Escuela Politécnica Superior

Universidad de Córdoba

Software Libre y Compromiso Social

Memoria de prácticas

Eduardo Arroyo Ramírez
i12arrae@uco.es
Curso 2019/2020

Índice

Índice de figuras

Parte I
Práctica 1

1. Ejercicio 1

Elabora una lista de 10 proyectos destacados de software libre y describir los siguientes aspectos de cada uno de esos proyectos:

1. *Nombre, página web, autor o autores principales.*
2. *Tamaño de la comunidad que lo mantiene. ¿Es mucho o poco activa dicha comunidad?.*
3. *¿Hay alguna empresa privada en torno al proyecto?.*
4. *Forma de financiación.*
5. *Tipo de licencia de software libre que utiliza el proyecto.*
6. *Modelo de desarrollo, lenguaje de programación, sistema operativo y/o plataforma, etc.*
7. *Descripción general del proyecto. Escribe 3 o 4 párrafos que describen el proyecto, su funcionalidad y su utilidad.*

1.1. Apache

1. Web: https://httpd.apache.org/.
   Autores principales: https://httpd.apache.org/contributors/.

2. Según la web de apache, la comunidad está formada por voluntarios y hay aproximadamente 7000 personas que han aportado código. Parece una comunidad muy activa ya que según reza la web, tienen más de 200 millones de líneas de código en revisión y más de 3 millones de commits.

3. Apache Software Foundation (ASF) es una una fundación creada para dar soporte a los proyectos de software bajo la denominación Apache, incluyendo el popular servidor HTTP Apache. La ASF se formó a partir del llamado Grupo Apache y fue registrada en Delaware (Estados Unidos), en junio de 1999.

4. La Apache Software Foundation tiene un programa de esponsorización. Entre los principales sponsors se encuentran empresas de primerísimo nivel como Google, Yahoo, Microsoft, Hewlett-Packard, Facebook, AMD, IBM... Cuentan con un apartado de agradecimiento a los sponsors en su página web. Además admiten donativos individuales.También agradecen a la red de mirrors que facilitan la distribución del software de Apache por todo el mundo y cuyo servicio ofrecen a la ASF sin coste alguno.

5. Apache 2.0. La licencia de software bajo la cual el software de la fundación Apache es distribuido es una parte distintiva de la historia de Apache HTTP Server y de la comunidad de código abierto. La Licencia Apache permite la distribución de derivados de código abierto y cerrado a partir de su código fuente original.

6. Utilizan un modelo de gestión de proyectos al que llaman “The Apache Way. Lo explican bastante en profundidad en este video. El lenguaje utilizado es C. Utilizan Subversion para el control de versiones, aunque también tienen repositorios Git. Existen versiones de Apache para Unix, Linux, BSD, Windows, macOS, y otras.

7. El servidor HTTP Apache es un servidor web HTTP de código abierto, para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual según la normativa RFC 2616. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPd 1.3, pero más tarde fue reescrito por

completo. La arquitectura del servidor Apache es muy modular. El servidor consta de una sección core y diversos módulos que aportan mucha de la funcionalidad que podría considerarse básica para un servidor web.

1.2. Git

1. Web: https://git-scm.com/ Autor: Linus Torvalds, Supervisor: Junio Hamano
2. Según Wikipedia, la comunidad está formada por alrededor de 280 programadores. La web principal tiene un apartado con instrucciones para la comunidad, que se comunica principalmente mediante listas de correo. A juzgar por las fechas de los mensajes en el archivo y los 58562 commits al proyecto principal en github, la comunidad es realmente activa.
3. En la página principal se cita una lista de empresas y proyectos que utilizan git. Entre los que destacan Google, Facebook, Microsoft, Twitter, LInkedIn o Netflix.
4. Según la página principal, Git es miembro de la Software Freedom Conservancy, que según su web es una organización sin ánimo de lucro que promueve, mejora, desarrolla y defiende proyectos software gratuitos, libres y de código abierto (FLOSS). A su vez, esta organización se financia mediante sponsors y donaciones de pequeñas empresas y particulares.
5. La propia web de git tiene un apartado que explica que Git se licencia bajo GPLv2 “to guarantee your freedom to share and change free software—to make sure the software is free for all its users”. También especifican que el uso del término “Git” y los logos está restringido.
6. Git está desarrollado en C y Perl principalmente, aunque existen implementaciones escritas en otros lenguajes. Existen versiones de Git para Linux, Windows, BSD, Solaris y macOS.
7. Git es un software de control de versiones diseñado pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos.

1.3. LibreOffice

1. Web: https://es.libreoffice.org/.
   Autores principales: https://es.libreoffice.org/comunidad/desarrolladores/
2. La comunidad se divide en varios equipos diferenciados: desarrollo, documentación, infraestructura, diseño, traducción, control de calidad y publicidad.
3. The Document Foundation es dirigida e impulsada por sus miembros. Su misión es apoyar y fomentar el desarrollo y el proyecto LibreOffice, así como la su representación.
4. Reciben donaciones a través de esta página para costear los servidores y la infraestructura.
5. El proyecto LibreOffice emplea una licencia dual LGPLv3 (o posterior) / MPL para nuevas contribuciones a fin de permitir que la licencia sea actualizada.
6. En su desarrollo se utilizan los lenguajes C++, Java y Python. Utilizan Git como sistema de control de versiones y el trabajo se administra con Gerrit. Existen versiones de la suite para Windows, macOS y GNU/Linux.
7. LibreOffice es un paquete de software de oficina. Se creó en 2010 como bifurcación de OpenOffice.org, otro antiguo proyecto de código abierto, que a su vez tenía como base inicial a la suite ofimática StarOffice, desarrollada por StarDivision, adquirida por Sun Microsystems en agosto de 1999.

1.4. Gimp

1. The GNU Manipulation Program. Web: s://www.gimp.org/. Autores principales: Originalmente Spencer Kimball and Peter Mattis. La persona actualmente a cargo es Michael Natterer. Øyvind Kolås es el desarrollador principal de GEGL, el núcleo de GIMP. En esta página se puede encontrar una lista completa de las personas que han contribuido al código de GIMP.

2. La comunidad se organiza en torno a una wiki y un canal de IRC. Además existen multitud de foros de usuarios. La comunidad de desarrollo es bastante activa a juzgar por las fechas de las entradas en la web de seguimiento en Gitlab. En la página sobre el desarrollo hay un gráfico que muestra una actividad constante en los últimos 5 años de commits, y además reza que hay un total de 72 contribuyentes activos.

3. GIMP forma parte del proyecto GNU. La fundación GNOME ayuda a GIMP canalizando las donaciones al proyecto GIMP a través de su organización..

4. El proyecto se financia mediante donaciones a través de plataformas de pago como Paypal. Además, Øyvind Kolås y Jehan Pagès tienen sendas páginas de mecenazgo donde reciben donaciones por su trabajo en GEGL y para el desarrollo de funcionalidades de animación en GIMP respectivamente.

5. GPLv3+ y LGPL

6. Está escrito en C y utiliza GTK. Utilizan Git como sistema de control de versiones, alojado por GNOME Foundation. Está disponible para Unix, GNU/Linux, FreeBSD, Solaris, Windows y macOS entre otros. El desarrollo se gestiona desde una wiki y se utiliza un canal de IRC (irc://irc.gimp.org/#gimp) para la comunicación directa entre desarrolladores.

7. GIMP (siglas en inglés de GNU Image Manipulation Program) es un programa de edición de imágenes digitales en forma de mapa de bits, tanto dibujos como fotografías. GIMP tiene herramientas que se utilizan para el retoque y edición de imágenes, dibujo de formas libres, cambiar el tamaño, recortar, hacer fotomontajes, convertir a diferentes formatos de imagen, y otras tareas más especializadas. Se pueden también crear imágenes animadas en formato GIF e imágenes animadas en formato MPEG usando un plugin de animación.

