CURSO LINUX

AREA 3: El núcleo

Indice

- Conceptos
- Arquitectura
- Método de desarrollo
- Nomenclatura
- Sistema de módulos
- Compilación del núcleo

Vuelta al principio

25 de agosto de 1991, 20:57:08 GMT comp.os.minix

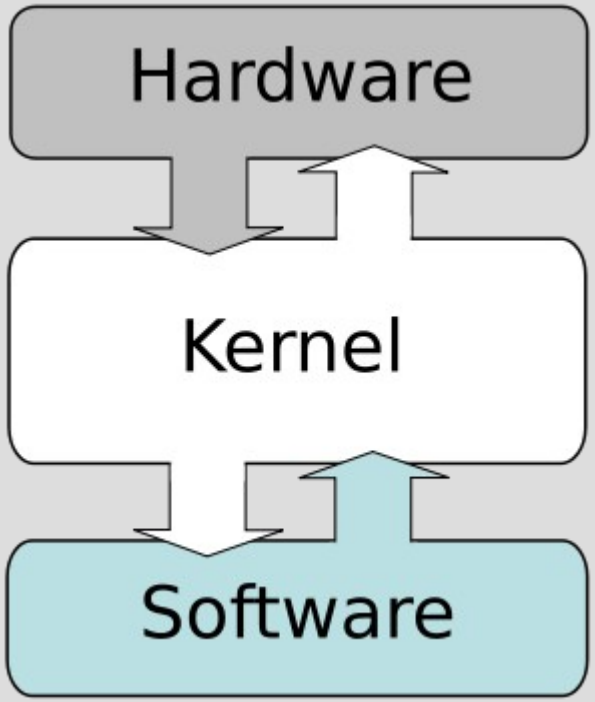
```
Hello everybody out there using minix -

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 Benedict Torvalds (torvalds@kruuna.helsinki.fi)

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

Linux es...

- Linux es el **núcleo** o **kernel** del sistema operativo denominado GNU/Linux
- El kernel es la parte fundamental de un sistema operativo
- Es el software responsable de facilitar a los distintos programas acceso seguro al hardware y responsable de gestionar los recursos.

