Índice general

Índice de figuras

Índice de cuadros

Capítulo 4

GNU's Not UNIX! (GNU)

4.1. UNIX

Era el año 1965 cuando los Laboratorios Bell[1] de AT&T[2], el Instituto Tecnológico de Massachusetts[3] y General Electric[4] comenzarón a trabajar en el Sistema Operativo Multics[5], acrónimo de "Multiplexed Information and Computing System"[6], diseñado para funcionar en una máquina Mainframe modelo GE-645[7].

El proyecto era base de las ideas de construir un Sistema Operativo que permitiese el acceso a distintos usuarios de modo simúltaneo, con el objetivo principal de compartir los recusos de computación[8] Laboratorio Bell de AT&T deciden abandonar el proyecto tras distintas versiones fallidas de Multics.

Ken Thomson[9], programador de los propios laboratorios, continuó trabajando sobre la misma arquitectura original de Multics, la computadora GE-6354[10], desarrollando un juego espacial llamado "Space Travel"[11]. El coste de cada partida se estimo en 75 dólares americanos de la época. El hecho del redimiento tan desfavorable origino que de nuevo, Ken Thomson, con la ayuda de Dennis Ritchie[12], reescribiese el juego para una computadora DEC PDP-7[13].

El trabajo invetido en esta nueva versión del juego, dio pie a la creación de un nuevo Sistema Operativo. Los dos programadores junto Rudd Canaday[14] crearon gran cantidad de Software de soporte. Entre otra utilidades crearon un nuevo sistema de fichero distribuido y un potente intérprete de órdenes.

El nuevo proyecto se denomino UNICS, acrónimo de "Uniplexed Information and Computing System"[15], que era un juego de palabras sobre el citado MULTICS. Finalmente paso a denominarse UNIX por influencia del hacker de MULTICS Brian Kernighan[16].

El proyecto UNIX finalmente fue financiado por los Laboratorio Bell, debido a que se trabajó para máquinas superiores de la propia familia PDP[17]. En concreto sobre el conjunto [PDP-11, PDP-20].

Corría el año 1970 y UNIX era una realidad. Destaca de las primeras versiones el editor de textos `rundoff`[18]. Todas las versiones de UNIX escritas antes de 1972 estaban codificadas en Lenguaje Ensamblador de las propias computadoras. Este año se produjo un hito histórico para el Sistema UNIX, y es la reescritura prácticamente completa del propio sistema a Lenguaje C[19].

La protabilidad de UNIX era una realidad, y el propio AT&T dotó a universidades y empresas de es nuevo sistema. Destaca de las versiones de la matriz UNIX, la distribución BSD (Berkley Software Distribution). El propio AT&T creo un departamentp para comercializar UNIX. Su desarrollo, en distintas versiones, se descontinuó con la versión 7, en el año 1979. A

principios de la década de los 80, AT&T inició el proyecto Plan 9 del que ya era Software base el Sistema X-Window del MIT.

Le siguió un intento comercial sobre UNIX 7 denominada UNIX System III. Por aquella época la dispersión de fabricantes era enorme.

En 1993, la empresa Novell adquirió los laboratorios de UNIX de AT&T. En ese mismo año la versión BSD evolucionada en el tiempo, era constantemente demandada debido a problemas de patentes.

En 1995 Novell creo su propia versión de UNIX llamada UNIX-Ware.

4.1.1. Arquitectura

El Sistema Operativo UNIX fue concebido con dos ideas clave en su diseño:

i. Para UNIX todo es un fichero es decir, tanto lo creado y modificado por el usuario/desarrollador/administrador como cualquier dispositivo de Entrada/Salida.

   Todo es un flujo de datos o bien continuo o bien por bloques.

ii. Cada programa del Sistema tiene un único cometido, su función está perfectamente definida, por ello la tarea debe realizar será ejecutada de forma óptima.

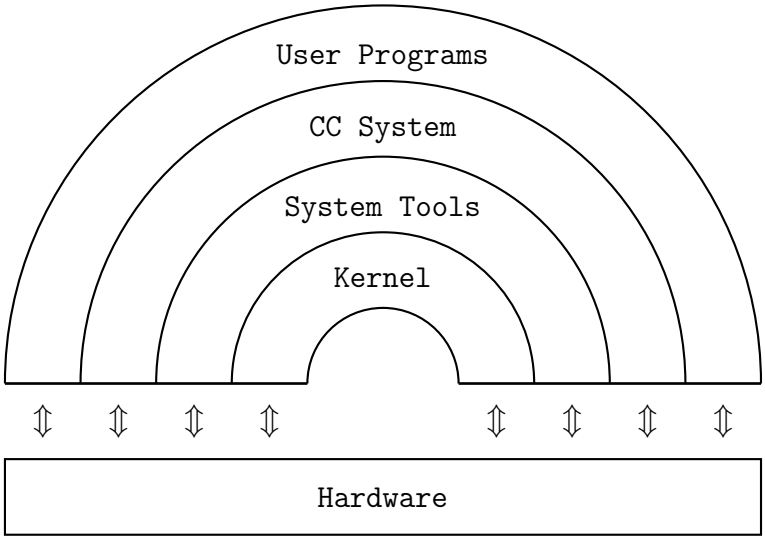

Figura 4.1: Arquitectura de UNIX.

UNIX se puede resumir a nivel arquitectónico como: "un conjunto de partes que hacen un todo". La pieza fundamental del puzzle la constitulle el Kernel[1]. El Kernel es el programa más cercano a la máquina, al propio Hardware, y es el encargado de organizar y mantener la coherencia entre las distintas piezas físicas de la computadora y los programas naturales del porpio Sistema UNIX. Parte de su cometido es normalizar y estandarizar la gestión de los dispositivos como flujos (anteriormente citados) para que un programa sea capaz de utilizar ese recurso indepenciente del frabricante y sus características añadidas.

