Introducción al Sistema Operativo LINUX

S. Candela

CREACIÓN

- 1969 Ken Thompson y Dennis Ritchie en los laboratorios de Bell **UNIX**

  + características, especificaciones y funcionamiento

- 1987 Andrew S. Tanenbaum **Minix**

  + estructura y código del kernel

- 1991 Linus Torvalds Linus's Unix = **Linux Kernel** (www.kernel.org)
  (kernel actual 1,5 millones de líneas de código)

GNU - GPL

- 1983 Richard Stallman GNU (Gnu's Not Unix) proyecto para generar software libre, (editores Emacs, compiladores gcc, interprete de comandos bsh, sistema operativo Hurd, aplicaciones …) bajo la licencia publica general GPL
- (General Public License), Copy Left, se puede usar, copiar, distribuir y modificar (con las mismas condiciones). Se conserva la firma del autor. Se puede cobrar.
- GNU/Linux (distribución de Linux) = Kernel (1,5 millones de líneas de código) + otras capas del sistema operativo y utilidades (GNU) = LINUX

CARACTERISTICAS DEL KERNEL

- Su código es de libre uso
- Escrito en lenguaje C, compilado con el GNU compilador gcc, primera capa en ensamblador
- Ejecutable en varias plataformas hardware
- Se ejecuta en máquinas con arquitectura de 32 bits y 64 bits
- Estaciones de trabajo y servidores
- Código y funcionamiento escrito bajo la familia de estándares POSIX (Portable Operating System Interface)
- Soporta CPU's con uno o varios microprocesadores (SMP) symmetric multiprocessing
- Multitarea
- Multiusuario

CARACTERISTICAS DEL KERNEL

- Gestión y protección de memoria
- Memoria virtual
- Varios sistemas de ficheros
- Comunicación entre procesos (señales, pipes, IPC, sockets)
- Librerías compartidas y dinámicas
- Permite trabajar en red TCP/IP
- Soporte grafico para interfase con el usuario
- Estable, veloz, completo y rendimiento aceptable
- Funcionalmente es muy parecido a UNIX
- Mas de 100.000 usuarios
- Actualizado, mejorado, mantenido y ampliado por la comunidad de usuarios (modelo “bazar”, contrapuesto al modelo “catedral”)

ARQUITECTURA DEL KERNEL

OBJETIVOS DE DISEÑO

- CLARO
- COMPATIBLE
- PORTABLE
- ROBUSTO
- SEGURO
- VELOZ

ARQUITECTURA DEL KERNEL

CLARO

- Los diseñadores suelen sacrificar claridad por velocidad
- La claridad complementa la robustez del sistema
- Este objetivo esta en contraposición con el objetivo velocidad
- Facilita la realización de cambios y mejoras
- Reglas de estilo fichero /usr/src/Documentation/CodingStyle

CARACTERISTICAS DEL KERNEL

COMPATIBLE

- Soporta ejecutar ficheros Java
- Ejecuta aplicaciones DOS, con el emulador DOSEMU
- Ejecuta algunas aplicaciones Windows a través del proyecto WINE
- Compatibilidad de ficheros Windows a través de los servicios SAMBA
- Soporta varios sistemas de ficheros
  ext2 y ext3 (sistemas de ficheros nativos)
- ISO-9660 usado sobre los CDROMs
- Diseñado bajo la normativa POSIX
- MSDOS

CARACTERISTICAS DEL KERNEL

COMPATIBLE

- NFS (Network File System)
- Soporta protocolo de redes TCP/IP
- Soporta protocolo de AppleTalk (Macintosh)
- Protocolos de Novell IPX (Internetwork Packet Exange)
- SPX (Sequenced Packet Exange)
- NCP (NetWare Core Protocol)
- IPv6 la nueva versión de IP
- Compatibilidad con una variedad de dispositivos hardware

CARACTERISTICAS DEL KERNEL

MODULAR

- El kernel define una interfaz abstracta a los subsistemas
- VFS (Virtual File System Interface),permite implementar nuevos sistemas de ficheros
- Interfase abstracta para manejadores binarios
- Permite soportar nuevos formatos de ficheros ejecutables como Java

- PORTABLE

- Debido a la separación entre código fuente dependiente de la arquitectura y código fuente independiente de la arquitectura
- Capacidad de ejecutar Linux en diversas plataformas
- Intel, Alpha, Motorola, Mips, PowerPC, Sparc, Macintoshes, Sun, etc
- Portatiles, y PAD's

CARACTERISTICAS DEL KERNEL

ROBUSTO Y SEGURO

- Fuentes abiertos permite a la comunidad de usuarios modificar los errores detectados y mantenerlo actualizado.
- Mecanismos de protección para los programas y usuarios
- Cortafuego para protección de intrusos.