Kernel

- Funciones:
  - Carga y ejecución de procesos
  - Entrada / salida
  - Interfaz entre el espacio del núcleo y el espacio de usuario



Kernel... Más y más

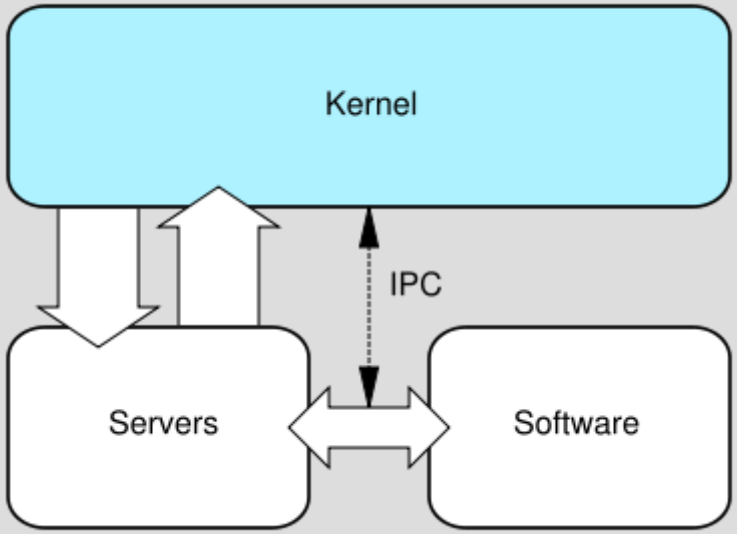
- Otras funciones añadidas al kernel
  - Sistemas de ficheros
  - Funciones de red
  - Servicios
  - Sistemas gráficos
  - ...
- Se añaden al núcleo por rendimiento

Tipos de núcleo

- Micronúcleos : (microkernel)
- Núcleo mixto
- Exonúcleo
- Núcleo monolítico

Micronúcleo

- Abstracción muy simple sobre el hw para:
  - Gestión de hilos
  - Espacio de direccionamiento
  - Comunicación de procesos
- El resto son servidores en espacio de usuario



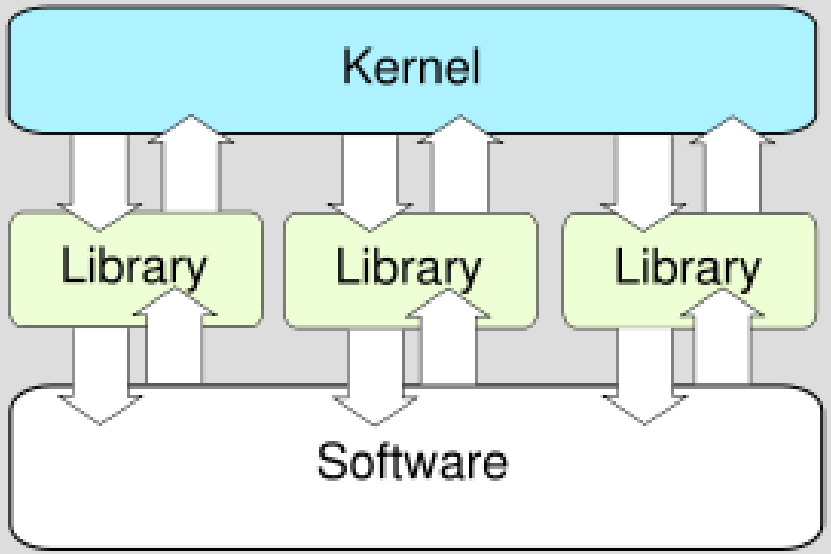
Micronúcleos

Núcleo Mixto

- También denominado híbrido