[1]Traducido directamente como Núcleo.

El Kernel tiene una misión crítica y es casi impermeable en cuanto a su acceso. Pone a disposición del programador un conjunto de primitivas universales que reciben el nombre de "System Calls[2]" y siempre gestionadas por el propio núcleo. Son tan necesarias para el correcto funcionamiento de la computadora a nivel Hardware y Software.

Las "System Calls" son el acceso directo para los Programas Base[3] de UNIX y los distintos compiladores (aunque su lenguaje principal es C[4]). A través de estas herramientas se generan conjuntos de Bibliotecas que operan que operan sobre las primitivas del Kernel y que a su vez dan soporte a los programas de usuario.

4.1.2. Entorno de Usuario

4.1.2.1. Sistemas de Archivos UNIX FILE SYSTEM (UFS)

El Sistema de Ficheros UNIX (UFS) se basa en seis principios:

I. Estructura Jerárquica: Como hemos visto previamente en el Capítulo (), UFS sigue una disciplina de árbol con raíz:

   i. Profundidad:

   ii. Anchura:

   El índice (Root) es representado mediante el símbolo "/" y través del mismo es posible acceder a cualquier carpeta o archivo siempre y cuando se tengan los permisos necesarios.

II. Consistencia con el tratamiento de los datos: UNIX trata los ficheros como un flujo de datos que a su vez es gestionado por el Kernel. Con ello, se asegura que su uso no depende exclusivamente de un programa o un fabricante por contra, el control es siempre del Sistema y en última instancia del administrador del mismo.

III. Fácil acceso a las operaciones de manipulación de ficheros: Existen un conjunto de programas de bajo nivel (Baselayout) en la arquitectura modular del Sistema. Estos son diseñados con una única finalidad: creación, manipulación y destrucción de datos que a su vez son usadas como herramientas estandar por programas de alto nivel.

IV. Crecimiento y gestión dinámica de ficheros: Se fundamenta en la estructura de bloque de un fichero (ver Figura x). Dicha estructura recibe el nombre de *i-nodo* que presenta la característica de ser altamente direccionable (por ello su crecimiento es dinámico y deslocalizado).

   Un *i-nodo* se trata un meta-bloque de información para que el Gestor de Sistema de Archivos sea capaz de gestionar y localizar la información. Cada fichero en UNIX tiene asociado su correspondiente *i-nodo* que se compone de los siguientes bloques:

   i. Identificador del propietario de fichero (`UID`).

   ii. Tipo de fichero.

[2]Llamadas al Sistema.
[3]Denominada Baselayout.
[4]`cc` C Compiler.

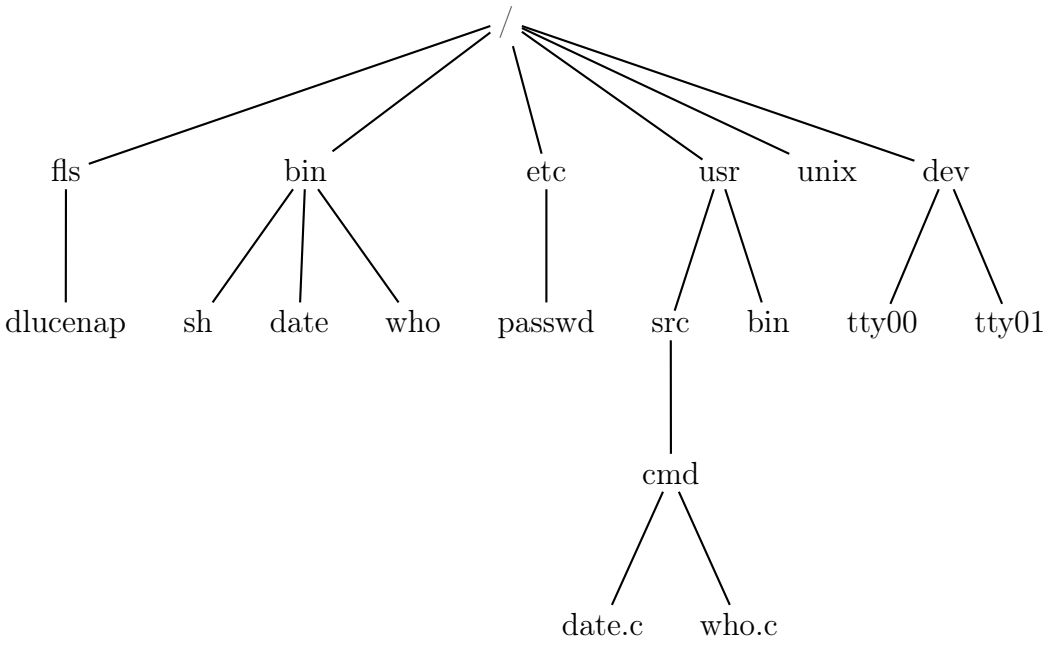

Figura 4.2: Ejemplo de Sistema de Fichero UNIX.

iii. Tipo de acceso al fichero.

iv. Tiempos de acceso al fichero.

v. Número de enlaces del fichero.

vi. Entradas para los bloque de dirección de los datos del fichero.

vii. Tamaño del fichero.

V. Protección de los datos de usuario: UNIX basa sus permisos en tres categorias:

i. Permisos para usuario propietario.

ii. Permisos para grupo propietario.

iii. Permisos para otros usuarios.

El Administrador (Root) posee la característica de poder modificar cualquier fichero del sistema al igual que acceder a su información siempre y cuando sea legible.

Cada uno de estas categorias tiene potencialmente tres accesos:

i. `r`: Modo exclusivo de lectura.

ii. `w`: Modo de lectura/escritura.

iii. `w`: Capacidad de ejecutar el fichero.

