Entendiendo el Kernel de Linux

Jesús Espino García
Samuel Rodríguez Sevilla

Grupo de Usuarios de Linux
Universidad Carlos III de Madrid.

13 de Marzo de 2009

Indice

- Núcleo del sistema operativo GNU/Linux.
- GPL.
- Estable.
- Maduro.
- Moderno.
- Eficiente.
- Versátil.
- Configurable.
- y mucho mas!!

Indice

- En Abril de 1991 Linus Torvals empieza el desarrollo de Linux.
- En Septiembre del 1991 lanza la versión 0.01
- En Diciembre del 1991 libera la primera versión ”self hosted”.
- En Marzo del 1994 aparece la versión 1.0.0.
- En 1996 aparece Tux como mascota de Linux y la versión 2.0 del mismo.
- En 1999 lanza la versión 2.2
- En 2001 lanza la versión 2.4
- En 2003 lanza la versión 2.6 (la actual).

Indice

Hasta la versión 2.5

- El primer numero indicaba la versión de la rama del kernel.
- El segundo indicaba si era estable o no (par estable, impar inestable).
- El tercero era la minor-release de esa rama.

versión 2.6.0 y posteriores

- El primer y segundo numero se han estabilizado (no se esperan cambios a corto plazo).
- El tercer numero indica la versión del kernel.
- Si esta versión esta seguida por -rcX significa que es una versión de desarrollo.
- Si esta versión esta seguida por un .X significa que es estable y que esta en el bugfix X.

