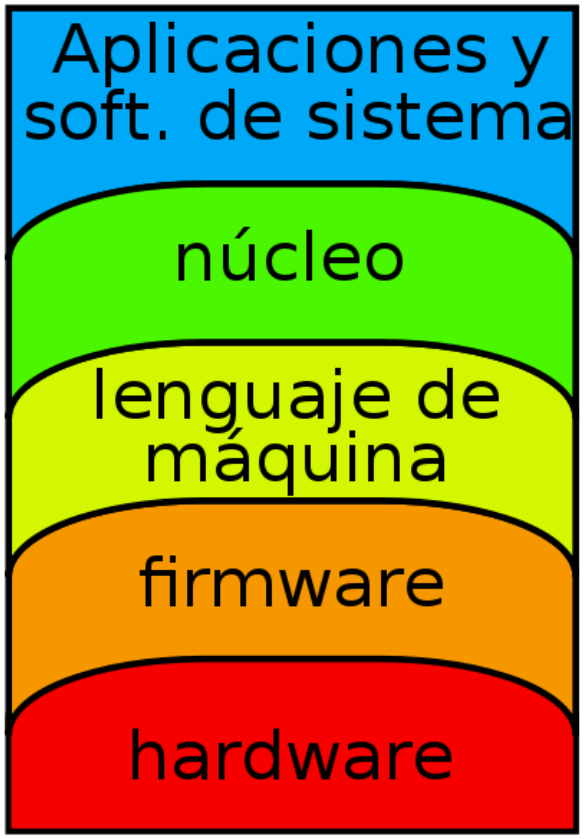
KERNEL o NÚCLEO

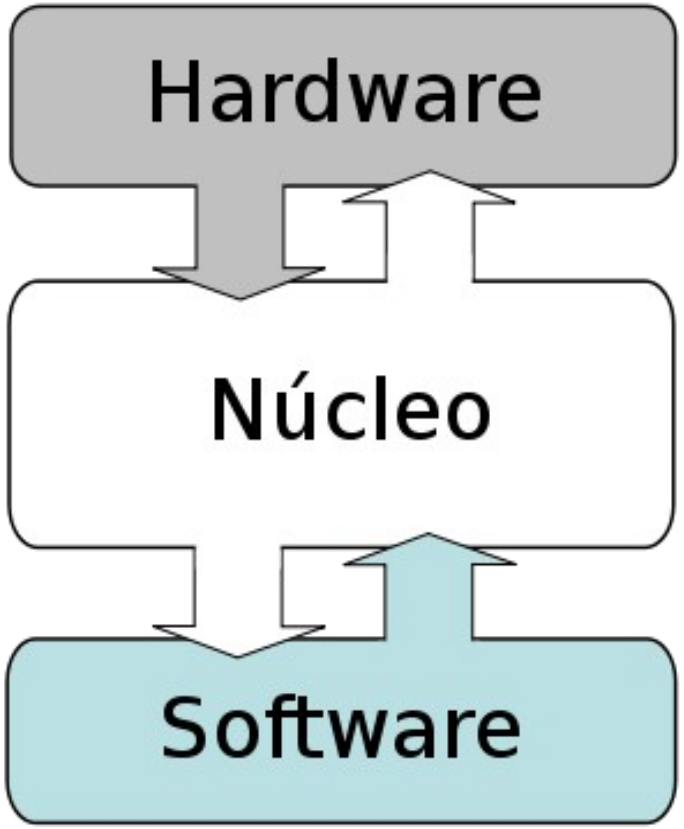
Definición

- Parte fundamental de un sistema operativo
- Es el ***software*** responsable de facilitar a los distintos programas ***acceso seguro al hardware***, es el encargado de gestionar recursos a través de servicios de ***llamada al sistema***
- Los núcleos implementan una serie de ***abstracciones del hardware***. Permitiendo ocultar la complejidad, proporcionando una interfaz limpia y uniforme al hardware subyacente, lo que **facilita su empleo**.
- Es la parte que se carga primero y permanece en memoria principal