A modo general los permisos se estructurán:

$$\underbrace{rwx}_{user}\underbrace{rwx}_{group}\underbrace{rwx}_{others} \tag{4.1}$$

La pertenecia a un determinado grupo de un usuario es tarea exlusiva del Administrador. Dichos grupo y propiedades se encuentran en el fichero de texto: `/etc/group`. Cada uno de estos grupo posee un GID *Group Identifier* que simplifica el trabajo del sistema.

Por otra parte, los usuarios (que sólo pueden ser dados de alta o baja por el Administrador) se encuentran en el fichero de texto `/etc/shadow` donde además de su datos de sistema se encuentra su contraseña cifrada por `SH1` y su correspondiente UID *User Identifier*.

VI. Tratamiento de Hardware Entrada/Salida como flujos de datos:

UNIX diferenica entre dos clases de dispositivos:

i. Dispositivos de caracteres: Se tratan de dispositivos que admiten como argumento flujos de Bytes.

ii. Dispositivos por bloques: Se trata de dispositivos que tratan su entrada en bloques de Bytes.

iii. Dispositivos de red: Dispositivos de retroalimentación de datos. Funciona como entrada y salida.

4.1.3. Entorno de Ejecución

Los programas en UNIX son ficheros ejecutables. Un programa en ejecución es una instacia del mismo sobre un bloque de metadatos gestionado por el kernel. Por ello, los programas en ejecución reciben el nombre de proceso.

Dado que UNIX nacio con la idea de ser multipropósito y multiusuario, siempre ha tenido la necesidad de tener una disciplina sobre el conjunto de programas en ejecución. Era y sigue siendo necesario organizarlos y darles los recursos necesarios para realizar su cometido: tiempo de proceso, asignación de espacios en la memoria, acceso a dispositivos externos, etc.

UNIX clásico tiene un modo de arranque por interruptores o niveles de ejecución "run levels" Es necesario por ello un proceso madre (con PID 0) llamado textttinit que llame al resto de programas y servicios necesarios para el ambiente de ejecución del Sistema Operativo. Por esto, el propio kernel dota de la primitiva `fork();` para generar nuevos procesos.

```
 - init (0)
 | --> RUN LEVEL 0 (JNB): Apagado de la máquina.
 | -> RUN LEVEL 1 (monouser): Arranque del sistema sin interfaces de red ni
servicios.
 | --> RUN LEVEL 6 (reboot): Reinicio de la máquina.
```

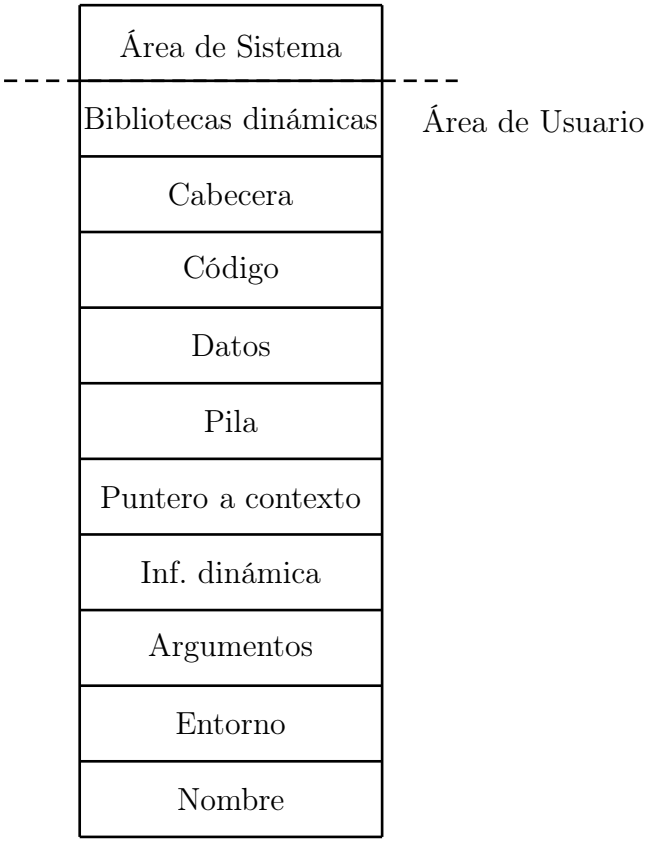

Figura 4.3: Modelo de proceso en UNIX.

4.1.3.1. Shell (Intérprete de órdenes)

Shell o Intérprete de órdenes es un programa encargado de dotar al usuario de un ambiente de trabajo sencillo y potente.

Además de proporcionar un entorno de ejecución y edición, contiene valores añadidos con el objetivo de ceder un mayor gobierno del sistema al propio usuario.

Sus aspectos más destacables son:

i. Control sobre conjuntos de ficheros: La Shell de UNIX utiliza los patrones clásicos de las Expresiones Regulares con el fin de que el usuario sea capaz de gestionar grandes volúmenes de información. Dichas expresiones se resumen en ().

ii. Capacidad de redireccionar la Entrada/Salida:

iii. Dotar al programador de herramientas Software: Igualmente y como hemos visto en los dos anteriores puntos, la Shell proporciona una serie de operadores y directivas para el entorno.

   En el caso de ejecución de programas, el hecho más resaltable es que mediante el operador `&` sea posible lanzar varios programas en ejecucion a través de un mismo comando:

   ```
   program1 & program2 & ... & programn-esimo
   ```

   Por último recalcar que el ambiente Shell es gobernado por una serie de variables generales y propias de cada usuario (`/home/user/.profile`). Es resaltable por su función la variable

Expresión	Significado
'.'	Representa cualquier carácter excepto '\n'.
'*'	∅ o más copias de la expresión que le precede.
'[ ]'	Operador de conjunto de caracteres. También se usa para expresar rangos.
'^'	Operador para indicar el comienzo de la expresión.
'$'	Indica el final de una expresión regular.
'{ }'	Indica el número de ocurrencias para el carácter que le precede.
'\'	Operador de "escapé" para caracteres restringidos.
'+'	Se utiliza con rangos, indica 1 o más copias de la expresión que le precede.
'?'	Indica ∅ o una ocrrencia.
'|'	Operador de disyunción.
"..."	Maraca el texto entrecomillado como literal.
'/'	Maraca como literal los caracteres que le preceden.
'( )'	Operador para subconjuntos de expresiones regulares.

