Capítulo 1

Debian GNU/Linux

En este capítulo —nada técnico— veremos de dónde viene el sistema operativo Debian, su filosofía y características. También veremos algo de la historia de sus precursores: el proyecto GNU, el núcleo Linux y el sistema UNIX. Aunque es una historia conocida, nos parece interesante incluir aquí un pequeño resumen de lo que es una parte crucial de la historia de la informática y que ayuda a entender cómo hemos llegado hasta aquí.

El relato nos lleva directamente a los albores de la informática y de la incipiente industria del software. Empecemos.

1.1. UNIX

Corría el año 1969, cuando un pequeño equipo de investigadores de Bell Labs (AT&T), liderado por Ken Thompson y Dennis Ritchie, empezó a trabajar en un nuevo sistema operativo inspirado en el proyecto Multics, en el que Bell Labs había participado, pero abandonó. Como un juego de palabras, llamaron UNIX al nuevo sistema ya que pretendía ser más sencillo que Multics.

El primer prototipo de UNIX, como era lo habitual entonces, se desarrolló en ensamblador. Para facilitar la experimentación, que era una parte fundamental de la labor de Bell Labs, decidieron reescribir UNIX en un nuevo lenguaje de programación que el mismo equipo estaba creando: el lenguaje C [2]. Desde entonces y hasta la actualidad, más de 50 años después, los SO se siguen escribiendo principalmente en C.

La decisión de usar C le proporcionó a UNIX un nivel de portabilidad sin precedentes, y eso a pesar de que no era el objetivo principal de ese cambio. Eso facilitó que fuera adaptado con rapidez a diferentes arquitecturas por parte de Universidades y otros centros de investigación, lo que resultó clave en su gran éxito y popularidad.

También fue determinante un conjunto de pautas bien definidas, conocidas como *filosofía Unix*. Algunos de sus principios más importantes son:

- Cada programa hace solo una cosa y la hace bien.
- Los programas trabajan con flujos. La salida de un programa es la entrada de otro.
- El texto plano es un formato de intercambio universal.
- Las tareas repetitivas son fáciles de automatizar (*scripting*).
- Minimiza la interacción con el usuario.
- Todo se trata como un fichero, con las mismas primitivas.
- Falla rápido e informa claramente del error.

Los distintos Unix[1] que surgieron dominaron la informática profesional desde entonces hasta comienzos del siglo XXI. De entre los muchos Unix que surgieron como AIX, HP-UX, SunOS, etc., uno de los más influyentes fue BSD, creado por la Universidad de California en Berkeley en 1977. BSD añadió funcionalidades muy importantes como el soporte para los protocolos TCP/IP o la abstracción de socket, y herramientas muy populares como el editor `vi` o la C Shell (`csh`). Además, su licencia mucho más permisiva que la del UNIX de AT&T, permitió que tanto su diseño arquitectural como su código fuera reutilizado incluso en productos comerciales privativos. De BSD surgieron variantes como FreeBSD, OpenBSD y NetBSD que siguen existiendo a día de hoy, y su influencia llega a otros SO de uso doméstico que surgieron después como Apple macOS o Microsoft Windows.

1.2. El fin del software libre

Ya antes de UNIX, empresas como IBM vendían y mantenían los grandes y caros computadores que se utilizaban en facultades e instituciones de investigación. Como parte de sus servicios, proporcionaban a los clientes el código fuente de los programas que se ejecutaban en sus computadores[2] como requisito para su funcionamiento. Los programadores e investigadores podían estudiar, modificar y mejorar esos programas. Los propios fabricantes fomentaban esa práctica, redistribuyendo esas modificaciones al resto de sus clientes, bajo la convicción de que el valor de su negocio estaba únicamente en el hardware.

[1] «UNIX» es la marca comercial propiedad de AT&T —hoy un estándar de Open Group— mientras que «Unix» es un término genérico para referirse a los sistema de este tipo.

[2] Como hacía AT&T con UNIX.

En 1976, un estudiante de 20 años del Harvard College, escribió una breve carta abierta titulada «Open Letter to Hobbyists»[3] [3] en la que argumentaba que los programadores que estaban compartiendo sus aplicaciones y modificaciones perjudicaban a las empresas que comercializaban software, siendo según él, una actividad equivalente al robo. Abogaba por cerrar el código, distribuir únicamente binarios ejecutables, licenciar el uso y prohibir legalmente la modificación y redistribución.

La carta no tuvo mucha repercusión en un primer momento, pero el año siguiente (1977) el periodista Andrew Pollack publicó un artículo en «The New York Times» apoyando abiertamente estas ideas. Su difusión en un periódico tan influyente marcó un punto de inflexión que propició que rápidamente empresas como Apple, MITS, Tandy o Commodore comenzaran a cerrar su software. Posteriormente esos cambios llegaron también a la legislación de protección industrial e intelectual. El nombre del estudiante autor de la carta era William Henry Gates III, mas conocido como Bill Gates.

