Universidad
Inca Garcilaso de la Vega
Nuevos Tiempos. Nuevas Ideas

FACULTAD DE INGENIERÍA DE SISTEMAS, CÓMPUTO Y TELECOMUNICACIONES
PROGRAMA DE EDUCACION A DISTANCIA

Tarea Académica – 2019.1

ARQUITECTURA GNU LINUX

CURSO: SISTEMAS OPERATIVOS

APELLIDOS: ARMAS BENAVIDES
NOMBRES: RAÚL MARCELO
CÓDIGO: 732624420
CARRERA PROFESIONAL: INGENIERÍA DE SISTEMAS Y CÓMPUTO

DATOS DE LA ASIGNATURA

ASIGNATURA : SISTEMAS OPERATIVOS
PROFESOR : SANTIAGO GONZÁLES SÁNCHEZ
FECHA DE PRESENTACIÓN : 27/03/2019

ÍNDICE

Contenido

1 INTRODUCCIÓN

El presente artículo es un estudio de revisión en donde se ha pesquisado distintas

fuentes bibliográficas con la finalidad de resumir la arquitectura GNU/LINUX.
Los sistemas GNU/Linux [Joh98] ya no son una novedad; cuentan con una amplia variedad de usuarios y de ámbitos de trabajo donde son utilizados. Su origen se remonta al mes de agosto de 1991, cuando un estudiante finlandés llamado Linus Torvalds anunció, en el newsgroup comp.os.minix que había creado su propio núcleo de sistema operativo y lo ofreció a la comunidad de desarrolladores para que lo probara y sugiriera mejoras para hacerlo más utilizable. Este sería el origen del núcleo (o kernel) del operativo que, más tarde, se llamaría Linux. Por otra parte, la FSF (Free Software Foundation), mediante su proyecto GNU, producía software desde 1984 que podía ser utilizado libremente, debido a lo que Richard Stallman (miembro de la FSF) consideraba software libre: aquel del que podíamos conseguir sus fuentes (código), estudiarlas y modificarlas, y redistribuirlo sin que nos obliguen a pagar por ello. En este modelo, el negocio no está en la ocultación del código, sino en el software complementario añadido, en la adecuación del software a los clientes y en los servicios añadidos, como el mantenimiento y la formación de usuarios (el soporte que les ofrezcamos), ya sea en forma de material, libros y manuales, o en cursos de formación. La combinación (o suma) del software GNU y del kernel Linux es la que nos ha traído a los actuales sistemas GNU/Linux. Actualmente, tanto los movimientos Open Source, desde diferentes organizaciones (como FSF) y empresas como las que generan las diferentes distribuciones Linux (Red Hat, Canonical Ubuntu, Mandrake, Novell SuSe...), pasando por grandes empresas (como HP, IBM o Sun, que proporcionan apoyos y/o patrocinios), han dado un empujón muy grande a los sistemas GNU/Linux hasta situarlos al nivel de poder competir, y superar, muchas de las soluciones propietarias cerradas existentes.

2 DEFINICIÓN DE LINUX

Los sistemas GNU/Linux no son ya una novedad. El software GNU se inició a mediados de los ochenta, y el kernel Linux a principios de los noventa. Linux se apoya en tecnología probada de UNIX, con más de cuarenta años de historia.

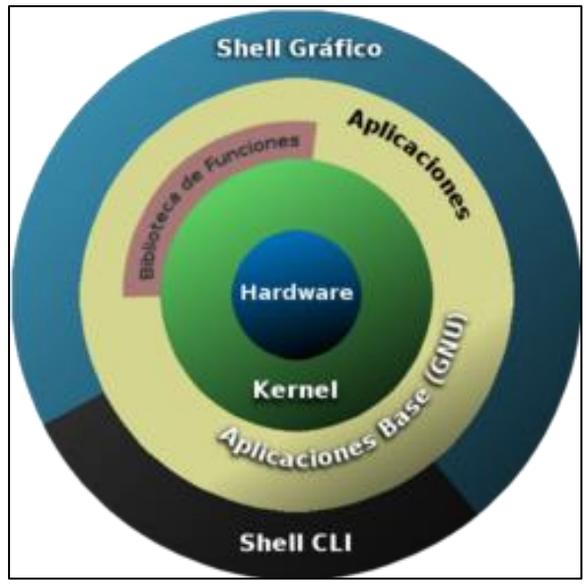
Contribución de GNU:

- El compilador de C y C++ (GCC)
- Shell bash (el más utilizado en LINUX)

3 ARQUITECTURA DE GNU/LINUX

El sistema operativo es, en sí, una interfaz de software que nos permite a nosotros, como humanos, interactuar con los dispositivos de hardware.

Esta interfaz está dividida en varias capas, cada una de las cuales tendrá una funcionalidad específica.



El hardware, como se ve, se encuentra en el centro de la figura, y tenemos varias “capas” de software que lo

envuelven, siendo el operador externo quien interactúa, desde afuera, con todo el esquema.

El hardware es el conjunto de dispositivos físicos del ordenador, llámese memoria RAM, almacenamiento secundario, discos, ssd, microprocesadores, puertos de comunicación, etc.

