posibilita el acceso público a su código fuente. Debido a esta característica, es posible realizar un estudio sobre que contiene dicho código fuente. Realizar un estudio sobre el código fuente de un software no libre, hubiera sido una tarea altamente costosa debido a las restricciones de licencias que ello conlleva. Por tanto, no hay mejor caso de estudio que el análisis del código de una distribución linux como es Debian GNU/Linux.

Capítulo 2

Antecedentes

En este capitulo se muestra inicialmente algunos de los estudios anteriores sobre el código fuente de GNU/Linux. En ellos se realizan diversos análisis sobre sistemas grandes de software, análogamente a este proyecto.
Estos estudios giran entorno a diversos objetivos muy interesantes como son el análisis cuantitativo de lineas de código, licencia, estudios de clones, etc, los cuales ha servido de base para realizar el estudio de semejanzas a nivel de fichero de este proyecto.
También en este capitulo se describen parte de los algoritmos de resumen, alguno de ellos usados en este proyecto, como son MD5, SHA-1 y NILSIMSA. Dichos algoritmos son muy útiles, ya que proveen una forma abreviada del objeto a analizar, repercutiendo directamente en la velocidad del análisis. Por ultimo se describe el lenguaje de programación Python, el cual ha sido seleccionado para este proyecto por su simplicidad y vertibilidad, entre otras razones.

2.1. Estudios sobre código fuente de distribuciones GNU/Linux

David A. Wheeler estudió en 2001 [Wheeler, 2001] el código fuente de la distribución GNU/Linux Red Hat 7.1.

Para ello usa el siguiente procedimiento:

- Instalar el código fuente en un formato sin comprimir.
- Contar el número de líneas de código fuente físicas (SLOC).
- Usar un modelo de estimación para estimar el esfuerzo y coste para desarrollar el mismo sistema de una forma propietaria.

- Determinar las licencias de software de cada componente para realizar estadísticas.

Para su estudio no se incluyo versiones antiguas de un mismo software. Con el fin de identificar y eliminar ficheros duplicados, Wheeler uso sus códigos MD5. Como vemos, no es la primera vez que se ha usado MD5 para identificar duplicados en el código fuente del software libre.

En este caso en concreto, el uso de MD5 se uso para discriminar ficheros en el estudio, sin embargo uno de los objetivos de este proyecto es justamente el contrario, encontrar ficheros iguales o semejante dentro de una distribución GNU/Linux.

Posteriormente se realizo un estudio sobre la mayor distribución de GNU/Linux Debian basándose en el trabajo de Wheeler para Red Hat. Contando Patatas: El tamaño de Debian 2.2 [Gonzz-Barahona et al., 2001] estudia la distribución Debian 2.2 , formada por mas de 56.000.000 líneas físicas de código, casi el doble que Red Hat 7.1.

En este estudio se detalla la distribución de Debian, dividida en dos archivos , normal y non-US. El archivo non-US incluye paquetes que tienen algún impedimento legal para ser exportado desde Estados Unidos. Cada archivo se divide en tres distribuciones, denominadas main, contrib y non-free.

En el estudio de Debian se especifica que solo ser estudiara la distribución main del archivo normal, ya que es donde esta la mayor parte del archivo, esta compuesto solo por software libre y no contiene restricciones de exportación. En nuestro proyecto también se realizara un estudio de sobre la distribución main del archivo normal, pero de la versión 3.1 de Debian, también denominada Sarge.

El procedimiento empleado para el estudio de Debian 2.2 es el siguiente:

- Determinar el listado de paquetes a analizar y donde acceder a ellos.
- Se descargan los ficheros secuencialmente, se desempaquetan , se analiza y se borra antes de pasar al siguiente.

- Análisis de los datos recogidos y obtención de estadísticas.

En Debian se dispone de los paquetes fuente para cada distribución del sistema, como ya hemos visto con anterioridad. A partir de Debian 2.0 se dispone de un fichero Sources.gz en el directorio source de cualquier mirror de ftp://archive.debian.org. En el fichero Sources.gz esta contenida toda la información relativa a los paquetes fuente de la distribución Debian.

Veamos un ejemplo, en el fichero Sources.gz incluye la información del paquete zsh:

```
Package: zsh
Binary: zsh, zsh-doc, zsh-static
versión: 4.2.5-7
Priority: optional
Section: shells
Maintainer: Clint Adams <schizo@debian.org>
Build-Depends: texinfo, groff-base, libncurses5-dev, texi2html, libcap-dev [!hurd-i386 !freebsd-i386], bsdm
Architecture: any
Standards-versión: 3.6.1
Format: 1.0
Directory: pool/main/z/zsh
Files:
 c6af31817783e823f2475f2d4063369d 690 zsh_4.2.5-7.dsc
 5ba6b82a11bae5a2b96d5b9aba4cdeaa 2624122 zsh_4.2.5.orig.tar.gz
 8e1de3e70d42627d0d702877344242ba 314685 zsh_4.2.5-7.diff.gz
```

Es posible que en una distribución Debian, existan varias versiones de un mismo paquete. El estudio El tamaño de Debian 2.2 [Gonzz-Barahona et al., 2001], ha seleccionado manualmente un representante de las versiones de un mismo paquete, dejando el resto a un lado del estudio. Esto es de esta forma, solo en paquetes que son una línea de evolución (como en emacs19 y emacs20) Para el estudio de este proyecto se ha analizado la totalidad del archivo Debian, ya que el foco esta puesto en determinar igualdades y semejanzas en el código fuente.

2.2. Estudios sobre búsqueda de clones de software

Existen otros estudios focalizados en la búsqueda de clones de software. Rainer Koschke [Koschke, 2006] divide los tipos de clones de software en tres:

- Tipo 1: es una copia exacta sin modificaciones (excepto por los espacios en blanco y comentarios).

- Tipo 2: es una copia sintácticamente idéntica, solo los identificadores de variable, tipo o función han sido cambiados.

- Tipo 3: es una copia con modificaciones, algunas líneas han sido cambiadas, añadidas o eliminadas.

Una consecuencia de clonar código es el aumento del esfuerzo de mantenimiento. Cualquier cambio debe realizarse númerosas veces si el código es redundante.

En el estudio de 1998 Clone Detection Using Abstract Syntax Trees [Baxter et al., 1998], se analiza un sistema de más de 400.000 líneas de código C. Se establece que el porcentaje de clonado medio es del 12,7 para todos sus subsistemas. Como podemos comprobar para este caso en concreto, es un porcentaje nada despreciable.

En el estudio de 2007 A Survey on Software Clone Detection Research [Roy and Cordy, 2007] se describe que numerosos estudios muestran que entre el 5 y el 20 por ciento de los sistemas de software pueden contener código duplicado, lo cual es básicamente el resultado de copiar fragmentos de código y usarlos, pegándolos con y sin modificaciones. Roy [Roy and Cordy, 2007] describe algunas técnicas que son aplicadas en el código fuente antes de iniciar las comparaciones necesarias para encontrar código clonado. Se describen a continuación:

- Borrado de comentarios: ignora todos los tipos de comentarios en el código fuente.

- Borrado de espacios en blanco: se borran tabuladores, nuevas líneas y otros espacios en blanco.

- Normalización: algunos tipos de normalizaciones básicas pueden ser aplicadas en el código fuente.

En este proyecto, se ha hecho uso de la técnica de borrado de espacios en blanco, con el fin de disminuir los posibles casos falsos positivos.

2.3. Algoritmos HASH

Un algoritmos HASH genera clave única a modo de resumen para un fichero. Todas las claves generadas con una función de HASH tienen el mismo tamaño, sea cual sea el mensaje utilizado como entrada. No es posible reconstruir el mensaje original a partir de su HASH. A continuacion, se describen un algoritmo HASH, MD5, y un algoritmo casi HASH, NILSIMSA.

2.3.1. MD5

Acrónimo de Message-Digest Algorithm 5, Algoritmo de Resumen del Mensaje 5. MD5 es un algoritmo de reducción desarrollado en 1991. Su uso esta ampliamente extendido para comprobar que un archivo de descargado de internet no ha sido alterado. Usa 128 bits en un número de 32 dígitos hexadecimales, denominado suma MD5. Cualquier cambio en un fichero , cambiara drásticamente su código MD5.

Para dos ficheros , los cuales se diferencian en un solo carácter de su contenido, su suma MD5 es completamente diferente como se puede comprobar en el siguiente ejemplo.

Para mostrar por pantalla el contenido de un fichero usamos el comando *cat*[1].

Este es el contenido de un fichero de ejemplo llamado *documento1.txt*.

```
alberto@debian:~$ cat documento1.txt
Esto es el contenido del fichero de documento1.txt.
Este contenido es exactamente igual en los ficheros de ejemplo,
excepto en la parte donde se hace referencia al nombre de fichero.
En este caso en concreto es el último carácter antes de la extensión .txt.
Este contenido es exactamente igual en los ficheros de ejemplo,
excepto en la parte donde se hace referencia al nombre de fichero.
En este caso en concreto es el último carácter antes de la extensión .txt.
Este contenido es exactamente igual en los ficheros de ejemplo,
excepto en la parte donde se hace referencia al nombre de fichero.
En este caso en concreto es el último carácter antes de la extensión .txt.
Este contenido es exactamente igual en los ficheros de ejemplo,
excepto en la parte donde se hace referencia al nombre de fichero.
En este caso en concreto es el último carácter antes de la extensión .txt.
```

Este es el contenido de un fichero de ejemplo llamado *documento2.txt*.

[1]*cat* es un programa de GNU/Linux para concatenar y mostrar ficheros.

```
alberto@debian:~$ cat documento2.txt
Esto es el contenido del fichero de documento2.txt.
Este contenido es exactamente igual en los ficheros de ejemplo,
excepto en la parte donde se hace referencia al nombre de fichero.
En este caso en concreto es el último carácter antes de la extensión .txt.
Este contenido es exactamente igual en los ficheros de ejemplo,
excepto en la parte donde se hace referencia al nombre de fichero.
En este caso en concreto es el último carácter antes de la extensión .txt.
Este contenido es exactamente igual en los ficheros de ejemplo,
excepto en la parte donde se hace referencia al nombre de fichero.
En este caso en concreto es el último carácter antes de la extensión .txt.
Este contenido es exactamente igual en los ficheros de ejemplo,
excepto en la parte donde se hace referencia al nombre de fichero.
En este caso en concreto es el último carácter antes de la extensión .txt.
```

Para mostrar por pantalla el valor MD5 de un fichero usamos el comando *MD5sum*[2].

```
alberto@debian:~$ MD5sum documento1.txt documento2.txt
1652cb5e30de0d607b2a0e12590b048d  documento1.txt
796bf5aacdae16c3a7cb78ed67206c1a  documento2.txt
```

Una vez realizado un simple cambio en un carácter dentro de dos ficheros idénticos, el valor MD5 es completamente diferente. En este caso se ha cambiado el carácter *1* por el carácter *2*.

2.3.2. SHA-1

Acrónimo de Secure Hash Algorithm, Algoritmo de Hash Seguro. SHA-1 fue desarrollado en 1995 dos años antes de SHA-0. Usa 160 bits y se basa en principios generales usados en MD5. Tiene limitaciones de longitud de 264 bit que MD5 no tiene. Es mas seguro pero más lento que MD5. SHA-1 es el estándar como función HASH.

2.3.3. NILSIMSA

Nilsimsa es casi un algoritmo HASH. La codificación de NILSIMSA de 256 bits es representada típicamente como un número de 64 dígitos hexadecimal. Un cambio moderado en un fichero , no cambia drásticamente su código NILSIMSA. Para dos ficheros, los cuales se diferencian en un solo carácter de su contenido, su NILSIMSA

[2] *MD5sum* es un programa de GNU/Linux para mostrar el valor MD5 de ficheros.

es bastante parecido como se puede comprobar en el siguiente ejemplo.

Este es el contenido de un fichero de ejemplo llamado *documento1.txt*, usado en la sección 2.3.1 .

```
alberto@debian:~$ cat documento1.txt
Esto es el contenido del fichero de documento1.txt.
Este contenido es exactamente igual en los ficheros de ejemplo,
excepto en la parte donde se hace referencia al nombre de fichero.
En este caso en concreto es el último carácter antes de la extensión .txt.
Este contenido es exactamente igual en los ficheros de ejemplo,
excepto en la parte donde se hace referencia al nombre de fichero.
En este caso en concreto es el último carácter antes de la extensión .txt.
Este contenido es exactamente igual en los ficheros de ejemplo,
excepto en la parte donde se hace referencia al nombre de fichero.
En este caso en concreto es el último carácter antes de la extensión .txt.
Este contenido es exactamente igual en los ficheros de ejemplo,
excepto en la parte donde se hace referencia al nombre de fichero.
En este caso en concreto es el último carácter antes de la extensión .txt.
```

Este es el contenido de un fichero de ejemplo llamado *documento2.txt*, usado en la sección 2.3.1 .

```
alberto@debian:~$ cat documento2.txt
Esto es el contenido del fichero de documento1.txt.
Este contenido es exactamente igual en los ficheros de ejemplo,
excepto en la parte donde se hace referencia al nombre de fichero.
En este caso en concreto es el último carácter antes de la extensión .txt.
Este contenido es exactamente igual en los ficheros de ejemplo,
excepto en la parte donde se hace referencia al nombre de fichero.
En este caso en concreto es el último carácter antes de la extensión .txt.
Este contenido es exactamente igual en los ficheros de ejemplo,
excepto en la parte donde se hace referencia al nombre de fichero.
En este caso en concreto es el último carácter antes de la extensión .txt.
Este contenido es exactamente igual en los ficheros de ejemplo,
excepto en la parte donde se hace referencia al nombre de fichero.
En este caso en concreto es el último carácter antes de la extensión .txt.
```

Para mostrar por pantalla el valor NILSIMSA de un fichero usamos el comando *NILSIMSA*[3].

```
alberto@debian:~$ NILSIMSA documento1.txt documento2.txt
95580b310c55808ca54b1a08123bb298a67b2cc945643cc4ab636fda0ed2aeaf documento1.txt
15580b310c55808ca54b1a08123bb298a67b2cc945643cc4ab636fda0ed2aeaf documento2.txt
```

[3]*NILSIMSA* es un programa para mostrar el valor NILSIMSA de ficheros.