1.5. Symphony

1. Web: https://symfony.com. Líder actual del proyecto: Fabien Potencier

2. Según la web del proyecto, hay más de 600.000 desarrolladores implicados. Parece una comunidad muy activa.

3. Symfony es patrocinado por SensioLabs, una compañía francesa que provee consultoría, servicios, formación sobre tecnologías open source.

4. Symfony está financiado por SensioLabs

5. Utiliza licencia MIT

6. Está desarrollado en PHP y se puede utilizar en plataformas Unix, GNU/Linux, macOS y Windows.

7. Symfony es un framework diseñado para desarrollar aplicaciones web basado en el patrón Modelo Vista Controlador. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

1.6. Node.js

1. Web: nodejs.org. Autor: Ryan Dahl

2. La comunidad es muy activa en los grupos de google como nodejs, nodejs-dev y también su canal de IRC #node.js en freenode. Se reúnen en la NodeConf.

3. OpenJS Foundation / Joyent

4. No he encontrado información acerca del modelo de financiación del proyecto.
5. Licencia MIT.
6. El cuerpo de operaciones de base de Node.js está escrito en JavaScript con métodos de soporte escritos en C++. Utiliza el motor javascript V8 originalmente creado para Google Chrome, que está escrito en C++. NodeJS está disponible para sistemas Unix, GNU/Linux, macOS, Windows...
7. Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación ECMAScript, asíncrono, con E/S de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google. Fue creado con el enfoque de ser útil en la creación de programas de red altamente escalables, como por ejemplo, servidores web.

1.7. Ubuntu

1. Web: https://ubuntu.com/. Autor
2. La comunidad se coordina desde https://discourse.ubuntu.com/. Ésta participa en el desarrollo, arreglo de bugs, traducción, documentación, etc. No he encontrado datos sobre el tamaño de la comunidad.
3. Canonical
4. Ubuntu se financia principalmente a través de la actividad de Canonical.
5. GPL y otras licencias libres
6. Según la guía de instalación, Ubuntu soporta las seis arquitecturas principales (x86, x64, ARM, ARM64, IBM POWER e IBM z, y muchas de sus variantes.
   Además de la arquitectura IBM/Motorola PowerPC, que tiene un port no oficial.
7. Ubuntu es un sistema operativo de código abierto y libre para computadores. Es una distribución de Linux basada en Debian. Actualmente corre en computadores de escritorio y servidores. Está orientado al usuario promedio, con un fuerte enfoque en la facilidad de uso y en mejorar la experiencia del usuario. Está compuesto de múltiple software normalmente distribuido bajo una licencia libre o de código abierto.

   Estadísitcas web sugieren que la cuota de mercado de Ubuntu dentro de las distribuciones Linux es, aproximadamente, del 52 %, y con una tendencia a aumentar como servidor web. Canonical, además de mantener Ubuntu, provee una versión orientada a servidores, Ubuntu Server, una versión para empresas, Ubuntu Business Desktop Remix, una para televisores, Ubuntu TV, otra versión para tabletas Ubuntu Tablet, también Ubuntu Phone y una para usar el escritorio desde teléfonos inteligentes, Ubuntu for Android.

1.8. Gnome

1. Web: https://www.gnome.org/ Autores: Miguel de Icaza y Federico Mena
2. La comunidad se comunica a través de IRC y listas de correo. No he encontrado un dato del tamaño de la comunidad, pero a juzgar por la lista de cambios recientes y la frecuencia de las actualizaciones, la comunidad de desarrolladores es muy activa.
3. GNOME Foundation (parte de GNU Project)
4. GNOME Se financia a través de sus patrocinadores como Canonical, Debian, Google, RedHat, EndlessOS, SUSE y otros. También recibe donaciones particulares.
5. GPLv2
6. GNOME está escrito principalmente en C y C++. GNOME está disponible en las principales distribuciones GNU/Linux, incluyendo Fedora, Debian, Ubuntu, Red Hat Linux, CentOS, Oracle Linux, Arch Linux y Gentoo.

7. GNOME provee un gestor de ventanas «intuitivo y atractivo» y una plataforma de desarrollo para crear aplicaciones que se integran con el escritorio. El Proyecto pone énfasis en la simplicidad, facilidad de uso y eficiencia. Tiene como objetivo la libertad para crear un entorno de escritorio que siempre tendrá el código fuente disponible para reutilizarse bajo una licencia de software libre.

   Desde GNOME 2, el enfoque fue puesto en la productividad. Con este fin, se crearon las Pautas de Interfaz Humana de GNOME (Human Interface Guidelines, HIG). Todos los programas de GNOME comparten un estilo coherente de interfaz gráfica de usuario (GUI), pero no están limitados a los mismos widgets de GUI. Por el contrario, el diseño de la GUI de GNOME está guiado por conceptos que se describen en GNOME HIG, que a su vez depende de la ergonomía cognitiva. Después de HIG, los desarrolladores pueden crear programas de GUI de alta calidad, consistentes y utilizables, ya que abordan todo, desde el diseño de la GUI hasta el diseño recomendado de widgets basado en píxeles.

1.9. GCC

1. Web: https://www.gnu.org/software/gcc/. Autor: Proyecto GNU
2. GCC mantiene una larguísima lista de colaboradores en su web https://gcc.gnu.org/onlinedocs/gcc/Contributors.html
3. Free Software Foundation
4. La FSF recibe financiación de sus “patrons” y de donaciones particulares.
5. GPLv3
6. Gcc está escrito en C
7. El GNU Compiler Collection (colección de compiladores GNU) es un conjunto de compiladores creados por el proyecto GNU. GCC es software libre y lo distribuye la Free Software Foundation (FSF) bajo la licencia general pública GPL. Estos compiladores se consideran estándar para los sistemas operativos derivados de UNIX, de código abierto y también de propietarios, como Mac OS X. GCC requiere el conjunto de aplicaciones conocido como binutils para realizar tareas como identificar archivos objeto u obtener su tamaño para copiarlos, traducirlos o crear listas, enlazarlos, o quitarles símbolos innecesarios. Originalmente GCC significaba GNU C Compiler (compilador GNU de C), porque solo compilaba el lenguaje C. Posteriormente se extendió para compilar C++, Fortran, Ada y otros.

1.10. React

1. Web https://reactjs.org/, Autor: Jordan Walke
2. La comunidad es grande y activa
3. Facebook Inc.
4. Es financiado por Facebook.
5. Actualmente MIT, aunque empezó con ApacheV2 y luego paso a BSD de 3 cláusulas.
6. Desarrollado en Javascript. React es una biblioteca javascript para web, por lo que es soportado por cualquiera de los navegadores modernos.
7. React (también llamada React.js o ReactJS) es una biblioteca Javascript de código abierto diseñada para crear interfaces de usuario con el objetivo de facilitar el desarrollo de aplicaciones en una sola página. Es mantenido por Facebook y la comunidad de software libre, han participado en el proyecto más de mil desarrolladores diferentes. React intenta ayudar a los desarrolladores a construir aplicaciones que usan datos que cambian todo el tiempo. Su objetivo es ser sencillo, declarativo y fácil de combinar.
   React sólo maneja la interfaz de usuario en una aplicación; React es la Vista en un contexto en el que se use el patrón MVC (Modelo-Vista-Controlador) o MVVM (Modelo-vista-modelo de vista). También puede ser utilizado con las extensiones de React-based que se encargan de las partes no-UI (que no forman parte de la interfaz de usuario) de una aplicación web.