3.1 El kernel Linux

Por su parte, el kernel del sistema es una capa de software que recubre al hardware, y que contiene muchas utilidades incorporadas.

Primero y principal, el kernel se comunica con el hardware por medio de controladores de dispositivos, o drivers. Estos controladores son programados por los propios fabricantes de los dispositivos, o, en su defecto, por programadores de Linux que realizan ingeniería inversa sobre drivers privativos.

El kernel también brinda otras utilidades, que podrían, groso modo, englobarse en los siguientes áreas:

- Administración de memoria virtual
- Administración de procesos
- Administración de Entrada/Salida
- Administración de red

Con estas características, nuestro kernel podrá planificar procesos en el proceasdor, gestionar recursos de almacenamiento principal, como ser memoria RAM, y espacio de intercambio (swap)[1], y gestionar las peticiones de entrada/salida con cualquier dispositivo e interfaz de red.

3.2 El sistema operativo GNU

Una capa externa al kernel es la del propio sistema operativo, y sus aplicaciones. El sistema operativo se compone, entre otras cosas, de utilidades como editores de texto (vi/vim, Emacs, nano), compiladores (**GCC**), intérprete de comandos **(Bash, sh, rsh**), entre otras. El sistema operativo es el encargado de brindar soporte a aplicaciones de usuario, mediante intérpretes de órdenes, como ser una shell de línea de comandos, o un entorno gráfico, ya sea un entorno de escritorio completo, o simples administradores de ventanas.

3.3 La interfaz de llamadas al sistema

Existe un elemento intermedio entre el kernel Linux y el sistema GNU, que no figura en la gráfica, y que se denomina Interfaz de llamadas al sistema, o “syscall”. Esta interfaz, conocida como POSIX.1, provee al sistema operativo un API, o interfaz de programación de aplicaciones, que no es otra cosa que una serie de funciones del kernel que pueden ser accedidas desde el sistema operativo mediante llamadas. Así es como el sistema operativo puede comunicarse con el kernel, enviarle instrucciones, y recibir resultados.