Capas de abstracción



Interacción Software, Núcleo y Hardware



Tipos de kernel

- Los kernels **monolíticos** facilitan abstracciones del hardware subyacente realmente potentes y variadas
- Los **microkernels** proporcionan un pequeño conjunto de abstracciones simples del hardware y usan las aplicaciones llamadas servidores para ofrecer mayor funcionalidad
- Los kernels **híbridos** son muy parecidos a los microkernels puros, excepto porque incluyen código adicional en el espacio de kernel para que se ejecute más rápidamente.
- Los **exokernels** no facilitan ninguna abstracción, pero permiten el uso de bibliotecas que proporcionan mayor funcionalidad gracias al acceso casi directo al hardware.

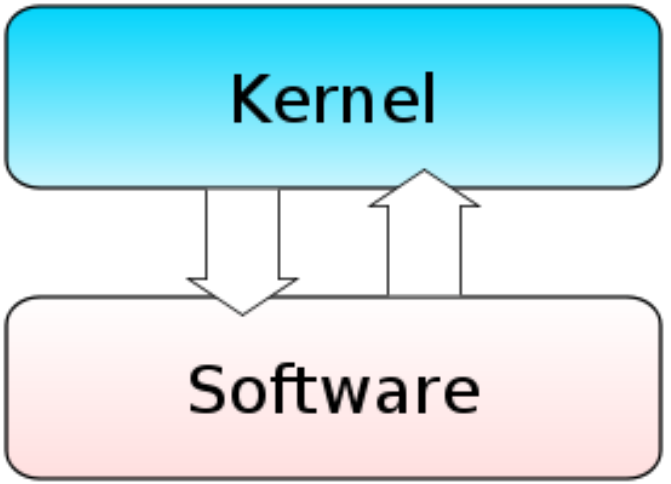
Kernel Monolítico

- ***"El gran embrollo"***
  Núcleo grande y complejo, engloba todos los servicios del sistema.
  Programado de forma no modular.
  Cualquier cambio a realizar en los servicios requiere la recompilación del núcleo y el reinicio del sistema para aplicar los nuevos cambios.

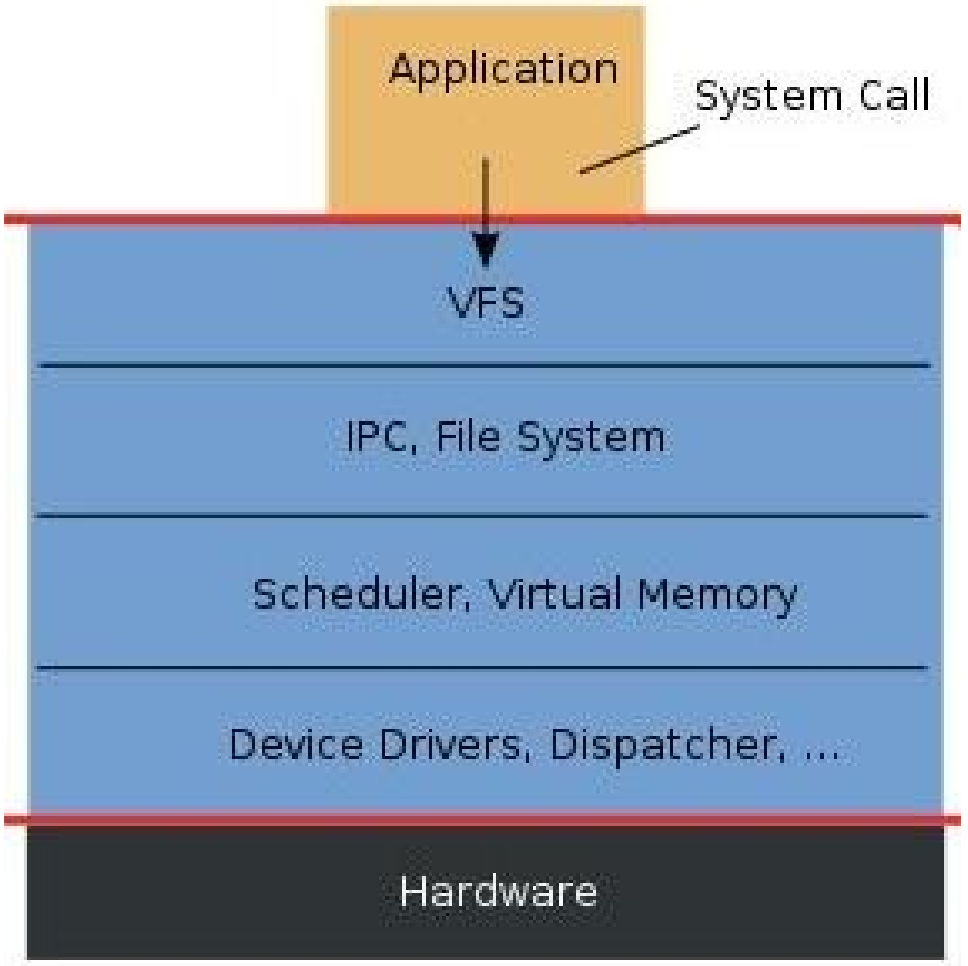
- Todo el sistema operativo trabaja en *kernel space* y en *modo supervisor*.