2. Ejercicio 2

Familiarízate con las comunidades de proyectos emblemáticos y también personajes del software libre como son:

1. *Free Software Movement. Breve descripción y describe alguna de sus iniciativas.*
2. *Open Source Initiative. Describe alguna de sus iniciativas.*
3. *Diferencias entre Free Software Foundation y GNU Project*
4. *Describe brevemente algunos de los principales proyectos que se traen entre manos en proyecto GNU*
5. *Linux distros: Debian, Ubuntu, Mint*
6. *Analiza y describe brevemente las figuras de Richard Stallman (GNU), Linus Torvalds (Linux), Eric S. Raymond ("La Catedral y el Bazar"). Añade dos figuras más (en total serían 5) que han aportado al mundo del Software libre y escribe una breve descripción de ellas.*

2.1. Free Software Movement

El movimiento del software libre es un movimiento social con el objetivo de obtener y garantizar las libertades que permiten a los usuarios de software ejecutarlo, estudiarlo, cambiarlo y redistribuir copias del mismo con o sin cambios.

Sobre la base de las tradiciones y filosofías de la cultura hacker y el mundo académico de los años 1970s, Richard Stallman fundó formalmente el movimiento en 1983, con el lanzamiento del Proyecto GNU. Stallman estableció la Fundación del Software Libre en 1985 para apoyar el movimiento. La meta del movimiento fue dar libertad a los usuarios, reemplazando el software con términos de licencia restrictivos, como el software privativo, por software libre.

2.2. Open Source Initiative

La Open Source Initiative es una organización (public benefit corporation) californiana fundada en el año 1998 por Bruce Perens y Eric S. Raymond, que promueve el uso de software de código abierto.

2.3. Diferencias entre la FSF y GNU Project

La Free Software Foundation es una organización sin ánimo de lucro con la misión de promover la libertad de los usuarios de computadoras. Defienden los derechos de todos los usuarios de software y centra su trabajo en asuntos legales, organizativos y promocionales en beneficio de la comunidad de usuarios de software libre. Por su parte, el Proyecto GNU es un proyecto colaborativo con el objetivo de crear un sistema operativo completamente libre, incluyendo sus herramientas de aplicación.

El objetivo inicial de la Free Software Foundation era recaudar fondos para ayudar a programar GNU.

2.4. Principales proyectos GNU

El Proyecto GNU consta de una serie de pequeños subproyectos mantenidos por voluntarios, empresas o combinaciones de ambos. Estos subproyectos también se denominan «Proyectos de GNU» o «Paquetes GNU». Algunos de los más conocidos son:

2.4.1. GCC

El GNU Compiler Collection (colección de compiladores GNU) es un conjunto de compiladores creados por el proyecto GNU. GCC es software libre y lo distribuye la Free Software Foundation (FSF) bajo la licencia general pública GPL. Estos compiladores se consideran estándar para los sistemas operativos derivados de UNIX, de código abierto y también de propietarios, como Mac OS X. GCC requiere el conjunto de aplicaciones conocido como binutils para realizar tareas como identificar archivos objeto u

obtener su tamaño para copiarlos, traducirlos o crear listas, enlazarlos, o quitarles símbolos innecesarios. Originalmente GCC significaba GNU C Compiler (compilador GNU de C), porque solo compilaba el lenguaje C. Posteriormente se extendió para compilar C++, Fortran, Ada y otros.

GCC es parte del proyecto GNU, y tiene como objetivo mejorar el compilador usado en todos los sistemas GNU, incluyendo la variante GNU/Linux. El desarrollo de GCC usa un entorno de desarrollo abierto y soporta muchas plataformas con el fin de fomentar el uso de un compilador-optimizador de clase global, que pueda atraer muchos equipos de desarrollo, y asegure que GCC y los sistemas GNU funcionen en diferentes arquitecturas y diferentes entornos, y más aún, para extender y mejorar las características de GCC.

2.4.2. GNOME

GNOME es un entorno de escritorio e infraestructura de desarrollo para sistemas operativos Unix, GNU/Linux y derivados Unix como BSD o Solaris; compuesto enteramente de software libre.

El proyecto fue iniciado por los programadores mexicanos Miguel de Icaza y Federico Mena en agosto de 1997 y forma parte oficial del proyecto GNU. Nació como una alternativa a KDE bajo el nombre de GNU Network Object Model Environment (Entorno de Modelo de Objeto de Red GNU). Actualmente, incluyendo al español, se encuentra disponible en 166 idiomas.

GNOME está disponible en las principales distribuciones GNU/Linux, incluyendo Fedora, Debian, Ubuntu, Red Hat Linux, CentOS, Oracle Linux, Arch Linux y Gentoo.

2.4.3. Bash

GNU Bash o simplemente Bash (Bourne-again shell) es un lenguaje de comandos y shell de Unix escrito por Brian Fox para el Proyecto GNU como un reemplazo de software libre para el shell Bourne. Lanzado por primera vez en 1989, se ha utilizado ampliamente como el shell de inicio de sesión predeterminado para la mayoría de las distribuciones de Linux y MacOS Mojave de Apple y versiones anteriores. Una versión también está disponible para Windows 10 y Android. También es el shell de usuario predeterminado en Solaris 11.

Bash es un procesador de comandos que generalmente se ejecuta en una ventana de texto donde el usuario escribe comandos que causan acciones. Bash también puede leer y ejecutar comandos desde un archivo, llamado script de shell. Al igual que todos los shells de Unix, es compatible con el agrupamiento de nombres de archivo (coincidencia de comodines), tuberías, here documents, sustitución de comandos, variables y estructuras de control para pruebas de condición e iteración. Las palabras reservadas, la sintaxis, las variables de ámbito dinámico y otras características básicas del lenguaje se copian de sh. Otras características, por ejemplo, el historial, se copian de csh y ksh. Bash es un shell compatible con POSIX, pero con varias extensiones.

2.4.4. GIMP

GIMP (siglas en inglés de GNU Image Manipulation Program) es un programa de edición de imágenes digitales en forma de mapa de bits, tanto dibujos como fotografías. Es un programa libre y gratuito. Forma parte del proyecto GNU y está disponible bajo la Licencia pública general de GNU y GNU Lesser General Public License.

Es el programa de manipulación de gráficos disponible en más sistemas operativos (Unix, GNU/Linux, FreeBSD, Solaris, Microsoft Windows y macOS, entre otros).

GIMP tiene herramientas que se utilizan para el retoque y edición de imágenes, dibujo de formas libres, cambiar el tamaño, recortar, hacer fotomontajes, convertir a diferentes formatos de imagen, y otras tareas más especializadas. Se pueden también crear imágenes animadas en formato GIF e imágenes animadas en formato MPEG usando un plugin de animación.

Los desarrolladores y encargados de mantener GIMP se esfuerzan en mantener y desarrollar una aplicación gráfica de software libre, de alta calidad para la edición y creación de imágenes originales, de fotografías, de íconos, de elementos gráficos tanto de páginas web como de elementos artísticos de interfaz de usuario.

2.5. Distribuciones Linux

2.5.1. Debian

Debian o Proyecto Debian, (en inglés, Debian Project) es una comunidad conformada por desarrolladores y usuarios, que mantiene un sistema operativo GNU basado en software libre. El sistema se encuentra precompilado, empaquetado y en formato deb para múltiples arquitecturas de computador y para varios núcleos.

El proyecto Debian fue anunciado inicialmente 1993 por Ian Murdock. El nombre Debian proviene de la combinación del nombre de su entonces novia (y posteriormente ex-esposa) Deborah y el suyo, por lo tanto, Deb(orah) e Ian. Debian 0.01 fue lanzado el 15 de septiembre de 1993, y la primera versión estable fue hecha en 1996.

Nació como una apuesta por separar en sus versiones el software libre del software no libre. El modelo de desarrollo del proyecto es ajeno a motivos empresariales o comerciales, siendo llevado adelante por los propios usuarios, aunque cuenta con el apoyo de varias empresas en forma de infraestructuras. Debian no vende directamente su software, lo pone a disposición de cualquiera en Internet, aunque sí permite a personas o empresas distribuirlo comercialmente mientras se respete su licencia.