VELOZ

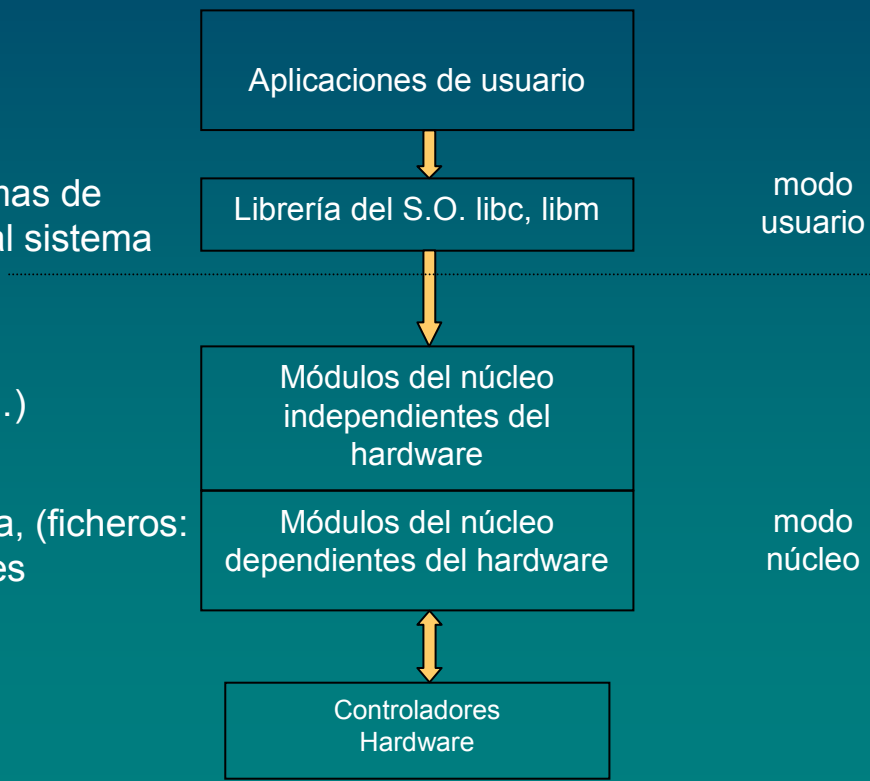
- Objetivo mas demandado por los usuarios
- Código optimizado, primera capa en ensamblador
- No suele ser un objetivo crucial.