Una vez realizado un simple cambio en un carácter dentro de dos ficheros idénticos, el valor NILSIMSA de ambos es casi idéntico. En este caso la diferencia en el código NILSIMSA de *documento1.txt* y *documento2.txt* esta únicamente en el primer carácter del código NILSIMSA, un 9 y un 1 en cada caso.

Al comparar dos ficheros entre sí, NILSIMSA nos va a devolver un valor de comparación NILSIMSA entre -128 y 128. La razón se explica a continuación.

Para dos códigos de 256 bits seleccionados al azar esperamos tener una media de 128 bits iguales. Por lo tanto, el valor de comparación NILSIMSA entre dos ficheros es igual al número de bits iguales de sus códigos NILSIMSA menos 128.

Teniendo esto en cuenta esto, tenemos los siguientes casos de valor de comparación:

- Valor de comparación NILSIMSA 128: todos los bits de los dos códigos NILSIMSA de dos ficheros son iguales. Esto quiere decir que los dos ficheros son iguales o muy parecidos. Es un caso extremo.
- Valor de comparación NILSIMSA 0: es el caso medio de NILSIMSA, 128 bits de los 256 bits de los códigos NILSIMSA de cada fichero son iguales.
- Valor de comparación NILSIMSA -128: ninguno de los bits de los dos códigos NILSIMSA de dos ficheros son iguales. Esto quiere decir que los dos ficheros son completamente diferentes o con muy poco en común. Es un caso extremo.

Para el desarrollador de NILSIMSA, si el valor de comparación NILSIMSA de dos ficheros es superior a 24 se puede afirmar que dichos ficheros no han sido generados independientemente. Es decir al menos son iguales 152 bits de los 256 bits de sus códigos NILSIMSA

El código NILSIMSA se calcula dividiendo un fichero en varios partes. A cada parte del fichero se le asigna uno de los 256 bits correspondientes al código NILSIMSA, y el algoritmo NILSIMSA calcula un 0 o un 1 para cada bit.

Vamos a comprobar brevemente este comportamiento por medio de dos ficheros, los cuales suponemos a priori, son muy parecidos. Comprobaremos un fichero del paquete fuente Emacs de Debian, y el mismo fichero del paquete fuente Xemacs de

Debian.

Emacs es un editor de texto muy popular, perteneciente al proyecto GNU. Xemacs es un editor de texto que funciona en modo gráfico y en modo consola, desarrollado a partir de Emacs. Debido a ello, su código fuente tiene bastante similitud y nos interesa esta característica para utilizar NILSIMSA sobre él.

Se selecciona el fichero data.c para emas y xemacs. El fichero data.c para emacs tiene una longitud de 3320 líneas, mientras que el mismo fichero para xemacs reduce su longitud a 2259 líneas. Aunque xemacs este basado en emacs, lógicamente sus ficheros no tienen porque ser iguales, ni en contenido ni en longitud, aunque muchos de ellos si serán similares. Vamos a obtener el código NILSIMSA para ambos ficheros.

```
alberto@debian:~$ NILSIMSA temp/emacs23-23.1+1/src/data.c temp2/xemacs21-21.4.22/src/data.c
3ed10ca08003984eb0c749b2545420e0e44a58309ad2fe37201a0494f6f6704d temp/emacs23-23.1+1/src/data.c
5fd006118213184eb5c64975748625d6e442983093d2ee650c122094b2e6780d temp2/xemacs21-21.4.22/src/data.c
```

Como podemos comprobar el código NILSIMSA de los dos ficheros no es exactamente igual, pero si que coinciden bastantes caracteres. Vamos a obtener su código de comparación NILSIMSA para ver como son de parecidos.

```
alberto@debian:~$ NILSIMSA -c temp/emacs23-23.1+1/src/data.c temp2/xemacs21-21.4.22/src/data.c
  76 temp2/xemacs21-21.4.22/src/data.c
```

Como podemos observar, su código de comparación NILSIMSA es 76, esto quiere decir 204 bits iguales entre su códigos NILSIMSA. Si comparamos el código de comparación NILSIMSA 76 con el recomendado por el desarrollador de NILSIMSA para afirmar que dos ficheros son parecidos , NILSIMSA 24, vemos que los ficheros son, lógicamente, muy coincidentes entre si.

Veamos algunos ejemplos de coincidencia dentro de los ficheros. Para ello se realiza una inspección manual para encontrar similitudes.

La cabecera del fichero data.c de emacs, correspondiente a la licencia, es la siguiente:

```
/* Primitive operations on Lisp data types for GNU Emacs Lisp interpreter.
   Copyright (C) 1985, 1986, 1988, 1993, 1994, 1995, 1997, 1998, 1999, 2000,
                 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009
                 Free Software Foundation, Inc.

This file is part of GNU Emacs.

GNU Emacs is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either versión 3 of the License, or
(at your option) any later versión.

GNU Emacs is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with GNU Emacs.  If not, see <http://www.gnu.org/licenses/>.  */
```

Ahora se inspecciona manualmente la cabecera del fichero data.c de xemacs, también correspondiente a la licencia:

```
/* Primitive operations on Lisp data types for XEmacs Lisp interpreter.
   Copyright (C) 1985, 1986, 1988, 1992, 1993, 1994, 1995
   Free Software Foundation, Inc.
   Copyright (C) 2000 Ben Wing.

This file is part of XEmacs.

XEmacs is free software; you can redistribute it and/or modify it
under the terms of the GNU General Public License as published by the
Free Software Foundation; either versión 2, or (at your option) any
later versión.

XEmacs is distributed in the hope that it will be useful, but WITHOUT
ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
for more details.

You should have received a copy of the GNU General Public License
along with XEmacs; see the file COPYING.  If not, write to
the Free Software Foundation, Inc., 59 Temple Place - Suite 330,
Boston, MA 02111-1307, USA.  */

/* Synched up with: Mule 2.0, FSF 19.30.  Some of FSF's data.c is in
```

```
   XEmacs' symbols.c. */

/* This file has been Mule-ized. */
```

Como se puede observar la parte de cabecera de ambos ficheros no es idéntica, pero es muy parecida.

Vamos inspeccionar otra parte de definición en el fichero data.c de emacs:

```
Lisp_Object Qnil, Qt, Qquote, Qlambda, Qsubr, Qunbound;
Lisp_Object Qerror_conditions, Qerror_message, Qtop_level;
Lisp_Object Qerror, Qquit, Qwrong_type_argument, Qargs_out_of_range;
Lisp_Object Qvoid_variable, Qvoid_function, Qcyclic_function_indirection;
Lisp_Object Qcyclic_variable_indirection, Qcircular_list;
Lisp_Object Qsetting_constant, Qinvalid_read_syntax;
Lisp_Object Qinvalid_function, Qwrong_number_of_arguments, Qno_catch;
Lisp_Object Qend_of_file, Qarith_error, Qmark_inactive;
Lisp_Object Qbeginning_of_buffer, Qend_of_buffer, Qbuffer_read_only;
Lisp_Object Qtext_read_only;

Lisp_Object Qintegerp, Qnatnump, Qwholenump, Qsymbolp, Qlistp, Qconsp;
Lisp_Object Qstringp, Qarrayp, Qsequencep, Qbufferp;
Lisp_Object Qchar_or_string_p, Qmarkerp, Qinteger_or_marker_p, Qvectorp;
Lisp_Object Qbuffer_or_string_p, Qkeywordp;
Lisp_Object Qboundp, Qfboundp;
Lisp_Object Qchar_table_p, Qvector_or_char_table_p;

Lisp_Object Qcdr;
Lisp_Object Qad_advice_info, Qad_activate_internal;

Lisp_Object Qrange_error, Qdomain_error, Qsingularity_error;
Lisp_Object Qoverflow_error, Qunderflow_error;

Lisp_Object Qfloatp;
Lisp_Object Qnumberp, Qnumber_or_marker_p;

Lisp_Object Qinteger;
static Lisp_Object Qsymbol, Qstring, Qcons, Qmarker, Qoverlay;
static Lisp_Object Qfloat, Qwindow_configuration, Qwindow;
Lisp_Object Qprocess;
static Lisp_Object Qcompiled_function, Qbuffer, Qframe, Qvector;
static Lisp_Object Qchar_table, Qbool_vector, Qhash_table;
static Lisp_Object Qsubrp, Qmany, Qunevalled;
Lisp_Object Qfont_spec, Qfont_entity, Qfont_object;

Lisp_Object Qinteractive_form;
```

```
static Lisp_Object swap_in_symval_forwarding P_ ((Lisp_Object, Lisp_Object));

Lisp_Object Vmost_positive_fixnum, Vmost_negative_fixnum;
```

Vamos a analizar la misma parte definición en el fichero data.c de xemacs:

```
Lisp_Object Qnil, Qt, Qquote, Qlambda, Qunbound;
Lisp_Object Qerror_conditions, Qerror_message;
Lisp_Object Qerror, Qquit, Qsyntax_error, Qinvalid_read_syntax;
Lisp_Object Qlist_formation_error;
Lisp_Object Qmalformed_list, Qmalformed_property_list;
Lisp_Object Qcircular_list, Qcircular_property_list;
Lisp_Object Qinvalid_argument, Qwrong_type_argument, Qargs_out_of_range;
Lisp_Object Qwrong_number_of_arguments, Qinvalid_function, Qno_catch;
Lisp_Object Qinternal_error, Qinvalid_state;
Lisp_Object Qvoid_variable, Qcyclic_variable_indirection;
Lisp_Object Qvoid_function, Qcyclic_function_indirection;
Lisp_Object Qinvalid_operation, Qinvalid_change;
Lisp_Object Qsetting_constant;
Lisp_Object Qediting_error;
Lisp_Object Qbeginning_of_buffer, Qend_of_buffer, Qbuffer_read_only;
Lisp_Object Qio_error, Qend_of_file;
Lisp_Object Qarith_error, Qrange_error, Qdomain_error;
Lisp_Object Qsingularity_error, Qoverflow_error, Qunderflow_error;
Lisp_Object Qintegerp, Qnatnump, Qsymbolp;
Lisp_Object Qlistp, Qtrue_list_p, Qweak_listp;
Lisp_Object Qconsp, Qsubrp;
Lisp_Object Qcharacterp, Qstringp, Qarrayp, Qsequencep, Qvectorp;
Lisp_Object Qchar_or_string_p, Qmarkerp, Qinteger_or_marker_p, Qbufferp;
Lisp_Object Qinteger_or_char_p, Qinteger_char_or_marker_p;
Lisp_Object Qnumberp, Qnumber_char_or_marker_p;
Lisp_Object Qbit_vectorp, Qbitp, Qcdr;

Lisp_Object Qfloatp;
```

Como se observa hay bastante grado de similitud pero no tanto como en la inspección de cabecera.

Vamos por último a inspeccionar las definiciones de algunas funciones dentro del fichero data.c de emacs:

```
DEFUN ("*", Ftimes, Stimes, 0, MANY, 0,
       doc: /* Return product of any number of arguments, which are numbers or markers.
usage: (* &rest NUMBERS-OR-MARKERS)  */)
     (nargs, args)
     int nargs;
     Lisp_Object *args;
```

```
{
  return arith_driver (Amult, nargs, args);
}

DEFUN ("/", Fquo, Squo, 2, MANY, 0,
       doc: /* Return first argument divided by all the remaining arguments.
The arguments must be numbers or markers.
usage: (/ DIVIDEND DIVISOR &rest DIVISORS)  */)
     (nargs, args)
     int nargs;
     Lisp_Object *args;
{
  int argnum;
  for (argnum = 2; argnum < nargs; argnum++)
    if (FLOATP (args[argnum]))
      return float_arith_driver (0, 0, Adiv, nargs, args);
  return arith_driver (Adiv, nargs, args);
}
```

Ahora inspeccionamos la definición de las mismas funciones pero en el fichero data.c de xemacs:

```
DEFUN ("*", Ftimes, 0, MANY, 0, /*
Return product of any number of arguments.
The arguments should all be numbers, characters or markers.
*/
       (int nargs, Lisp_Object *args))
{
  EMACS_INT iaccum = 1;
  Lisp_Object *args_end = args + nargs;

  while (args < args_end)
    {
      int_or_double iod;
      number_char_or_marker_to_int_or_double (*args++, &iod);
      if (iod.int_p)
iaccum *= iod.c.ival;
      else
{
  double daccum = (double) iaccum * iod.c.dval;
  while (args < args_end)
    daccum *= number_char_or_marker_to_double (*args++);
  return make_float (daccum);
}
    }

  return make_int (iaccum);
}
```

```
DEFUN ("/", Fquo, 1, MANY, 0, /*
Return first argument divided by all the remaining arguments.
The arguments must be numbers, characters or markers.
With one argument, reciprocates the argument.
*/
       (int nargs, Lisp_Object *args))
{
  EMACS_INT iaccum;
  double daccum;
  Lisp_Object *args_end = args + nargs;
  int_or_double iod;

  if (nargs == 1)
    iaccum = 1;
  else
    {
      number_char_or_marker_to_int_or_double (*args++, &iod);
      if (iod.int_p)
iaccum = iod.c.ival;
      else
{
  daccum = iod.c.dval;
  goto divide_floats;
}
    }

  while (args < args_end)
    {
      number_char_or_marker_to_int_or_double (*args++, &iod);
      if (iod.int_p)
{
  if (iod.c.ival == 0) goto divide_by_zero;
  iaccum /= iod.c.ival;
}
      else
{
  if (iod.c.dval == 0) goto divide_by_zero;
  daccum = (double) iaccum / iod.c.dval;
  goto divide_floats;
}
    }

  return make_int (iaccum);

 divide_floats:
  for (; args < args_end; args++)
    {
      double dval = number_char_or_marker_to_double (*args);
      if (dval == 0) goto divide_by_zero;
      daccum /= dval;
    }
```

```
  return make_float (daccum);

 divide_by_zero:
  Fsignal (Qarith_error, Qnil);
  return Qnil; /* not reached */
}
```

Como vemos las dos funciones tiene el mismo nombre y casi los mismos parámetros en ambos casos, pero su desarrollo es completamente distinto.