[1] El área de intercambio también se le conoce como memoria virtual

4 *NIX Y LINUX, DETALLES DE LA ARQUITECTURA

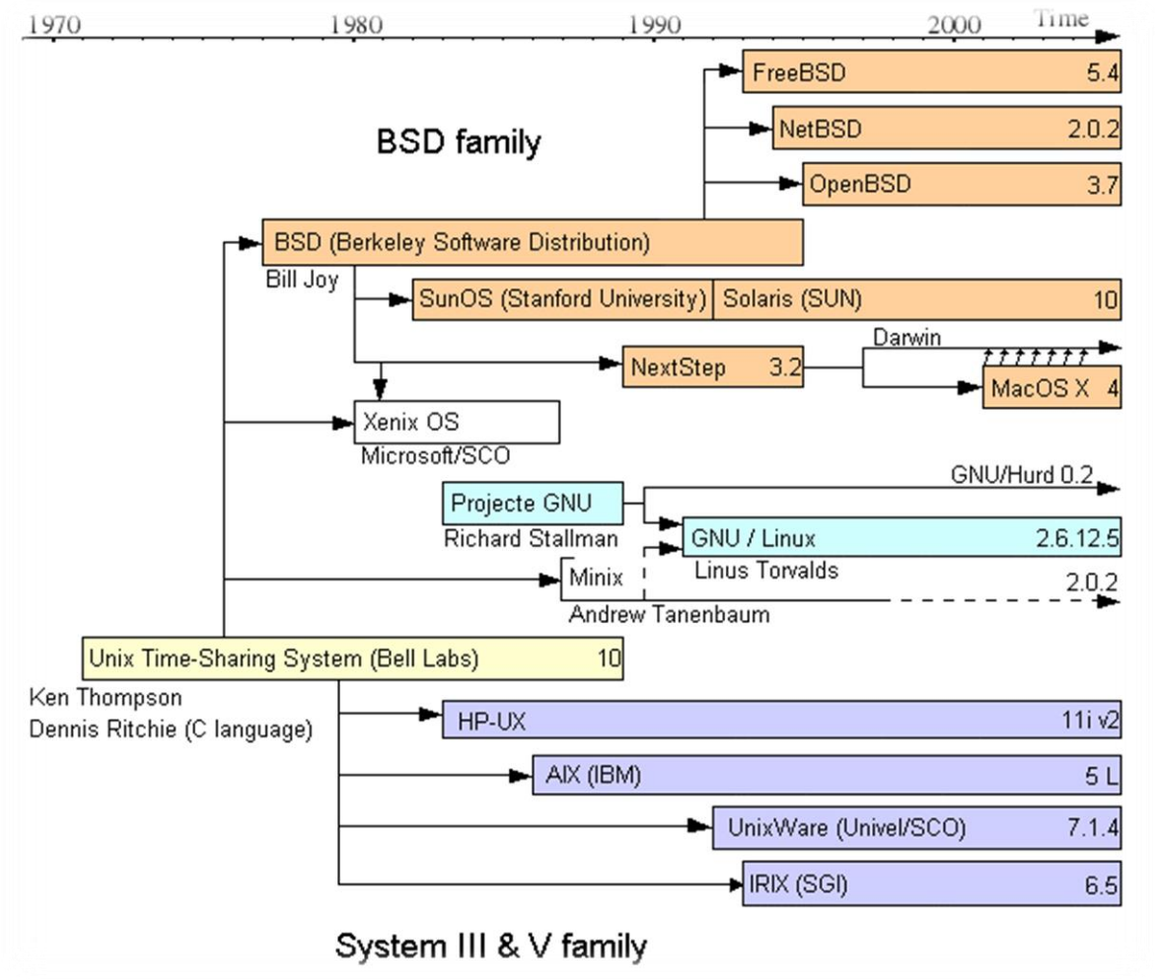
4.1 *Nix: Unix y Linux

Unix es un sistema operativo creado en los laboratorios AT&T Bell por Ken Thompson y Dennis Ritche entre otros, a fines de los años 60.

Recordemos que Ritche fue también el programador que creó, junto con Kernighan, el lenguaje de programación C, y juntos escribieron “The C Programming Language“, el libro en el que explican su sintaxis y estructuras.

Unix es, en fin, el precursor de todos los sistemas Linux y derivados, y por supuesto, el padre de todas las variantes de Unix disponibles en la actualidad (Solaris, BSD, …).

En la siguiente imagen podemos ver una cronología de desarrollo de versiones de Unix, entre las que se encuentra Linux, por supuesto. A todos estos sistemas se los conoce, en general, como sistemas operativos *nix.



4.2 C y *Nix

Todos los sistemas *nix están programados, en su mayor parte, por **código C**. Linux deriva de Unix, por lo que está mayormente programado en C también, junto con algunos segmentos de código **C++** (la variante orientada a objetos de C) y lenguajes de **scripting**. Linux en sí es el **kernel** del sistema operativo GNU en distribuciones GNU/Linux. GNU también está escrito en C en su mayor parte. Por lo general las aplicaciones que encontramos para Linux también están escritas en C o C++. El kernel, como hemos dicho tiene su mayor parte de código C, y controladores de bajo nivel escritos en lenguaje ensamblador.

El kernel del sistema operativo implementa muchos de los mecanismos originales utilizados por Unix para llevar a cabo sus tareas, y por supuesto, es código abierto, e incluso nosotros como programadores podemos escribir código utilizando estos mecanismos.

4.3 El Kernel Linux

El kernel de un sistema operativo tiene muchas tareas específicas, que pueden englobarse generalmente en las siguientes:

4.3.1 Gestión de Procesos

La administración de procesos hace referencia a cómo le kernel planificará los procesos en el procesador, cómo administrará las colas de procesos listos, en ejecución, esperando algún evento de entrada/salida, interrumpidos esperando alguna señal, etc.
Linux es, como Unix, un sistema operativo multitarea, por lo que puede ejecutar varios procesos simultáneamente. Para lograr este cometido, Linux arma una cola de procesos listos para entrar al procesador de la máquina, o a algún core libre del mismo. Determinar qué proceso de dicha lista será el siguiente en ocupar el recurso es tarea del planificador de procesos del kernel. El planificador determina ésto mediante un manejo de prioridades de tiempo real y de usuario, y una serie de bonus para evitar que un proceso de menor prioridad se mantenga fuera de ejecución por demasiado tiempo.

4.3.2 Gestión de Memoria

La administración de memoria se refiere a cómo el kernel gestionará la memoria principal del sistema, qué mecanismo implementará, memoria paginada, memoria segmentada, combinación de paginación y segmentación, administración de espacios de intercambio y memoria virtual, administración de segmentos de memoria compartida entre procesos, direcciones de memoria de nivel de usuario y de nivel de núcleo, etc.