- Se trata de un microkernel modificado para incluir funciones externas por motivos de rendimiento
- Ejemplos:
  - Windows NT / 2000 / XP
  - XNU
  - DragonFlyBSD
  - ReactOS

Exonúcleos

- El kernel solo controla la protección y el multiplexado
- El resto de funciones se realizan por librerías dinámicas
- En fase de investigación

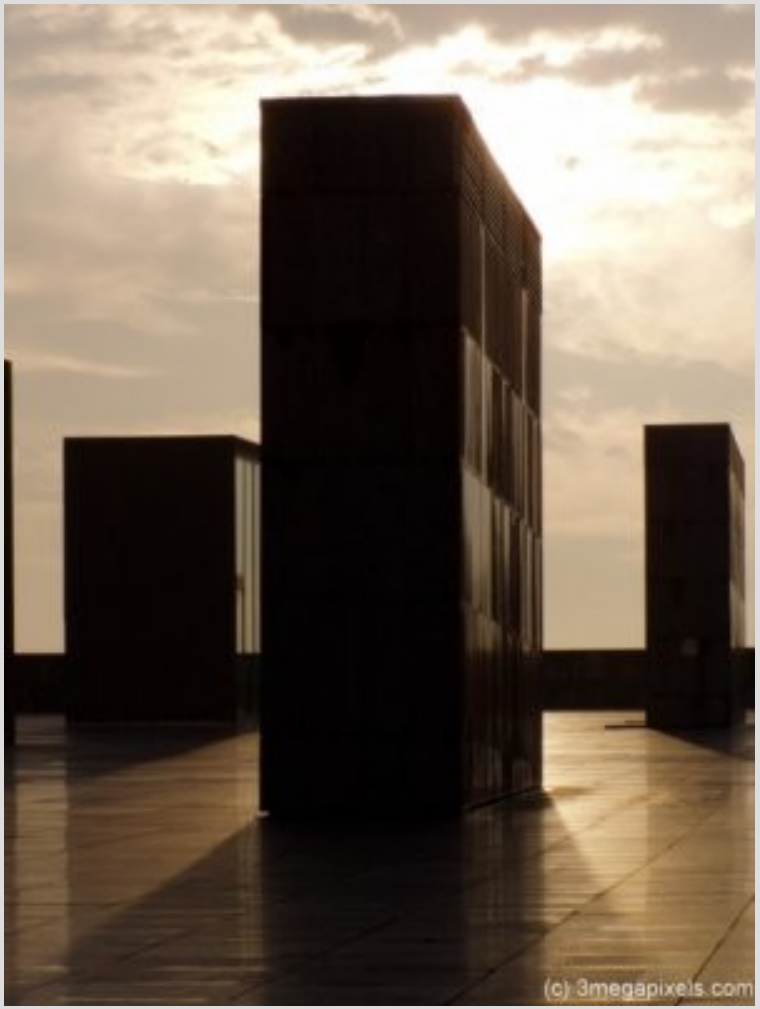


Núcleo monolítico

- El núcleo engloba todos los servicios del sistema
- Tiene mejor rendimiento que el microkernel
- Cualquier cambio a realizar en cualquier servicio del núcleo obliga a recompilarlo entero.
- Existen variaciones que permiten cargar módulos en tiempo de ejecución

Núcleos monolíticos

- Linux
- Unix
  - BSD
  - Solaris
- DOS
- VMS
- MacOS < 8.6