Hemos comprobado como entre dos ficheros existen diferentes grados de coincidencias o similitudes y NILSIMSA puede ayudarnos a medir ese grado de similitud para poder realizar comparaciones en un sistema realmente grande como es Debian con mas de 2.000.000 de ficheros en su código fuente.

2.4. Python

Python es un lenguaje de programación interpretado creado en 1990. Debido a esta característica, ahorra un tiempo considerable en el desarrollo de un amplio abanico de aplicaciones, siendo un excelente candidato para la realización del proyecto presente. Además incluye numerosos modulos para la realización de diversas tareas, tales como entrada y salida de ficheros, llamadas al sistema, e integración con sistemas de bases de datos. Permite dividir un programa en módulos reutilizables desde otros programas python. Es un lenguaje fácil de leer, fácil de usar y extremadamente útil y potente.

Capítulo 3

Diseño e Implementación

Durante este capitulo se describe brevemente los requisitos necesarios para la herramienta a desarrollar. También se resume las diferentes tecnologías usadas para su desarrollo y las diferentes fase de desarrollo de la herramienta.
Cada fase incluye un apartado especificación, con los objetivos planteados en esa fase. También incluye una fase de desarrollo con los detalles de implementación ejecutados.

3.1. Requisitos

Atendiendo a la descripción del problema desarrollado en la sección 1.2 la herramienta debe contar con los siguientes requisitos:

- Un repositorio de código fuente debe estar almacenado y accesible.
- La herramienta debe ser capaz de acceder a un repositorio de código fuente de software libre.
- La herramienta debe analizar el código fuente y procesarlo.
- Los resultados del análisis y proceso del código fuente deben ser almacenados en una base de datos relacional.
- La herramienta debe realizar consultas a la base de datos relacional.
- Las consultas realizadas sobre la base de datos relacional debe proporcionar información relativa a la reutilización del código fuente.

3.2. Tecnología usada

La tecnología que se ha usado esta totalmente condicionada por los objetivos descritos en el sección 1.2. Para el desarrollo de la herramienta se ha seleccionado el uso de software libre tanto por la versatilidad, disponibilidad y utilidad de las herramientas que lo compone. Análogamente a estas condiciones, se ha decidido analizar código fuente de software libre debido a su alta disponibilidad y alta complejidad requerida para la relevancia de este proyecto.

- Debian GNU/Linux[1]

  El sistema funciona bajo Debian GNU/Linux. El sistema de escritorio usado es GNOME, pero la herramienta no depende directamente de un sistema de ventanas, siendo perfectamente posible su ejecución bajo una interface de terminal. La versión de Debian GNU/Linux usada es Testing 5.0 Lenny.

  Debian esta disponible para varias plataformas, incluye numerosos paquetes y sirve de base para diversas distribuciones de GNU/Linux.

- Python[2]

  Python es un lenguaje de programación interpretador de código abierto creado por Guido van Rossum. La versión del lenguaje utilizada es la 2.5.2. Es considerado un rival de perl, aunque mucho más limpio y elegante. Permite dividir un programa en módulos reutilizables desde otros programas python. Incluye una gran colección de módulos estándar. Es un lenguaje interpretado, ahorra tiempo de desarrollo al no necesitar compilado ni enlazado. Es usado en como lenguaje de programación principal para analizar el código fuente de un repositorio de software libre, procesarlo y almacenarlo en una base de datos relacional. Es usado como interface para generar consultas a la base de datos relacional y obtener resultados.

- Postgresql[3]

  Motor de base de datos relacional libre, alternativo a mysql, firebird o MaxDB. Fue liberado bajo licencia BSD. La versión utilizada es la 8.3.4-2. Es usado para

[1] http://www.debian.org/
[2] http://www.python.org

almacenar la información que extrae la herramienta mediante el lenguaje Python y para proveer a este de una fuente de consultas.

- pyPgSQL[4]

  Es una interface de python para PostgreSQL. Se usa el módulo incluido PgSQL como interfaz para motor de base de datos PostgreSQL. La versión utilizada es la 2.5.1-2+b2. Es usado para acceder a la base de datos relacional PostgreSQL a través de procedimientos sencillos y totalmente documentados en python.

- MD5[5]

  Acrónimo de Message-Digest Algorithm 5, Algoritmo de Resumen del Mensaje 5. Es un algoritmo de reducción criptográfico de 128 bits ampliamente usado. La codificación del MD5 de 128 bits es representada típicamente como un número de 32 dígitos hexadecimal. Cualquier cambio en un fichero , cambiara drásticamente su código MD5. Una de sus aplicaciones determinar si un fichero ha sido alterado, por ejemplo un fichero descargado de internet. En este proyecto se ha usado para generar parte de la base de datos necesaria para analizar el código fuente del repositorio de software libre.

- Nilsimsa[6]

  La codificación de NILSIMSA de 256 bits es representada típicamente como un número de 64 dígitos hexadecimal. Un cambio moderado en un fichero , no cambia drásticamente su código NILSIMSA. Una de sus aplicaciones es detectar correo basura. La versión utilizada es la 0.2.4. En este proyecto se ha usado para generar parte de la base de datos necesaria para analizar el código fuente del repositorio de software libre.

- Psyco[7]

  Psyco es un compilador para Python. Genera código maquina en vez de realizar una interpretación del código python paso a paso. El resultado es un aumento de velocidad en los programas, que varia de 2 veces mas rápido a 100 veces mas rápido para proyectos python sin modificar. La versión utilizada es la 1.6-2. El

[3]http://www.postgresql.org/
[4]http://pypgsql.sourceforge.net/
[5]http://people.csail.mit.edu/rivest/Rivest-MD5.txt
[6]http://ixazon.dynip.com/ cmeclax/NILSIMSA.html

uso de este compilador fue determinante para el análisis del código fuente de Debian en un tiempo razonable.

3.3. Arquitectura

La herramienta CarpeDM se ejecuta en un sobre un único servidor. En dicho servidor se almacena los datos necesarios y se ejecutan las herramientas que usan dichos datos para obtener resultados. El sistema esta compuesto de varios elementos, representados en la Figura 3.1 y descritos seguidamente:

- En el servidor reside un repositorio con todo el código fuente de una distribución de software libre como es GNU/Linux.
- El sistema CarpeDM accede a ese repositorio de código fuente, extrae información y la procesa.
- El sistema CarpeDM almacena la información procesada en una base de datos relacional .
- Sobra la base de datos relacional el sistema CarpeDM genera consultas para obtener respuestas referentes al índice de semejanza en el código.
- Esas consultas son almacenadas en archivos para representadas en gráficos o analizadas por otras herramientas.

3.4. Desarrollo en espiral

Para la realización de este proyecto se ha usado el modelo de software llamado desarrollo en espiral. Este modelo fue definido por Barry Boehm en 1988 [Boehm, 1988] y es utilizado por la ingeniería de software. El desarrollo en espiral esta formado por varias iteraciones o bucles conformados a su vez por varias actividades. Dichas actividades se definen en función del análisis de riesgo correspondiente al bucle anterior.

Las cuatro actividades de cada bucle o iteración del desarrollo en espiral son:

- Determinar objetivos: se identifican y se especifican los objetivos del ciclo.

[7] http://psyco.sourceforge.net/

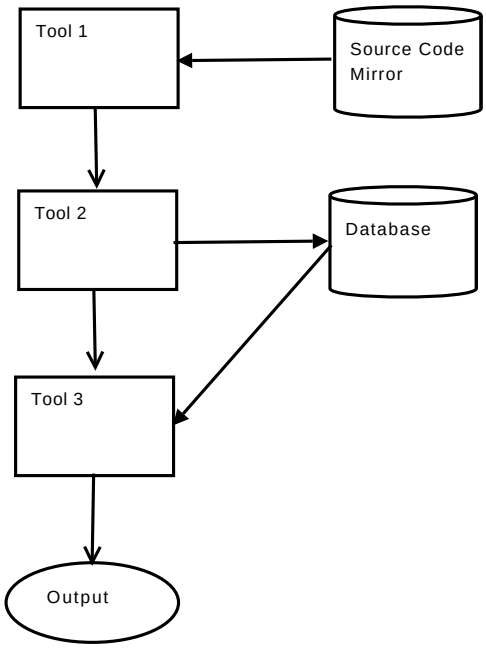

Figura 3.1: Arquitectura del sistema CarpeDM.

- Análisis de riesgos: se analizan los riesgos del ciclo y se intentar reducir lo máximo posible.

- Desarrollar, verificar y validar: el sistema es desarrollado y se realizan pruebas para vefiricar que se cumplen los objetivos fijados.

- Planificar: se revisa todo el proceso y se decide si se continua con el siguiente bucle de la espiral.

A continuación se detallan las diferentes fases o bucles que ha seguido este proyecto. Como se puede comprobar, los inicios de la espiral de desarrollo tienen unos objetivos muy básicos y sencillos. A medida que el proyecto se va desarrollando, la dificultad de desarrollo de los objetivos también aumenta.

3.4.1. Borrar caracteres

- **Especificación**: el objetivo de esta fase es buscar y eliminar caracteres que no nos interesan dentro de un fichero y que no serán necesarios para analizar las

semejanzas entre sí de los mismos, como hemos visto en la sección 1.1.

Los caracteres que se han determinado como no necesarios son espacios en blanco, tabuladores y líneas en blanco.

- **Desarrollo**: En esta fase se han seguido dos aproximaciones hacia nuestro objetivo.

  El primer desarrollo toma como entrada un fichero, analiza y elimina los caracteres no necesarios y guarda el resultado final en un fichero de salida. Con el fin de optimizar al máximo el rendimiento se ha usado la comparación carácter a carácter, una expresión regular propia del lenguaje Python y un filtrado de listas. Después de realizar pruebas se verificó que lo mas efectivo fue la comparación de caracteres.

  El segundo desarrollo toma como entrada un fichero, analiza y elimina los caracteres no necesarios y guarda el resultado final en una estructura de datos. Las dos estructuras de datos usados fueron listas y diccionarios, siendo los diccionarios sensiblemente mas efectivos que las listas.

3.4.2. MD5 y postgresql

- **Especificación**: el objetivo de esta fase es obtener el MD5 de las líneas de un fichero, eliminando caracteres no necesarios como ya se ha visto en la sección borrar-caract.

- **Desarrollo**: en esta fase se han seguido dos aproximaciones.

  El primer desarrollo toma como entrada un fichero, analiza y elimina caracteres no necesarios, y por cada línea de un fichero genera su código MD5. Dicho código se almacena en la estructura de datos diccionario de python.

El segundo desarrollo toma como entrada un fichero, analiza y elimina caracteres no necesarios y por cada línea de un fichero genera un código MD5. Dicho código es almacenado primeramente en un fichero.

Después de comprobar el correcto funcionamiento se realiza una mejora del segundo desarrollo.

Se elimina el almacenamiento en un fichero y se almacena el resultado en una base de datos postgresql.

Para ello se realiza una conexión a una base de datos postgresql existente, se genera una tabla con el campo número de línea y el campo código MD5. Por cada línea de un fichero se eliminan caracteres no necesarios, se genera el código MD5 de la línea y se inserta el valor del número de línea y del código MD5 de la línea en la tabla generada en la base de datos postgresql.

3.4.3. NILSIMSA

- **Especificación**: el objetivo de esta fase es generar el código NILSIMSA de un fichero.

- **Desarrollo**: Una primera aproximación hacia nuestro objetivo consiste en realizar una llamada al sistema a través del interfaz que python provee utilizando la librería os.

  Usando la librería os accedemos al programa NILSIMSA instalado en el ruta /usr/local/bin/NILSIMSA y le pasamos la ruta del fichero a analizar. Finalizado este paso, se captura el resultado que el programa NILSIMSA devuelve.

  En una segunda aproximación se toma como entrada un fichero, se analiza y elimina caracteres no necesarios y se genera un fichero de salida. Dicho fichero se usa como entrada para obtener su código NILSIMSA de forma análoga a la primera aproximación.

3.4.4. Base de datos

- **Especificación**: el objetivo de esta fase es generar, para un fichero, una entrada en una base de datos con la siguiente información:

  - ruta
  - nombre de fichero
  - nombre de paquete
  - código MD5 para el fichero con caracteres en blanco
  - código MD5 para el fichero sin caracteres en blanco
  - código NILSIMSA para el fichero con caracteres en blanco
  - código NILSIMSA para el fichero sin caracteres en blanco
  - número de líneas para un fichero con caracteres en blanco
  - número de líneas para un fichero sin caracteres en blanco
  - número de caracteres para un fichero con caracteres en blanco.
  - número de caracteres para un fichero sin caracteres en blanco.

- **Desarrollo**: Para la realización del objetivo marcado se generan una serie de variables para el almacenamiento temporal de la información requerida. Inicialmente se realiza una conexión a la base datos donde almacenaremos la información.
  La estructura de la tabla *codes* se muestra en el Cuadro 3.2 y esta formada por los siguientes campos:

  - **path**: es la ruta hacia el fichero analizado.
  - **name**: es el nombre del fichero analizado.
  - **package**: es el nombre del paquete al que corresponde el fichero analizado.
  - **MD5blank**: es la suma MD5 correspondiente al fichero analizado sin espacios en blanco.
  - **MD5**: es la suma MD5 correspondiente al fichero analizado.
  - **NILSIMSAblank**: es la clave NILSIMSA correspondiente al fichero analizado sin espacios en blanco.

- **NILSIMSA**: es la clave NILSIMSA correspondiente al fichero analizado.
- **linesblank**: es el número de líneas que contiene el fichero analizado sin espacios en blanco.
- **lines**: es el número de líneas que contiene el fichero analizado.
- **charactersblank**: es el número de caracteres que contiene el fichero analizado sin espacios en blanco.
- **characters**: es el número de caracteres que contiene el fichero analizado.