4.3.3 Gestión de entrada/salida

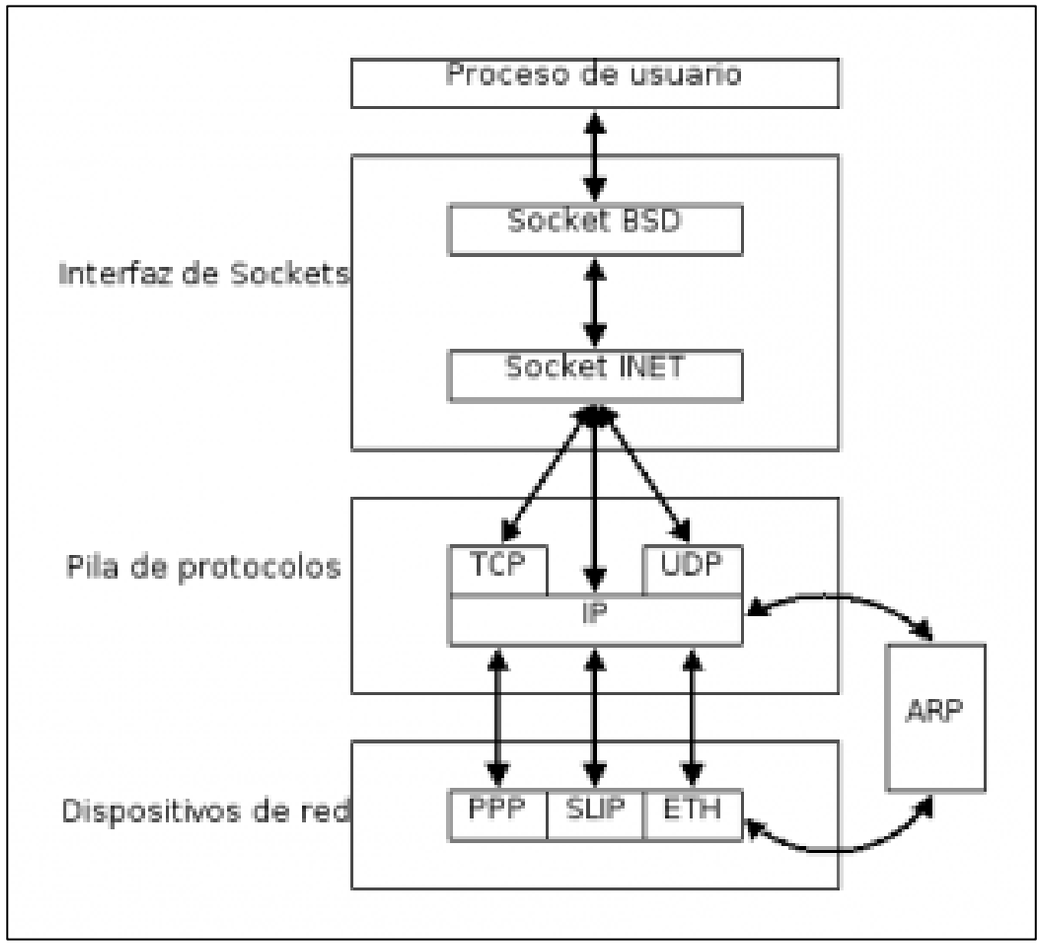
La gestión de entrada/salida hace referencia a cómo el kernel llevará a cabo tareas de comunicación con dispositivos externos, o periféricos, tales como teclados, mouses, o dispositivos de almacenamiento.

4.3.4 Gestión de red

La gestión de red podría considerarse un mecanismo de entrada/salida, pero la mayor parte de los sistemas operativos generalmente lo implementan en forma separada, como una tarea de comunicación de datos con equipos remotos independiente. Aquí tenemos también la administración de múltiples direcciones por interfaz de red (ip aliassing), administración de firewalls y filtrado a nivel de capa de red, y gestión de redes privadas virtuales soportadas por ip (IPSec).

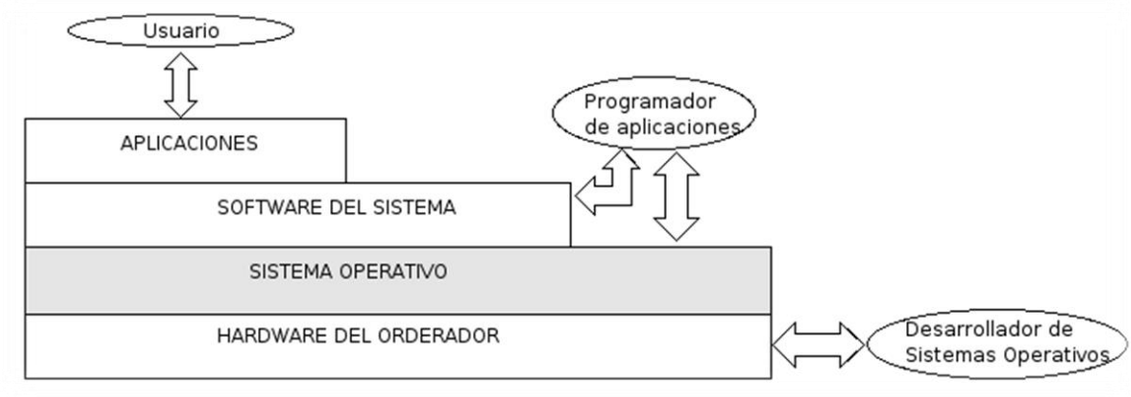
4.3.5 Gestión de IPC

La gestión de IPC, o mecanismos de comunicación entre procesos (Inter Process Communication) es una serie de herramientas que utiliza el kernel para establecer intercambios de datos y señalización entre procesos en el sistema operativo. Este es uno de los puntos importantes a destacar cuando decidimos incursionar en la programación *nix.



4.4 La arquitectura general de un *NIX

En general, la arquitectura de un sistema operativo está dada por la siguiente figura.