Cuadro 4.1: Operadores comunes para Expresiones Regulares en UNIX.

`PATH` que reconoce y redirecciona distintas rutas de programas ejecutables para estadarizar su llamada.

4.1.4. Servicios

Los Servicios en UNIX son controlados de forma automática por el propio Kernel. Los principales obejetivos de esta gestión son:

I. Control de Procesos:

II. Planificación de recursos Hardware:

III. Gestión de los recusos de memoria:

   i. Memoria Principal (RAM):

   ii. Memoria Secundaria (SWAP):

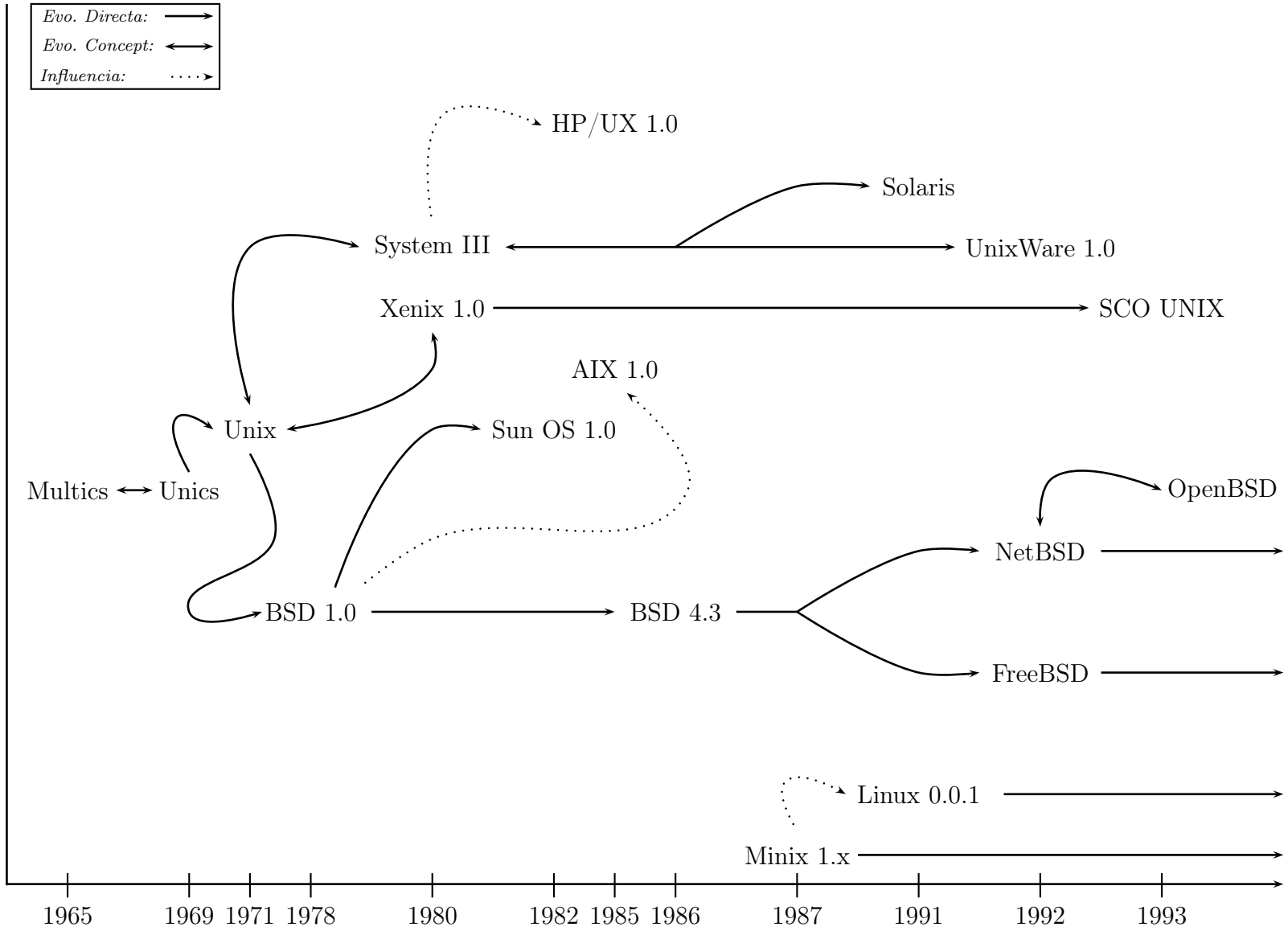

Figura 4.4: Evolución de UNIX y sus distintas versiones en el tiempo.

4.2. Proyecto GNU

Lo cierto es que la historia que contamos nace en 1983 con la creación de GNU[20] [Sta85] por parte de Richard Stallman[21]. Dos años después se crea la asociación "Free Software Fundation"[22] con el objetivo de promover el Software en torno a ciertas libertades, las cuales se consolidan con GPL[23], la licencia general y pública de GNU.