La comunidad de desarrolladores del proyecto cuenta con la representación de Software in the Public Interest (del inglés, "software de interés público"), una organización sin ánimo de lucro que da cobertura legal a varios proyectos de software libre, con el objetivo inicial de dar cobertura legal al proyecto Debian, tras el fin del patrocinio de la FSF (Free Software Foundation).

La primera adaptación del sistema Debian, siendo también la más desarrollada, es Debian GNU/-Linux, basada en el núcleo Linux, y como siempre utilizando herramientas de GNU. Existen también otras adaptaciones con diversos núcleos: Hurd (Debian GNU/Hurd); NetBSD (Debian GNU/NetBSD) y FreeBSD (Debian GNU/kFreeBSD).

2.5.2. Ubuntu

Ubuntu es un sistema operativo de código abierto y libre para computadores. Es una distribución de Linux basada en Debian. Actualmente corre en computadores de escritorio y servidores. Está orientado al usuario promedio, con un fuerte enfoque en la facilidad de uso y en mejorar la experiencia del usuario. Está compuesto de múltiple software normalmente distribuido bajo una licencia libre o de código abierto. Estadísticas web sugieren que la cuota de mercado de Ubuntu dentro de las distribuciones Linux es, aproximadamente, del 52 %, y con una tendencia a aumentar como servidor web.

Su patrocinador, Canonical, es una compañía británica propiedad del empresario sudafricano Mark Shuttleworth. Ofrece el sistema de manera gratuita, y se financia por medio de servicios vinculados al sistema operativo y vendiendo soporte técnico. Además, al mantenerlo libre y gratuito, la empresa es capaz de aprovechar los desarrolladores de la comunidad para mejorar los componentes de su sistema operativo. Extraoficialmente, la comunidad de desarrolladores proporciona soporte para otras derivaciones de Ubuntu, con otros entornos gráficos, como Kubuntu, Xubuntu, Ubuntu MATE, Edubuntu, Ubuntu Studio, Mythbuntu, Ubuntu GNOME y Lubuntu.Canonical, además de mantener Ubuntu, provee una versión orientada a servidores, Ubuntu Server, una versión para empresas, Ubuntu Business Desktop Remix, una para televisores, Ubuntu TV, otra versión para tabletas Ubuntu Tablet, también Ubuntu Phone y una para usar el escritorio desde teléfonos inteligentes, Ubuntu for Android.

Cada seis meses se publica una nueva versión de Ubuntu. Esta recibe soporte por parte de Canonical durante nueve meses por medio de actualizaciones de seguridad, parches para bugs críticos y actualizaciones menores de programas. Las versiones LTS (Long Term Support), que se liberan cada dos años, reciben soporte durante cinco años en los sistemas de escritorio y de servidor.

2.5.3. Mint

Linux Mint es una distribución de GNU/Linux comunitaria de origen franco-irlandesa basada en Debian y Ubuntu que tiene por objeto proveer ün sistema operativo moderno, elegante y cómodo que sea tan potente como fácil de usar". Linux Mint soporta multimedia al incluir software propietario y empaquetado con una variedad de aplicaciones gratuitas y de código abierto.

El proyecto fue concebido por Clément Lefèbvre y está siendo activamente desarrollado por el Equipo de Linux Team y la comunidad.

2.6. Figuras del Software Libre

2.6.1. Richard Stallman

Richard Matthew Stallman (Manhattan, Nueva York, 16 de marzo de 1953), con frecuencia abreviado como «rms», es un programador estadounidense y fundador del movimiento del software libre.

Entre sus logros destacados como programador se incluye la realización del editor de texto GNU Emacs, el compilador GCC, el depurador GDB, y el lenguaje de construcción GNU Make; todos bajo la rúbrica del Proyecto GNU. Sin embargo, es principalmente conocido por el establecimiento de un marco de referencia moral, político y legal para el software libre: un modelo de desarrollo y distribución alternativo al software privativo. Es también inventor del concepto de copyleft (aunque no del término): un método para licenciar obras contempladas por el derecho de autor, de tal forma que su uso y modificación (así como de sus derivados) permanezcan siempre permitidos.

2.6.2. Linus Torvalds

Linus Benedict Torvalds (Helsinki, Finlandia, 28 de diciembre de 1969) es un ingeniero de software finlandés-estadounidense, conocido por iniciar y mantener el desarrollo del kernel (en español, núcleo) Linux, basándose en el sistema operativo libre Minix creado por Andrew S. Tanenbaum y en algunas herramientas, varias utilidades y los compiladores desarrollados por el proyecto GNU. Actualmente es responsable de la coordinación del proyecto.

2.6.3. Eric S. Raymond

Eric Steven Raymond (nacido el 4 de diciembre de 1957), también conocido como ESR, es el autor de La catedral y el bazar, ("The Cathedral & the Bazaar", en inglés) y el responsable actual del Jargon File (también conocido como The New Hacker's Dictionary). Si bien con el Jargon File obtuvo fama como historiador de la cultura hacker, se convirtió después de 1997 en una figura líder en el Movimiento del Open Source y el Código abierto. Hoy día es uno de sus personajes más famosos y controvertidos.

Raymond es un neopagano, un confeso anarcocapitalista, y un defensor del derecho a poseer y utilizar armas de fuego. Tiene un gran interés en la ciencia ficción. Es músico amateur y cinturón negro de taekwondo.

2.6.4. Guido van Rossum

Guido van Rossum es un informático, conocido por ser el autor del lenguaje de programación Python. Nació y creció en los Países Bajos.

En el ambiente de los desarrolladores del lenguaje Python también se le conoce por el título BDFL (Benevolent Dictator for Life), teniendo asignada la tarea de fijar las directrices sobre la evolución de Python, así como la de tomar decisiones finales sobre el lenguaje que todos los desarrolladores acatan. Van Rossum tiene fama de ser bastante conservador, realizando pocos cambios al lenguaje entre versiones sucesivas, intentando mantener siempre la compatibilidad con versiones anteriores. El 12 de julio de 2018, con un mensaje enviado a la lista de python-committers, anunció su retiro de los procesos de decisión.

En el año 2001 recibió el FSF Award for the Advancement of Free Software como reconocimiento por su trabajo. En diciembre de 2005 fue contratado como desarrollador por la empresa estadounidense Google. Después de siete años en Google, a principios de diciembre de 2012, anuncia su retirada de la empresa norteamericana para incorporarse en enero de 2013 a la plantilla de la compañía Dropbox.

2.6.5. Ian Murdock

Ian Ashley Murdock (Konstanz, Alemania, 28 de abril de 1973-28 de diciembre de 2015) fue un informático estadounidense y fundador del proyecto de software libre Debian.

En 1993 escribió el Manifiesto Debian mientras estudiaba en la Purdue University, donde en 1996 obtuvo su licenciatura. Fue fundador, también, de la empresa Progeny Linux Systems. Fue CTO de la Linux Foundation y líder del Proyecto Indiana cuando trabajaba para Sun Microsystems.

Parte II
Práctica 2

3. Ejercicio 1

> *Explica brevemente qué es un sistema de control de versiones distribuido y sus diferencias con respecto a uno centralizado. Entra en la web de Git y revisa la documentación. Busca manuales, tutoriales, etc. de Git. Describe y compara con Git algún otro sistema de control de versiones distribuido (Mercurial, etc.).*

Un **sistema de control de versiones distribuido** es una forma de control de versiones en la cuál el código, incluyendo el histórico, está replicado en las computadoras de cada desarrollador [2]. Esto permite a cada desarrollador trabajar en el código sin necesidad de estar conectado a un servidor central [3]. Las ventajas que ofrece un sistema distribuido frente a uno centralizado son:

- No hay una copia canónica.
- Soporta operaciones desconectadas: La mayoría de operaciones comunes como confirmaciones, consultas del historial, diff, y revertir cambios son rápidas, porque no se necesita conexión con el sistema central.
- Cada copia de trabajo es una copia de seguridad.
- Ramas experimentales: crear y eliminar ramas es rápido y simple.
- Fomenta la colaboración entre desarrolladores, permitiendo por ejemplo compartir cambios sin necesidad de confirmarlos en el servidor central.