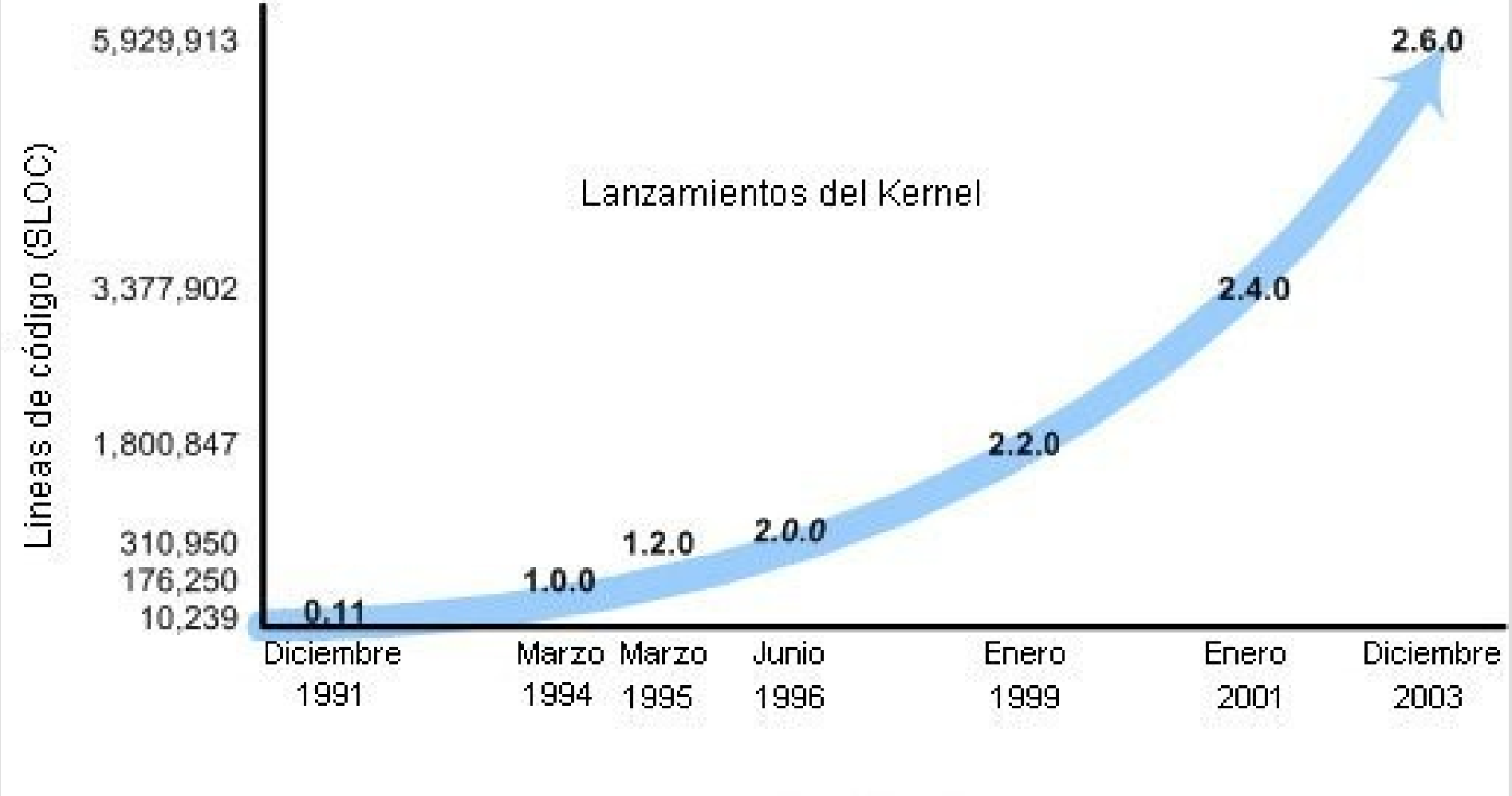


Así pues

- Linux es un núcleo monolítico
- Con módulos cargables en tiempo de ejecución
- Muuuy grande en cantidad de código
- > 6.000.000 líneas de código
- 99% escrito en C
- Open Source
- Y con un pinguino..

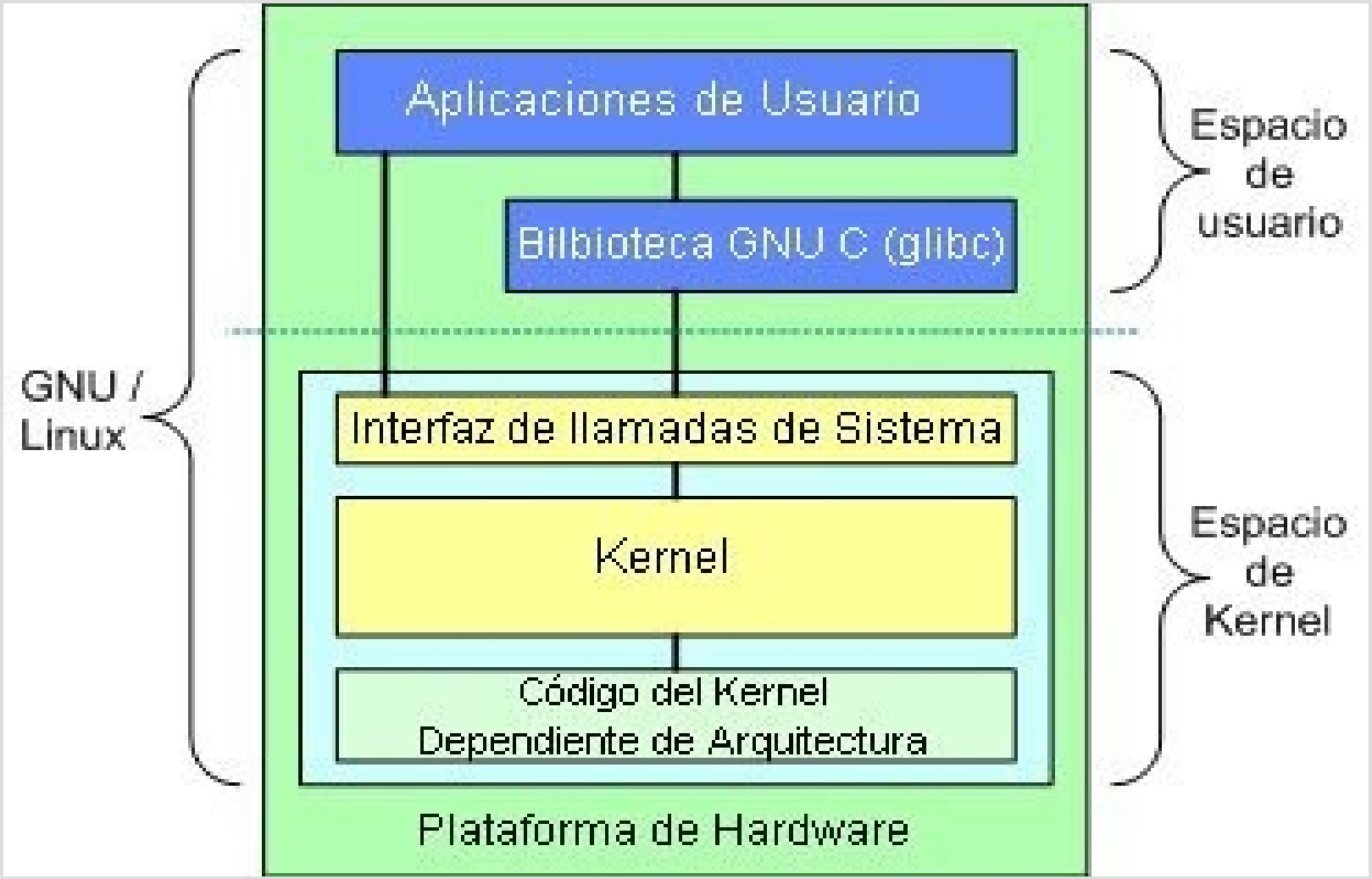
TUX

Evolución Linux

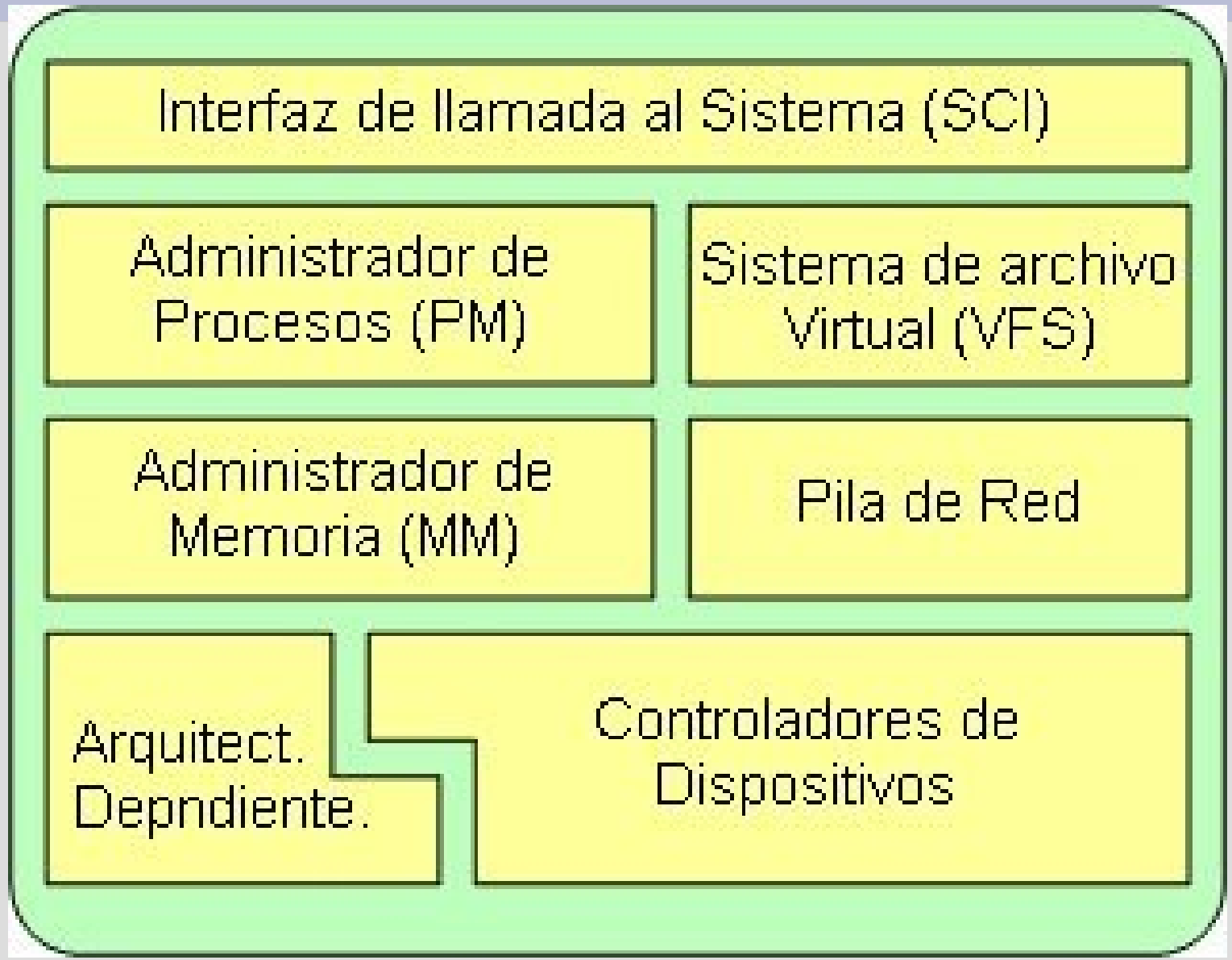


Arquitectura interna

Arquitectura básica



Subsistemas



Interfaz de llamadas del sistema

- La interfaz de llamadas del sistema (SCI) es una pequeña capa que provee el significado para realizar llamadas de funciones del espacio de usuario hacia el núcleo.
- Dependiente de la arquitectura, incluso dentro de la propia familia de procesador.
- Servicio de llamada de función multiplexora y demultiplexora.