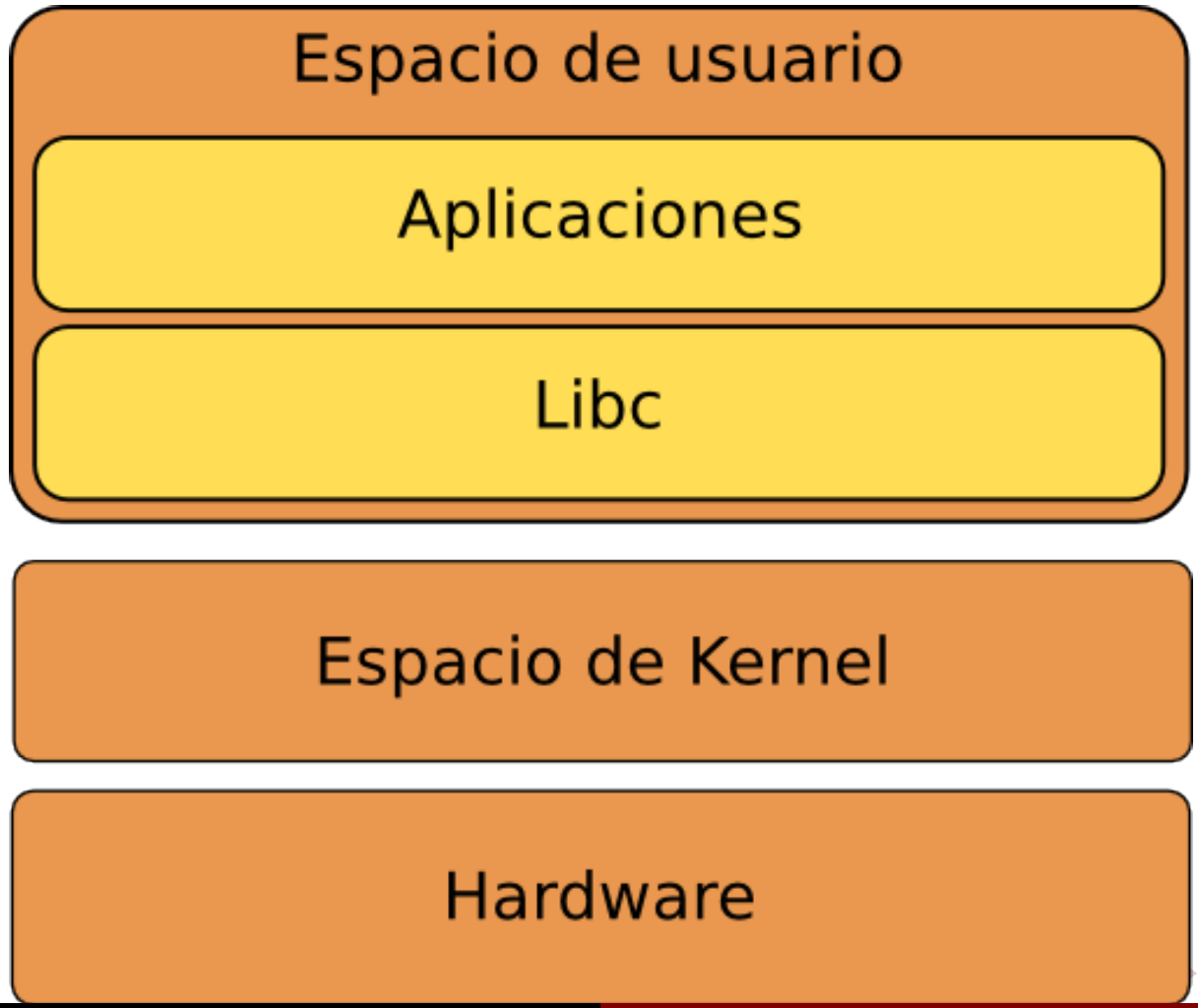
Indice

Git

- Sistema de control de versiones del kernel de Linux.
- Distribuido.
- Desarrollado directamente por Linus Torvals.
- Apareció como una alternativa a BitKeeper.
- Existen muchas ramas desarrolladas de manera distribuida.

Indice

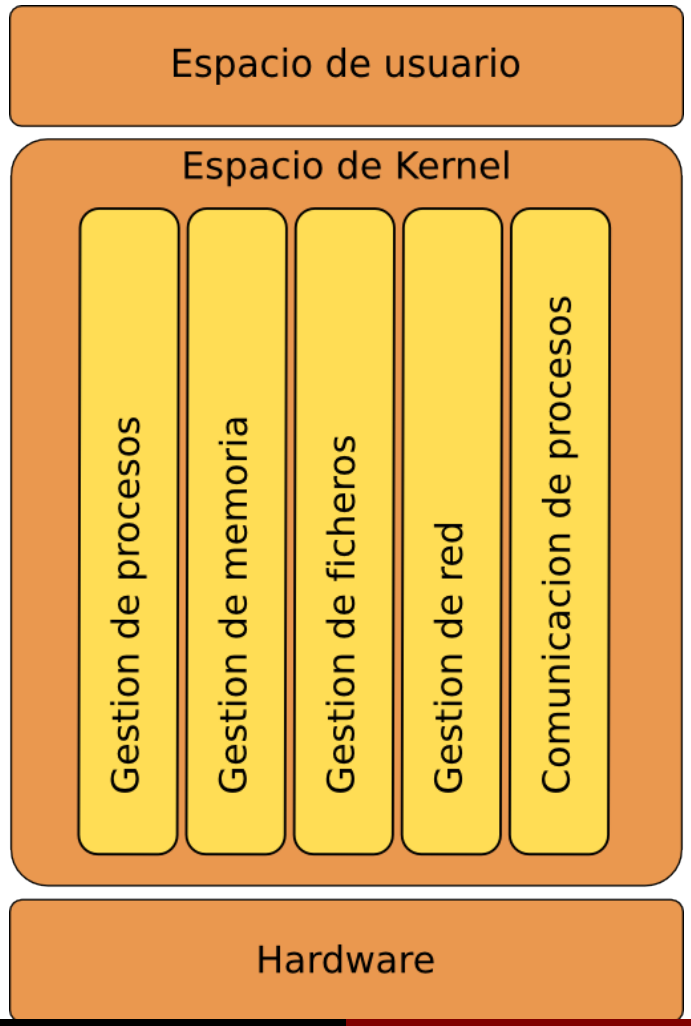
A vista de pájaro



Capas


Espacio de usuario
Espacio de Kernel
Interfaz (Syscalls y ficheros)
Interfaz Virtual
Puente de subsistemas
Logica del kernel
Control de dispositivos
Interfaz hardware (Drivers)
Hardware

Subsistemas



Indice

Tipos de Direcciones

- Dirección lógica: Dirección del segmento mas el offset.
- Dirección lineal (o virtual): Dirección en toda la memoria.
- Dirección física: Dirección de acceso a la RAM.

¿Qué es un segmento?

- Conjunto de memoria.
- Se utiliza para establecer unos permisos a la memoria.
- Se accede a sus contenidos a través de dirección del segmento + offset.
- Esta implementado en hardware.

¿Cómo lo usa Linux?