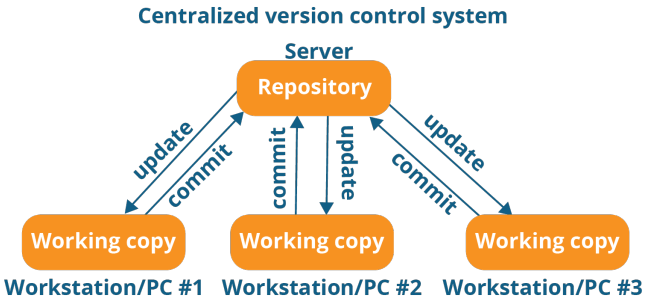

Figura 1: Esquema de control de versiones centralizado

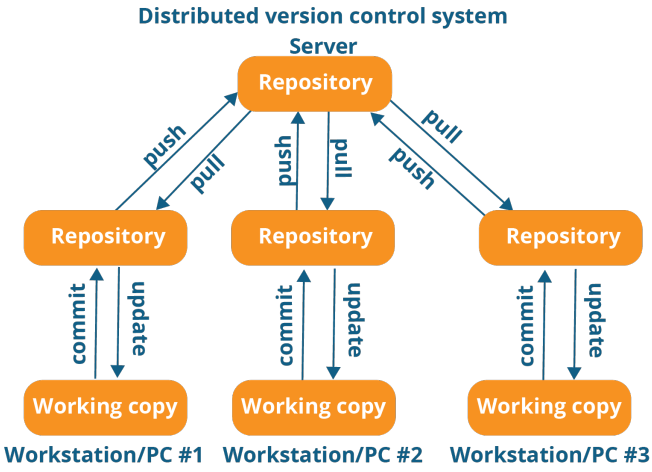

Figura 2: Esquema de control de versiones distribuido

La web de Git tiene un apartado de documentación con enlaces a los documentos siguientes:

- Manual de referencia.
- Algunos cheatsheets
- Libro online Pro Git
- Algunos videos básicos sobre Git.
- Una recopilación de tutoriales, libros, videos y cursos sobre Git.

Aparte de Git, existen otros sistemas de control de versiones distribuidos como por ejemplo Mercurial. Está implementado principalmente haciendo uso del lenguaje de programación Python, pero incluye una implementación binaria de diff escrita en C. Mercurial fue escrito originalmente para funcionar sobre GNU/Linux. Ha sido adaptado para Windows, Mac OS X y la mayoría de otros sistemas tipo Unix. Mercurial es, sobre todo, un programa para la línea de comandos. Todas las operaciones de Mercurial se invocan como opciones dadas a su programa motor, hg (cuyo nombre hace referencia al símbolo químico del mercurio) [4].

Según Intland Software, las principales diferencias entre Git y Mercurial son las siguientes [1]:

- En ambos sistemas, la historia tiene forma de grafo acíclico. Sin embargo, Mercurial ofrece un histórico lineal simple que puede causar confusión debido a la falta de información. Git, por el contrario, permite seguir el historial hacia atrás, pero es complicado de hacer.
- A menudo se piensa que Git maneja las ramas mejor que Mercurial. La estructura de ramas de Git ayuda a evitar errores en los "merges" de código.
- Git refuerza la excelencia técnica.
- Git es más potente para proyectos grandes.

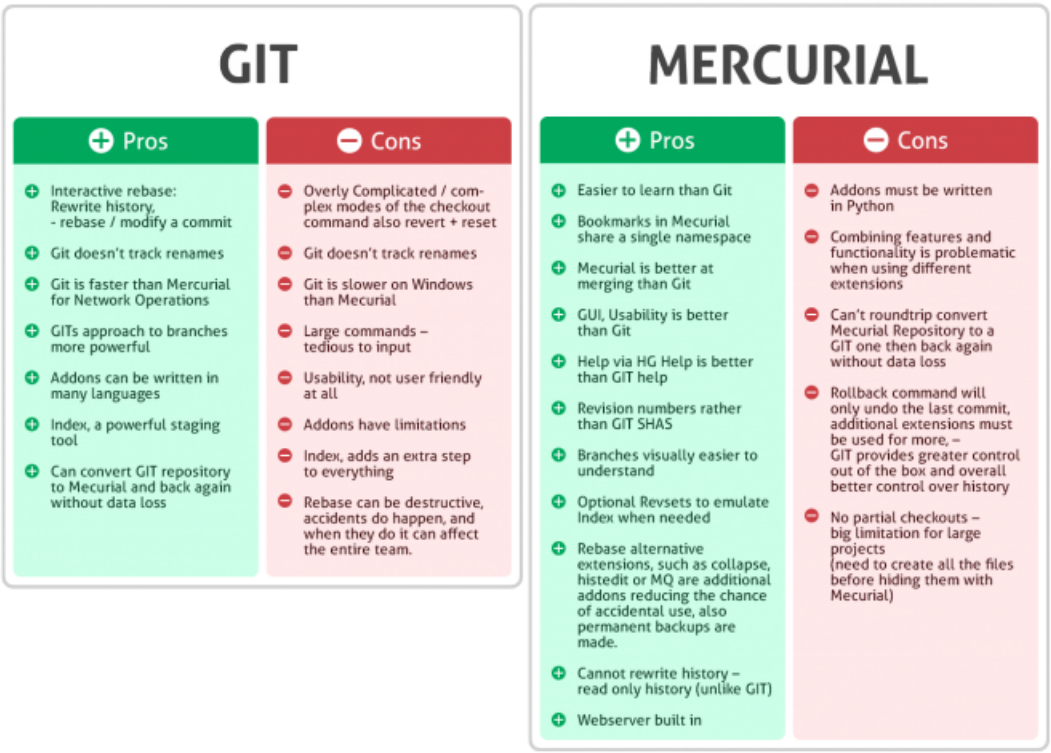

Figura 3: Pros y contras de Git y Mercurial[1]

4. Ejercicio 2

Establece tu nombre de usuario y dirección de correo electrónico. Esto es importante porque las confirmaciones de cambios (commits) en Git usan esta información, y es introducida de manera inmutable en los commits que envías. Utiliza tu usuario y correo de la UCO.

Se han utilizado los comandos siguientes sin que se produzca ninguna salida por consola.

```
$ git config --global user.name "i12arrae"
$ git config --global user.email "i12arrae@uco.es"
```

5. Ejercicio 3

Vamos a crear un nuevo repositorio desde cero (puedes usar uno tuyo o seguir el breve ejemplo que se describe a continuación). Crea en tu cuenta un nuevo directorio que se llame "reposlycs". Entra en dicho directorio e inicializa el repositorio (`$git init`). Comprueba la creación del directorio .git. ¿Qué función tiene este directorio?

Crear un directorio en `home` llamado "prueba"

```
$ cd ~
$ mkdir reposlycs
$ cd reposlycs
$ git init
Creado repositorio Git local en /home/eduarroyo/reposlycs/.git/
```

El comando ha creado un directorio llamado `.git`. Este directorio contiene todos los datos que Git necesita para el repositorio local. A destacar:

- Directorio objects: contiene todos los objetos del repositorio comprimidos y encriptados.
- Archivo config: contiene la configuración de Git para el proyecto
- Directorio refs: mantiene las referencias a las direfentes etiquetas y ramas del repositorio.
- Archivo HEAD: mantiene una referencia a la rama actual.