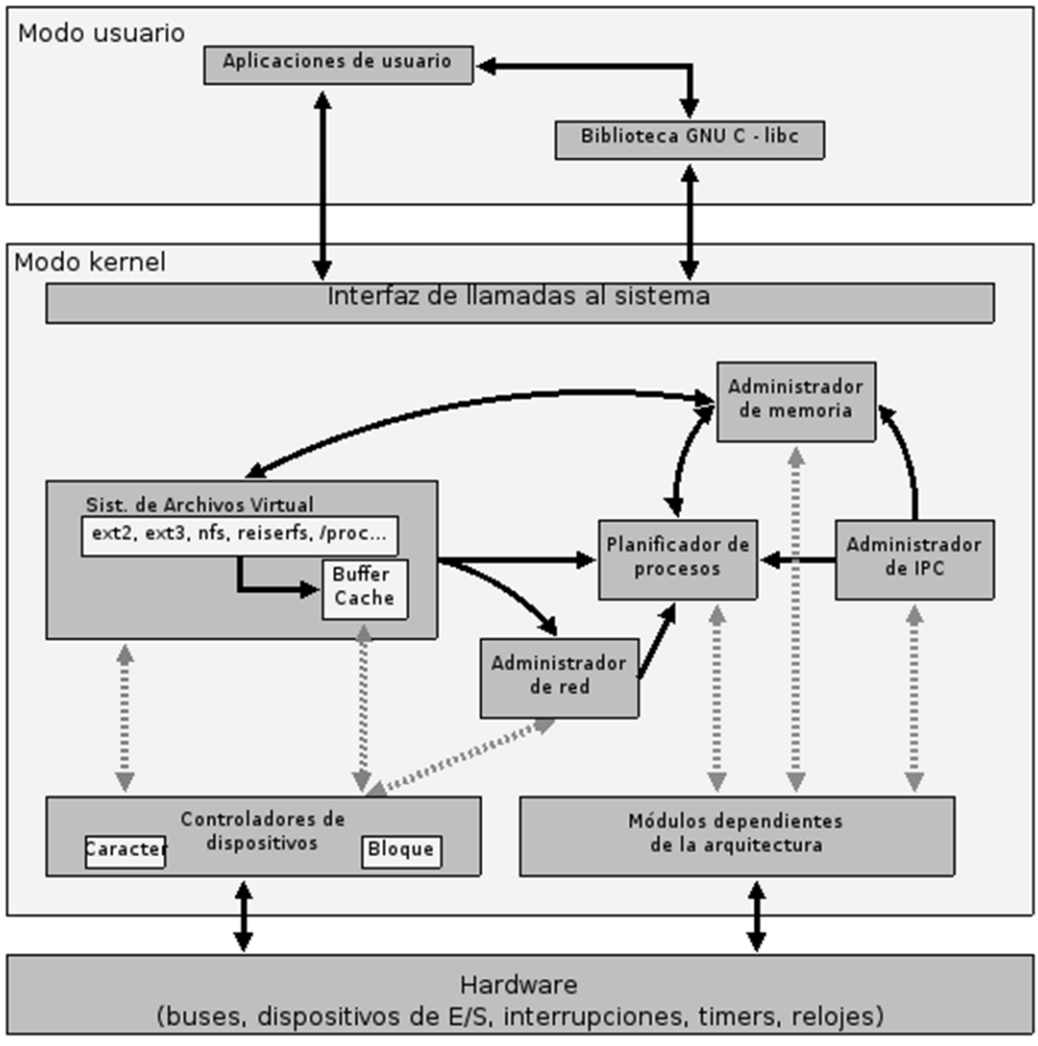
En ella podemos apreciar que un sistema operativo puede dividirse en diferentes capas. La capa superior, en la que trabaja el usuario final, es la que contiene las aplicaciones instaladas. Debajo de las aplicaciones encontramos la capa de software de sistema, que contiene herramientas de desarrollo y API's de medio/bajo nivel para el acceso a los recursos del sistema operativo. Debajo de esta capa encontramos directamente el sistema operativo. El sistema operativo oficia de interfaz entre el kernel y el entorno de usuario final.

Dentro del sistema operativo como concepto general encontramos al kernel. Lo correcto es considerar al kernel una pieza totalmente aislada del sistema, puesto que tienen funcionalidades y objetivos diferentes. El kernel se encarga de dar acceso a los recursos de hardware al sistema operativo. El kernel mantiene una serie de controladores y módulos de dispositivos que le permiten al sistema operativo, y por consiguiente, a las

aplicaciones de usuario, acceder a los dispositivos de hardware.
Existe una capa que separa al kernel del resto del sistema, llamada Interfaz de llamadas al sistema, o **SysCall Interface**. La syscall es una API, o interfaz de programación que ofrece al sistema, y por consiguiente, a todo programador de sistemas operativos, o de utilidades para el mismo, funciones de acceso de medio y bajo nivel a las rutinas del kernel.
Esto le permite al programador administrar sus procesos, memoria, y mecanismos de comunicación, de una manera eficiente, logrando que las utilidades y herramientas disponibles brinden una excelente flexibilidad y un enorme espectro de posibilidades.

4.5 Organización del Kernel Linux

El kernel Linux divide el espacio de direccionamiento en dos. Uno es el modo usuario, en el que corren aplicaciones de usuario y bibliotecas de C, como el imprescindible componente de un sistema GNU/Linux, libc.
En el modo kernel corre todo el kernel del sistema, y ésto se debe a que Linux es un sistema de arquitectura monolítica. En el espacio del núcleo encontramos los administradores de los que hemos estado hablando más arriba, como el planificador de procesos, el gestor de memoria, y los administradores de entrada/salida, redes, y sistema de archivos virtual.
En el modo núcleo también se encuentran los controladores de dispositivos y los módulos de administración de hardware.
Todo este kernel, como vimos, corre sobre el hardware del equipo, y lo administra de una manera extremadamente eficiente.
En la siguiente figura podemos ver una relación entre los diferentes componentes del kernel Linux, sus comunicaciones, y su organización dentro de la arquitectura de un sistema operativo.