El campo **package** en este caso llevara el valor *unknow* debido a que no estamos analizando una distribución de debian completa, sino que solo estamos analizando un fichero en concreto. En posteriores desarrollos el campo llevara un valor acorde al paquete al que pertenezca.

Para cada fichero se obtiene la información requerida y se almacena en variables temporales. Dichas variables son usadas para insertar una línea en la tabla *codes*. Una vez insertada línea en la tabla se cierra la conexion con la base de datos.

La siguiente tabla muestra la estructura de la tabla codes *codes* en la base de datos PostgreSQL.

Nombre del Campo	Tipo
path	varchar(500)
name	varchar(100)
package	varchar(100)
MD5blank	varchar(32)
MD5	varchar(32)
NILSIMSAblank	varchar(64)
NILSIMSA	varchar(64)
linesBlank	integer
lines	integer
charactersblank	integer
character	integer

Cuadro 3.1: Estructura de la tabla *codes* de PostgreSQL.

3.4.5. Directorio

- **Especificación**: El objetivo de esta fase analizar todos los ficheros contenidos dentro de una ruta especificada y generar entradas en una tabla de una base de datos tal y como se describió en la sección 3.4.4.

- **Desarrollo**: El desarrollo de este objetivo pasa de nuevo por el uso de la librería os de python. Dicha librería dispone de la función walk en la cual nos apoyamos para realizar la búsqueda de todos los ficheros dentro de la ruta especificada. Para cada fichero encontrado se realiza una llamada a la herramienta desarrollada en la sección 3.4.4 para generar una entrada en la tabla de la base de datos utilizada donde queda almacenada para su posterior uso.

3.4.6. Consulta

- **Especificación**: El objetivo de esta fase es realizar una serie de consultas a la base de datos PostgreSQL generada.

- **Desarrollo**: Se realiza una conexión a la base de datos PostgreSQL.
  Una vez conectado a la base de datos se realiza una de las siete siguiente consultas:

  - **consulta 1**: obtiene el código NILSIMSA y la ruta para los códigos NILSIMSA que se repiten.
  - **consulta 2**: obtiene el código MD5 y la ruta para los códigos MD5 que se repiten.
  - **consulta 3**: obtiene el código NILSIMSA y el nombre para los códigos NILSIMSA que se repiten
  - **consulta 4**: obtiene el nombre para los nombres que se repiten
  - **consulta 5**: obtiene el nombre para todos los nombres
  - **consulta 6**: obtiene el número de líneas y el nombre donde las líneas estan entre 10000+10

- **consulta 7**: obtiene todos los códigos NILSIMSA y se comparan con un fichero.

Los resultados de estas consultas son mostrados por pantalla.

3.4.7. Debian

- **Especificación**: Debian es la aplicación que recopila toda la información necesaria, la procesa y almacena.

- **Desarrollo**: Para ello lo primero que realiza la aplicación es una conexión a la base de datos PostgreSQL para crear una tabla donde almacenar la información necesaria. Dicha conexión se realiza a través del módulo PgSQL, el cual provee un interface a Python para actuar sobre la base de datos.

  Con anterioridad se ha hecho una copia del código fuente del repositorio de software libre analizado, Debian 3.1, haciendo uso de la siguiente herramienta.

```
#!/bin/sh

exec > /home/jjamor/urjc/mirror-debian/mirror-debian.sh.log 2>&1

/usr/bin/debmirror -v --arch=i386,powerpc --dist=sarge,stable --source --method=http
--host=ftp.debian.org --root=/ --progress /home/jjamor/urjc/mirror-debian/mirror
```

  Para cada fichero del código fuente del repositorio de software libre se almacena un conjunto de datos en una tabla en la base de datos PostgreSQL llamada *codes*. La estructura de la tabla *codes* se muestra en el Cuadro 3.2 y esta formada por los siguientes campos:

  - **path**: es la ruta hacia el fichero analizado una vez descomprimido de su paquete.
  - **name**: es el nombre del fichero analizado.
  - **package**: es el nombre del paquete al que corresponde el fichero analizado.

- **MD5blank**: es la suma MD5 correspondiente al fichero analizado sin espacios en blanco.
- **MD5**: es la suma MD5 correspondiente al fichero analizado.
- **NILSIMSAblank**: es la clave NILSIMSA correspondiente al fichero analizado sin espacios en blanco.
- **NILSIMSA**: es la clave NILSIMSA correspondiente al fichero analizado.
- **linesblank**: es el número de líneas que contiene el fichero analizado sin espacios en blanco.
- **lines**: es el número de líneas que contiene el fichero analizado.
- **charactersblank**: es el número de caracteres que contiene el fichero analizado sin espacios en blanco.
- **characters**: es el número de caracteres que contiene el fichero analizado.

Seguidamente la herramienta recupera un listado de los paquetes de software libre a analizar. Todo repositorio de software libre de Debian dispone de un archivo llamado *Sources* que contiene información relativa a los paquetes de software libre que existen en el repositorio. La herramienta analiza dentro de ese fichero los nombres de los paquetes buscados y los almacena en un diccionario.

La herramienta hace una llamada al sistema para recuperar el código fuente por cada paquete almacenado en el diccionario, generando una carpeta temporal con todos los ficheros relativos a un paquete. La herramienta procesa cada fichero de un paquete para obtener los datos del Cuadro 3.2 y los almacena en la base de datos PostgreSQL.

3.4.8. Query

- **Especificación**: Query es la aplicación que genera consultas SQL sobre la base de datos PostgreSQL que previamente Debian ha generado.

Nombre del Campo	Tipo
path	varchar(500)
name	varchar(100)
package	varchar(100)
MD5blank	varchar(32)
MD5	varchar(32)
NILSIMSAblank	varchar(64)
NILSIMSA	varchar(64)
linesBlank	integer
lines	integer
charactersblank	integer
character	integer

Cuadro 3.2: Estructura de la tabla *codes* de PostgreSQL.

- **Desarrollo**: Esas consultas dan como resultado ficheros que pueden ser analizados directamente, o por otras herramientas, y generar gráficas con gnuplot.

  Las consultas que Query realiza sobre la base de datos PostgreSQL en la tabla *codes* son las siguientes:

  - **consulta 1**: obtiene el número total de ficheros existentes.
  - **consulta 2**: obtiene el número de paquetes totales existentes.
  - **consulta 3**: obtiene el número total de códigos MD5.
  - **consulta 4**: obtiene el número total de grupos de códigos MD5 repetidos.
  - **consulta 5**: obtiene el número total de grupos de códigos MD5 sin espacios repetidos.
  - **consulta 6**: obtiene el número total de códigos NILSIMSA.
  - **consulta 7**: obtiene el número total de grupos de códigos NILSIMSA repetidos.
  - **consulta 8**: obtiene el número total de grupos de códigos NILSIMSA sin espacios repetidos.
  - **consulta 9**: obtiene el número total de ficheros con códigos MD5 repetidos.
  - **consulta 10**: obtiene el número total de ficheros con códigos MD5 sin espacios repetidos.

- **consulta 11**: obtiene el número total de ficheros con códigos NILSIMSA repetidos.
- **consulta 12**: obtiene el número total de ficheros con códigos NILSIMSA sin espacios repetidos.
- **consulta 13**: obtiene el número de rutas repetidas.
- **consulta 14**: obtiene el número total de nombres de ficheros.
- **consulta 15**: obtiene el número total de ficheros con el nombre repetido.
- **consulta 16**: obtiene el número total de ficheros con el nombre y el código MD5 repetido.
- **consulta 17**: obtiene el número total de ficheros con el nombre y el código NILSIMSA repetido.
- **consulta 18**: obtiene el número total de ficheros con el nombre, el código MD5 y el código NILSIMSA repetido.
- **consulta 19**: obtiene el número total de ficheros con el código MD5 y el código NILSIMSA repetido.

3.4.9. Contador NILSIMSA

- **Especificación**: Contador NILSIMSA se encarga, entre otras funciones, de comparar todos los códigos NILSIMSA de Debian entre si.

- **Desarrollo**: Para realizar la operación de comparación de todos los códigos NILSIMSA de Debian son necesarios varios procesos.

  Primeramente se realiza una consulta a la tabla codes de la base de datos postgreSQL.
  La consulta devuelvo todos códigos NILSIMSA de la distribución , se realiza un salvado a un diccionario y posteriormente se salva el diccionario a un fichero de salida.

  Se recupera el diccionario almacenado en el fichero de salida anterior, y se genera una lista ordenada de códigos NILSIMSA.

A este lista se la aplica la función unhexlify para transformar números hexadecimales a binario.

Cada código NILSIMSA es comparado con todos los demás códigos de la distribución Debian.

Se cuenta el número de veces que se cumple una resolución NILSIMSA de 64, 32, 16, 8 y 4 para cada código NILSIMSA, dentro del conjunto de todos los códigos NILSIMSA de Debian.

Es decir se cuenta el numero de veces, si se cumple la condición, que el código NILSIMSA estudiado tiene al menos 64+128 bits iguales, 32+128 bits iguales, 16+128 bits iguales, 8+128 bits iguales y 4+128 bits iguales.

El resultado se guarda en un diccionario, y posteriormente ese diccionario a fichero de salida.

3.4.10. Contador parcial

- **Especificación**: Contador parcial es el encargado de almacenar para un código NILSIMSA en concreto, la suma de cada valor de comparación NILSIMSA con respecto a todos los códigos NILSIMSA de Debian.

- **Desarrollo**: Se genera una diccionario que pueda almacenar los valores de comparación NILSIMSA de -128 a 128, para código NILSIMSA comparado.

  Se lee el fichero con todos los códigos NILSIMSA de Debian y se genera una lista ordenada y transformada a binario para realizar comparaciones.

  Para el código NILSIMSA especificado se compara todos los códigos NILSIMSA de Debian, sumando los valores de comparación NILSIMSA recibidos en cada caso.

Se genera un fichero de salida con la suma final de la comparación, listo para ser usado en una gráfica.

Capítulo 4

Resultados

En este capitulo se describen los resultados obtenidos con la herramienta desarrollada.

Algunos de estos resultados nos muestran la siguiente información para la distribucion GNU/Linux Debian 3.1:

- El número total de ficheros analizados es 2.403.211.
- El número total de paquetes analizados es 8.716
- El número total de nombres de ficheros analizados es 1.122.788.
- El número total de códigos MD5 es 1.903.326.
- El número total de códigos NILSIMSA es 1.862.348.
- El número total de grupos de códigos MD5 repetidos, eliminando espacios en blanco, es 193.459.
- El número total de grupos de códigos NILSIMSA repetidos, eliminando espacios en blanco, es 204.525.

Durante el capitulo se estudiaran gráficas y tablas que muestran los datos obtenidos.

Como resultado final veremos que se ha estimado una similitud entre los ficheros de Debian 3.1, de un 3 por ciento, aproximadamente.

4.1. Resultados de consultas SQL: gráficas de barras

La herramienta CarpeDM ha obtenido los siguientes resultados sobre el repositorio de código fuente de software libre Debian GNU/Linux.

El repositorio esta compuesto por un total de 8.716 paquetes de software que reúnen un total de 2.403.211 de ficheros. Gracias a la herramienta CarpeDM se han obtenido un conjunto de gráficas que se describen a continuación.

- La figura 4.1 y el Cuadro 4.1 muestran el número total de ficheros del sistema analizado (2.403.211), respecto al número total de códigos MD5 distintos que generan dichos ficheros (1.903.326). También se muestran el número total de códigos NILSIMSA distinto que generan los ficheros (1.862.348). Debido al algoritmo que posee NILSIMSA, vemos que para un mismo conjunto de ficheros genera mas códigos iguales que el algoritmo MD5.

  - files es el número total de ficheros analizados.

  - MD5 es el número total de códigos MD5 que genera Debian.

  - NILSIMSA es el número total de códigos NILSIMSA que genera Debian.

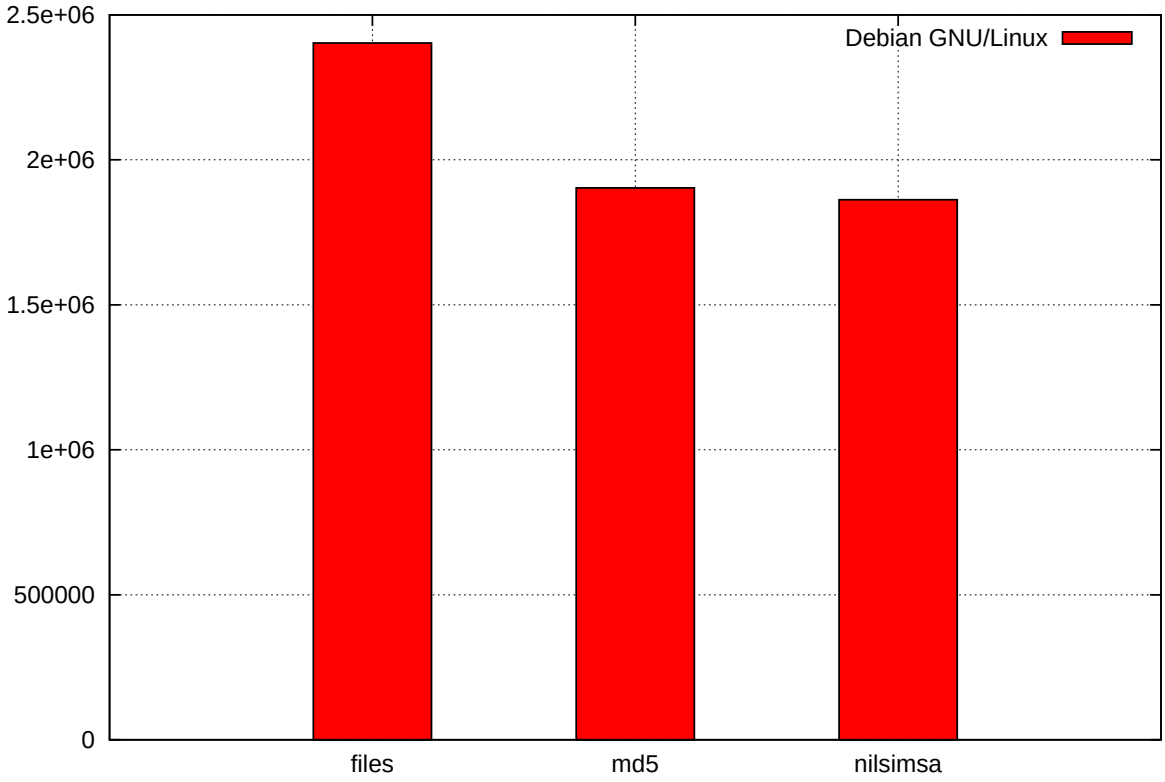

Figura 4.1: Comparación de códigos MD5 y NILSIMSA diferentes con respecto el número de ficheros.