VISTA MODULAR DEL KERNEL
Interfase entre los usuarios y el hardware

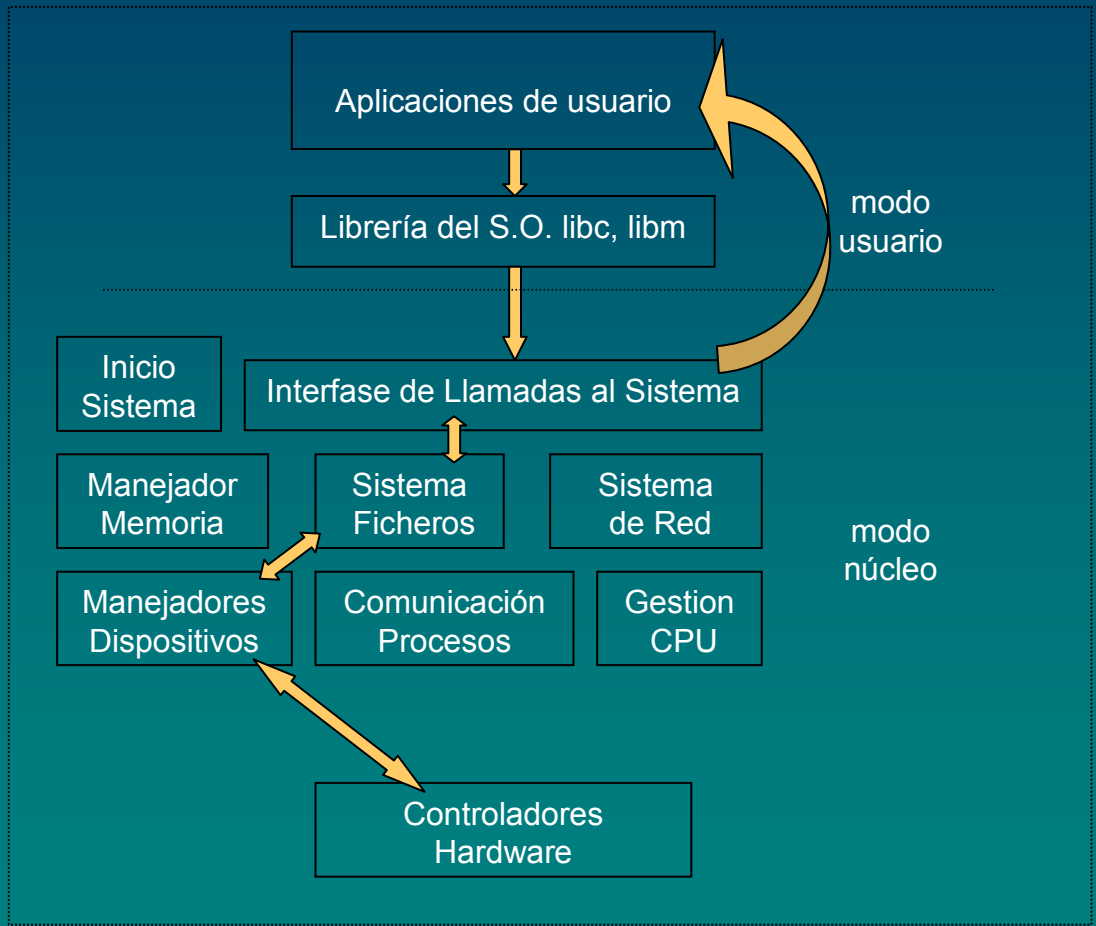
librería del sistema libc:
forma usual de comunicarse los programas de usuarios con el s.o. a través de las llamadas al sistema (system call).

código independiente del hardware (fs, mm, …)

código dependiente del hardware; primera capa, (ficheros: arch/i386/…, include/asm-i386/…); manejadores