- Lo usa de un modo muy limitado.
- Se prefiere el uso de páginas.
- Linux crea 4 segmentos que abarca toda la memoria (cada uno).
  - usercode
  - userdata
  - kernelcode
  - kerneldata
- Como todos los segmentos empiezan en 0, la posición se define únicamente por el offset.
- Linux solo usa segmentos cuando lo requiere la arquitectura.

¿Qué son las páginas?

- Conjunto de memoria.
- Se utiliza para establecer unos permisos a la memoria.
- Se accede a su contenido a través de una dirección compuesta de varios elementos.
- Los elementos de la dirección designan posiciones en unas tablas de índices.
- Esta implementado en hardware.

¿Cómo las usa Linux?

- Se usa un modelo con 4 tablas de direccionamiento (para poder albergar mucha memoria):
  - Page Global Directory
  - Page Upper Directory
  - Page Middle Directory
  - Page Table
- Se usa el mismo modelo para 32 y 64 bits.
- Cuando se usa 32 bits y no se usa PAE, Linux elimina las tablas Upper y Middle.
- Cuando se usa 32 bits y se usa PAE, Linux elimina la tabla Upper.

Indice

Buddy System

- Sistema encargado de reservar las páginas y hacer un seguimiento de las páginas libres.
- Soluciona el problema de la fragmentación externa.
- Se almacenan las páginas en 11 listas dependiendo del grupo de páginas contiguas disponibles.
- Linux utiliza varios Buddy System, uno para cada "zona" de memoria (DMA, normal, high).

Slab Allocator

- El Buddy System reserva memoria como mínimo del tamaño de la página.
- El Slab Allocator permite reservar bloques de tamaños arbitrarios mas pequeños que una página.
- Funciona encima del Buddy System.
- Tiene un sistema de caches que permite reutilizar bloques de memoria.
- La reutilización mejora el rendimiento de Linux que utiliza los slab con frecuencia.

Reclamado de páginas

- Al hacer una petición al Buddy System y no encontrar páginas apropiadas, este hace un reclamado.
- El sistema de reclamado divide las páginas en 4 tipos:
  - Unreclaimable: Paginas que no pueden o no es necesario reclamar.
  - Swappable: Paginas que pueden pasarse a la swap.
  - Syncable: Paginas que pueden ser sincronizadas a disco.
  - Discardables: Paginas que pueden ser descargadas.
- Dependiendo del tipo de páginas hace una acción u otra.
- El reclamado de páginas también se ejecuta periódicamente mediante kswapd.

Indice

Tipos de interrupciones

- Excepciones: Interrupciones del propio procesador.
- Interrupciones: Señales (normalmente por IRQ) del hardware.

Indice

IRQs y APIC

IRQ

- IRQ es una señal especial que va directamente a la CPU.
- Tiene su propio canal de comunicación (normalmente incluido en el bus del sistema).
- Este bus permite mandar señales al procesador en cualquier momento.

APCI

- APIC es un sistema de IRQs en 2 niveles (normalmente) para sistemas SMP.
- Permite discriminar el envío de IRQs a un procesador u otro.
- Cada CPU tiene su sistema de APIC local para gestionar sus interrupciones.

Indice

Excepciones

- Indica un error emitido por la CPU.
- Normalmente es una situación anómala producida por la programación.
- Ocurre durante la ejecución del proceso y no hay cambio de contexto.

Indice

Interrupciones

- Indica una solicitud al procesador de que se haga algo.
- Normalmente las interrupciones las producen el hardware.
- Puede ser urgente o no.
- En caso de ser urgente se interrumpe la ejecución y lanza el manejador.
- En caso de no ser urgente se espera al momento idóneo para lanzar el manejador.
- Para manejar una interrupción se produce un cambio de contexto.
- Se pueden emitir interrupciones por software (softirq).
- Las interrupciones pueden ser interrumpidas por otras.

Indice

¿Qué es un proceso?

- Estructura de datos en el SO.
- Proceso completo o hilo.
- Instancia de tarea a realizar por la CPU.
- Utiliza, reserva y comparte recursos.

Tipos de procesos

- Procesos: Instancia de un programa en ejecución.
- Procesos ligeros: Instancia del "contador de programa".

Descriptor de proceso

Información que contiene

- Estado.
- Memoria asignadas.
- Ficheros abiertos.
- Directorio actual.
- Padre.
- Señales recibidas.
- y mucho más.