Objeto	Unidades
files	2.403.211
MD5	1.903.326
NILSIMSA	1.862.348

Cuadro 4.1: Comparación files, MD5, NILSIMSA.

- La figura 4.2 y el Cuadro 4.2 muestran el número total de ficheros del sistema analizado (2.403.211), respecto al número total de códigos MD5 repetidos (194.172), el número total de códigos MD5 repetidos sin espacios en blanco (193.459), el número total de códigos NILSIMSA repetidos (202.177) y el número total de códigos NILSIMSA repetidos sin espacios en blanco (204.525).

  - files es el número total de ficheros analizados.

- MD5 es el número ficheros con MD5 repetido.
- MD5 noblank es el número ficheros, sin espacios en blanco, con MD5 repetido.
- NILSIMSA es el número de ficheros con NILSIMSA repetido.
- NILSIMSA noblank es el número de ficheros, sin espacios en blancos, con NILSIMSA repetido.

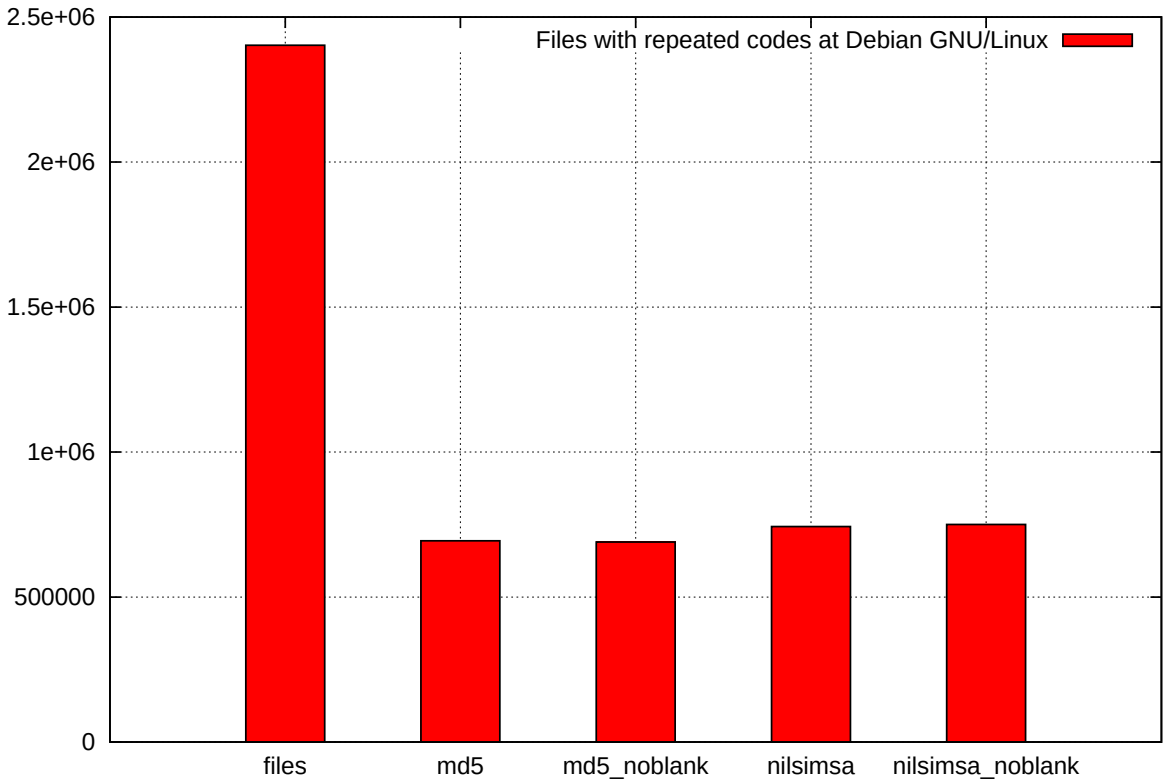

Figura 4.2: Comparacion de códigos MD5 y NILSIMSA repetidos con respecto el número de ficheros.

- La figura 4.3 y el Cuadro 4.3 muestran el número total de grupos de códigos MD5 (1.903.326) del sistema, respecto al número total de grupos de códigos MD5 repetidos (194.172). También se muestran el número total de códigos MD5 repetidos con respecto a los mismos ficheros quitando los espacios en blanco (193459).

Objeto	Unidades
files	2.403.211
MD5	694.057
MD5 noblank	689.531
NILSIMSA	743.040
NILSIMSA noblank	749.984

Cuadro 4.2: Comparación files, MD5, MD5 noblanck, NILSIMSA, NILSIMSA noblank.

- groups es el número total de códigos MD5 del sistema Debian.
- repeated es el número total de grupos de ficheros con MD5 repetido.
- noblank es el número total de grupos de ficheros , sin espacios en blanco, con MD5 repetido.

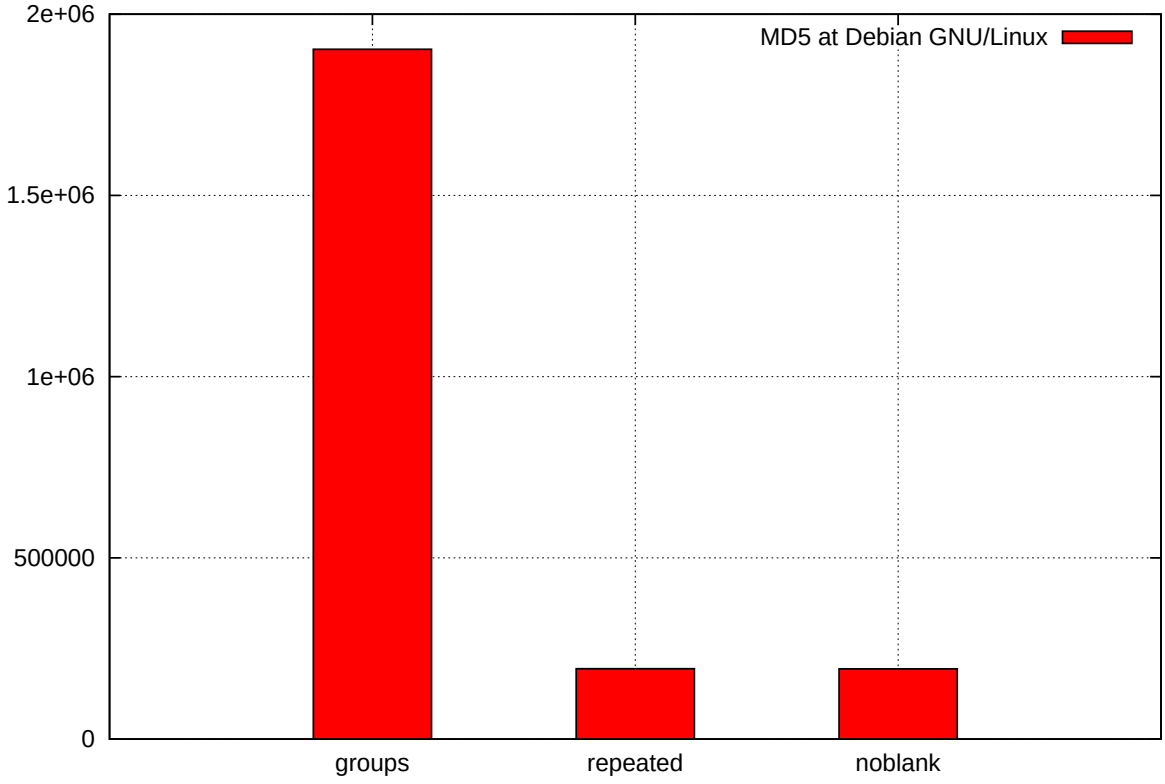

Figura 4.3: Comparación de grupos de códigos MD5, grupos MD5 repetidos y grupos MD5 sin espacios en blanco.

Objeto	Unidades
groups	1.903.326
repeated	194.172
noblank	193.459

Cuadro 4.3: Comparación groups, repeated, noblank para MD5.

- La figura 4.4 y el Cuadro 4.4 muestran el número total de grupos de códigos NILSIMSA (1.862.348) del sistema, respecto al número total de grupos de códigos NILSIMSA repetidos (202.177). También se muestran el número total de códigos NILSIMSA repetidos con respecto a los mismos ficheros quitando los espacios en blanco (193459).

  - groups es el número total de códigos NILSIMSA del sistema Debian.

  - repeated es el número total de grupos de ficheros con NILSIMSA repetido.

  - noblank es el número total de grupos de ficheros , sin espacios en blanco, con NILSIMSA repetido.

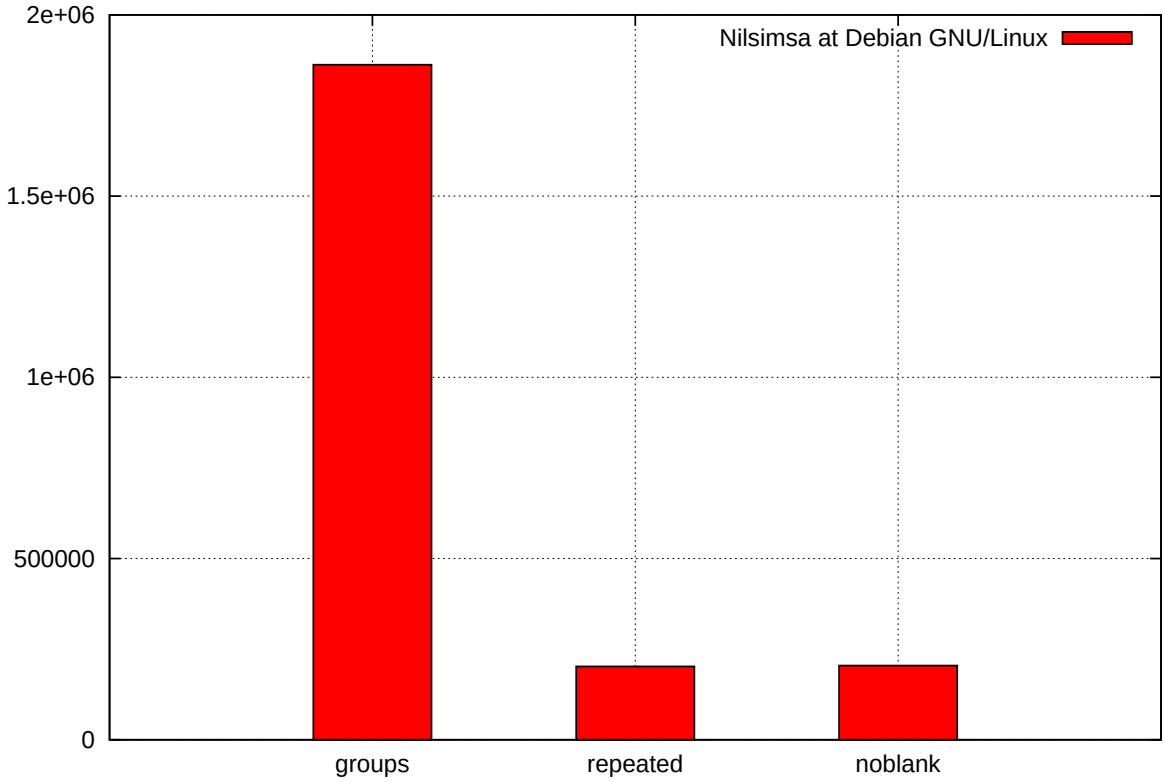

Figura 4.4: Comparación de grupos de códigos NILSIMSA, grupos NILSIMSA repetidos y grupos NILSIMSA sin espacios en blanco.

Objeto	Unidades
groups	1.862.348
repeated	202.177
noblank	204.525

Cuadro 4.4: Comparación groups, repeated, noblank para NILSIMSA.

4.2. Resultados de consultas SQL: gráficas de puntos

- La figura 4.5 representa cuantas veces se repite un mismo código MD5 enfrentado al número de grupos que cumplen dicha repetición (194.172) . También se representa cuantas veces se repite un mismo código NILSIMSA enfrentado al número de grupos que cumplen dicha repetición (202.177). Como se puede

apreciar la mayoría de los grupos se repiten muy pocas veces excepto en algunos casos en concreto.

Como se estudiara posteriormente, con MD5 solo podemos comparar ficheros entre si para conocer si son iguales o no. NILSIMSA servirá de herramienta para identificar ficheros similares entre si, y cuanto de similares son.