1.3. GNU

Una de las instituciones que empezaron a notar el cambio de mentalidad de la industria fue el Laboratorio de Inteligencia Artificial del MIT. Uno de los investigadores que trabajaba allí, Richard Stallman, vio su forma de trabajar amenazada por los recientes cambios en la legislación y en el modo de operar de las empresas de software. Stallman decidió iniciar el proyecto GNU [4] con el objetivo de crear un SO completo que permitiera a los usuarios recuperar la libertad perdida: la de usar, estudiar, modificar y distribuir el software sin restricciones.

El nombre GNU es un acrónimo recursivo. Significa GNU is Not Unix (GNU no es Unix) a la vez que juega con la pronunciación de la palabra *new*, que es prácticamente igual. El nombre plasma la intención de crear un SO similar a Unix, pero completamente libre, con el objetivo de que nadie tuviera ni quisiera usar ningún Unix propietario, terminando por reemplazarlo completamente.

Así nació la idea del software libre, aunque en realidad no era más que un intento por recuperar lo que algunos entendían que era la forma original y natural de entender el software. La decisión de crear un sistema similar a Unix no fue tanto por su superioridad técnica, sino porque era el SO

[3] *hobbyist* se refiere al 'ecosistema hobbyist', una comunidad de aficionados cuyo máximo exponente fue el Homebrew Computer Club (California, 1975).

más popular en ese momento, y Stallman entendió que eso ayudaría a su adopción.

Stallman y su equipo crearon la FSF para coordinar y promover el desarrollo del sistema GNU, y desarrollaron la licencia GPL, que constituye la base legal para proteger la libertad del software.

Con los años, el proyecto GNU fue creciendo y creando todo tipo de herramientas importantes como editores, compiladores, librerías, etc., A finales de los 80, el sistema tenía todos los componentes necesarios salvo uno: el núcleo. Trabajaban activamente en un núcleo llamado Hurd, pero su diseño era muy ambicioso y complejo, y su desarrollo se retrasaba constantemente.

1.4. Linux

Linus Torvalds, otro estudiante, este de la Universidad de Helsinki, estaba estudiando Minix, un pequeño sistema operativo con fines educativos basado en la filosofía de Unix creado por Andrew Tanenbaum. Aunque Torvalds admiraba Minix como herramienta didáctica, encontraba limitaciones en su diseño y era crítico con ciertas decisiones que afectaban a su rendimiento. Frustrado por la limitaciones que Tanenbaum imponía a la modificación y evolución de Minix, en 1991 anunció a la comunidad de Minix [5] que estaba trabajando en Linux, su propio núcleo.

Pronto Torvalds decidió liberar Linux bajo la licencia GPL, si bien sus motivaciones siempre fueron más técnicas y pragmáticas que filosóficas o éticas. La FSF y el proyecto GNU adoptaron el núcleo Linux como una solución temporal hasta tener lista una versión estable de Hurd, que nunca llegó. FSF desarrolló la librería `glibc`, un gran hito poco valorado, que permite que el sistema GNU pueda ejecutarse sobre el núcleo Linux, y también adaptaron muchas otras herramientas.

1.5. GNU/Linux

La combinación del sistema GNU y el núcleo Linux dio lugar a lo que conocemos como GNU/Linux, un SO completo y libre. Sin embargo, instalar un sistema así no era nada sencillo y solo personas con conocimientos técnicos avanzados podían conseguirlo. Implicaba compilar el núcleo y la mayoría de las herramientas.

Para simplificar esta tarea aparecieron las primera «distribuciones» de GNU/Linux. Una distribución es una colección organizada de paquetes, precompilados y mantenidos por un equipo de desarrolladores con el objetivo

de conseguir una coherencia entre ellos. Una distribución suele incluir el núcleo Linux —normalmente compilada con un objetivo específico—, el sistema GNU, un programa de instalación y una selección de programas y herramientas. Conforme todos esos componentes van evolucionando y se ofrecen versiones nuevas, la distribución define también nuevas versiones (*releases*). De ese modo los usuarios pueden instalar y mantener un sistema funcional con ciertas garantías en cuanto a estabilidad.

A lo largo de la década de los 90, GNU/Linux fue ganando popularidad y para principios de los 2000 era ya una alternativa viable a nivel industrial. Rápidamente se convirtió en la plataforma dominante para servidores en Internet, al mismo tiempo que copaba el Top 500 de los supercomputadores a nivel mundial, un nicho tradicionalmente dominado por los Unix comerciales. En 2010, con la compra de Sun Microsystems por parte de Oracle, Solaris que era de facto el último Unix propietario relevante, prácticamente desaparece.