Estados de un proceso

- TASK_RUNNING: Proceso en la CPU o a espera de CPU.
- TASK_INTERRUPTIBLE: Proceso durmiendo a espera de algo.
- TASK_UNINTERRUPTIBLE: Como TASK_INTERRUPTIBLE, pero sin poder ser interrumpido.
- TASK_STOPPED: Se ha parado la ejecución del proceso.
- TASK_TRACED: Se ha parado la ejecución del proceso por un debugger.
- TASK_ZOMBIE: Se ha terminado el proceso pero no hay ningún wait esperándole.
- TASK_DEAD: El proceso esta siendo eliminado definitivamente.

Cambio de proceso

- Muchos cambios de proceso por segundo.
- Actividad necesariamente rápida.
- Se deben a las interrupciones.
- La interrupción por tiempo permite hacer planificación de procesos.
- Consiste en almacenar el estado del proceso actual y sustituirlo por otro.
- El kernel (casi en su totalidad) es ”reentrant”.

Crear procesos

Procesos

- Proceso 0: El swapper o idle, se crea al inicio de manera ”artesanal”.
- Proceso 1: El init, se crea como copia del proceso 0 y ejecuta el programa init (ya en modo usuario).
- Resto de los procesos: Se crean como copia del proceso padre (normalmente init).

Estrategias

- Copy On Write: Padre e hijo comparten memoria para lectura y en el momento de la escritura se reserva una nueva pagina.
- Procesos ligeros: Padre e hijo comparte memoria.

Destruir procesos

- Recibe la syscall _exit().
- Pasa del estado en que este a TASK_ZOMBIE.
- Su padre hace un wait, y pasa al estado TASK_DEAD.
- Se eliminan todos los datos del proceso.
- Se elimina su descriptor del sistema.
- Los compiladores normalmente incluyen _exit() de manera automática al final del binario.

Hilos del kernel

- Son procesos creados dentro del kernel.
- Algunos se crean al inicio.
- Otros bajo demanda.
- Algunos ejemplos son:
  - kapmd: Gestiona eventos relacionados con APM.
  - kswapd: Reclama memoria periódicamente.
  - pdflush: Vacía los buffers "sucios" a disco.

Indice

Algoritmos

O(1)

- Scheduler hasta el 2.6.22
- Basado en listas de prioridad.
- Calculo del quamto basado en prioridad estática.
- Calculo del sucesor basado en prioridad dinámica.
- La prioridad estática viene dada por el tipo de proceso (normal o tiempo real) y el nice.
- La prioridad dinámica se calcula a partir de la prioridad estática y el "average sleep" del proceso.

Algoritmos

CFS (Completely Fair Scheduler)

- Scheduler desde 2.6.23
- Mas simple
- Reparte el tiempo del procesador de manera equitativa.
- Puntúa mas a los procesos que han usado menos el procesador.
- Penaliza a los que más.
- Intenta que el reparto sea justo.

Indice

Memoria de procesos

- Un proceso usar un conjunto de direcciones (regiones de memoria).
- Cada proceso cuenta con una pila que puede usar para albergar memoria dinámica.
- Cada región de memoria tiene unos permisos de acceso para el proceso.
- Varios procesos pueden acceder a las mismas posiciones de memoria (bibliotecas, shared memory...).
- El intento de acceso a cualquier otra posición de memoria genera una excepción.
- El intento de uso no debido de las posiciones de memoria asignadas genera una excepción.
- Al acceder a su memoria puede ser que no este la pagina (por diversos motivos), por lo cual se produce una demanda de pagina.

Indice

syscalls

- Son el mecanismo que tienen los procesos para comunicarse con el núcleo.
- Todas devuelven un valor de tipo *long* para indicar si ha habido éxito.
- Están identificadas por un valor que utiliza la aplicación cuando le quiere indicar al núcleo qué *syscall* desea utilizar.
- Todas se encuentran en una tabla denominada *sys_call_table* que es dependiente de la arquitectura. La tabla suele estar en el fichero *entry.S* y en los i386 se encuentra en *arch/i386/kernel*.

syscalls

- Se utiliza una interrupción software para indicarle al núcleo que se va a hacer uso de una *syscall*. En la arquitectura i386 se utiliza la interrupción 0x80.
- Las *syscalls* tienen parámetros, que en el caso de la arquitectura i386 se pasan por registro. Si hay más de 5 parámetros, éstos serán pasados a través de un buffer de memoria. Núnca se debe acceder a dicho buffer directamente, se copia con *copy_from_user* a espacio de núcleo y si se le tiene que pasar información se usa *copy_to_user*.

syscalls

El formato de una *syscall* es el siguiente:

```
asmlinkage long sys_nombre(/*parametros*/)
{
   return 0;
}
```

Además se registran en *asm/unistd.h* con un define de formato __NR_nombre. Y con *_syscalln* se crea una función que hará la llamada a la *syscall*.