- Define por si mismo una interfaz virtual de alto nivel (API) sobre el hardware subyacente, mediante un conjunto de *llamadas al sistema* para implementar todos los servicios del sistema operativo

- La mayoría de los "sistemas operativos monolíticos" modernos pueden cargar (y descargar) dinámicamente **módulos ejecutables** en *runtime*.
  Mejor que bootear una nueva imagen de kernel cada vez



Monolithic Kernel based Operating System



Kernel Monolítico: Ejemplos

Unix-like kernels

- Linux (http://en.wikipedia.org/wiki/Linux)
- BSD (FreeBSD, NetBSD, OpenBSD)
- Solaris and OpenSolaris
- AIX
- Multics (http://en.wikipedia.org/wiki/Multics)

DOS

- DR-DOS
- MS-DOS

- **Microsoft Windows** 9x series (95, 98, Windows 98SE, Me)
- Mac OS kernel, up to Mac OS 8.6
- OpenVMS
- XTS-400

Kernel Monolítico: Caso MULTICS

- **Multi**plexed Information and **C**omputing **S**ervice
- Fue un sistema operativo de tiempo compartido escrito en PL/I
  http://en.wikipedia.org/wiki/PL/I
- Se inició como proyecto de investigación en el MIT, General Electric y Bell Labs
  - La investigación comenzó en 1964 a cargo del profesor Fernando J. Corbató
  - En 1969 comenzó a utilizarse como sistema proveedor de servicios académicos para todo el campus del MIT
  - Su concepción fue como producto comercial – Sistema 7x24
  - Fue diseñado para las *utilites* (electricidad y telefonía), tuvo numerosas características para proveer alta disponibilidad y seguridad – Apróx. 85 sites
  - Las investigaciones terminarion en los 70, el desarrollo en 1985, el soporte en 1988
  - La última instalación se dio de baja en el año 2000
  - En noviembre de 2007 MIT libera el código de MULTICS

Kernel Monolítico: Caso MULTICS

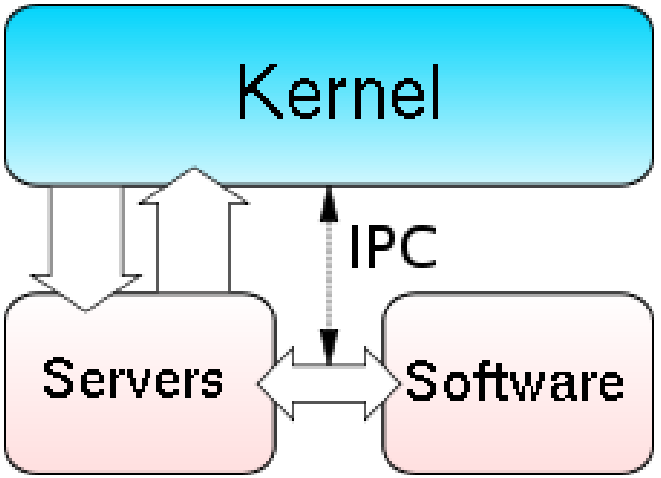
- Características:
  - Implementación en lenguaje de alto nivel
  - Reconfiguración online (CPUs, memoria, controladoras I/O, discos)
  - Memoria virtual con segmentos y páginas
  - Primer sistema de archivos jerárquico
  - Linkeo dinámico y llamadas a función por nombre
  - Seguridad y anillos (rings)