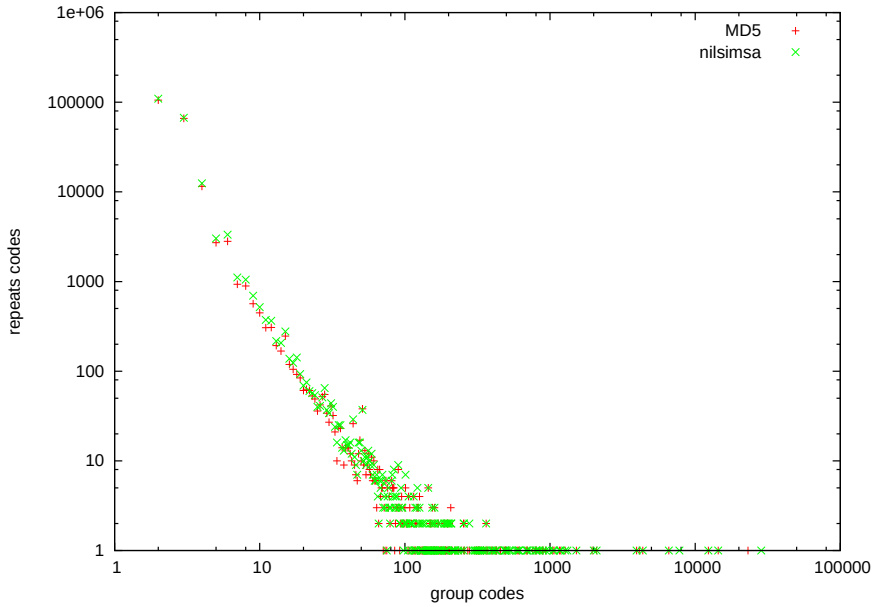

Figura 4.5: Comparación de repetición de grupos de códigos MD5 y NILSIMSA.

- La figura 4.6 representa cuantas veces se repite un mismo código MD5 enfrentado al número de grupos que cumplen dicha repetición (193.459) . También se representa cuantas veces se repite un mismo código NILSIMSA enfrentado al número de grupos que cumplen dicha repetición (204.525). Como se puede apreciar la mayoría de los grupos se repiten muy pocas veces excepto en algunos casos en concreto.

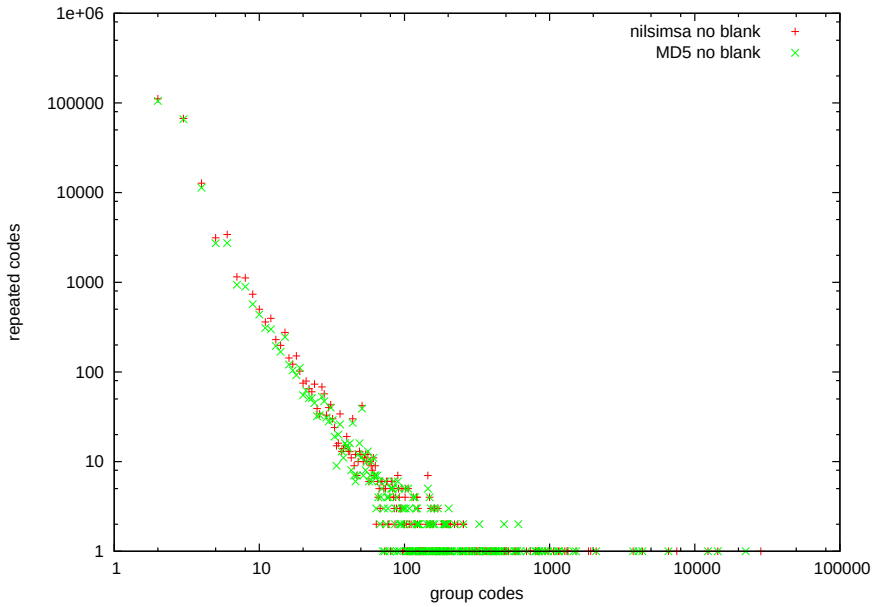

Figura 4.6: Comparacion de repetición de grupos de códigos MD5 y NILSIMSA sin espacios en blanco.

- La figura 4.7 representa cuantas veces se repite un mismo nombre enfrentado al número de grupos que cumplen dicha repetición (250.765) . También se representa cuantas veces se repite un mismo nombre y código MD5 enfrentado al número de grupos que cumplen dicha repetición (181.814). Como se puede apreciar la mayoría de los grupos se repiten muy pocas veces excepto en algunos casos en concreto.

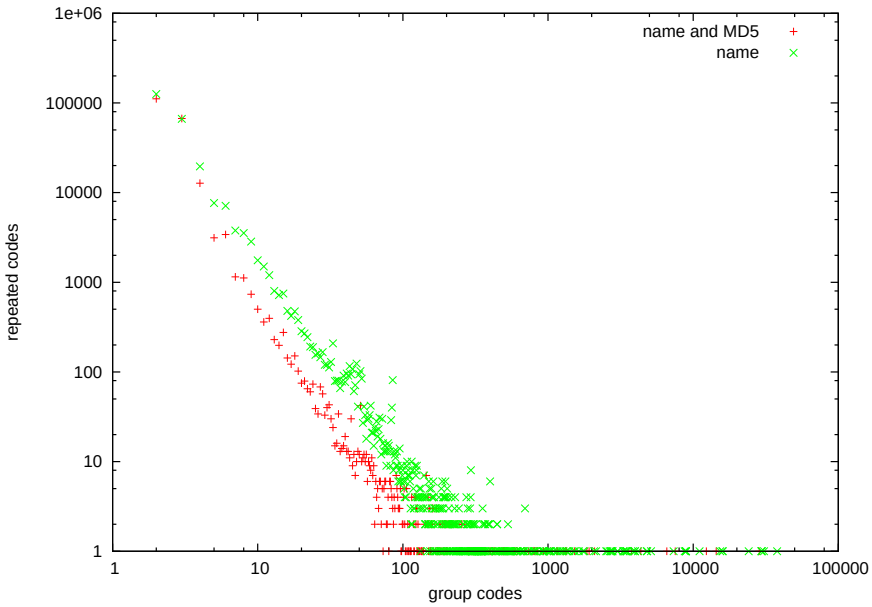

Figura 4.7: Comparación de repetición de grupos de códigos con el mismo nombre y con el mismo nombre y MD5.

- La figura 4.8 representa cuantas veces se repite un mismo nombre enfrentado al número de grupos que cumplen dicha repetición (250.765) . También se representa cuantas veces se repite un mismo nombre y código NILSIMSA enfrentado al número de grupos que cumplen dicha repetición (188.560). Como se puede apreciar la mayoría de los grupos se repiten muy pocas veces excepto en algunos casos en concreto.

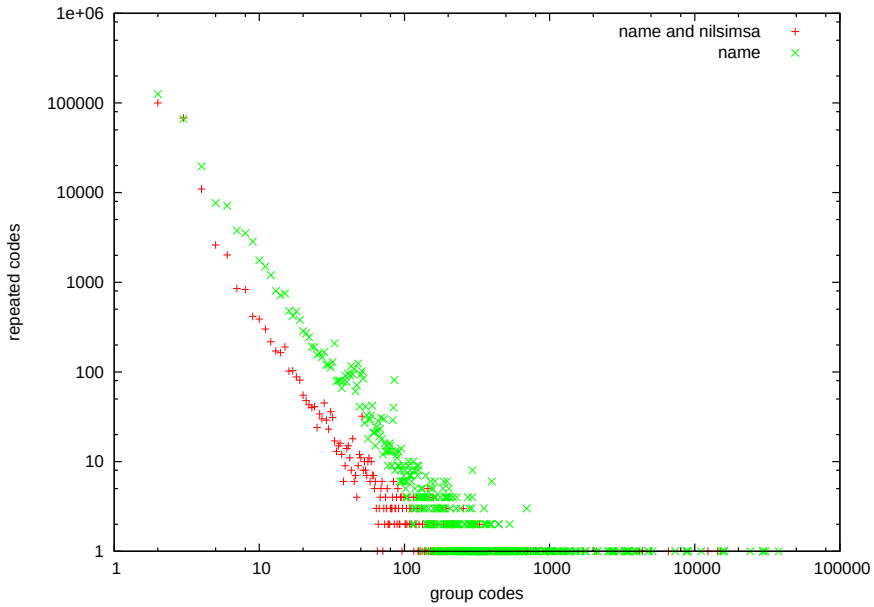

Figura 4.8: Comparación de repetición de grupos de códigos con el mismo nombre y con el mismo nombre y NILSIMSA.

- La figura 4.9 representa cuantas veces se repite un mismo nombre enfrentado al número de grupos que cumplen dicha repetición (250.765) . También se representa cuantas veces se repite un mismo nombre, código MD5 y código NILSIMSA enfrentado al número de grupos que cumplen dicha repetición (181.814). Como se puede apreciar la mayoría de los grupos se repiten muy pocas veces excepto en algunos casos en concreto.

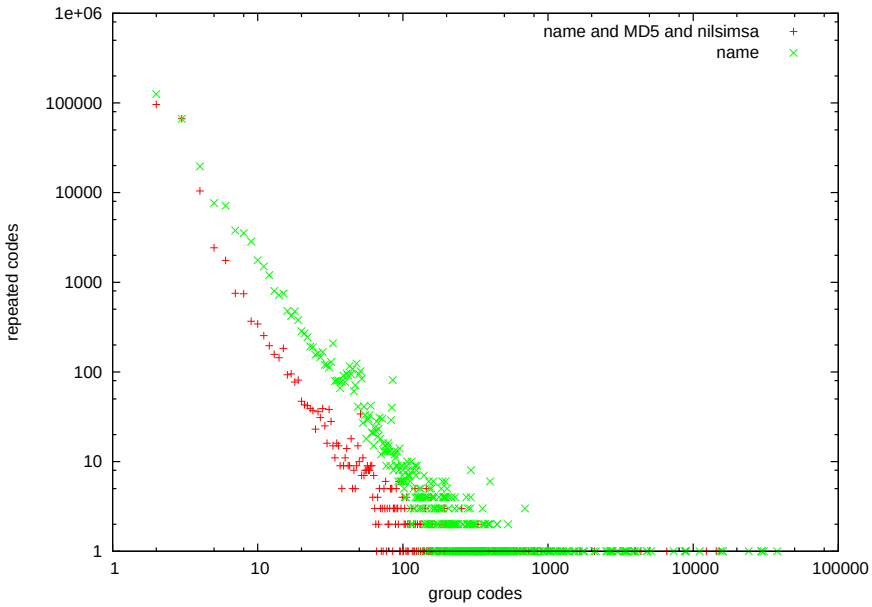

Figura 4.9: Comparación de repetición de grupos de códigos con el mismo nombre y con el mismo nombre y NILSIMSA.

- La figura 4.10 representa cuantas veces se repite un mismo nombre enfrentado al número de grupos que cumplen dicha repetición (250.765) . También se representa cuantas veces se repite un mismo código MD5 y código NILSIMSA enfrentado al número de grupos que cumplen dicha repetición (194.172). Como se puede apreciar la mayoría de los grupos se repiten muy pocas veces excepto en algunos casos en concreto.

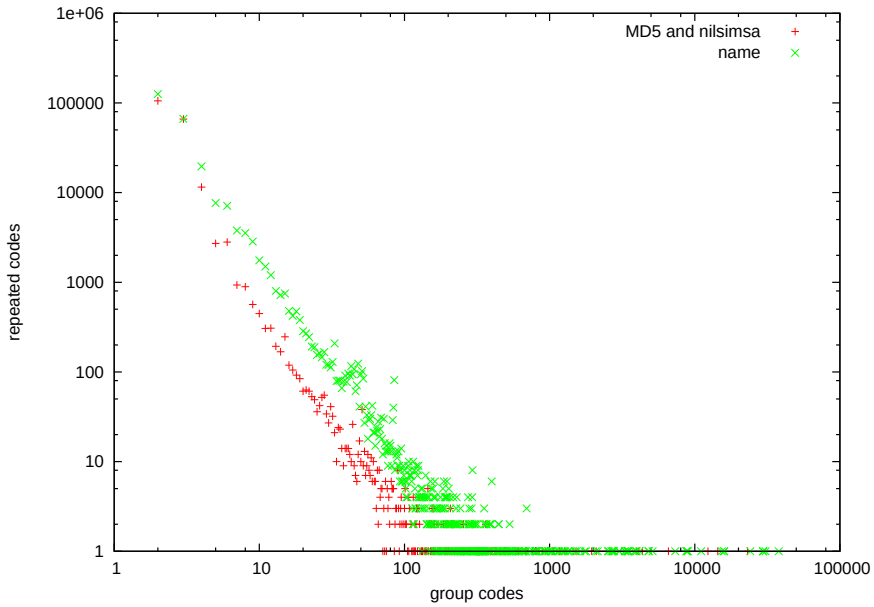

Figura 4.10: Comparación de repetición de grupos de códigos con el mismo nombre y con el mismo MD5 y NILSIMSA.

- La figura 4.11 muestra la comparación de los 1.862.348 códigos NILSIMSA de Debian situados en el eje x, frente al número de códigos situados en el eje y, que cumplen las resoluciones NILSIMSA especificadas de 4, 8, 16, 32 y 64 bits. Analizando los resultados de la figura, se observa que a medida que aumenta el número de bits de resolución, disminuye el número de códigos NILSIMSA que cumplen dicha resolución.

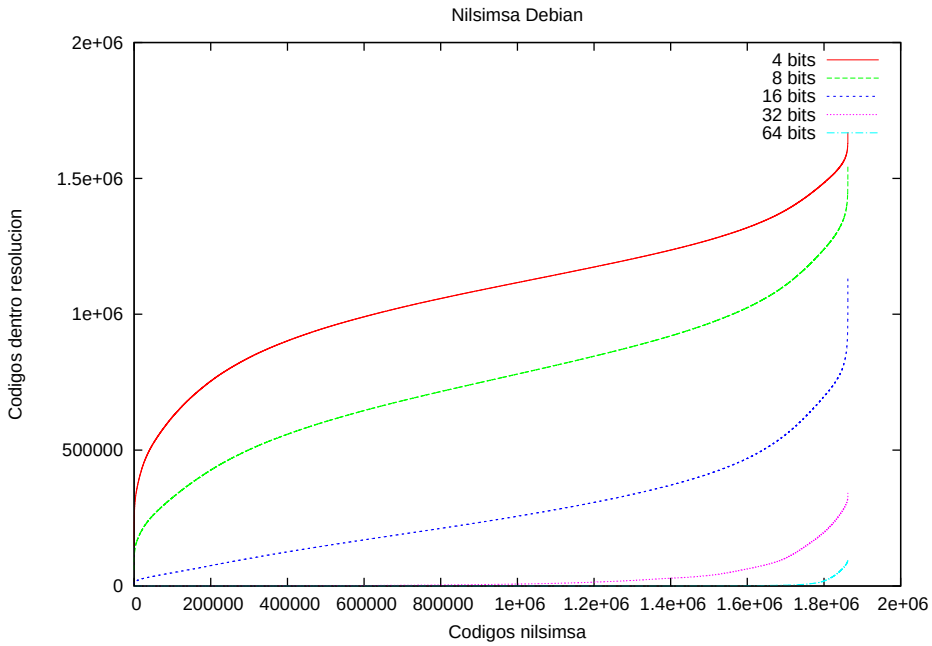

Figura 4.11: Comparación de los códigos NILSIMSA de Debian que estan dentro de las resoluciones 4, 8, 16, 32 y 64 bits

- Tomando como base la figura 4.11, se han seleccionado 3 casos:

  - El caso 1 corresponde a un código NILSIMSA del entorno inicial de las curvas, el mas cercano a 0 códigos NILSIMSA repetidos.
  - El caso 2 corresponde a un código NILSIMSA del entorno intermedio de las curvas.
  - El caso 3 corresponde a un código NILSIMSA del entorno final de las curvas, el mas cercano a 1.862.348 códigos NILSIMSA repetidos.