Indice

Señales

- Las señales son mensajes que se envían a los procesos o grupos de procesos para informarles de ciertos eventos, como alarmas, fallos de segmentación de memoria, etc.
- Es un sistema bastante complejo internamente que requiere de más de 6 tipos diferentes de estructuras.
- Se puede obtener una lista de las señales en *man 7 signal*.
- Hay dos señales que no pueden ser capturadas por los procesos: SIGKILL y SIGSTOP.
- El núcleo divide la gestión de señales en dos partes:
  - Generación de señales, dónde se actualizan las tablas del proceso para representar que se ha enviado la señal, y
  - Despacho de señales, dónde se fuerza al proceso a reaccionar ante la señal.
- El núcleo comprueba constantemente el estado de las señales.

Señales

Los procesos pueden realizar alguna de las siguientes acciones con las señales:

- Ignorarlas
- Ejecutar una de las acciones por defecto, que son:
  - Terminar, que hace que el proceso no siga ejecutándose.
  - Realizar un volcado de memoria (generar un fichero *core*) y luego termina al proceso.
  - Ignorar.
  - Parar, que deja al procesos suspendido hasta que reciba una señal.
  - Continuar, que en caso de que el proceso se encontrase parado lo reanuda.
- Capturar la señal y tratarla con un manejador de señal.

Indice

Dispositivos

- Los dispositivos son todos los elementos de un ordenador que permiten realizar operaciones de entrada/salida. Por ejemplo los discos duros, el teclado, etc.
- Los dispositivos se representan en *sysfs* con un sistema de *kobjects* (se encuentra declarado en *linux/kobject.h*). Estos *kobjects* describen la información sobre un dispositivo para que puede ser fácilmente consultable.
- Para poder acceder a los dispositivos se necesitan los ficheros de dispositivo (típicamente se encuentran en */dev/*). Estos vienen descritos por dos números denominados *major* y *minor* y por su tipo, que puede ser caracter o bloque.

Dispositivos

- El *major* identifica el tipo de dispositivo y el *minor* identifica un dispositivo concreto de ese tipo.
- Los dispositivos permiten realizar al programador una serie de acciones (*open*, *read*, *lseek*, *ioctl*, etc.).
- Los controladores de dispositivo deben estar registrados en el sistema y utilizan un descriptor de tipo *device_driver* que debe encontrarse en dentro de una estructura de tipo *device*.
- Los controladores de dispositivo compilados estáticamente en el núcleo se registran durante el proceso de inicio. El resto deberán llamar a la función de registro que le corresponda (por ejemplo *pci_register_driver*).
- Casi todos los controladores de dispositivo siguen una política de asignación de recursos tardía. De este modo, asignan la memoria cuando se les solicita hacer algo (normalmente por una llamada a *open*) y la liberan cuando ya no es necesaria (no hay ninguna aplicación utilizando sus recursos).

Indice

Arquitectura

Hay dos tipos de dispositivos:

- Dispositivos de caracter: leen y/o escriben un flujo de datos. Su acceso es secuencial. Ejemplos de este tipo de dispositivos son el ratón y el teclado.
- Dispositivos de bloque: leen y/o escriben bloques de datos y permiten el acceso aleatorio a los mismos. Ejemplos de este tipo de dispositivos tenemos los discos duros, DVDs, etc.

Arquitectura

Los dispositivos de bloques entrañan más complejidad.

- Los dispositivos se dividen en sectores.
- Los sectores se agrupan en bloques.
- Los bloques se agrupan en segmentos (un segmento es la cantidad de información que puede mandarse por un canal DMA).
- Los segmentos se agrupan en páginas (pueden tener bloques que no se encuentran en un segmento).

Arquitectura

- Los bloques se leen a memoria en áreas denominadas buffers.
- Los buffers está definidos por la estructura *buffer_head* que se encuentra en *linux/buffer_head.h*.
- Esta estructura controla los datos sobre a qué bloque corresponde el buffer, de qué dispositivo, en qué estado se encuentra, cuantas referencias hay a él, etc.