Gran cantidad de Software se desarrolló entre la última etapa de la década de los ochenta y principios de los noventa del siglo XX. Era tal, dicha cantidad, que se podría construir un clon de UNIX a falta de un núcleo en proyecto llamado Hurd[24]. El problema surgía de una mala planificación y la insistencia por crear un MiniKernel, que conllevaba a una excesiva burocracia en la etapa de solución para cualquier posible fallo. Y es que, por entonces, un estudiante finlandés, Linus Torvals[25], primero trabajando con grandes máquina y poco después con el 8386 dónde a partir de Minix[26], adaptándolo a sus necesidades escribe y extiende funcionalidades que finalmente constituirían un nuevo núcleo muy discreto.

4.2.1. Licencia GPL

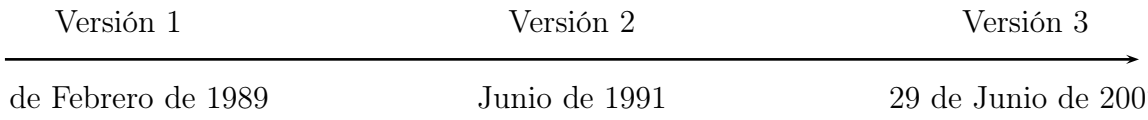

Figura 4.5: Evolución de la Licencia GPL.

La General Public License (Licencia Pública de GNU) se trata de un marco legal para desarrolladores y Software creada por Free Software Foundation (Fundación de Software Libre).

Su primera versión se hizo pública el 25 de Febrero de 1989. Con este nuevo "marco legal" se pretendía consolidar el proyecto GNU que a su vez tenía su esencia espiritual en la cultura Hacker de los años sesenta del siglo XX. Los desarrolles que decidieran distribuir su Software bajo Copyright de GNU permitian que se derivasen copias del mismo sin tener por ello una remuneración. Al igual, se obligaba a que aquellos que mejorasen o ampliasen las características de un programa con la licencia GPL publicasen dichas mejoras para que la comunidad se siguiera enriqueciendo y creciendo.

La licencia GPL es compatible con otro tipo de licencias "Libres" partiendo de la base de que estas últimas se basan en los postulados de la propia GPL.

De igual manera se han creado licencias basadas en GPL que añaden restriciones para el uso y compartición del Software. Es la respuesta de la industria ante la proliferación de la cultura GNU.

La licencia GPL ha tenido tres versiones oficiales hasta la fecha:

I. GPL versión 1: Como hemos citado anteriomente fue publicada el 25 de Febrero de 1989 bajo las siguientes ideas:

i. Dado el problema que plantea el Software comercial con su política de distribución de Software en formato ejecutable (binario), la licencia GPL determina que los programas que se distribuyan con su Copyrights deben ir acompañados del código fuente.

ii. El segundo problema que intenta solucionar es el de la compatibilidad con Software comercial, no tanto para oponerse a el, sino para proteger los programas GPL. Por ello, la versiones modificadas de un Software GPL deben acerse públicas.

II. GPL versión 2: Fue publicada en Junio de 1991 con los objetivos:

i. Hacer más restrictiva la publicación del Software, añadiendo la imperiosa necesidad de acompañar a un fichero ejecutable de su código fuente.

ii. De igual manera y basado en los problemas que presentaban las Bibliotecas GNU con Software comercial, se creo una licencia específica para las misma; *Library General Public License* (Licencia Pública General de Bibliotecas) bajo la misma idea de que se deben abrir sus códigos fuente.

III. GPL versión 3: Tras un largo periodo de trabajo (el proyecto comienza en 2005) finalmente el 29 de Junio de 2007 se hizo pública la versión 3 de la licencia GPL.

Entre sus novedades destacan:

i. Nuevas clausulas para formatos industriales restrictivos, como DRM (Gestión Digital de Derechos).

ii. Resolución de ambigüedades.

iii. Adaptar la GPL a los marcos jurídicos de los distintos países.

iv. Proteger a desarrolles de Software Libre frente a las patentes.

4.3. Linux

Parece ser que el punto de partida se fija el 25 de agosto de 1991, 20:57:08 GMT, con el siguiente correo electrónico[27]:

```
From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Newsgroups: comp.os.minix
Subject: What would you like to see most in minix?
Summary: small poll for my new operating system
Message-ID:
Date: 25 Aug 91 20:57:08 GMT
Organization: University of Helsinki


Hello everybody out there using minix -

I'm doing a (free) operating system (just a hobby, won't be big and
professional like gnu) for 386(486) AT clones.  This has been brewing
```

```
since april, and is starting to get ready.  I'd like any feedback on
things people like/dislike in minix, as my OS resembles it somewhat
(same physical layout of the file-system (due to practical reasons)
among other things).

I've currently ported bash(1.08) and gcc(1.40), and things seem to work.
This implies that I'll get something practical within a few months, and
I'd like to know what features most people would want.  Any suggestions
are welcome, but I won't promise I'll implement them :-)

                Linus (torvalds@kruuna.helsinki.fi)

PS.  Yes - it's free of any minix code, and it has a multi-threaded fs.
It is NOT protable (uses 386 task switching etc), and it probably never
will support anything other than AT-harddisks, as that's all I have :-(.
```

4.3.1. Historia

Linux nace a principio de los años noventa del siglo XX. Fue el resultado de enormes esfuerzos por parte de la cultura "Hacker" para tener la posibilidad de ejecutar un Sistema Operativo fiable y abierto como alternativa a los privativos y comerciales: MS-DOS de Microsoft y Mac OS de Apple.

La base de para Linux fue el código de Minix[28] que era a su vez, un prototipo funcional de UNIX con alrededor de 2000 líneas de código y disponible para cualquier estudiante que tuviera acceso al manual "*The Desing of Operating Systems*" del profesor Tanenbaum.

Además y como hemos citado anteriormente, el Proyecto GNU proporcionaba un potente compilador GCC (GNU C Compiler) además de otras herramientas para y el desarrollo y una Licencia que daba al propio proyecto un "marco legal".

Se podría afirmar que Linux no era más que un hobby de un estudiante de segundo curso de Informátiva en la Universidad de Helsinki.