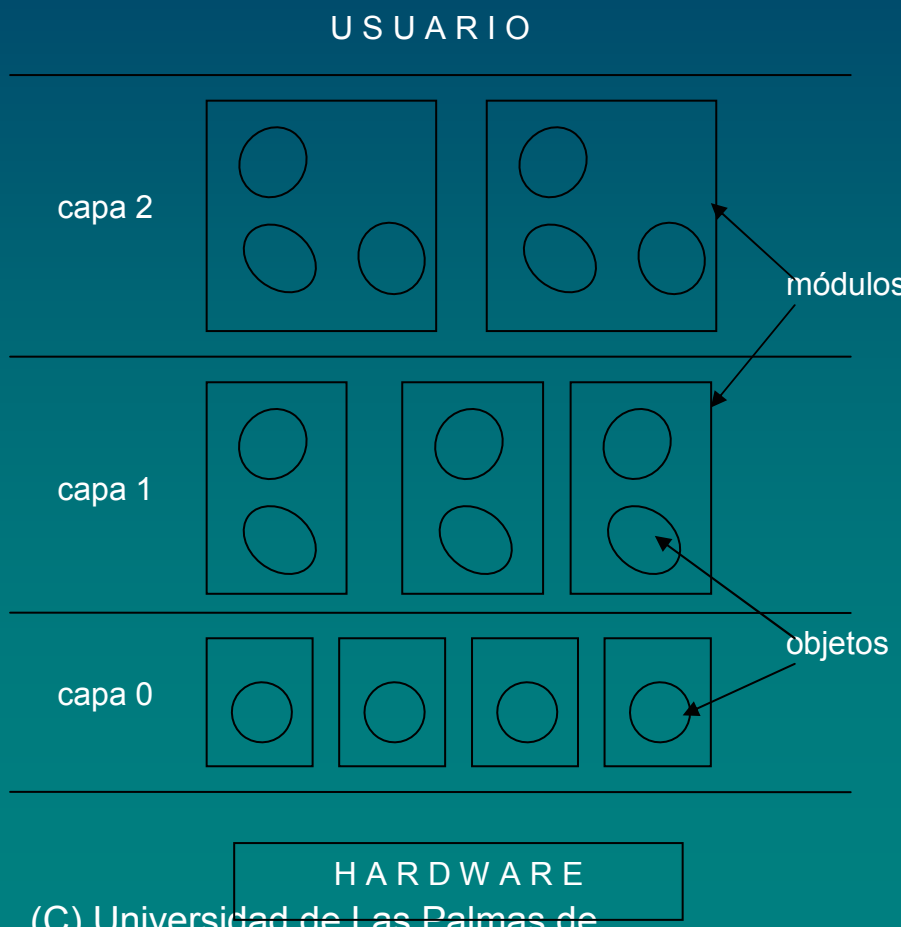
ANATOMIA DEL KERNEL
a nivel funcional



Pasos de una solicitud de un servicio por parte de un usario

1. El usuario en su programa solicita una función de la librería libc.
2. La función de libc, hace una llamada al kernel a través de una llamada (syscall).
3. El kernel recibe esa llamada mediante una función (system call).
4. El kernel redirige esta llamada a los módulos implicados FS, MM, RED
5. Estos llaman a otros hasta llegar al hardware y realizar la solicitud del usuario.
6. El kernel hace llegar a la aplicación del usuario el resultado de su solicitud.

CAPAS, MODULOS, OBJETOS, COMPONENTES



Estructura por Capas

- cada capa proporciona una serie de funciones a las capas superiores.
- las funciones en cada capa se construyen con recursos de la propia capa o con funciones de capas inferiores.
- cada capa tiene su lenguaje de descripción.
- el lenguaje de descripción y uso de los recursos aumenta de nivel a medida que subimos hacia el usuario.
- en el nivel mas bajo el nivel de descripción es el nivel maquina: in, out, lock, …, es el que entienden los controladores hardware.
- en el nivel mas alto o de usuario el lenguaje de descripción se asemeja al lenguaje natural: imprimir, 🖶 icono de una impresora, 🗣 voz “imprimir”.
- se pueden implementar mecanismos de seguridad por capas.

Módulos – Objetos - Componentes

- MÓDULOS

- ofrecen funciones.
- separan interfase de uso de su implementación.
- ocultan a los usuarios aspectos de implementación.
- se pueden cambiar módulos sin afectar a otros módulos o al sistema.
-
- OBJETOS – COMPONENTES

- Es una forma más estructurada de implementar los módulos.
- Se encapsulan estructuras de datos con los procedimientos que las manejan.
- Componentes un refinamiento del concepto de objetos.

Estructura Microkernel vs Monolítica o Macrokernel

ESTRUCTURA DE MICROKERNEL (Minix, Mach)

- El kernel se implementa mediante varios procesos o módulos separados kernel, mm, fs, net.
- Los procesos se ejecutan en modo privilegiado y se comunican mediante mensajes.
- Tiene ventajas en el diseño y en la actualización de un modulo.
- Módulos que no se necesitan no tienen que ser cargados.
- Desventaja, la utilización de un recurso de otro modulo se solicita por mensajes lo que lo hace mas lento.

ESTRUCTURA MONOLÍTICA O MACROKERNEL (Unix)

- El kernel es un único gran proceso.
- La utilización de un procedimiento se llama directamente, no necesita mensajes, por lo que es mas rápido.
- Actualizaciones, implican recompilar todo el kernel.

Estructura de Linux

- Linux mantiene una estructura monolítica, pero admite módulos cargables.
- Linux implementa una estructura por capas con modulos.
- No esta implementada una estructura de objetos o componentes
- Módulos pueden ser manejadores de dispositivos, que se cargan cuando se necesitan.

Recorrido por el Directorio de los Fuentes de Linux.

Los fuentes de Linux, se encuentran en el directorio /usr/src/linux estructurados en varios subdirectorios con estructura jerárquica en arbol.

Aproximadamente 2 millones de líneas de código.

- documentation.
- arch (arch-itecture).
- dirvers.
- fs.
- include.
- init.
- ipc.
- kernel.
- lib.
- mm.
- net.
- scripts.

Fuentes de Linux: documentación, arch

documentation.

documentación relativa a la configuración del núcleo y funcionamiento de los módulos.

arch (arch-itecture)

código dependiente de la arquitectura hardware (400.000 líneas de código, 25 % del total del código) arch/i386, /arch/alpha, arch/m68k, arch/mips, arch/sparc, …

- arch/i386

  código para la arquitectura Intel y compatibles (AMD, Cyrix, IDT) (50.000 líneas de código) contiene los subdirectorios:
- arch/i386/kernel

  manejo de señales, SMP (simetric multiprocesing).
- arch/i386/lib

  librería rápida genérica con funciones como strlen, memcpy.
- arch/i386/mm

  procedimientos del manejador de memoria dependientes de la arquitectura.