Arquitectura

- Las operaciones de I/O que se están realizando se encuentran definidas por la estructura *bio* que se encuentra en *linux/bio.h*.
- Esta estructura representa las operaciones sobre páginas.
- La operación de I/O puede afectar a más de una página. Se controla con el array *bio_vec* de *bio*.
- Las operaciones se encolan un una cola de peticiones pendientes.

IO Scheduler

- Las operaciones de I/O no son óptimas en tiempo.
- El sistema operativo planifica estas operaciones para mejorar el rendimiento.
- La planificación se lleva a cabo siguiendo alguna de las siguientes políticas (en el núcleo se denominan *elevators*):
  - *Linus Elevator*
  - *Deadline I/O Scheduler*
  - *Anticipatory I/O Scheduler*
  - *Complete Fair Queuing I/O Scheduler*
  - *Noop I/O Scheduler*

Indice

Indice

Inicio del sistema

- Es el proceso que permite poner en funcionamiento al sistema operativo.
- Comienza por la BIOS, que realiza una comprobación del hardware y pasa el control a un programa que típicamente se encuentra en el sector de inicio del disco duro.
- El cargador de arranque se encarga de cargar el núcleo en memoria y pasarle el control junto con una serie de parámetros (se llama a la función *setup( )* que se encuentra desplazada 0x200 bytes del inicio del núcleo).

Indice

Proceso de inicio

El proceso de arranque realiza las siguientes tareas:

- Se invoca a la BIOS para saber la cantidad de RAM del sistema o, en sistemas ACPI, obtener un mapa físico de la RAM.
- Ajusta la la velocidad del repetición del teclado.
- Inicializa el adaptador de vídeo.
- Reinicia el controlador de disco y obtiene los parámetros de los discos.
- Comprueba que haya un bus IBM Micro Channel.
- Comprueba si hay un ratón PS/2.
- Comprueba si la BIOS soporta APM (*Advanced Power Management*).
- En caso de que la BIOS soporte EDD (*Enhanced Disk Drive Services*), se solicita crear una tabla en memoria con la descripción de los discos duros.

Proceso de inicio

- Según si el núcleo era sin comprimir o comprimido, procede a continuar o desplaza el núcleo en memoria para colocarlo donde le corresponde.
- Se activa el pin A20 en el controlador de teclado 8042.
- Se crea una tabla de descriptores de interrupciones (IDT) provisional y una tabla global de descripción (GDT) provisional.
- Reinicia la unidad de coma flotante (FPU).
- Se reprograma el controlador programable de interrupciones (PIC) para enmascarar todas las interrupciones menos la de IRQ2.
- Se pone la CPU en modo protegido.
- Se llama a la función *startup_32( )*.

Indice

La función startup_32 (I)

- Inicializa los registros de segmentación y una pila provisional.
- Pone a 0 los bits del registro *eflags*.
- Inicializa con ceros el área de datos del núcleo que se encuentran en *_edata* y *_end*.
- Llama a *decompress_kernel( )*.
- Una vez descomprimido el núcleo se llama a un nuevo método *startup_32 ( )*.

La función startup_32 (y II)

El nuevo *startup_32( )* hace las siguiente operaciones:

- Inicializa los registros de segmentación con los valores finales.
- Rellena el segmento *bss* del núcleo con ceros.
- Inicializa la tabla de páginas provisional del núcleo.
- Almacena la dirección del *Page Global Directory* en el registro *cr3*.
- Asigna la pila de modo núcleo al proceso 0.
- Vuelve a poner a 0 el registro *eflags*.
- Llama a *setup_idt( )*.
- Almacena los parámetros obtenidos de la BIOS en el primer *frame* de memoria.
- Identifica el modelo del procesador.
- Carga los registros de la tabla GDT e IDT.
- salta a la función *start_kernel( )*.

Indice

La función start_kernel

Llama a las siguientes funciones:

- *sched_init( ).*
- *build_all_zonelists( ).*
- *page_alloc_init( )* y *mem_init( ).*
- *trap_init( ).*
- *softirq_init( ).*
- *time_init( ).*
- *kmem_cache_init( ).*
- *calibrate_delay( ).*
- *kernel_thread( ).*

Referencias

- `/usr/src/linux/Documentation`: Documentación del kernel.
- http://www.kernelnewbies.org: Pagina para iniciarse en el mundo del kernel de Linux.
- http://www.kernel.org: Pagina oficial del kernel de Linux.
- http://www.makelinux.net/kernel_map: Mapa interactivo del kernel.

Dudas

. . .

Fin