Tras el correo escribo por Linus a la propia comunidad de Minix ver apartado (4.3). fue publicada la versión 0.0.1 en Septiembre de 1991 que a su vez sirvio como recurso publitario para una comunidad que empezaba a usar Internet y que era capaz de conectar grandes distancias en un tiempo muy corto. El trabajo inicialmente era una mejora de Minix, las versiones 0.10 y 0.11 (sobretodo está última) comenzador a dar forma a una pieza de un Sistema Operativo, un nuevo núcleo para Sistemas GNU.

Dichas versiones incluían soporte para: VGA, EGA, controladoras Floppy y múltiple "keymaps" fruto del trabajo de la comunidad.

El trabajo intenso de los desarrolladores provocó que la siguiente versión a la 0.12 fuera la 0.95. Linux fue duramente criticado por instituciones académicas dado que heredaba la arquitectura monolítica de Minix. Por ello y tras diversas discursiones se modifico la arquitectura del propio Kernel.

No debemos olvidar que gran parte de su éxito se debe al empuje comercial y la creación de importantes empresas en torno al propio Linux y otro Software GNU.

Debemos destarcar importantes empresas como Red Hat o Caldera. De igual manera, programadores y aficionados de todo el mundo comenzarón a trabajar es su propias versiones

GNU/Linux por ejemplo: Ian Murdock con Debian o Peter Volkerding con Slackware.

Gracias a la participación del miles de desarrolladores, Linux es hoy día uno de los productos con capacidad de ejecutarse en múltiples dispositivos con arquitecturas muy distintas.

En Marzo de 1994 se publicó la versión 1.0 estable del Kernel.

4.3.2. Arquitectura

4.3.3. Componentes

```
 - initramfs
 -> init (0)
 | --> RUN LEVEL 0 (JNB): Apagado de la máquina.
 | -> RUN LEVEL 1 (mono-user): Modo administración.
 | -> RUN LEVEL 2 (multi-user): Arranque del sistema en modo multiusuario sin
interfaces de red ni
servicios.
 | -> RUN LEVEL 3 (multi-user & net): Arranque del sistema en modo multiusuario
y con interfaces de red.
 | -> RUN LEVEL 4 (not-used): Sin definir.
 | -> RUN LEVEL 5 (multi-user, net & display manager): Arranque del sistema en
modo multiusuario, con interfaces de red y gestor de vertanas (entorno gráfico).
 | --> RUN LEVEL 6 (reboot): Reinicio de la máquina.
```

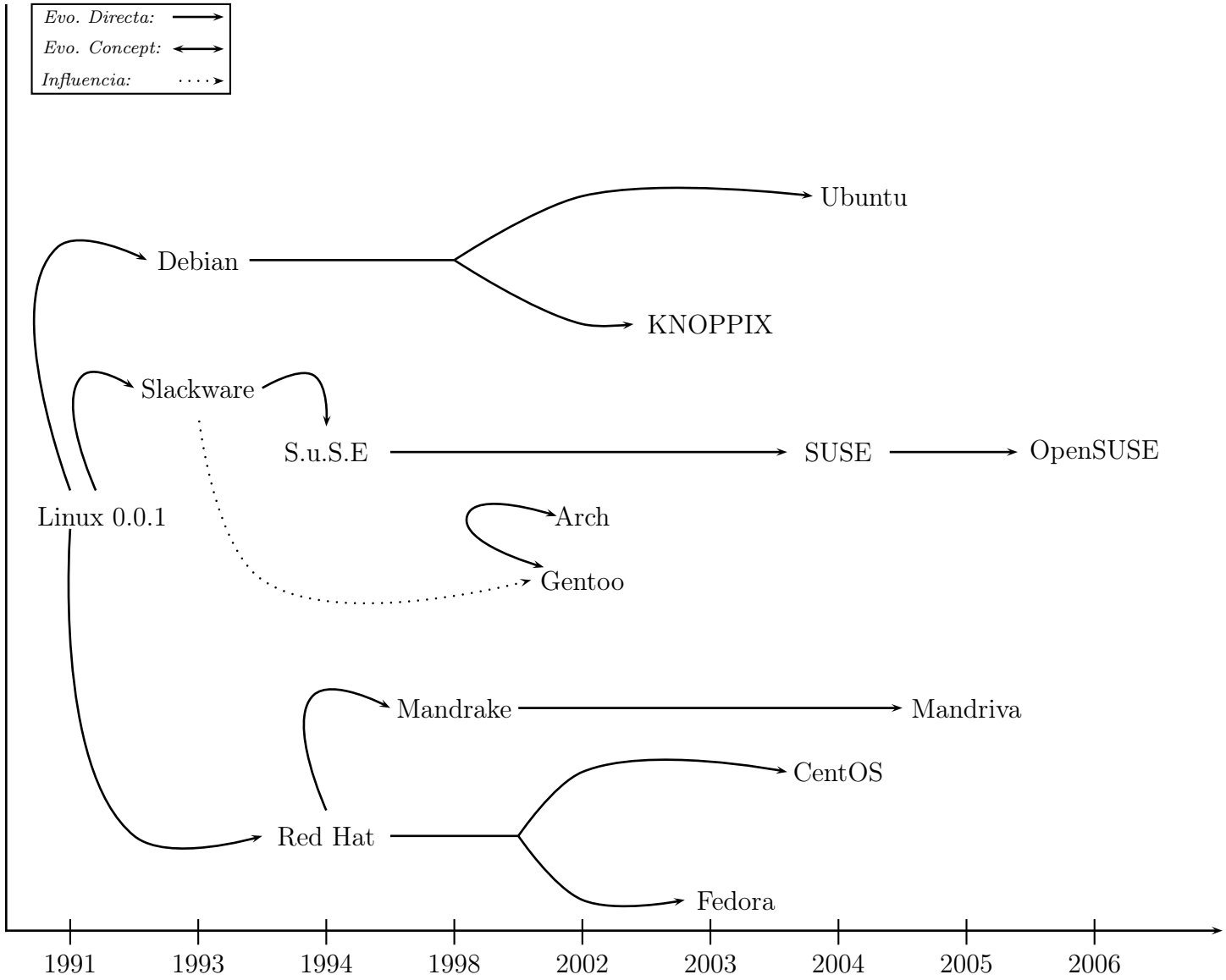

Figura 4.6: Evolución de Linux y distintas distribuciones en el tiempo.