5 Tareas del administrador

Según hemos descrito, podríamos separar las tareas de un administrador GNU/ Linux (o UNIX en general) en dos partes principales: administración del sistema y administración de red. En los siguientes subapartados, mostramos de forma resumida en qué consisten en general estas tareas en los sistemas GNU/ Linux (o UNIX). Trataremos la mayor parte del contenido con cierto detalle en estos módulos y en los asociados a administración avanzada. Otra parte de las tareas, por cuestiones de espacio o complejidad, la explicaremos superficialmente o no la trataremos. Las tareas de administración engloban una serie de conocimientos y técnicas de los cuales aquí sólo podemos ver la "punta del iceberg". En la bibliografía adjunta a cada módulo aportamos referencias para ampliar todos los temas a tratar. Como veréis, hay una amplia bibliografía para casi cualquier punto en el que queráis profundizar.

5.1.1 Tareas de administración local del sistema

- ✓ **Arranque y apagado del sistema:** cualquier sistema basado en UNIX tiene unos sistemas de arranque y apagado ajustables, de manera que podemos configurar qué servicios ofrecemos en el arranque de la máquina y cuándo hay que pararlos, o programar el apagado del sistema para su mantenimiento.

- ✓ **Gestión de usuarios y grupos**: dar cabida a los usuarios es una de las principales tareas de cualquier administrador. Habrá que decidir qué usuarios podrán acceder al sistema, de qué forma y bajo qué permisos, y establecer

comunidades mediante los grupos. Un caso particular será el de los usuarios de sistema, pseudousuarios dedicados a tareas del sistema.

- ✓ **Gestión de recursos del sistema:** qué ofrecemos, cómo lo ofrecemos y a quién damos acceso.

- ✓ **Gestión de los sistemas de ficheros:** el ordenador puede disponer de diferentes recursos de almacenamiento de datos y dispositivos (disquetes, discos duros, ópticos, etc.) con diferentes sistemas de acceso a los ficheros. Pueden ser permanentes, extraíbles o temporales, con lo cual habrá que modelar y gestionar los procesos de montaje y desmontaje de los sistemas de ficheros que ofrezcan los discos o dispositivos afines.

- ✓ **Cuotas del sistema:** cualquier recurso que vaya a ser compartido tiene que ser administrado, y según la cantidad de usuarios, habrá que establecer un sistema de cuotas para evitar el abuso de los recursos por parte de los usuarios o establecer clases (o grupos) de usuarios diferenciados por mayor o menor uso de recursos. Suelen ser habituales sistemas de cuotas de espacio de disco, o de impresión, o de uso de CPU (tiempo de computación usado).

- ✓ **Seguridad del sistema:** seguridad local, sobre protecciones a los recursos frente a usos indebidos, accesos no permitidos a datos del sistema, o a datos de otros usuarios o grupos.

- ✓ **Backup y restauración del sistema:** es necesario establecer políticas periódicas (según importancia de los datos), de copias de seguridad de los sistemas. Hay que establecer periodos de copia que permitan salvaguardar nuestros datos de fallos del sistema (o factores externos) que puedan provocar pérdidas o corrupción de datos.

- ✓ **Automatización de tareas rutinarias:** muchas de las tareas frecuentes de la administración o del uso habitual de la máquina pueden ser fácilmente automatizadas, ya debido a su simplicidad (y por lo tanto, a la facilidad de repetirlas), como a su temporalización, que hace que tengan que ser repetidas en periodos concretos. Estas automatizaciones suelen hacerse bien mediante programación por lenguajes interpretados de tipo script (shells, Perl, etc.), o por la inclusión en sistemas de temporalización (cron, at...).

- ✓ **Gestión de impression y colas:** los sistemas UNIX pueden utilizarse como sistemas de impresión para controlar una o más impresoras conectadas al sistema, así como para gestionar las colas de trabajo que los usuarios o aplicaciones puedan enviar a las mismas.

- ✓ **Gestión de módems y terminales:** estos dispositivos suelen ser habituales en entornos no conectados a red local ni a banda ancha:

  - Los módems permiten una conexión a la Red por medio de un intermediario (el ISP o proveedor de acceso), o bien la posibilidad de conectar a nuestro sistema desde el exterior por acceso telefónico desde cualquier punto de la red telefónica.

  - En el caso de los terminales, antes de la introducción de las redes solía ser habitual que la máquina UNIX fuese el elemento central de cómputo, con una serie de terminales "tontos", que únicamente se dedicaban a visualizar la información o a permitir la entrada de información por medio de teclados externos. Solía tratarse de terminales de tipo serie o paralelo. Hoy en día, todavía suelen ser habituales en entornos industriales, y en

nuestro sistema GNU/Linux de escritorio tenemos un tipo particular, que son los terminales de texto "virtuales", a los que se accede mediante las teclas Alt+Fxx.