6. Ejercicio 4

Crea un directorio que cuelgue del anterior llamado "include".

```
$ mkdir include
```

7. Ejercicio 5

En el directorio "reposlycs" escribe un fichero llamado "helloworld.c" que contenga el siguiente código:

```c
#include <stdio.h>
int main(void)
{
    printf("Hello World!");
}
```

y un fichero "include/myinclude.h" que contenga:

```c
include <stdio.h>
void f()
{
    printf("Hello World \n");
}
```

Los archivos se crean con `touch` y luego se modifican con el editor de textos:

```
$ touch helloworld.c
$ touch include/myinclude.h
```

8. Ejercicio 6

*Añade estos ficheros a tu repositorio y comprueba el estado del repositorio con "git status". Añade los cambios al stage (**$git add ...**). Haz el primer commit con el mensaje "Initial commit" y vuelve a comprobar el estado de tu repositorio.*

Antes de hacer nada, compruebo el estado del repositorio:

```
$ git status
En la rama master

No hay commits todavía

Archivos sin seguimiento:
  (usa "git add <archivo>..." para incluirlo a lo que se será
  confirmado)

        helloworld.c
        include/

no hay nada agregado al commit pero hay archivos sin seguimiento
presentes (usa "git add" para hacerles seguimiento)
```

A continuación se añaden los dos archivos al seguimiento:

```
$ git add helloworld.c
$ git add include/myinclude.h
```

Primera confirmación de código:

```
$ git commit -m "añadir helloworld.c y myinclude.h"
[master (commit-raíz) 6f325ad] añadir helloworld.c y
myinclude.h 2 files changed, 10 insertions(+)
 create mode 100644 helloworld.c
 create mode 100644 include/myinclude.h
```

Comprobar de nuevo el estado del repositorio:

```
$ git status
En la rama master
nada para hacer commit, el árbol de trabajo está limpio
```

9. Ejercicio 7

Compila y comprueba lo que ocurre con el ejecutable generado. ¿Se podría añadir el ejecutable al staging y luego al commit?. Haz una prueba. ¿Tendría sentido hacer commit de un ejecutable?

Compilar:

```
$ gcc helloworld.c -o helloworld
```

La compilación ha producido el ejecutable helloworld. Compruebo el estado del repositorio:

```
$ git status
En la rama master
Archivos sin seguimiento:
  (usa "git add <archivo>..." para incluirlo a lo que será
  confirmado)

        helloworld

no hay nada agregado al commit pero hay archivos sin seguimiento
presentes (usa "git add" para hacerles seguimiento)
```

Podría agregar el ejecutable al seguimiento con `$ git add helloworld` pero no tiene sentido hacerlo ya que es dependiente del código fuente. Cada desarrollador debe compilar su código local para obtener un ejecutable. Creo el archivo .gitignore con el contenido helloworld para que git no tenga en cuenta el ejecutable. Añado .gitignore al seguimiento y hago un commit.

10. Ejercicio 8

Abre una rama que se llama "branchA" y modifica el fichero de la función f() con lo necesario para que f() quede así:

```
1 void f()
2 {
3     char c1[100]= "Hello world";
4     char c2[100]= ", I am <your name here>";
5     printf("%s\n", strcat(c1, c2));
6     return 0;
7 }
```

Creo la rama local "branchA":

```
$ git checkout -b branchA
Cambiado a nueva rama 'branchA'
```

A continuación se realiza la modificación en el archivo en el editor de texto.

11. Ejercicio 9

Realiza el commit con los cambios mencionados en la branchA y el mensaje "Now with strcat()".

Primero se añade el archivo al seguimiento, luego se realiza la confirmación:

```
$git add include/myinclude.h
$ git commit -m "Now with strcat()"
[branchA ed2ee6a] Now with strcat()
 1 file changed, 1 insertion(+)
```

12. Ejercicio 10

Cambia a la rama master (`$git checkout master`). En la rama master añade un comentario al inicio del fichero "helloworld.c" con la fecha y tu nombre como autor del fichero. Realiza el commit de este cambio en la rama master. Con el mensaje "Now with date and author"

```
$ git checkout master
Cambiado a rama 'master'
```

Modifico el fichero helloworld.c añadiendo un comentario al principio y confirmo el cambio:

```
$ git add helloworld.c
$ git commit -m "Now with date and author"
[master 6bfa915] Now with date and author
 1 file changed, 1 insertion(+)
```

13. Ejercicio 11

Desde la rama master, fusiona la rama master con la rama branchA. En este caso no hay ningún conflicto, ambas ramas han modificado distintos archivos. Es un merge sin conflicto.

Para fusionar las ramas se utiliza el comando `merge`:

```
$ git merge branchA
Merge made by the 'recursive' strategy.
 .gitignore           | 1 +
 include/helloworld.h | 5 ++++-
 2 files changed, 5 insertions(+), 1 deletion(-)
 create mode 100644 .gitignore
```

14. Ejercicio 12

Repite el proceso del punto anterior pero ahora creando un conflicto (merge con conflicto).

Estando en `master`, creo una rama nueva con los comandos

```
$ git branch branchB
eduarroyo@ganimedes:~/reposlycs$ git checkout branchB
Cambiado a rama 'branchB'
```

A continuación introduzco una línea en helloworld.c y confirmo los cambios:

```
$ git add helloworld.c
$ git commit -m "Marca desde BranchB"
[branchB c272698] Marca desde BranchB
    1 file changed, 3 insertions(+), 1 deletion(-)
```

Vuelvo a rama master e introduzco un cambio en el helloworld.c y confirmo:

```
$ git checkout master
git checkout master
-- Edito el archivo helloworld.c
$ git add helloworld.c
$ git commit -m "Marca desde master"
[master 8395c14] Marca desde master
 1 file changed, 3 insertions(+), 1 deletion(-)
```

Ahora voy a hacer el merge de la rama BranchB:

```
$ git merge BranchB
```

15. Ejercicio 13

Sube este proyecto a GitHub.

Primero he creado un repositorio nuevo en mi cuenta de github. Luego he ejecutado el siguiente comando:

```
$ git push --set-upstream https://github.com/eduarroyo/prueba1
.git master
Username for 'https://github.com': eduarroyo
Password for 'https://eduarroyo@github.com':
Enumerando objetos: 18, listo.
Contando objetos: 100% (18/18), listo.
Compresión delta usando hasta 4 hilos
Comprimiendo objetos: 100% (15/15), listo.
Escribiendo objetos: 100% (18/18), 1.84 KiB | 1.84 MiB/s,
listo.
Total 18 (delta 1), reusado 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/eduarroyo/prueba1.git
 * [new branch]      master -> master
Rama 'master' configurada para hacer seguimiento a la rama
remota 'master' de 'https://github.com/eduarroyo/prueba1.git'.
```

16. Ejercicio 14

> *Bájate un proyecto de GitHub de uno de tus compañeros (o de cualquier otra persona) a tu cuenta local.*

Desde el directorio `home`, se clona el repositorio donde están las prácticas de la asignatura en LaTeX:

```
$ cd ~
$ git clone https://github.com/eduarroyo/slcs-practicas.git
Clonando en 'slcs-practicas'...
remote: Enumerating objects: 73, done.
remote: Counting objects: 100% (73/73), done.
remote: Compressing objects: 100% (40/40), done.
remote: Total 73 (delta 24), reused 65 (delta 16),
pack-reused 0
Desempaquetando objetos: 100% (73/73), listo.
```

Se ha creado el direcotrio slcs-practicas con todo el contenido del repositorio remoto.

Parte III
Práctica 3

Siguiendo las instrucciones del enunciado, la práctica 3 se ha realizado en un documento de LibreOffice Writer y se puede descargar del repositorio de github.