Notas del capítulo

[1] http://www.alcatel-lucent.com/wps/portal/BellLabs
[2]http://www.att.com/
[3]http://web.mit.edu/
[4]http://www.ge.com/
[5]http://www.multicians.org/
[6]"Sistema de Computación e Información Multiplexada"
[7]**GE-645**
[8]En aquella época el coste de computo era realmente elevado debido en gran parte al consumo energético.
[9]**Ken Thomson**, nacio en Nueva Orleans (New Orleans, EEUU) el 4 de Febrero de 1943. Se trata de un prestigioso científico de la computación conocido historicamente como uno de los creadores del Sistema Opreativo UNIX. Thomson se gradua en 1966 como Ingeniero Electrónico y de Ciencias de la Computación por la Universidad de Berkley (California). Comenzó a trabajar para los Laboratorio Bell en el proyecto Multics bajo la suprevisión de Elwyn Berlekamp.

A comienzos de los años sesenta del siglo XX, Thomson en compañia de Ritchie continuaron en solitario (sin financiación) una nueva versión de Multics. La base de esta nueva implementación gira en torno a las utilidades Software creadas para el juego "Space Travel". En 1969 se publica la primera versión de este Sistema Operativo llamado UNICS (posteriormente UNIX).

Thomson también ha trabajado en el editor QED, que hace un potente uso de las expresiones regulares, base de posteriores lenguajes como Perl.

Otro hito de la historia de Ken Thomson es su participación en el desarrollo de UTF-8 junto a Rob Pike en 1992.

En el año 2000 Thomson deja los Laboratorios Bell para en 2006 trabajar como ingeniero distinguido en la empresa Google Inc.

Entre todos sus reconocimientos por su trabajo, destacamos los siguientes:

- En 1983, junto a Ritchie recibe el prestigios premio Turing.
- En 1990, de nuevo con Denis Ritchie recibe el premio IEEE Richard W. Hamming Medal del instituto IEEE (Institute of Electrical and Electronics Engineers).
- En 1997 entra a formar parte del Museo Historico de la Computación (Computer History Museum) con su compañero Ritchie.
- En 1999 Thomson es elegido por IEEE para recibir el primer premio Tsutomu Kanai Award.

[10]**GE-6354**
[11]"Viaje Espacial"
[12]**Dennis MacAlistair Ritchie** (también conocido como drm), nacido en 9 de Septiembre de 1941 en Bronxville Nueva York (New York, EEUU). Es conocido por se el padre del Lenguaje de Programación C.

Ritchie trabajó para los Laboratorio Bell de AT&T donde junto a su compañero Ken Thomson creo el Sistema Operativo UNIX (1969).

Junto a Ken Thomson ha sido galardonado con pretigiosos premios como: el Premio Turing (1983) por ACM, la Medalla Hamming (1990) por IEEE y la Medalla de la Tecnología (1990) de manos del presidente Bill Clinton.

Ritchie se retiró en el año 2007. El 12 de Octubre de 2011, a la edad de 70 años, falleció por un fallo cardiaco (tras pedecer un largo cancer) en Nueva Jersey (New Jersey, EEUU).
[13]DEC PDP-7
[14]
[15]"Sistema de Computación e Información Uniplexada"
[16]**Brian Wilson Kernighan** (también conocido como 'K'), nació en Toronto (Canada) en 1942. Es uno de los científicos más influyentes en la computación de la última mitad del siglo XX.

Trabajó en los Laboratorios Bell de AT&T inicialmente en el Proyecto Multics para después colaborar activamente en el desarrollo de UNIX. A Kernighan se le debe el nombre de UNIX, puesto que el Sistema Operativo creado por Ritchie y Thomson originalmente recibia el nombre de UNICS, basandose en el acrónimo MULTICS.

Kernighan es un autor muy prolífico. Su firma está en obras y Software como:

1. “C Programming Language" con Dennis Ritchie, aunque sus palabras dicen lo contrario: “it’s entirely Dennis Ritchie’s work".
2. Es coautor de los lenguajes: AWK y AMPL.
3. Ha trabajado con Shen Lin en métodos de optimización de problemas NP-complete, gráficos particionados y problemas del viajero.
4. Es también conocido por la frase: (WYSIAYG) “What You See Is All You Get"(Lo que ves es todo lo que obtienes), que deriva de: (WYSIWYG) “What You See Is What You Get"(Lo que ves es lo que obtienes).

Actualmente Kernighan es profesor en el Departamento de Ciencis de la Computación de la Universidad de Princenton (Princeton University).

[17]Programmed Data Processor (PDP) was a series of minicomputers made and marketed by the Digital Equipment Corporation from 1957 to 1990. The name ’PDPíntentionally avoided the use of the term ’computer’because, at the time of the first PDPs, computers had a reputation of being large, complicated, and expensive machines, and the venture capitalists behind Digital (especially Georges Doriot) would not support Digital’s attempting to build a çomputer"; the word "minicomputer"had not yet been coined. So instead, Digital used their existing line of logic modules to build a Programmable Data Processor and aimed it at a market which could not afford the larger computers.