Fuentes de Linux: drivers, fs

drivers

manejadores de los dispositivos hardware, (64.000 líneas de código para una arquitectura).

- cha.
- block.
- cdrom.

fs

contienen subdirectorios con todos los sistemas de fichero soportados por Linux.

- ext2 para dispositivos localizados en el equipo.
- NFS para dispositivos accedidos a través de la red.
- proc seudo sistema de ficheros para acceder a variables y estructuras del kernel.

Fuentes de Linux: include

include

contiene ficheros cabecera .h, para la compilación del núcleo y de las aplicaciones y se divide en subdirectorios

- include/asm-*

  existe uno para cada arquitectura, include/asm-i386

  contienen macros para el preprocesador y funciones para la arquitectura, muchas de estas funciones están implementadas en lenguaje ensamblador para mayor rapidez
- include/linux

  básicamente define constantes y estructuras del kernel
- include/net

  ficheros cabecera para el sistema de red
- include/SCSI

  ficheros cabecera para el controlador SCSI
- include/video

  ficheros cabecera para placas de video

Fuentes de Linux: init, ipc, kernel

init

su principal fichero es main.c para la inicialización del kernel.

ipc

herramientas para la comunicación entre procesos norma system V.

kernel

es la capa mas interna del kernel que no depende de la arquitectura.

contiene los procedimientos basicos como, manejo de la cpu, crear y terminar procesos, etc.

Fuentes de Linux: lib, mm, net, scripts

lib

contiene dos partes

- lib/inflate.c

  descomprime y comprime el kernel cuando el sistema esta arrancando
- librería C estandar manejo de string, memoria, etc.

mm

procedimientos y estructuras del manejador de memoria que son independientes de la arquitectura.

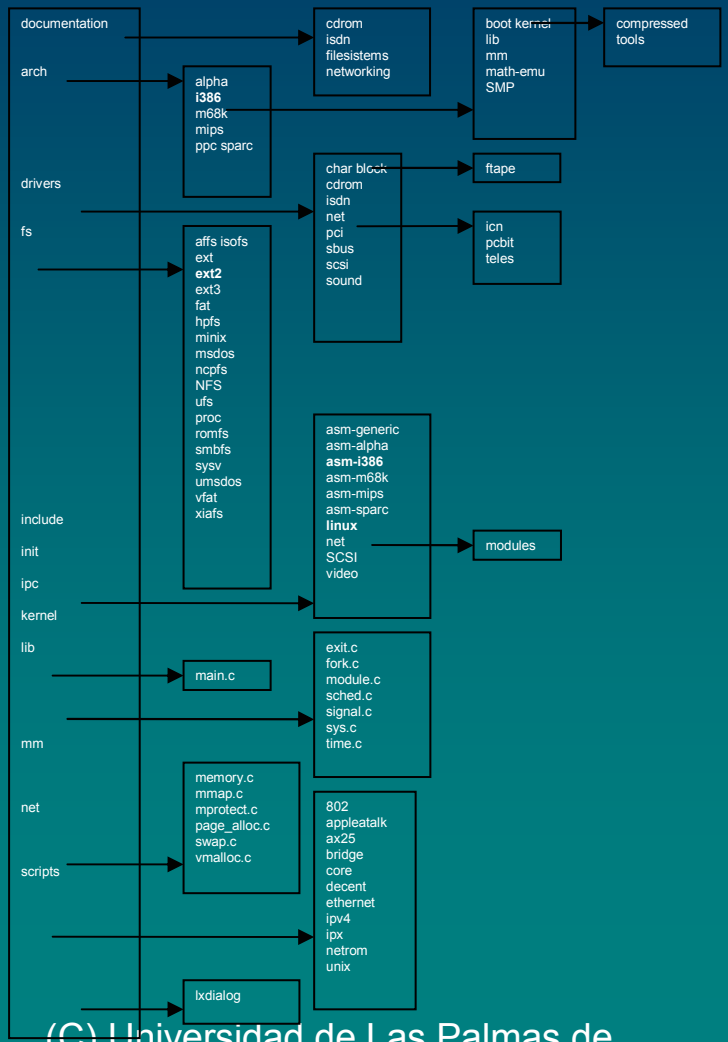
net

procedimientos y estructuras de subsistema de red independiente de la arquitectura con los protocolos de red TCP/IP, IPX, AppleTalk, etc.