  Se ha comparado cada uno de los 3 códigos NILSIMSA con todos los códigos NILSIMSA de Debian. En el eje x, se muestra la resolución NILSIMSA de -128 a 128. En el eje y, se muestra el número de códigos NILSIMSA que cumplen la resolución del eje x, para cada uno de los tres casos. La figura 4.12 muestra la comparación de las resoluciones de 3 códigos NILSIMSA con respecto al repositorio completo de Debian.

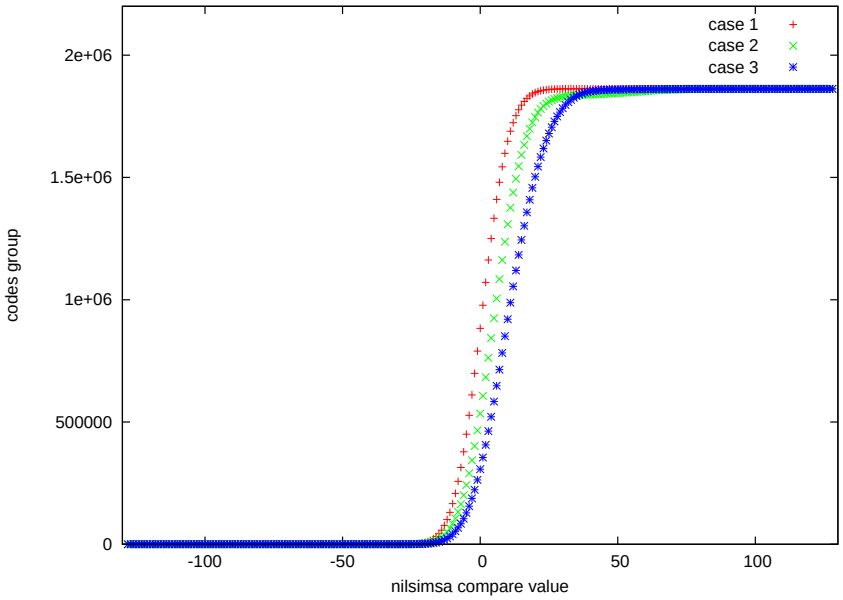

Figura 4.12: Comparación de las resoluciones de 3 códigos NILSIMSA de Debian con respecto al repositorio completo de Debian

- De la figura 4.12 se realiza una figura de detalle en el entorno NILSIMSA de 0 a 40 en la figura 4.13.

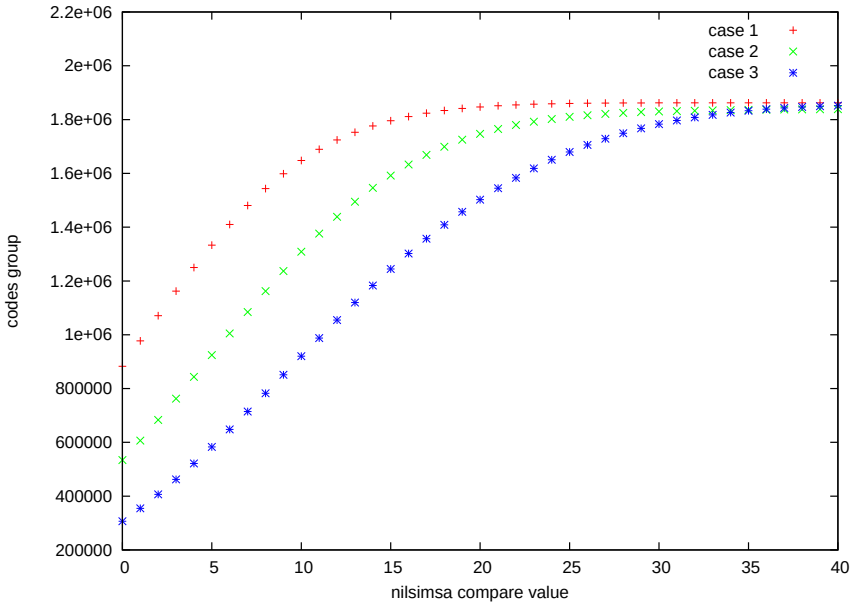

Figura 4.13: Detalle de comparación de las resoluciones de 3 códigos NILSIMSA de Debian con respecto al repositorio completo de Debian

Se ha estimado que al alcanzar la cifra de 1.800.000 códigos, estamos hablando de la mayoría de ellos, dejando menos del 5 por ciento del total de 1.862.348 códigos fuera del estudio.

En el caso 1, del entorno mas cercano a 0 códigos NILSIMSA repetidos, se alcanza 1.800.000 al llegar a la resolución NILSIMSA 16.

En el caso 2, del entorno intermedio de códigos NILSIMSA repetidos, se alcanza 1.800.000 al llegar a la resolución NILSIMSA 24.

En el caso 3, del entorno mas cercano a 1.862.348 códigos NILSIMSA repetidos, se alcanza 1.800.000 al llegar a la resolución NILSIMSA 32.

Tomando como base el caso intermedio(caso 2) y atendiendo a que el desarrollador de NILSIMSA considera que dos códigos de comparación NILSIMSA por encima de 24 no esta independientemente generados, se concluye lo siguiente:

Se calcula que 60.206 códigos NILSIMSA están por encima de la resolución NILSIMSA 24 para el caso número 2. Es decir podemos afirmar que para el caso medio número 2, hay aproximadamente un 3 por ciento de semejanza con respecto a todos los códigos NILSIMSA de Debian.

Se puede tomar como referencia este caso medio, debido a que como muestra la figura 4.11, la mayoría de los códigos NILSIMSA de Debian se mueven en ese entorno. Estudiando la curva para los casos mas extremos de inicio y final de curva, podemos suponer que se compensan entre si.

Como vemos , en una distribución tan extensa y diversa como Debian, hablar de un 3 por ciento de semejanza es un dato bastante significativo sobre su composición.

Capítulo 5

Conclusiones y trabajo futuro

El trabajo realizado ha dado como resultado una herramienta eficiente en el análisis del código fuente de un sistema de software libre. Se han expuesto resultados de análisis que cubren parte de la incógnita que ha perseguido este trabajo. La parte mas compleja del análisis fue ejecutado por la herramienta en menos de una semana. Esta parte consistió en comparar entre si todo los códigos NILSIMSA de la distribución GNULinux/Debian 3.1 Sarge.

El trabajo realizado ha seguido la metodología en espiral explicada con anterioridad, lo cual ha permitido diversos estudios y aproximaciones hacia el resultado final, con el fin de buscar el análisis mas preciso que se ha podido ofrecer para un estudio tan extenso como una distribución de software libre.

En este trabajo se ha aprendido a usar y conocer herramientas y sistemas pertenecientes al proyecto GNU/Linux, los cuales han facilitado muy sensiblemente el buen desarrollo de los objetivos marcados.

Se ha comprobado que las semejanzas en el código fuente del software libre, existen, son reales y pueden llegar a analizarse. Este análisis arroja respuestas claras y visuales sobre un sistema de software en concreto, Debian 3.1. Basándose en estudios anteriores, se ha realizado una herramienta que analiza y contabiliza la similitud a nivel de ficheros de un sistema realmente grande como es Debian, obteniendo respuestas de su similitud que anteriormente eran desconocidas.
El análisis de similitud es un análisis mas complejo que un estudio de igualdad entre

ficheros, el cual requiere un nivel de abstracción mas extenso y esta abierto a numerosas interpretaciones. La organización del sistema a analizar, las herramientas usadas y los algoritmos de generación y análisis de datos han seguido el criterio de la máxima eficiencia posible, ya que es obligatorio una eficiencia alta al tratar con un volumen de datos tan extenso.

A partir de este análisis se abren vías de investigación que pueden resuelver mas incógnitas o aumenten las respuestas que se exponen en este trabajo.

Algunos de los posibles trabajos futuros a raíz de este trabajo son:

- Estudio y comparación con otras versiones anteriores y posteriores de Debian, como pueden ser Debian 2, 4 y 5.

- Estudio y comparación con otras distribuciones GNU/Linux, como pueden ser Red Hat.

- Aumento o disminución de la granularidad del estudio, para estudiar el software a nivel de líneas de códigos o de paquetes o subsistemas.

Bibliografía

[Baxter et al., 1998] Baxter, I. D., Yahin, A., Moura, L., Sant'Anna, M., and Bier, L. (1998). *Clone Detection Using Abstract Syntax Trees*. icsm.

[Boehm, 1988] Boehm, B. W. (1988). *A spiral model of software development and enhancement*. IEEE Computer 21.

[Gonzz-Barahona et al., 2001] Gonzz-Barahona, J. M., Ortuz, M. A., de las Heras Quiros, P., Centeno Gonzz, J., and MatellOlivera, V. (2001). Counting potatoes: the size of Debian 2.2. *Upgrade Magazine*, II(6):60–66.

[Kamiya et al., 2002] Kamiya, T., Kusumoto, S., and Inoue, K. (2002). *CCFinder: a multilinguistic token-based code clone detection system for large scale source code*. IEEE Transactions on Software Engineering.

[Koschke, 2006] Koschke, R. (2006). *Survey of Research on Software Clones*. Universitat Bremen, Germany.

[Roy and Cordy, 2007] Roy, C. K. and Cordy, J. R. (2007). *A Survey on Software Clone Detection Research*. Queen's School of Computing TR.

[Wheeler, 2001] Wheeler, D. A. (2001). More than a gigabuck: Estimating GNU/Linux's size.
http://www.dwheeler.com/sloc/redhat71-v1/redhat71sloc.html.

Apéndice A

Licencia Reconocimiento-CompartirIgual 2.5 España

Licencia

LA OBRA (SEGÚN SE DEFINE MÁS ADELANTE) SE PROPORCIONA BAJO LOS TÉRMINOS DE ESTA LICENCIA PÚBLICA DE CREATIVE COMMONS ("CCPL" O "LICENCIA"). LA OBRA SE ENCUENTRA PROTEGIDA POR LA LEY ESPAÑOLA DE PROPIEDAD INTELECTUAL Y/O CUALESQUIERA OTRAS NORMAS RESULTEN DE APLICACIÓN. QUEDA PROHIBIDO CUALQUIER USO DE LA OBRA DIFERENTE A LO AUTORIZADO BAJO ESTA LICENCIA O LO DISPUESTO EN LAS LEYES DE PROPIEDAD INTELECTUAL.

MEDIANTE EL EJERCICIO DE CUALQUIER DERECHO SOBRE LA OBRA, USTED ACEPTA Y CONSIENTE LAS LIMITACIONES Y OBLIGACIONES DE ESTA LICENCIA. EL LICENCIADOR LE CEDE LOS DERECHOS CONTENIDOS EN ESTA LICENCIA, SIEMPRE QUE USTED ACEPTE LOS PRESENTES TÉRMINOS Y CONDICIONES.

1. Definiciones

1. La "obra" es la creación literaria, artística o científica ofrecida bajo los términos de esta licencia.

2. El "autor" es la persona o la entidad que creó la obra.

3. Se considerará “obra conjunta” aquella susceptible de ser incluida en alguna de las siguientes categorías:

   a) “Obra en colaboración”, entendiendo por tal aquella que sea resultado unitario de la colaboración de varios autores.

   b) “Obra colectiva”, entendiendo por tal la creada por la iniciativa y bajo la coordinación de una persona natural o jurídica que la edite y divulgue bajo su nombre y que esté constituida por la reunión de aportaciones de diferentes autores cuya contribución personal se funde en una creación única y autónoma, para la cual haya sido concebida sin que sea posible atribuir separadamente a cualquiera de ellos un derecho sobre el conjunto de la obra realizada.

   c) “Obra compuesta e independiente”, entendiendo por tal la obra nueva que incorpore una obra preexistente sin la colaboración del autor de esta última.

4. Se considerarán “obras derivadas” aquellas que se encuentren basadas en una obra o en una obra y otras preexistentes, tales como: las traducciones y adaptaciones; las revisiones, actualizaciones y anotaciones; los compendios, resúmenes y extractos; los arreglos musicales y, en general, cualesquiera transformaciones de una obra literaria, artística o científica, salvo que la obra resultante tenga el carácter de obra conjunta en cuyo caso no será considerada como una obra derivada a los efectos de esta licencia. Para evitar la duda, si la obra consiste en una composición musical o grabación de sonidos, la sincronización temporal de la obra con una imagen en movimiento (“synching”) será considerada como una obra derivada a los efectos de esta licencia.

5. Tendrán la consideración de “obras audiovisuales” las creaciones expresadas mediante una serie de imágenes asociadas, con o sin sonorización incorporada, así como las composiciones musicales, que estén destinadas esencialmente a ser mostradas a través de aparatos de proyección o por cualquier otro medio de comunicación pública de la imagen y del sonido, con independencia de la naturaleza de los soportes materiales de dichas obras.