- ✓ **Accounting(olog) de sistema:** para poder verificar el funcionamiento correcto de nuestro sistema, es necesario llevar políticas de log que nos puedan informar de los posibles fallos del sistema o del rendimiento que se obtiene de una aplicación, servicio o recurso hardware. O bien permitir resumir los recursos gastados, los usos realizados o la productividad del sistema en forma de informe.

- ✓ **System performance tunning:** técnicas de optimización del sistema para un fin dado. Suele ser habitual que un sistema esté pensado para una tarea concreta y que podamos verificar su funcionamiento adecuado (por ejemplo, mediante logs), para examinar sus parámetros y adecuarlos a las prestaciones que se esperan.

- ✓ **Personalización del sistema:** reconfiguración del kernel. Los kernels, por ejemplo en GNU/Linux, son altamente personalizables, según las características que queramos incluir y el tipo de dispositivos que tengamos o esperemos tener en nuestra máquina, así como los parámetros que afecten al rendimiento del sistema o que consigan las aplicaciones.

5.2 Tareas de administración de red

- ✓ **Interfaz de red y conectividad:** el tipo de interfaz de red que utilizamos, ya sea el acceso a una red local, la conexión a una red mayor, o conexiones del tipo banda ancha con tecnologías ADSL, RDSI, u ópticas por cable. Además, el tipo de conectividades que vamos a tener en forma de servicios o peticiones.

- ✓ **Routing de datos:** los datos que circularán, de dónde o hacia dónde se dirigirán, dependiendo de los dispositivos de red disponibles y de las funciones de la máquina en red; posiblemente, será necesario redirigir el trafico desde/hacia uno o más sitios.

- ✓ **Seguridad de red:** una red, sobre todo si es abierta a cualquier punto exterior, es una posible fuente de ataques y, por lo tanto, puede comprometer la seguridad de nuestros sistemas o los datos de nuestros usuarios. Hay que protegerse, detectar e impedir posibles ataques con una política de seguridad clara y eficaz.

- ✓ **Servicios de nombres:** en una red hay infinidad de recursos disponibles. Los servicios de nombres nos permiten nombrar objetos (como máquinas y servicios) para poderlos localizar y gestionar. Con servicios como el DNS, DHCP, LDAP, etc., se nos permitirá localizar servicios o equipos.

- ✓ **NIS (network information service):** las grandes organizaciones han de tener mecanismos para poder organizar, de forma efectiva, los recursos y el acceso a ellos. Las formas habituales en UNIX estándar, como los logins de usuarios con control por passwords locales, son efectivos con pocas máquinas y usuarios, pero cuando tenemos grandes organizaciones, con estructuras jerárquicas, usuarios que pueden acceder a múltiples recursos de forma unificada o separada por diferentes permisos, etc., los métodos UNIX sencillos se muestran claramente insuficientes o imposibles. Entonces se necesitan sistemas más eficaces para controlar toda esta estructura. Servicios como NIS, NIS+ y LDAP nos permiten organizar de modo adecuado toda esta complejidad.

- ✓ **NFS (network fyle systems**): a menudo, en las estructuras de sistemas en red es necesario compartir informaciones (como los propios ficheros) por parte de todos o algunos de los usuarios. O sencillamente, debido a la distribución física

de los usuarios, es necesario un acceso a los ficheros desde cualquier punto de la red. Los sistemas de ficheros por red (como NFS) permiten un acceso transparente a los ficheros, independientemente de nuestra situación en la red. Y en algunos casos, como Samba/CIFS, nos ofrecen soporte para el acceso por parte de plataformas hardware/software diferentes, e independientes de las configuraciones de clientes o servidores.