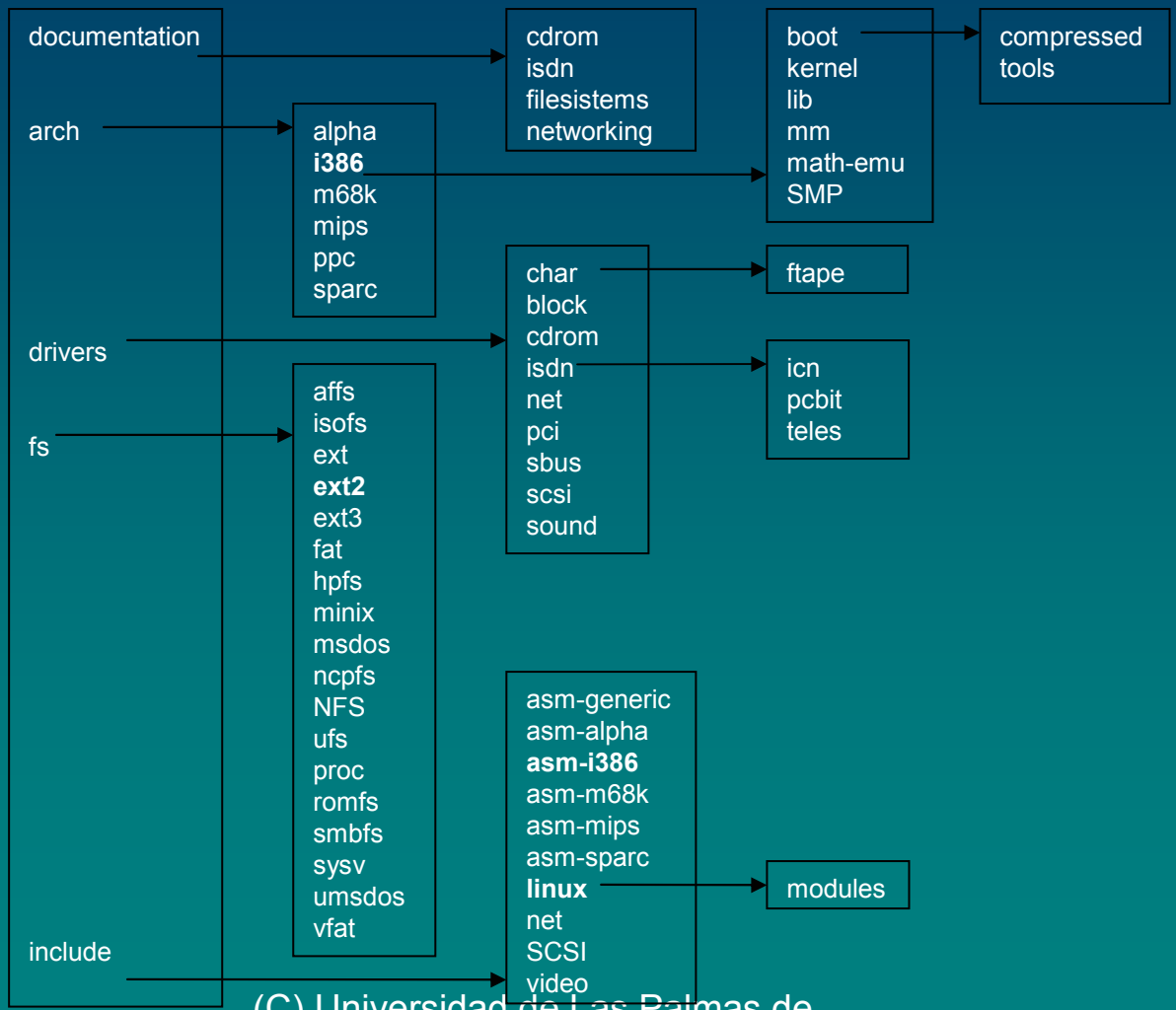
scripts

este subdirectorio no contiene código, contiene scritps para configurar el kernel.

Fuentes: Estructura de directorio



Fuentes: Estructura de directorio



Fuentes: Estructura de directorio