- `linux/kernel`

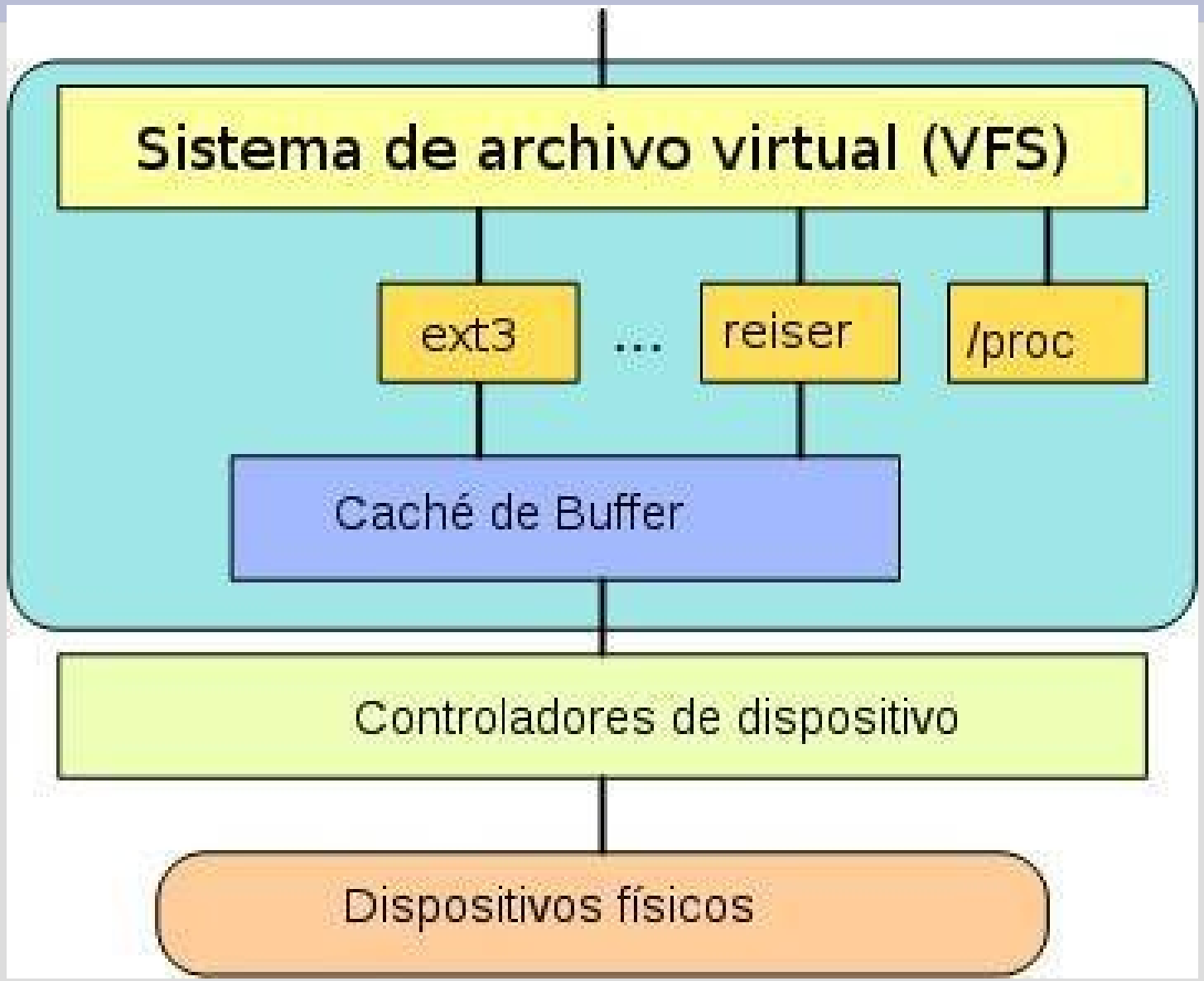
Administrador de procesos

- Se centra en la ejecución de procesos
- Dentro del nucleo se organiza en threads
- Proporciona un API con el SCI para:
  - crear nuevos procesos
  - parar procesos
  - sincronizar procesos
- Implementa el algoritmo de scheduling
- Soporta SMP (Multiproceso simétrico)
- `linux/kernel + linux/arch`

Administrador de memoria

- Otro recurso a compartir (importante)
- Gestiona memoria virtual en páginas de 4K
- Tiene registro de las páginas libres, usadas y llenas
- Puede crecer o disminuir dinámicamente
- Páginas pueden moverse a disco cuando se agota la memoria (swapping)
- `linux/mm`

Sistema de archivo virtual



VFS

- Proporciona una interfaz común a todos los sistemas de archivo soportados por linux
- VFS es una capa de intercambio de datos
- VFS tiene un API para abrir, cerrar, leer, escribir
- Bajo ese API existen plugins para cada sistema de archivo
- Buffer de caché común
- `linux/fs`

Sistemas de archivo

- Estructuran la información guardada en un dispositivo físico
- Separación en bloques (sectores)
- Como mínimo:
  - crear
  - mover
  - renombrar