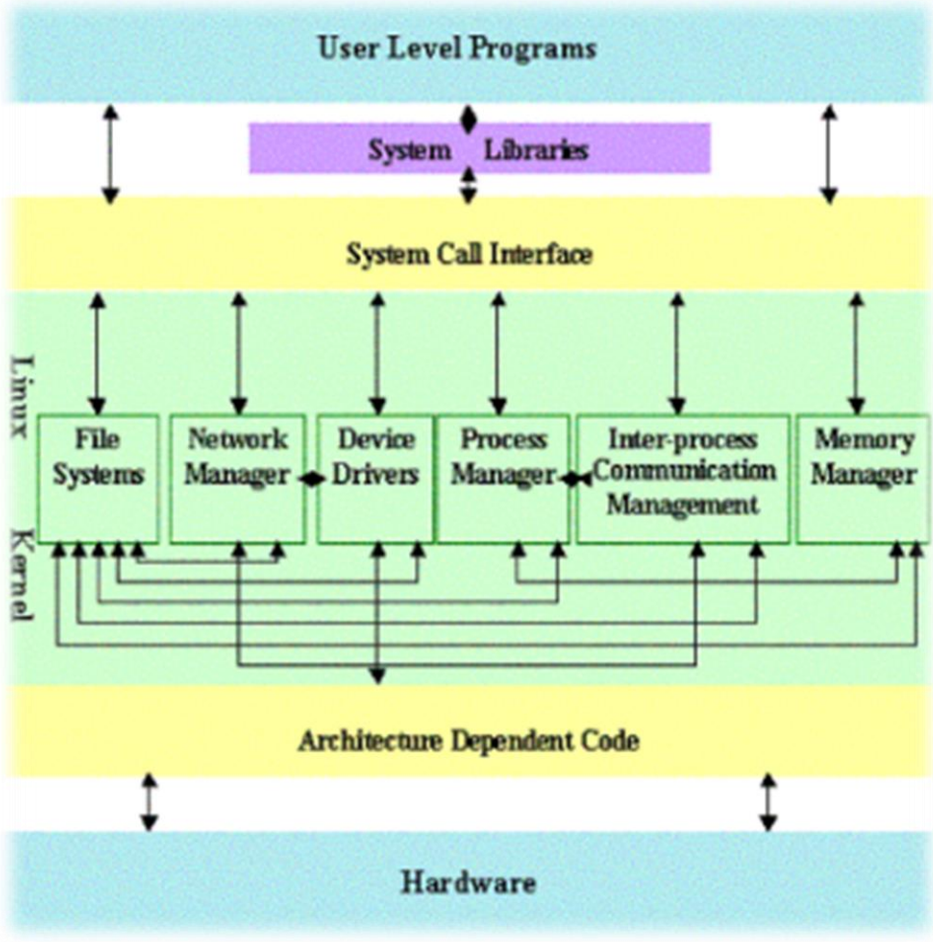
- ✓ **UNIX remote commands:** UNIX dispone de comandos transparentes a la red, en el sentido de que, independientemente de la conexión física, es posible ejecutar comandos que muevan información por la red o permitan acceso a algunos servicios de las máquinas. Los comandos suelen tener una "r" delante, con el sentido de 'remoto', por ejemplo: rcp, rlogin, rsh, rexec, etc., que permiten las funcionalidades indicadas de forma remota en la red.

- ✓ **Aplicaciones de red:** aplicaciones de conexión a servicios de red, como telnet (acceso interactivo), ftp (transmisión de ficheros), en forma de aplicación cliente que se conecta a un servicio servido desde otra máquina. O bien que nosotros mismos podemos servir con el servidor adecuado: servidor de telnet, servidor ftp, servidor web, etc.

- ✓ **Impresión remota:** acceso a servidores de impresión remotos, ya sea directamente a impresoras remotas o bien a otras máquinas que ofrecen sus impresoras locales. Impresión en red de forma transparente al usuario o aplicación.

- ✓ **Correo electrónico:** uno de los primeros servicios proporcionados por las máquinas UNIX es el servidor de correo, que permite el almacenamiento de correo, o un punto de retransmisión de correo hacia otros servidores, si no iba dirigido a usuarios propios de su sistema. Para el caso web, también de forma parecida, un sistema UNIX con el servidor web adecuado ofrece una plataforma excelente para web. UNIX tiene la mayor cuota de mercado en cuanto a servidores de correo y web, y es uno de los principales mercados, donde tiene una posición dominante. Los sistemas GNU/Linux ofrecen soluciones de código abierto para correo y web y conforman uno de sus principales usos.

- ✓ **XWindow:** un caso particular de interconexión es el sistema gráfico de los sistemas GNU/Linux (y la mayor parte de UNIX), X Window. Este sistema permite una transparencia total de red y funciona bajo modelos cliente servidor; permite que el procesamiento de una aplicación esté desligado de la visualización y de la interacción por medio de dispositivos de entrada, por lo que éstos se sitúan en cualquier parte de la red. Por ejemplo, podemos estar ejecutando una determinada aplicación en una máquina UNIX cuando desde otra visualizamos en pantalla los resultados gráficos, y entramos datos con el teclado y ratón locales de forma remota. Es más, el cliente, llamado cliente X, es tan sólo un componente software que puede ser portado a otros sistemas operativos, permitiendo ejecutar aplicaciones en una máquina UNIX y visualizarlas en cualquier otro sistema. Un caso particular son los llamados terminales X, que son básicamente una especie de terminales "tontos" gráficos que sólo permiten visualizar o interactuar (por teclado y ratón) con una aplicación en ejecución remota.



Arquitectura de Linux

6 CONCLUSIONES

- Hemos introducido conceptos fundamentales de sistemas operativos Unix, en particular, detalles de la administración y organización interna de Linux, para de esta forma tener una visión más cercana al funcionamiento y relaciones entre componentes.

7 BIBLIOGRAFÍA

- Tanenbaum, A. S. (2009). *Sistemas operativos modernos* (tercera ed.). Ciudad de México, México: Pearson.

- Corbet, J.; Rubini, A.; Kroah-Hartman, G. (2005). Linux Device Drivers (3.a ed.). O'Reilly

- Bulma. "Bulma Linux User Group". Documentación general y comunidades de usuarios.

- https://juncotic.com/gnulinux-arquitectura-basica-del-sistema/
- http://openaccess.uoc.edu/webapps/o2/bitstream/10609/61286/1/Administraci%C3%B3n%20de%20sistemas%20GNU_Linux_M%C3%B3dulo1_Introducci%C3%B3n%20al%20sistema%20operativo%20GNU_Linux.pdf