- Unix (Unics) ¿=? hacks de Multics
  - Denis Ritchie y Ken Thompson trabajaban en el proyecto MULTICS en Bell Labs
  - En 1969 MULTICS pasa de Bell Labs a GE (comienza el proyecto Unix)

Microkernel (nanokernel, picokernel)

- Define una abstracción muy simple del hardware, con un conjunto de llamadas al sistema que implementan servicios mínimos de sistema.

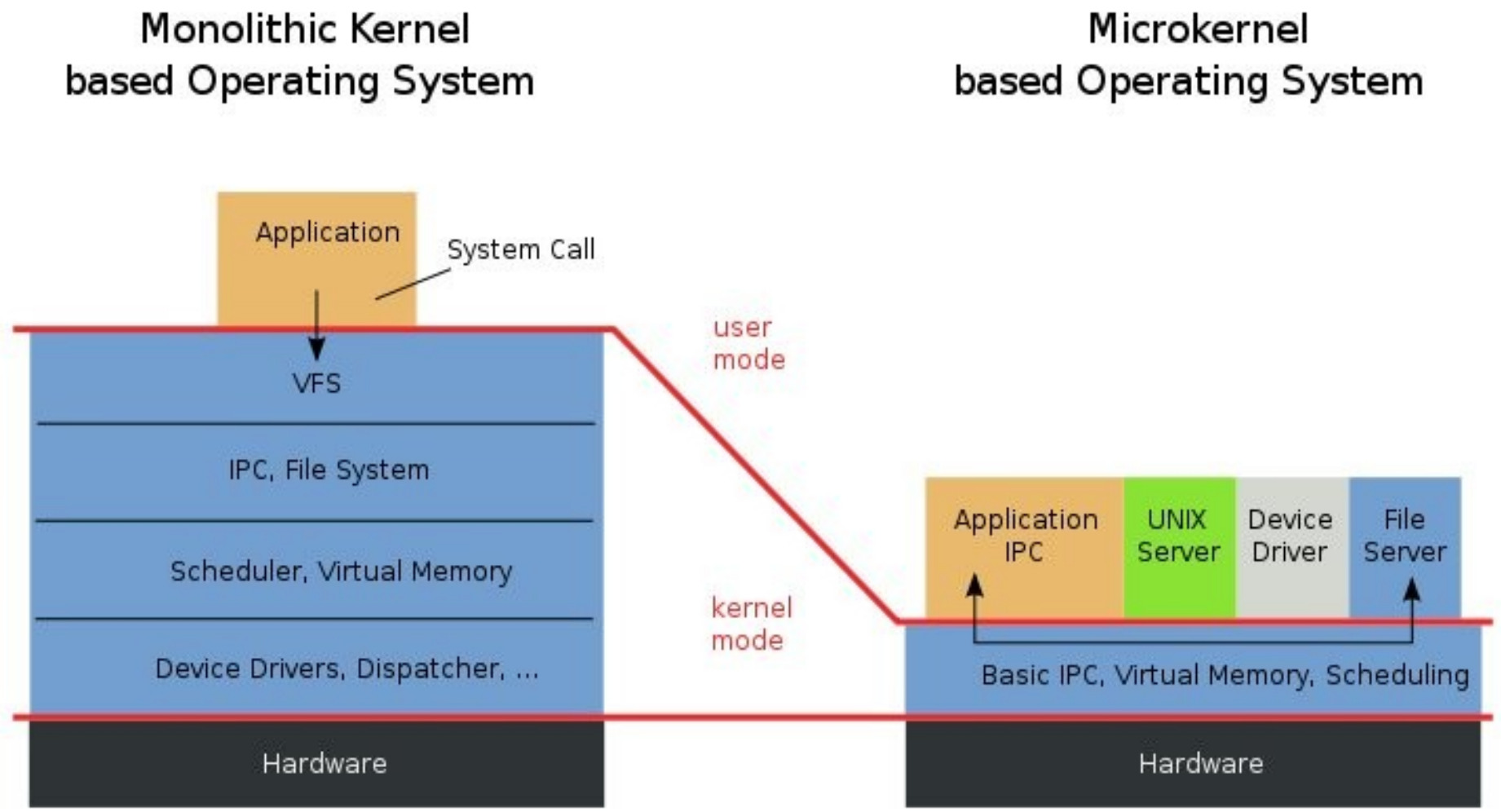
- Debe permitir la construcción arbitraria de servicios sobre él proveyendo:
  - *mecanismos para administrar la memoria*
  - *mecanismos para administrar el uso de CPU*
  - *comunicación entre procesos (IPC)*