1.6. Debian GNU/Linux

En 1993, otro estudiante de la Universidad de Texas, Ian Murdock, anunció el proyecto Debian [6]. Su objetivo era crear una nueva distribución de GNU/Linux que hiciera énfasis en la libertad.

Debian fue una de las primeras grandes distribuciones de GNU/Linux[4], y se estableció rápidamente como una de las más profesionales y respetadas por la comunidad hasta la actualidad. En su compromiso por la libertad, Debian diferencia claramente entre software libre y no libre, de modo que los usuarios puedan tomar decisiones informadas sobre lo que instalan.

Uno de los aspectos distintivos de Debian es su Contrato Social [5], un documento que establece los principios éticos y los compromisos que el proyecto asume con sus usuarios. Los podemos resumir en sus epígrafes:

1. Debian permanecerá 100 % libre.
2. Contribuiremos con la comunidad de software libre.
3. No ocultaremos los problemas.
4. Nuestra prioridad son nuestros usuarios el software libre.
5. Asumimos que algunos usuarios necesitan software no libre.

También define las Directrices de Software Libre de Debian (DFSG) que establecen los criterios que deben cumplir los programas que se incluyen en la distribución:

[4] Solo por detrás de Slackware

[5] `https://www.debian.org/social_contract.es.html`

1. Libre redistribución.
2. Disponibilidad del código fuente.
3. Permiso par realizar trabajos derivados.
4. Integridad del código fuente del autor.
5. No discriminación contra personas o grupos.
6. No discriminación en función de la finalidad.
7. Distribución de la licencia.
8. La licencia no debe ser específica para Debian.
9. La licencia no debe contaminar otro software.

Aunque su esfuerzo por mantener y promover la libertad del software es ciertamente destacable, no es el único aspecto que hace de este un proyecto de referencia entre las distribuciones de GNU/Linux. En el plano organizativo, Debian se basa principalmente en su comunidad de *desarrolladores oficiales*[6], en su mayoría voluntarios. Convertirse en desarrollador oficial está abierto a cualquier persona, pero requiere superar un proceso de selección bastante riguroso.

Existe un conjunto de normas públicas que regulan todos los procesos importantes del proyecto, como pueden ser la adopción o eliminación de paquetes oficiales, cambios en la *policy*[7] o la elección del líder. Todas estas decisiones se toman mediante votaciones públicas con debates en los que participa la comunidad. La ausencia de una empresa tras el proyecto y la gran transparencia del proceso son aspectos muy valorados y muy poco habituales en la industria del software, incluso en la del software libre.

Debian hace mucho énfasis en la estabilidad y el soporte a largo plazo. Para ir incorporando nuevas versiones de los programas, mantiene un ciclo de desarrollo basado en tres ramas diferentes: *stable*, *testing* y *unstable*[8]. La rama *stable* corresponde con las versiones oficiales (*releases*) y está pensada para usar en servidores y entorno de producción. Las versiones estables se publican aproximadamente cada 2 años y tienen un soporte de unos 3 años[9]. La rama *testing* es el entorno de pruebas para preparar la siguiente versión estable y, aunque puede seguir incorporando nuevas versiones de los paquetes, debe estar justificado. La rama *unstable* es la versión de desarrollo, y en ella pueden aparecer versiones nuevas prácticamente a diario.

Aunque es perfectamente posible utilizar *unstable* es un equipo de escritorio, existe corre un riesgo no despreciable de quedar con paquetes o dependencias rotas en momento puntuales. Esto permite a cada usuario decidir el

[6] Abreviado como DD (Debian Developer)

[7] Se refiere a la *Debian Policy Manual*: `https://www.debian.org/doc/debian-policy/`

[8] Y una mucho más inestable llamada *experimental*

[9] Puedes ver la lista oficial de versiones estables en `https://www.debian.org/releases/`

equilibrio de estabilidad/novedad que quiere para su sistema de una manera muy sencilla, y además el posible tener un sistema mixto con paquetes de diferentes ramas.

Debian, además, es una de las distribuciones que mantiene soporte para más arquitecturas. En el momento de escribir esto la última versión estable publicada (Debian 13: Trixie) cuenta con soporte para 9 arquitecturas. Esto da una idea clara de su compromiso para ofrecer al usuario un sistema operativo realmente universal.

A lo largo de sus muchos años de vida, Debian ha influido e influye en la industria del software. No solo por las muchas distribuciones derivadas[10], algunas de ellas muy populares como Ubuntu, Kali o Raspbian, que siguen bastante integradas en sus procesos de desarrollo, tomando los paquetes actuales de Debian como base. También son referencia sus prácticas de desarrollo y empaquetado (especialmente su gestor `apt`), sus procesos de mantenimiento y despliegue, y su esquema de versiones y ciclo de desarrollo.

[10] Unas 120 según Distrowatch.com