  - eliminar
- tanto archivos como directorios

Sistemas de archivo

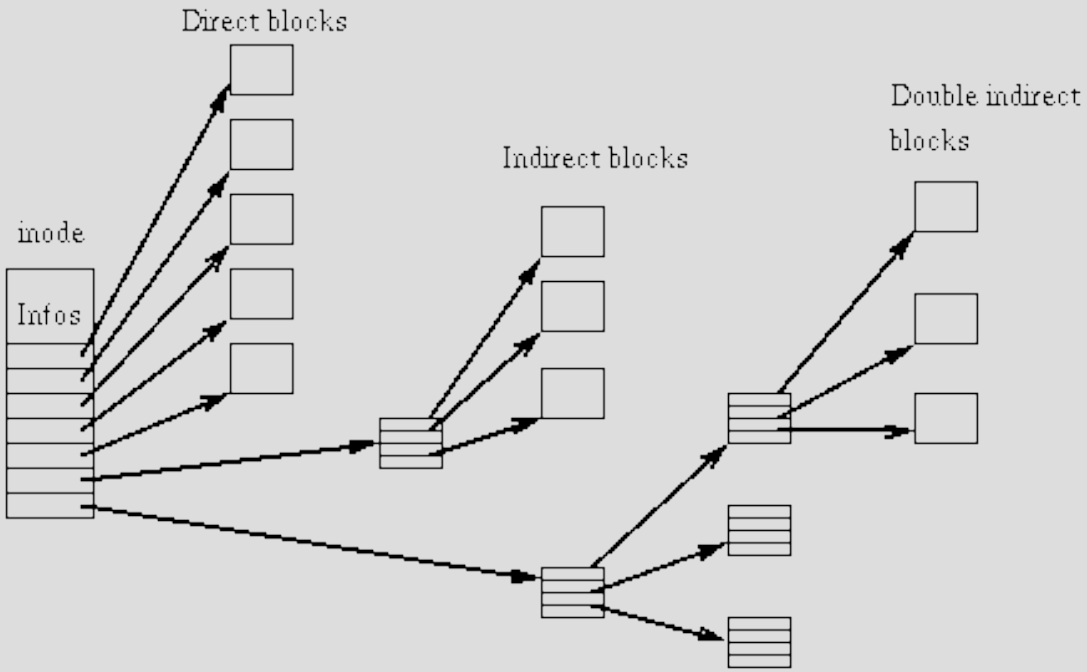
- Funcionalidades adicionales
  - Enlaces (fisicos o virtuales)
- Seguridad de acceso
  - privilegios
  - lista de acceso
- Prevención de segmentación
- Integridad del sistema

Sistemas Linux

- ext2
- ext3
- JFS
- ReiserFS
- XFS
- .. y algunos especiales
- jffs2

ext2

- Second extended filesystem
- Basado en inodos
- Límites: 2TB Archivo – 4TB Partición



ext3

- Sucesor de ext2 y compatible con él
- Utiliza un sistema de journaling
- Permite transacciones
- Es más robusto que ext2
- Permite montarse como ext2, desactivando el journaling
- Tiene peor rendimiento que los siguientes

JFS

- IBM Journaling File System
- FS de 64 bit, con licencia GPL
- Introducido en AIX 3.1
- Eficiente en el sistema de journaling
- Limites: 1PiB Archivo 32PiB Volumen
- (PiB = 20^50 bits)

ReiserFS

- Desarrollado por NameSys (Hans Reiser)
- Incluye journaling
- Será sustituido por Raiser4
- Reparticionado en caliente
- 10-15 veces más rápido que ext2
- Limites: 8 TiB Archivo 16 TiB Volumen
- Algunos problemas (corrupción del arbol, operaciones asíncronas, etc.)