- Si el hardware provee múltiples niveles de privilegio, el microkernel es el único software ejecutándose en el nivel más privilegiado
  Los servicios reales del sistema operativo, como device drivers, stacks de protocolos, file systems, etc. están contenidos en espacio de usuario.

- Este paradigma tuvo una gran relevancia académica durante los 80 y principios de los 90

- **Microkernel** (nanokernel, picokernel)



Micokernel

- **Ventajas**
  - administración del código más simple gracias a la división de los servicios en espacio de usuario
  - incrementar la seguridad y estabilidad dado que existe menos código involucrado
  - descentralización de los fallos (un fallo en una parte del sistema no lo colapsaría por completo)

- **Desventajas**
  - complejidad para la sincronización de todos los “módulos” que lo componen (se emplea pasaje de mensajes)
  - acceso de los módulos a la memoria
  - la anulación de las ventajas de Zero Copy (`http://es.wikipedia.org/wiki/Zero-copy`)
  - menor rendimiento (copia de variables que se realiza en la comunicación entre módulos)

Micokernel: Ejemplos

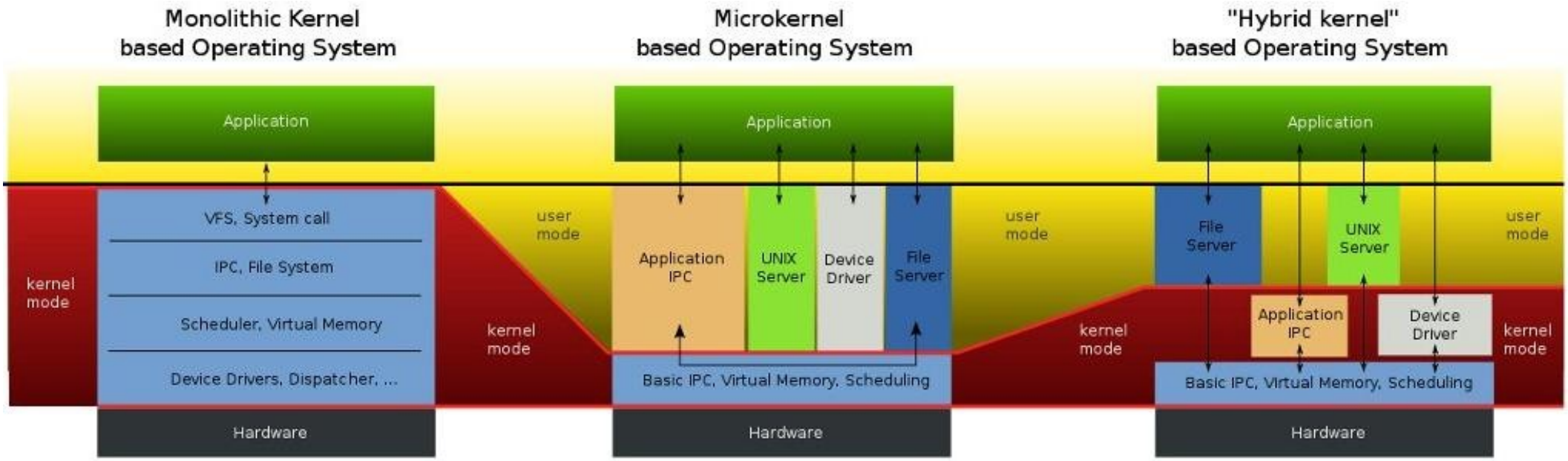
- La familia de micronúcleos L4 (Sistema Operativo Fiasco)
- El micronúcleo Mach, usado en GNU Hurd y en Mac OS X
- BeOS
- Minix
- MorphOS
- QNX
- RadiOS
- Hurd