Parte IV
Práctica 4

17. Ejercicio 1

Uso extendido de GNU/Linux en escritorio y en la industria. Últimos avances. Lectura recomendada:

- *Snaps: ¿El método más simple para instalar apps en Linux?*

Cuestiones:

1. *Porcentaje de uso de los principales SO de escritorio.*
2. *¿Cómo crees que podría mejorarse el uso de GNU/Linux en escritorios?*
3. *¿Qué es snap y snapcraft.io? ¿para qué sirve y qué diferencias aporta al sistema tradicional de instalación de paquetes de GNU/Linux?*
4. *Buscar cómo se instala snap en mi distribución en la documentación de snap.*
5. *Instalar y probar snap*
6. *¿Crees que mejora en el sentido de facilitar el uso de GNU/Linux?*

17.1. Pregunta 1

Estos gráficos muestran los datos de abril de 2020 sobre el porcentaje de uso de los principales SO de escritorio en todo el mundo [5] y en España [6].

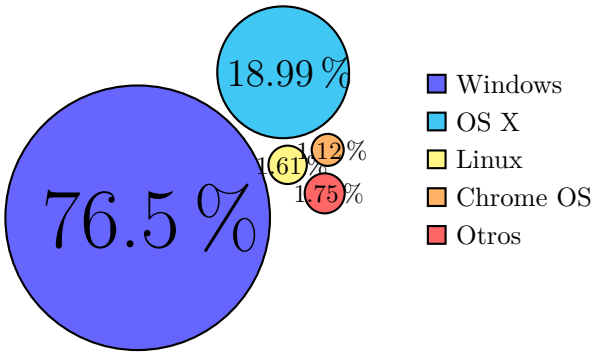

Figura 4: Uso de los principales SO de escritorio en abril de 2020 en el mundo.

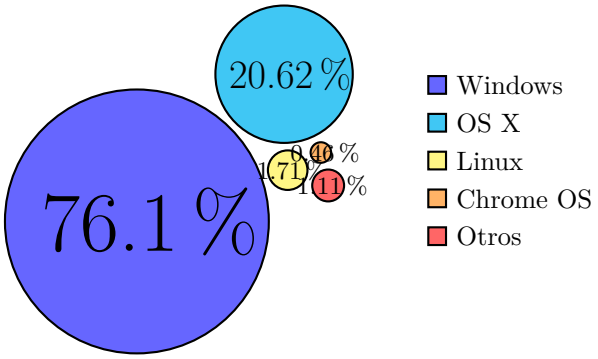

Figura 5: Uso de los principales SO de escritorio en abril de 2020 en España.

17.2. Pregunta 2

Creo que el porcentaje de uso de GNU/Linux aumentaría significativamente a largo plazo si las administraciones lo adoptasen ampliamente, sobre todo en el ámbito de la educación. Por otra parte, la mayoría de los ordenadores que se venden incluyen una licencia de Windows como si no hubiera opción. Si se acabase con esta práctica y en su lugar se permitiese al comprador elegir, los usuarios percibirían mejor el valor y el precio del producto que están comprando y, sin duda, muchos elegirían GNU/Linux aunque sólo fuera por el precio.

En mi opinión, la dificultad de uso no es un factor que frene la adopción de GNU/Linux para el usuario medio de hoy. Si algo detiene al usuario a la hora de decidir utilizar un sistema operativo diferente es hábito, el aferrarse a lo que conoce y rechazar el cambio.

17.3. Pregunta 3

«Snap» es un sistema de gestión de paquetes paquete de aplicación para escritorio, nube o IoT, creado y diseñado por Canonical. snapcraft.io es la herramienta que utilizan los desarrolladores para empaquetar sus programas en formato snap. La principal ventaja de snap frente a otros sistemas de gestión de paquetes es ser independiente de la distribución en la que se instala. Los propios snaps no dependen de ninguna tienda de aplicaciones y se pueden obtener de cualquier forma. El inconveniente de Snap es el mayor tamaño de sus paquetes, ya que cada uno viene con sus dependencias incluidas.

17.4. Pregunta 4

Según las instrucciones para la instalación de snap en Ubuntu[1], snap se instala por defecto en Ubuntu 20.04, que es la distribución que tengo actualmente instalada.

17.5. Pregunta 5

Como snap se instaló automáticamente con mi Ubuntu, paso a comprobar la instalación siguiendo las instrucciones que dan en snapcraft.io[1]. En la figura 6 se puede ver la ejecución del comando snap sin parámetros, la instalación del paquete `hello-world` y la ejecución posterior del programa instalado.

```
eduarroyo@ganimedes:~/Documentos/slcs/slcs-practicas$ snap
El comando snap permite instalar, configurar, actualizar y eliminar snaps.
Los snaps son paquetes que funcionan sobre muchas distribuciones Linux
diferentes, permitiendo la entrega y funcionamiento seguro de las
últimas aplicaciones y utilidades.

Uso: snap <comando> [<opciones>...]

Las órdenes pueden clasificarse de esta manera:

       Básicos: find, info, install, list, remove
        ...más: refresh, revert, switch, disable, enable
     Historial: changes, tasks, abort, watch
      Demonios: services, start, stop, restart, logs
       Ordenes: alias, aliases, unalias, prefer
  Configuración: get, set, unset, wait
        Cuenta: login, logout, whoami
      Permisos: connections, interface, connect, disconnect
   Instantáneas: saved, save, check-snapshot, restore, forget
          Otro: version, warnings, okay, ack, known, model, create-cohort
    Desarrollo: run, pack, try, download, prepare-image

Para obtener más información sobre alguna orden, ejecute «snap help <orden>».
Para un breviario de todas las órdenes, ejecute «snap help --all».
eduarroyo@ganimedes:~/Documentos/slcs/slcs-practicas$ sudo snap install hello-world
[sudo] contraseña para eduarroyo:
Se ha instalado hello-world 6.4 por Canonical✓
eduarroyo@ganimedes:~/Documentos/slcs/slcs-practicas$ hello-world
Hello World!
eduarroyo@ganimedes:~/Documentos/slcs/slcs-practicas$
```

Figura 6: Comprobación de funcionamiento de snap

[1] https://snapcraft.io/docs/installing-snap-on-ubuntu

17.6. Pregunta 6

Al menos en Ubuntu, la tienda de aplicaciones que conocía, junto con `apt`, hacían bastante fácil la instalación de aplicaciones en GNU/Linux. Sin duda, el carácter multiplataforma de snap contribuye a una mayor homogeneización del software entre las distintas distribuciones, pero sobre todo facilita el trabajo a los desarrolladores, que sólo tienen que empaquetar sus aplicaciones en un sistema.

18. Ejercicio 2

Free software and GNU/Linux publications:

- *LJ. Linux Journal.* https://www.linuxjournal.com/
- *LM. Linux Magazine.* http://www.linux-magazine.com/
- *Linux Weekly News.* https://lwn.net/
- http://linux.com
- *nixCraft.* https://www.cyberciti.biz/
- *slashdot y barrapunto*
- *OMG! Ubuntu!. We follow the Ubuntu and GNOME development.* https://www.omgubuntu.co.uk/
- *Sección "Linux" en:* http://hipertextual.com
- https://ayudalinux.com/
- https://lamiradadelreplicante.com/

Cuestiones:

1. *Ordena tus preferencias de 1 a 5.*
2. *¿Has encontrado contenido GNU/Linux o de free software en otras publicaciones?¿Cuáles?*

18.1. Pregunta 1

Aunque para valorar las distintas publicaciones haría falta un seguimiento más detenido de cada web, diría que las cinco que más me han gustado son estas:

1. OMG! Ubuntu! Tiene un aire desenfadado y artículos no tan técnicos, pero muy interesantes, con novedades y proyectos curiosos.
2. ayudalinux.com está orientado a quienes se inician con Linux, con muchos tutoriales y artículos para solucionar problemas comunes.
3. Linux Magazine tiene contenido técnico muy detallado y artículos clasificados por áreas como administración, desarrollo, hardware, servidor, programación..., el problema es que muchos de sus artículos son de pago.
4. slashdot es una web donde los usuarios envían noticias de temas diversos y las comentan. Tienen un apartado específico con temas de FOSS.
5. linux.com Es una web de recopilación de noticias clasificadas por temas como Open Source, Seguridad, Administración de sisteams, IoT, Hardware, Linux...