6. El “licenciador” es la persona o la entidad que ofrece la obra bajo los términos de esta licencia y le cede los derechos de explotación de la misma conforme a lo dispuesto en ella.

7. “Usted” es la persona o la entidad que ejercita los derechos cedidos mediante esta licencia y que no ha violado previamente los términos de la misma con respecto a la obra, o que ha recibido el permiso expreso del licenciador de ejercitar los derechos cedidos mediante esta licencia a pesar de una violación anterior.

8. La “transformación” de una obra comprende su traducción, adaptación y cualquier otra modificación en su forma de la que se derive una obra diferente. Cuando se trate de una base de datos según se define más adelante, se considerará también transformación la reordenación de la misma. La creación resultante de la transformación de una obra tendrá la consideración de obra derivada.

9. Se entiende por “reproducción” la fijación de la obra en un medio que permita su comunicación y la obtención de copias de toda o parte de ella.

10. Se entiende por “distribución” la puesta a disposición del público del original o copias de la obra mediante su venta, alquiler, préstamo o de cualquier otra forma.

11. Se entenderá por “comunicación pública” todo acto por el cual una pluralidad de personas pueda tener acceso a la obra sin previa distribución de ejemplares a cada una de ellas. No se considerará pública la comunicación cuando se celebre dentro de un ámbito estrictamente doméstico que no esté integrado o conectado a una red de difusión de cualquier tipo. A efectos de esta licencia se considerará comunicación pública la puesta a disposición del público de la obra por procedimientos alámbricos o inalámbricos, incluida la puesta a disposición del público de la obra de tal forma que cualquier persona pueda acceder a ella desde el lugar y en el momento que elija.

12. La “explotación” de la obra comprende su reproducción, distribución, comunicación pública y transformación.

13. Tendrán la consideración de “bases de datos” las colecciones de obras ajenas, de datos o de otros elementos independientes como las antologías y las bases de datos propiamente dichas que por la selección o disposición de sus contenidos constituyan creaciones intelectuales, sin perjuicio, en su caso, de los derechos que pudieran subsistir sobre dichos contenidos.

14. Los “elementos de la licencia” son las características principales de la licencia según la selección efectuada por el licenciador e indicadas en el título de esta

licencia: Reconocimiento de autoría (Reconocimiento), Compartir de manera igual (CompartirIgual).

2. Límites y uso legítimo de los derechos. Nada en esta licencia pretende reducir o restringir cualesquiera límites legales de los derechos exclusivos del titular de los derechos de propiedad intelectual de acuerdo con la Ley de Propiedad Intelectual o cualesquiera otras leyes aplicables, ya sean derivados de usos legítimos, tales como el derecho de copia privada o el derecho a cita, u otras limitaciones como la derivada de la primera venta de ejemplares.

3. Concesión de licencia. Conforme a los términos y a las condiciones de esta licencia, el licenciador concede (durante toda la vigencia de los derechos de propiedad intelectual) una licencia de ámbito mundial, sin derecho de remuneración, no exclusiva e indefinida que incluye la cesión de los siguientes derechos:

1. Derecho de reproducción, distribución y comunicación pública sobre la obra.

2. Derecho a incorporarla en una o más obras conjuntas o bases de datos y para su reproducción en tanto que incorporada a dichas obras conjuntas o bases de datos.

3. Derecho para efectuar cualquier transformación sobre la obra y crear y reproducir obras derivadas.

4. Derecho de distribución y comunicación pública de copias o grabaciones de la obra, como incorporada a obras conjuntas o bases de datos.

5. Derecho de distribución y comunicación pública de copias o grabaciones de la obra, por medio de una obra derivada.

6. Para evitar la duda, sin perjuicio de la preceptiva autorización del licenciador, y especialmente cuando la obra se trate de una obra audiovisual, el licenciador renuncia al derecho exclusivo a percibir, tanto individualmente como mediante una entidad de gestión de derechos, o varias, (por ejemplo: SGAE, Dama, VEGAP), los derechos de explotación de la obra, así como los derivados de obras derivadas, conjuntas o bases de datos, si dicha explotación pretende principalmente o se encuentra dirigida hacia la obtención de un beneficio mercantil o la remuneración monetaria privada.

Los anteriores derechos se pueden ejercitar en todos los medios y formatos, tangibles o intangibles, conocidos o por conocer. Los derechos mencionados incluyen el derecho a efectuar las modificaciones que sean precisas técnicamente para el ejercicio de los derechos en otros medios y formatos. Todos los derechos no cedidos expresamente por el licenciador quedan reservados.

4. Restricciones. La cesión de derechos que supone esta licencia se encuentra sujeta y limitada a las restricciones siguientes:

1. Usted puede reproducir, distribuir o comunicar públicamente la obra solamente bajo los términos de esta licencia y debe incluir una copia de la misma, o su Identificador Uniforme de Recurso (URI), con cada copia o grabación de la obra que usted reproduzca, distribuya o comunique públicamente. Usted no puede ofrecer o imponer ningún término sobre la obra que altere o restrinja los términos de esta licencia o el ejercicio de sus derechos por parte de los cesionarios de la misma. Usted no puede sublicenciar la obra. Usted debe mantener intactos todos los avisos que se refieran a esta licencia y a la ausencia de garantías. Usted no puede reproducir, distribuir o comunicar públicamente la obra con medidas tecnológicas que controlen el acceso o uso de la obra de una manera contraria a los términos de esta licencia. Lo anterior se aplica a una obra en tanto que incorporada a una obra conjunta o base de datos, pero no implica que éstas, al margen de la obra objeto de esta licencia, tengan que estar sujetas a los términos de la misma. Si usted crea una obra conjunta o base de datos, previa comunicación del licenciador, usted deberá quitar de la obra conjunta o base de datos cualquier crédito requerido en el apartado 4c, según lo que se le requiera y en la medida de lo posible. Si usted crea una obra derivada, previa comunicación del licenciador, usted deberá quitar de la obra derivada cualquier crédito requerido en el apartado 4c, según lo que se le requiera y en la medida de lo posible.

2. Usted puede reproducir, distribuir o comunicar públicamente una obra derivada solamente bajo los términos de esta licencia, o de una versión posterior de esta licencia con sus mismos elementos principales, o de una licencia iCommons de Creative Commons que contenga los mismos elementos principales que esta licencia (ejemplo: Reconocimiento-CompartirIgual 2.5 Japón). Usted debe incluir una copia de la esta licencia o de la mencionada anteriormente, o bien su Identificador Uniforme de Recurso (URI), con cada copia o grabación de la obra que

usted reproduzca, distribuya o comunique públicamente. Usted no puede ofrecer o imponer ningún término respecto de las obras derivadas o sus transformaciones que alteren o restrinjan los términos de esta licencia o el ejercicio de sus derechos por parte de los cesionarios de la misma. Usted debe mantener intactos todos los avisos que se refieran a esta licencia y a la ausencia de garantías. Usted no puede reproducir, distribuir o comunicar públicamente la obra derivada con medidas tecnológicas que controlen el acceso o uso de la obra de una manera contraria a los términos de esta licencia. Lo anterior se aplica a una obra derivada en tanto que incorporada a una obra conjunta o base de datos, pero no implica que éstas, al margen de la obra objeto de esta licencia, tengan que estar sujetas a los términos de esta licencia.

3. Si usted reproduce, distribuye o comunica públicamente la obra o cualquier obra derivada, conjunta o base datos que la incorpore, usted debe mantener intactos todos los avisos sobre la propiedad intelectual de la obra y reconocer al autor original, de manera razonable conforme al medio o a los medios que usted esté utilizando, indicando el nombre (o el seudónimo, en su caso) del autor original si es facilitado, y/o reconocer a aquellas partes (por ejemplo: institución, publicación, revista) que el autor original y/o el licenciador designen para ser reconocidos en el aviso legal, las condiciones de uso, o de cualquier otra manera razonable; el título de la obra si es facilitado; de manera razonable, el Identificador Uniforme de Recurso (URI), si existe, que el licenciador especifica para ser vinculado a la obra, a menos que tal URI no se refiera al aviso sobre propiedad intelectual o a la información sobre la licencia de la obra; y en el caso de una obra derivada, un aviso que identifique el uso de la obra en la obra derivada (e.g., "traducción castellana de la obra de Autor Original,.º "guión basado en obra original de Autor Original"). Tal aviso se puede desarrollar de cualquier manera razonable; con tal de que, sin embargo, en el caso de una obra derivada, conjunta o base datos, aparezca como mínimo este aviso allá donde aparezcan los avisos correspondientes a otros autores y de forma comparable a los mismos.

4. En el caso de la inclusión de la obra en alguna base de datos o recopilación, el propietario o el gestor de la base de datos deberá renunciar a cualquier derecho relacionado con esta inclusión y concerniente a los usos de la obra una vez extraída de las bases de datos, ya sea de manera individual o conjuntamente con

otros materiales.

5. Exoneración de responsabilidad.

A MENOS QUE SE ACUERDE MUTUAMENTE ENTRE LAS PARTES, EL LICENCIADOR OFRECE LA OBRA TAL CUAL (ON AN .ᴬS-IS"BASIS) Y NO CONFIERE NINGUNA GARANTÍA DE CUALQUIER TIPO RESPECTO DE LA OBRA O DE LA PRESENCIA O AUSENCIA DE ERRORES QUE PUEDAN O NO SER DESCUBIERTOS. ALGUNAS JURISDICCIONES NO PERMITEN LA EXCLUSIÓN DE TALES GARANTÍAS, POR LO QUE TAL EXCLUSIÓN PUEDE NO SER DE APLICACIÓN A USTED.

6. Limitación de responsabilidad.

SALVO QUE LO DISPONGA EXPRESA E IMPERATIVAMENTE LA LEY APLICABLE, EN NINGÚN CASO EL LICENCIADOR SERÁ RESPONSABLE ANTE USTED POR CUALQUIER TEORÍA LEGAL DE CUALESQUIERA DAÑOS RESULTANTES, GENERALES O ESPECIALES (INCLUIDO EL DAÑO EMERGENTE Y EL LUCRO CESANTE), FORTUITOS O CAUSALES, DIRECTOS O INDIRECTOS, PRODUCIDOS EN CONEXIÓN CON ESTA LICENCIA O EL USO DE LA OBRA, INCLUSO SI EL LICENCIADOR HUBIERA SIDO INFORMADO DE LA POSIBILIDAD DE TALES DAÑOS.

7. Finalización de la licencia.

1. Esta licencia y la cesión de los derechos que contiene terminarán automáticamente en caso de cualquier incumplimiento de los términos de la misma. Las personas o entidades que hayan recibido obras derivadas, conjuntas o bases de datos de usted bajo esta licencia, sin embargo, no verán sus licencias finalizadas, siempre que tales personas o entidades se mantengan en el cumplimiento íntegro de esta licencia. Las secciones 1, 2, 5, 6, 7 y 8 permanecerán vigentes pese a cualquier finalización de esta licencia.

2. Conforme a las condiciones y términos anteriores, la cesión de derechos de esta licencia es perpetua (durante toda la vigencia de los derechos de propiedad intelectual aplicables a la obra). A pesar de lo anterior, el licenciador se reserva el derecho a divulgar o publicar la obra en condiciones distintas a las presentes, o de retirar la obra en cualquier momento. No obstante, ello no supondrá dar por concluida esta licencia (o cualquier otra licencia que haya sido concedida,

o sea necesario ser concedida, bajo los términos de esta licencia), que continuará vigente y con efectos completos a no ser que haya finalizado conforme a lo establecido anteriormente.

8. Miscelánea.

1. Cada vez que usted explote de alguna forma la obra, o una obra conjunta o una base datos que la incorpore, el licenciador original ofrece a los terceros y sucesivos licenciatarios la cesión de derechos sobre la obra en las mismas condiciones y términos que la licencia concedida a usted.

2. Cada vez que usted explote de alguna forma una obra derivada, el licenciador original ofrece a los terceros y sucesivos licenciatarios la cesión de derechos sobre la obra original en las mismas condiciones y términos que la licencia concedida a usted.

3. Si alguna disposición de esta licencia resulta inválida o inaplicable según la Ley vigente, ello no afectará la validez o aplicabilidad del resto de los términos de esta licencia y, sin ninguna acción adicional por cualquiera las partes de este acuerdo, tal disposición se entenderá reformada en lo estrictamente necesario para hacer que tal disposición sea válida y ejecutiva.

4. No se entenderá que existe renuncia respecto de algún término o disposición de esta licencia, ni que se consiente violación alguna de la misma, a menos que tal renuncia o consentimiento figure por escrito y lleve la firma de la parte que renuncie o consienta.

5. Esta licencia constituye el acuerdo pleno entre las partes con respecto a la obra objeto de la licencia. No caben interpretaciones, acuerdos o términos con respecto a la obra que no se encuentren expresamente especificados en la presente licencia. El licenciador no estará obligado por ninguna disposición complementaria que pueda aparecer en cualquier comunicación de usted. Esta licencia no se puede modificar sin el mutuo acuerdo por escrito entre el licenciador y usted.

Creative Commons no es parte de esta licencia, y no ofrece ninguna garantía en relación con la obra. Creative Commons no será responsable frente a usted o a cualquier parte, por cualquier teoría legal de cualesquiera daños resultantes, incluyendo, pero no

limitado, daños generales o especiales (incluido el daño emergente y el lucro cesante), fortuitos o causales, en conexión con esta licencia. A pesar de las dos (2) oraciones anteriores, si Creative Commons se ha identificado expresamente como el licenciador, tendrá todos los derechos y obligaciones del licenciador.

Salvo para el propósito limitado de indicar al público que la obra está licenciada bajo la CCPL, ninguna parte utilizará la marca registrada “Creative Commons” o cualquier marca registrada o insignia relacionada con “Creative Commons” sin su consentimiento por escrito. Cualquier uso permitido se hará de conformidad con las pautas vigentes en cada momento sobre el uso de la marca registrada por “Creative Commons”, en tanto que sean publicadas su sitio web (website) o sean proporcionadas a petición previa.

Puede contactar con Creative Commons en: `http://creativecommons.org/`.