Webs

- El debate Tanenbaum – Torvalds (1992)
  http://en.wikipedia.org/wiki/Tanenbaum%E2%80%93Torvalds_debate
  http://oreilly.com/catalog/opensources/book/appa.html (mails)
- MINIX
  http://en.wikipedia.org/wiki/Minix
- Mach
  http://en.wikipedia.org/wiki/Mach_(kernel)

Kernel Híbrido

- Arquitectura de kernel basada en los kernels monolíticos y microkernel
- Categoría controversial dada la similitud con los kernels monolíticos, el término fue desestimado por algunos aduciendo que es “simple marketing”



Kernel Híbridos: Ejemplos

- Plan 9 (Inferno)
- ReactOS

Webs

- http://en.wikipedia.org/wiki/Plan_9_from_Bell_Labs
- http://en.wikipedia.org/wiki/ReactOS
- http://en.wikipedia.org/wiki/Syllable_(operating_system)

Exokernel

- Desarrollado con fines de investigación por el grupo de Sistemas Operativos y Paralelos y Distribuidos del MIT

- Reduce su función a la multiplexación segura de los recursos físicos

- En general los diseños de kernel ocultan los recursos de hardware requiriendo que los programas accedan a los mismos a través de algún *modelo conceptual* o *abstracción* (file systems, memoria virtual, schedulers, sockets, etc.)
  Las abstracciones predefinidas facilitan el desarrollo de programas pero limitan la performance y reprimen la experimentación de nuevas abstracciones

- El concepto de exokernel es un compromiso:
  "dejar que el kernel administre los recursos físicos de hardware básicos para múltiples aplicaciones y dejar que las mismas decidan que hacer con estos recursos"

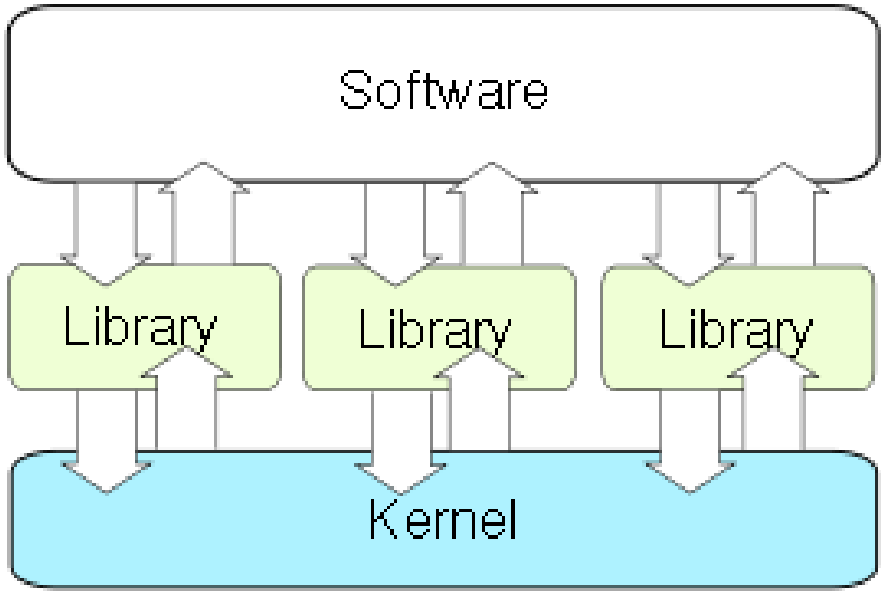
Exokernel

- A pesar de su reducida funcionalidad, la asignación y revocación de recursos está implementada en el mismo núcleo, aunque las *aplicaciones* pueden participar en las políticas de asignación y revocación