[18] http://www.gnu.org/software/groff/

[19]Decir que partes de UNIX para aquel entonces y aun hoy en día se mantienen en código nativo de cada procesador.

[20]http://www.gnu.org/

[21]**Richard Matthew Stallman** nacio el 16 de Marzo de 1953 en la Ciudad de Nueva York (New York City, EEUU). Stallman es un reconocido activista a favor de los derechos (en cuestion de ciertas libertades) de los usuarios de computadoras.

Richard Stallman se gradua como físico por la Universidad de Harvar en 1974. Poco después trabaja y realiza un post-grado en el prestigioso MIT. Allí conocé: "MIT’s hacker culture"(La cultura Harcker del MIT) donde los códigos fuente de los programas eran compartidos por losd desarrolladores. El problema de acceso a las fuentes del programa de cierto Hardware durante su trabajo en el MIT, incita a Stallman, a crear una organización que se basase en los principios de que el Software es .ªbiertos. Dicho de otro modo, el Software debe ser accesible para los usuarios, respetando su/sus autor/autores, pero siempre con la idea de que teniendo este código, será mucho más sencillo adaptar el Hardware a los distintas necesidades de un usuario.

Con esta idea crea en 1983 "GNU Project"(Proyecto GNU) para crear un clon libre de UNIX. Con el objeto de asentar una cultura "para el movimiento libre del Software", en 1985 funda "The Free Software Foundation"(Fundación del Software Libre).

Durante estos años fue uno de los mayores contribuyentes del código para el Proyecto GNU. Dado quem como dice textualmente: "Yo no puedo crear Leyes", decide que el Software de GNU sea distribuido bajo una licencia llamada "GNU General Public License"(Licencia Pública General de GNU).

El 25 se Febrero de 1989 publica la primera versión de la denominada, GPL, que ha sufrido a lo largo de su historia, tres revisiones.

[22]http://www.fsf.org/

[23] http://www.gnu.org/copyleft/gpl.html

[24] http://www.gnu.org/software/hurd/

[25]**Linus Benedict Torvalds** nació el 28 de Diciembre de 1969 en Helsinki (Finlandia). Linus es conocido en el mundo entero por ser el primer creador y actual máximo responsable del Kernel Linux.

Además Linus es cocreador del Software de Control de Versiones Git. Durante los años 1988 y 1996 se gradua en Ciencias de la Computación por la Universidad de Helsinki.

Su formación académica fue interrumpida por distintos motivos, entre ellos el servicio militar. Poco después de finalizar su preparación obligatoria militar y ya de nuevo en la universidad tiene sus primeros contactos con el Sistema Operativo UNIX.

Primero se forma con grandes máquinas para después, y sobre una computadora PC-Compatible con un procesador 80386 de Intel, empezar a mejorar una copia de MINIX. Su Software base era el propio MINIX y el compilador GCC.

Finalmente el 5 de Junio de 1991 anuncia en el grupo çomp.os.minix"su nuevo sistema e invita a otros usuarios a participen en el para que a través de sus sugerencias, Linus haga evolucionar el primitivo Linux (en su famoso correo electrónico aclara que no promete incorporar en el Software todas las propuestas).

Linus originalmente llama a su trabajo "Freax"(un juego de palabras entre "Free, Librez "X, en alusión a UNIX".

Torvals decide licenciar al ya denominado Linux bajo el copyriht de GNU (GPL) en base a la idea de que el compilador que usa es el popular GCC.

Linus es cocreador de Git, un popular sistema de control de versiones que tuvo su primera versión disponible para el público el 7 de Abril de 2005. El origen del proyecto es parte de las insistentes críticas de "Linux Foundationz el propio Torvals de CVS (un popular Sistema de Control de Versiones gratuito).

[26]http://www.minix3.org/

[27]

```
Hola a todos aquellos que usan Minix -

Estoy haciendo un sistema operativo (libre) (sólo un hobby, no será grande y
profesional como GNU) para 386 (486) AT clones. Esta ha sido la cerveza
desde abril, y está empezando a prepararse. Me gustaría algún comentario sobre
cosas que la gente le gusta / disgusta de minix, como mi sistema operativo se asemeja un poco
(la misma disposición física del sistema de archivos (por razones prácticas)
entre otras cosas).

Me he portado bash (1.08) y gcc (1.40), y las cosas parecen funcionar.
Esto implica que tendré algo práctico dentro de unos meses, y
Me gustaría saber qué características la mayoría de la gente quiere. cualquier sugerencia
son bienvenidos, pero no voy a prometer voy a ponerlas en práctica :-)

               Linus (torvalds@kruuna.helsinki.fi)

PS. Sí - es libre de cualquier código de minix, y tiene un sistema de archivos multi-hilo.
NO es protable (386 utiliza la conmutación de tareas, etc), y probablemente nunca
apoyará nada que no sea AT-discos duros, ya que es todo lo que he :-(.
```

[28]Minix es un proyecto...

Bibliografía

[Bac86] Maurice J. Bach. *The Desing of the Unix Operating System*. Prentice/Hall International, Inc, 1986.

[KP87] Brian W. Kerninghan and Rob Pike. *El Entorno de Programación UNIX*. Prentice-Hall, 1987.

[Mau08] Wolfgang Mauerer. *Professional Linux® Kernel Architecture*. Wiley Plublishing, Inc, 2008.

[MNN05] Marshall Kirk Mckusick and George V. Neville-Neil. *The Desing and Implementation of the FreeBSD Operating System*. Addison-Wesley, 2005.

[rhc12] rhcsoftware.com. About RHC Software. http://www.rhcsoftware.com/, 2012.

[Sta85] Richard Stallman. The GNU manifesto. *j-DDJ*, 10(3):30–??, mar 1985.

[Vah96] Uresh Vahalia. *UNIX Internals: The New Frontiers*. Prentice-Hall, 1996.

Índice alfabético