XFS

- Sistema con journaling de SGI
- Limites: 8 exabytes (10^18 bytes)
- Sistema multihilo en operaciones concurrentes
-
- http://oss.sgi.com/projects/xfs/

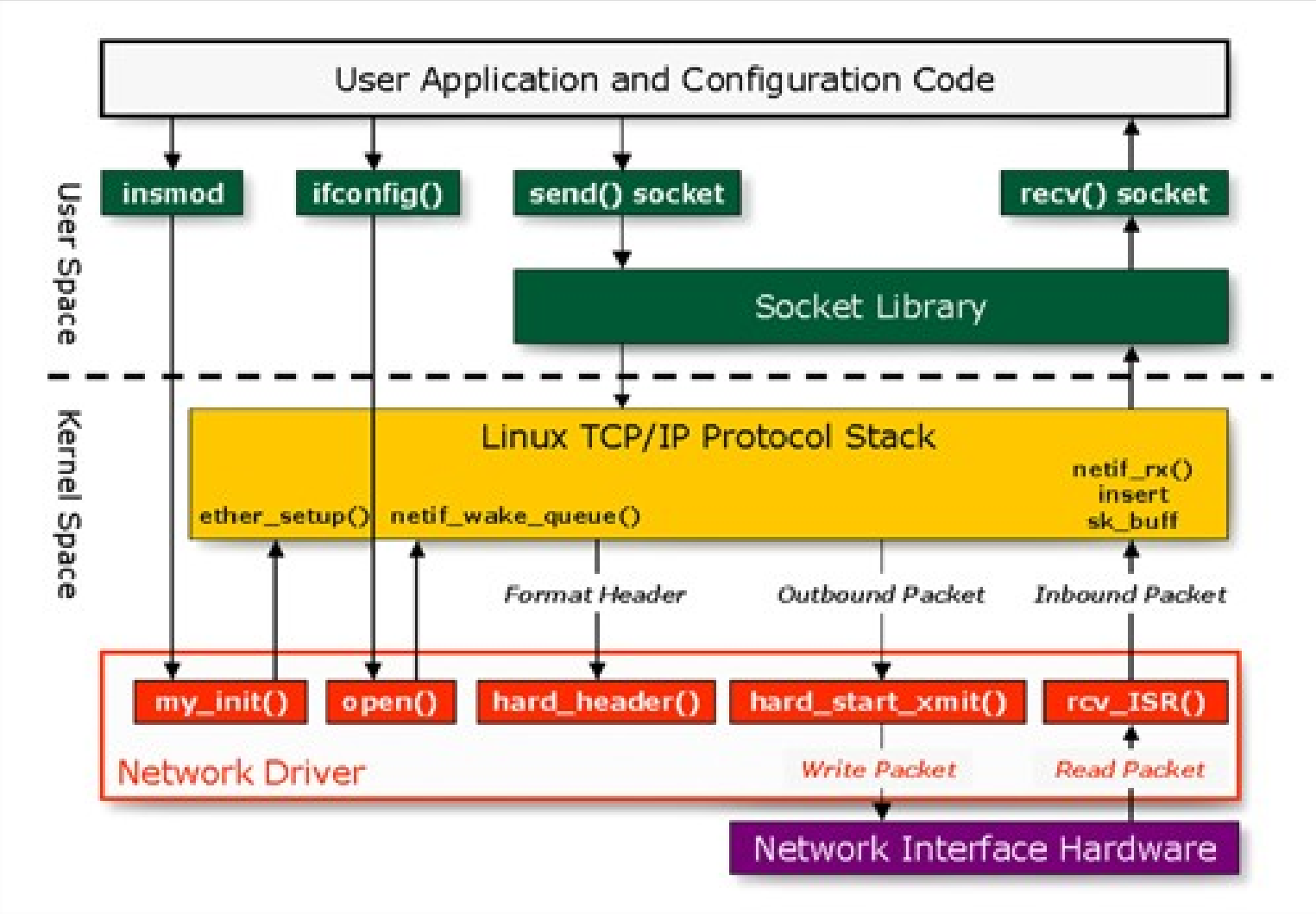
jffs2

- Journaling Flash File System 2
- Especializado en memorias flash
- Con journaling
- Acceso a dispositivos NAND
- Compresión (zlib, rubin, rtime)
- Recolector de basura
- Problemas de rendimiento con > 1Gb

Pila de red

- Core: `net/core`
- IP: `net/ipv4`
- Routing: `net/ipv4`
- Udp: `net/ipv4`
- Forwarding: `net/ipv4`
- Bridging: `net/bridge`
- `linux/net`

Pila de red (2)


User Application and Configuration Code
insmod
ifconfig()
send() socket
recv() socket
User Space
Socket Library
Kernel Space
Linux TCP/IP Protocol Stack
ether_setup()
netif_wake_queue()
netif_rx()
insert
sk_buff
Format Header
Outbound Packet
Inbound Packet
my_init()
open()
hard_header()
hard_start_xmit()
rcv_ISR()
Network Driver
Write Packet
Read Packet
Network Interface Hardware

Drivers

- La mayor parte del código de linux son drivers
- `linux/drivers`
- Generalmente cargados como módulos

```
acorn        clocksource  firmware    leds        mtd         ps3         tc
acpi         connector    hid         lguest      net         rapidio     telephony
amba         cpufreq      hwmon       macintosh   nubus       rtc         uio
ata          cpuidle      i2c         Makefile    of          s390        usb
atm          crypto       ide         mca         oprofile    sbus        video
auxdisplay   dca          ieee1394    md          parisc      scsi        virtio
base         dio          infiniband  media       parport     serial      w1
block        dma          input       message     pci         sh          watchdog
bluetooth    edac         isdn        mfd         pcmcia      sn          xen
cdrom        eisa         Kconfig     misc        pnp         spi         zorro
char         firewire     kvm         mmc         power       ssb
```

Código dependiente arquitectura

- Existe una parte del código que depende de la arquitectura
- `linux/arch`
- Cada subdirectorio de arquitectura contiene los archivos necesarios para que el kernel funcione en ella
- Parte del código en ensamblador .S, mínimo con respecto al c

Arquitecturas soportadas

- x86-based PCs
- Alpha AXP,
- SPARC, UltraSPARC
- Motorola 68000,
- PowerPC / 64
- ARM
- Hitachi SuperH
- Cell
- IBM S/390,
- MIPS,
- HP PA-RISC,
- Intel IA-64,
- DEC VAX
- AXIS CRIS,
- Xtensa, AVR32
- Renesas M32R

Otras características

- Portabilidad
- Eficiencia
- Permite incluir nuevos protocolos fácilmente
- Kernel dinámico (módulos cargables)
- Inclusión de virtualización
- Método de desarrollo muy especial

El método linux

- Modelo de desarrollo abierto
- Muchos desarrolladores “autoasignados” trabajando en paralelo en las áreas de interés
- Unos pocos coordinadores de área
- Un coordinador central (Linus Torvalds)
- Sistema de pruebas muy costoso (diferentes arquitecturas)

El método linux (2)

- Sistema basado en parches (patch)
- Control de versiones mediante git (escrito por torvalds para el nucleo)
- Casi todo el trabajo se realiza por internet (no hay contacto entre desarrolladores)
- Uso intensivo email y listas de correo
- Basado en el respeto mútuo y en netiqueta (no siempre funciona)

Desarrollo del kernel 2.6

- 2.6.x Linus Torvalds
  - Ventana de release 2 semanas -> rc1
  - Nuevo rc cada semana
  - Cuando está maduro se lanza
- 2.6.x.x -stable
  - Correcciones criticas de seguridad y regresiones
  - Ciclo de revisión
- Kernels -mm (Andrew Morton)
- Kernels -git

Nomenclatura

- Actualmente dispone de 4 digitos 2.X.X.X
- El primer dígito es el más importante, solo ha cambiado 2 veces en la historia
- El segundo 2.1, 2.2, 2.4, 2.6 ... Distinguia inicialmente entre pares o impares (estable o testing)
- El tercero es la release concreta y cambia con adiciones importantes release torvalds
- El 4º es el stable, mantenido por otro equipo

Sistema de módulos

- Linux es monolítico
- Era complicado que era disponer de un núcleo nuevo con cada hardware instalado
- Solución: Loadable Kernel Módules (LKM)
- Código a añadir al núcleo en tiempo de ejecución
-

Usos LKM

- Drivers dispositivos
- Driver sistemas de archivos
- Nuevas llamadas al sistema
- Drivers de red (protocolos, etc.)