- Las aplicaciones son llamadas *library operating systems*; pueden solicitar direcciones de memoria, bloques de disco, etc
  El kernel asegura que el recurso esté libre y que la aplicación tiene permitido accederlo

.

  Este acceso al hardware de bajo nivel permite al programador implementar abstracciones customizadas y omitir aquellas innecesarias y elegir el nivel de abstracción deseado



Exokernel: Ejemplos

- Nemesis
- ExOS

Webs

- The Exokernel Operating System Architecture
  http://pdos.csail.mit.edu/exo/theses/engler/thesis.ps

- Exokernel: An Operating System Architecture for Application-Level Resource Management
  http://delivery.acm.org/10.1145/230000/224076/p251-engler.pdf?key1=224076&key2=2519598511&coll=&dl=ACM&CFID=15151515&CFTOKEN=6184618

- Nemesis at Cambridge An operating system with principles
  http://www.cl.cam.ac.uk/research/srg/netos/old-projects/nemesis/

CASO DE ESTUDIO: Kernel Monolítico 4.4BSD

Notas

- BSD = Berkeley Software Distribution (non-Bell & non-AT&T)
- 4.4BSD fue liberado en Junio de 1994
  4.4BDS-Lite → 4.4BDS-Lite Rel2 → FreeBSD, OpenBSD, NetBSD, DragonFly BSD
- Kernel Code = machine-independent code + machine-dependent code

Organización del kernel

- La mayor parte implementa servicios del sistema que las apps usan a través de llamadas al sistema (características independientes de la plataforma):
  - **Basic kernel facilities**: timer and system-clock handling, descriptor management, and process management
  - **Memory-management support**: paging and swapping
  - **Generic system interfaces**: the I/O, control, and multiplexing operations performed on descriptors
  - **The filesystem**: files, directories, pathname translation, file locking, and I/O buffer management
  - **Terminal-handling support**: the terminal-interface driver and terminal line disciplines
  - **Interprocess-communication facilities**: sockets
  - **Support for network communication**: communication protocols and generic network facilities, such as routing

CASO DE ESTUDIO: Kernel Monolítico 4.4BSD

Organización del kernel

Las características dependientes de la plataforma están aislada del fuente principal. Nada del código independiente de la plataforma depende el hardware subyacente.

Cuando una acción independiente de la plataforma es necesaria, el código dependiente de la plataforma invoca a una función dependiente de la plataforma ubicada en el código dependiente.

- Low-level system-startup actions
- Trap and fault handling
- Low-level manipulation of the run-time context of a process
- Configuration and initialization of hardware devices
- Run-time support for I/O devices

CASO DE ESTUDIO: Kernel Monolítico 4.4BSD

Código independiente de la plataforma

Category	Lines of code	Percentage of kernel
headers	9,393	4.6
initialization	1,107	0.6
kernel facilities	8,793	4.4
generic interfaces	4,782	2.4
interprocess communication	4,540	2.2
terminal handling	3,911	1.9
virtual memory	11,813	5.8
vnode management	7,954	3.9
filesystem naming	6,550	3.2
fast filestore	4,365	2.2
log-structure filestore	4,337	2.1
memory-based filestore	645	0.3
cd9660 filesystem	4,177	2.1
miscellaneous filesystems (10)	12,695	6.3
network filesystem	17,199	8.5
network communication	8,630	4.3
internet protocols	11,984	5.9
ISO protocols	23,924	11.8
X.25 protocols	10,626	5.3
XNS protocols	5,192	2.6
total machine independent	**162,617**	**80.4**