18.2. Pregunta 2

- Linux Today

- TecMint
- Softpedia: Linux
- Linux Insider
- nixCraft

19. Ejercicio 3

GNU/Linux en redes sociales:

- *@Linux 269mil*
- *@usemoslinux 24mil*
- *@MuyLinux 23mil*
- *@linuxhispano 7mil*
- *@andalinux 3mil*
- *@AulaSL 634*

Best Linux hashtags, o hashtags relacionadas con Linux que más se usan en RRSS: #linux #programming #python #coding #hacking #technology #programmer #technology #java #hacker #computerscience #code #coder #javascript #html #developer #cybersecurity #webdeveloper #css #computer #kalilinux #software #php #webdevelopment #webdesign #programmers #geek #softwaredeveloper #windows #bhfyp
Cuestiones:

1. *Anota las cuentas de Twitter que más te gusten de las anteriores o de otras que tú consultes en orden preferente de 1 a 5.*
2. *¿Has encontrado contenido GNU/Linux o de free software en otras RRSS? ¿Cuáles?*

Lectura recomendada en el portal Linux Journal sobre su 25 aniversario:

- *25 Years Later: Interview with Linus Torvalds*

19.1. Pregunta 1

1. @linuxhispano comparte los artículos de su web `http://linuxhispano.net` y también su tira cómica.
2. @andalinux es muy activo en twitter y comparte noticias interesantes y variadas.
3. @muylinux comparte las noticias de su web `http://muylinux.com`.
4. @ubuntizando
5. @gvanrossum

19.2. Pregunta 2

- Cuentas de Twitter
  - @gvanrossum Guido van Rossum es el creador de Python.
  - @LibbyMClark Editora de contenido digital en la Linux Foundation. Periodista tecnológica. Centrada en contenido sobre FOSS.
  - @JenWike Editora jefe en @opensourceway.

- @migueldeicaza Fundador de Gnome y de Xamarin y otros proyectos Mono.
- @ubuntizando Comparte noticias y trucos para ubuntu.

- Canales de Youtube
  - Aula de Software Libre de la EPSC
  - Riba Linux
  - Linux Scoop
  - The Linux Experiment
  - Level1Linux
  - The Linux Foundation
  - Nos gusta linux

Parte V
Práctica 5

Comienza leyendo los siguientes textos:

1. *Artículo "Mensajería instantánea libre y responsable", por Óscar Martín, de Ingeniería sin Fronteras Andalucía.*
2. *Acerca de riseup.net y Political Principles.*
3. *¿Qué hace especial al correo de riseup.net?*

Una vez leído, puedes realizar las siguientes actividades:

1. *Instala en tu móvil las aplicaciones de mensajería instantánea sugeridas en el artículo 1. Reflexiona sobre puntos fuertes y débiles de cada una, y en qué se puede basar la popularidad de cada una.*
2. *Responde a la pregunta ¿es Telegram totalmente libre?*
3. *Investiga sobre el protocolo OTR (Off the record messaging), ¿podrías utilizar este cifrado en alguna de las redes de mensajería instantánea que usas?*
4. *Investiga sobre GNU Privacy Guard (GnuPG o GPG). Estudia cómo podrías usarlo en tu correo electrónico (existen implementaciones a nivel de navegador y a nivel de cliente de correo).*

Otros textos de interés:

- *Defensa personal del correo electrónico, infografía de la FSF (Free Software Foundation)*
- *Defensa personal del correo electrónico, guía de la FSF (Free Software Foundation)*
- *Tor and HTTPS, de la EFF (Electronic Frontier Foundation)*
- *NSA Spying on Americans, de la EFF*

20. Pregunta 1

He instalado Xabber en mi teléfono y he creado una cuenta XMPP desde la misma app. También he instalado Pidgin en mi ordenador y he estado haciendo algunas pruebas. En mi opinión, el éxito o el fracaso de una red de mensajería instantánea no depende tanto del cliente que se utilice sino de la capacidad para crear una comunidad de usuarios. Al fin y al cabo, la gente irá a la red donde están sus contactos y tras la llegada de los smartphones, la primera red en hacerse con el mercado de la mensajería fue Whatsapp, por lo que todo el mundo está allí y muy poca gente se va a mover. Por otra parte, ante las ventajas evidentes que proporcionan otras redes de MI más respetuosas con los usuarios, de nuevo bajo mi punto de vista, el usuario promedio permanece indiferente. Le importa más quién está en la red que las ventajas que ofrezca.

21. Pregunta 2

Aunque tanto las aplicaciones cliente como la biblioteca TDLIB para construcción de clientes o el mismo el protocolo MTPROTO de Telegram están licenciados bajo licencias FOSS como GPL, X11(MIT) o Boost, Telegram no se puede considerar como un sistema de MI libre ya que el software del servidor no es libre. En mi opinión, las libertades de los usuarios de su software no se encuentran entre las prioridades de Telegram FZ-LLC, por lo que, de abrir el código del servidor, su sistema sería en todo caso Open Source, pero no libre.

22. Pregunta 3

OTR es un sistema de encriptación para MI que permite a los usuarios tener sesiones de chat seguras. OTR proporciona cuatro beneficios principales:

- Encriptación extremo a extremo: el mensaje se encripta en el lado del remitente y sólo se desencripta en el lado del destinatario. Nada ni nadie entre ellos dos puede leer el mensaje.
- "Forward secrecy": este sistema de encriptación permite proteger los mensajes pasados contra futuros compromisos de claves de cifrado o contraseñas.
- Autenticación mutua: garantiza que la persona con la que chateas es realmente ella y viceversa.
- Autenticación denegable: hace imposible para un tercero probar la autoría de un determinado mensaje.

Ninguna de las redes de mensajería que utilizo implementa este protocolo pero, según parece, algunas de ellas están soportadas por plugins de Pidgin y este cuenta con un plugin[2] que aplica este cifrado sobre cualquier protocolo que soporte el cliente.

23. Pregunta 4

GNU Privacy Guard (GnuPG) es una implementación libre y completa del estándar OpenPGP y permite encriptar y firmar tus comunicaciones. Para utilizar este sistema con mi cuenta de correo personal primaria (gmail), necesitaría una herramienta para el navegador y para Android. Para el navegador he encontrado varios plugins como FlowCrypt[3] o Mailevelope[4] que son compatibles con Gmail. En el caso de Android, tendría que dejar de usar la app oficial de Gmail para utilizar un cliente de correo que lo soportase este cifrado, como FairEmail[5], OpenKeychain[6] o k-9 Mail[7].

[2] https://otr.cypherpunks.ca/
[3] https://flowcrypt.com/
[4] https://www.mailvelope.com/en/
[5] https://email.faircode.eu/
[6] https://www.openkeychain.org/
[7] https://k9mail.app/

Referencias

[1] E. J. Intland Software, "Pros and cons: Is git better than mercurial?." https://content.intland.com/blog/sdlc/why-is-git-better-than-mercurial, Enero 2015. Accedido: 23/3/2020.

[2] "Distributed version control." https://en.wikipedia.org/wiki/Distributed_version_control, marzo 2020. Accedido: 23/3/2020.

[3] "Primeras impresiones: Dvcs mercurial." www.juancarlosfernandez.net/2010/10/evaluando-el-dvcs-de-mercurial.html, octubre 2010. Accedido: 23/3/2020.

[4] "Mercurial." https://es.wikipedia.org/wiki/Mercurial, septiembre 2019. Accedido: 23/3/2020.

[5] "Desktop operating system market share worldwide." https://gs.statcounter.com/os-market-share/desktop/worldwide/. Accedido: 8/5/2020.

[6] "Desktop operating system market share worldwide." https://gs.statcounter.com/os-market-share/desktop/spain/#monthly-201904-202004. Accedido: 8/5/2020.