- Intérpretes ejecutables
- TTY

Utilidades LKM

- `insmod` : inserta un nuevo LKM
- `rmmod` : elimina un LKM del núcleo
- `depmod` : determina interdependencias
- `kerneld` : demonio del kernel para LKM
- `ksyms` : muestra símbolos exportados por el kernel para LKM
- `lsmod` : lista los LKMs cargados
- `modprobe` : inserta o elimina LKMs de manera segura

Ejemplo

- `lsmod`
- Revisar los módulos presentes y sus dependencias
- Mirar también `/proc/modules`
- Los módulos que se pueden cargar están en
  - /lib/modules/<version_kernel>
- Cada vez que cambia el kernel hay que cambiar los módulos asociados

Secuencia de carga de linux

- Gestor de arranque (grub o lilo)
- Especificación del archivo que contiene el kernel
- Especificación del archivo inicial en un disco ram (initrd)

Compilación del núcleo

- Primero, las fuentes
- Nos las bajamos de ftp.kernel.org
- En nuestro caso la versión 2.6.24 (si, los hay más modernos)
- linux-2.6.24.tar.bz2
- Descomprimimos:
  - tar xjvf linux-2.6.24.tar.bz2

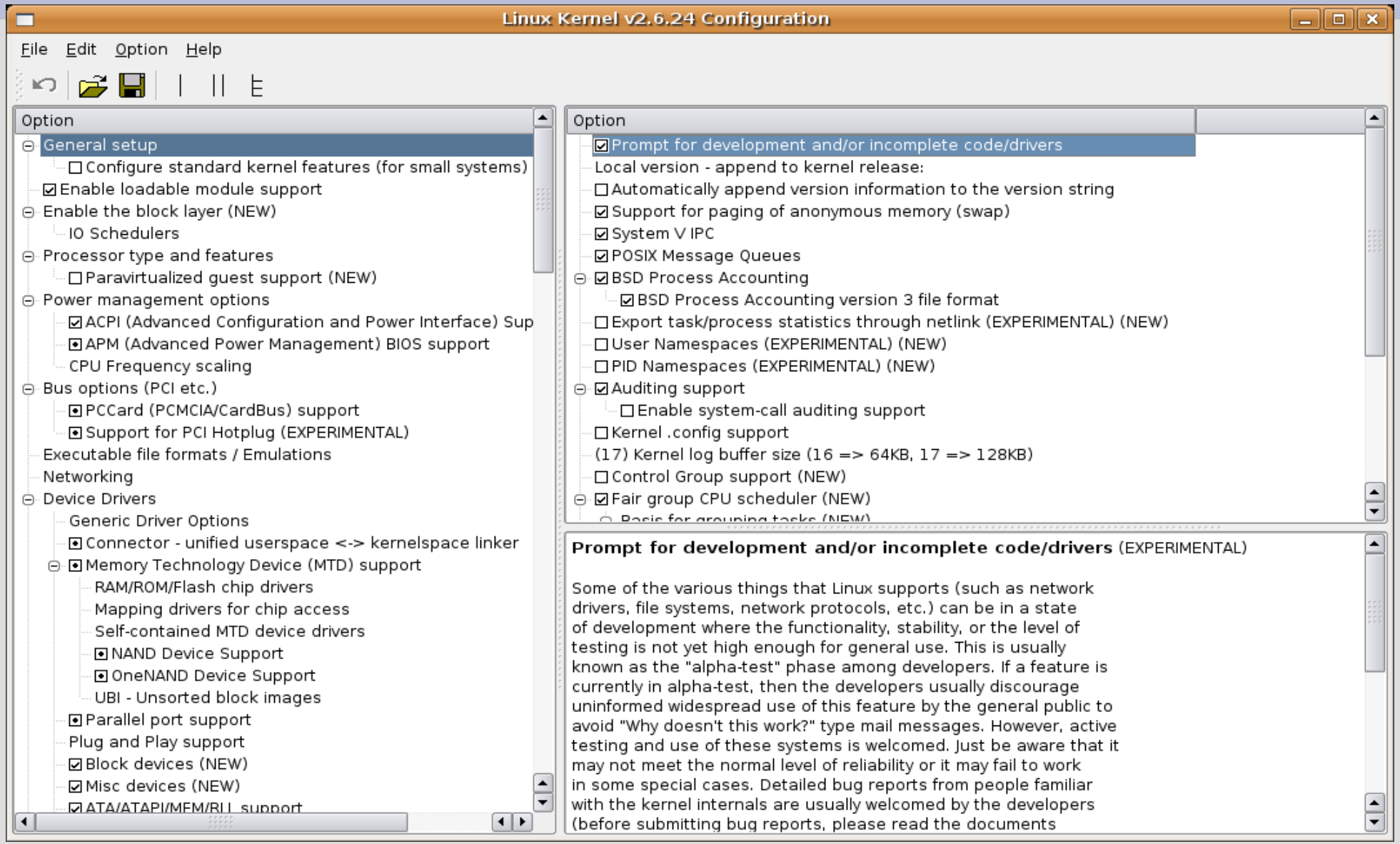
Echemos un vistazo a las fuentes

- Revisad las notas anteriores y mirar dentro de los archivos relevantes
- **NO MODIFIQUEIS NADA**todavía..

Configuremos

- cd linux
- make mrproper
- cp /boot/config-`uname -r` ./.config
- make xmenuconfig (versión gráfica)
- make menuconfig (versión menús texto)

Configuración



¿Cómo descubrir dispositivos?

- lsmod
- lspci
- lsusb (si está instalado)
- ¿documentación?
-

Compilación

- make
- .... (esto tarda un rato)
- make modules_install (asegurarse que es otra versión del nucleo)
- ... (otro ratito)
- Tendremos el núcleo en:
  - `arch/x86/boot/bzImage`

Instalación

- cp System.map /boot/System.map-2.6.24
- cp bzImage /boot/vmlinuz-2.6.24
- mkinitrd -k”vmlinuz-2.6.24” -i”System.map-2.6.24” -M /boot/System.map-2.6.24
- Abrimos YaST
  - Sistema -> Configuracion Cargador Arranque
  - clonar selección
  - Poner nuevos datos
  - Colocar en la 2a posición y revisar

Arrancamos...

- Seleccionamos nuevo kernel
- ... si todo va bien
- Abrimos consola
  - uname -a
- (ya somos los felices usuarios de un nuevo kernel)

Consideraciones

- El tipo de filesystem que tenga nuestra partición de arranque debe ser compilado con en núcleo (o habría que incorporar el módulo en la imagen de arranque)
- Cuantos más elementos se compilen con el núcleo, más grande será este.
- Siempre conservar el núcleo anterior para tener donde volver

Referencias

- jorsol.blogspot.com/2008/02/anatoma-del-kernel-linux.html
- developer.axis.com/old/software/jffs/index.html
- README del nucleo
- www.digitalhermit.com/linux/Kernel-Build-HOWTO.html
- gicl.cs.drexel.edu/people/sevy/network/Linux_network_stack_walkthrough.html
- linux.tar.bz/articles/2.6-development_process
- www.tldp.org/HOWTO/Module-HOWTO/x73.html
- se.uwaterloo.ca/~mctanuan/cs746g/LinuxCA.html