Code:
C
Header files
Assembly

C 98%
Assembly 2%

CASO DE ESTUDIO: Kernel Monolítico 4.4BSD

Código dependiente de la plataforma (HP300)

Category	Lines of code	Percentage of kernel
machine dependent headers	1,562	0.8
device driver headers	3,495	1.7
device driver source	17,506	8.7
virtual memory	3,087	1.5
other machine dependent	6,287	3.1
routines in assembly language	3,014	1.5
HP/UX compatibility	4,683	2.3
total machine dependent	**39,634**	**19.6**

Code:
Sin cotar HP/UX Compatibility y device support

machine-dependent es solo el 6.9 %

http://en.wikipedia.org/wiki/HP_300

CASO DE ESTUDIO: Kernel Monolítico 4.4BSD en HP300

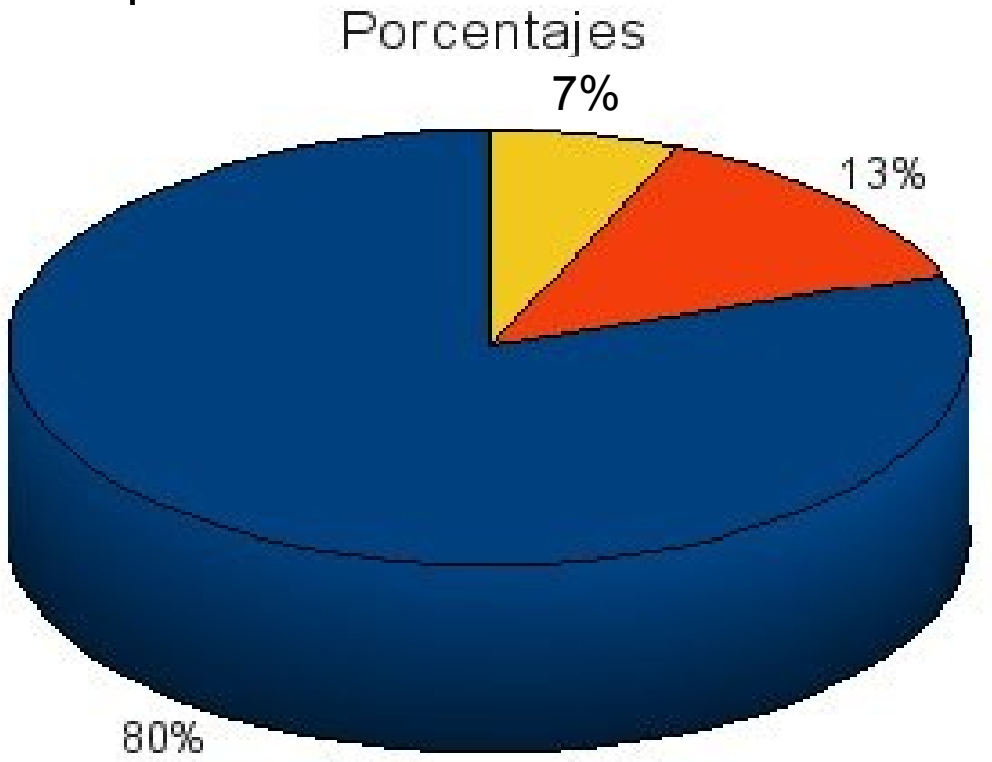
Porcentajes de líneas del kernel

80% independiente de la plataforma

20% dependiente de la plataforma

13% dispositivos y compatibilidad con HP300

7% funcionalidades propias del sistema operativo



Curiosidades:

Estadísticas del Kernel de Linux
http://www.schoenitzer.de/lks/lks_en.html

Donde descargar el Kernel de Linux?
http://www.kernel.org

Discusión Microkernel (Tanembaum) vs Kernel Monolítico (Linus Torvalds) 1992
http://www.dina.dk/~abraham/Linus_vs_Tanenbaum.html
http://en.wikipedia.org/wiki/Tanenbaum%E2%80%93Torvalds_debate