Sistemas Operativos II

Microkernel

Universidad Arturo Jauretche
Ingeniería Informática

Docentes:

Ing. Jorge Osio

Ing. Eduardo Kunysz

Motivaciones

- Se desea poner lo **menos posible** en el Kernel, debido a que los errores en él pueden paralizar el sistema de inmediato.
- Se demostró que (aprox.):

1000 líneas de código ⟶ 10 errores

=> **SO tradicionales:**

5.000.000 líneas de código ⟶ 50.000 errores

La mayoría no son críticos, sin embargo son tantos que los fabricantes ponen botones de **reset**

Definición

- Un **microkernel** es un pequeño núcleo de sistema operativo que proporciona bases para **ampliaciones modulares**.

- Enfoque popularizado por el SO "Mach" y su implementación en computadoras Next.

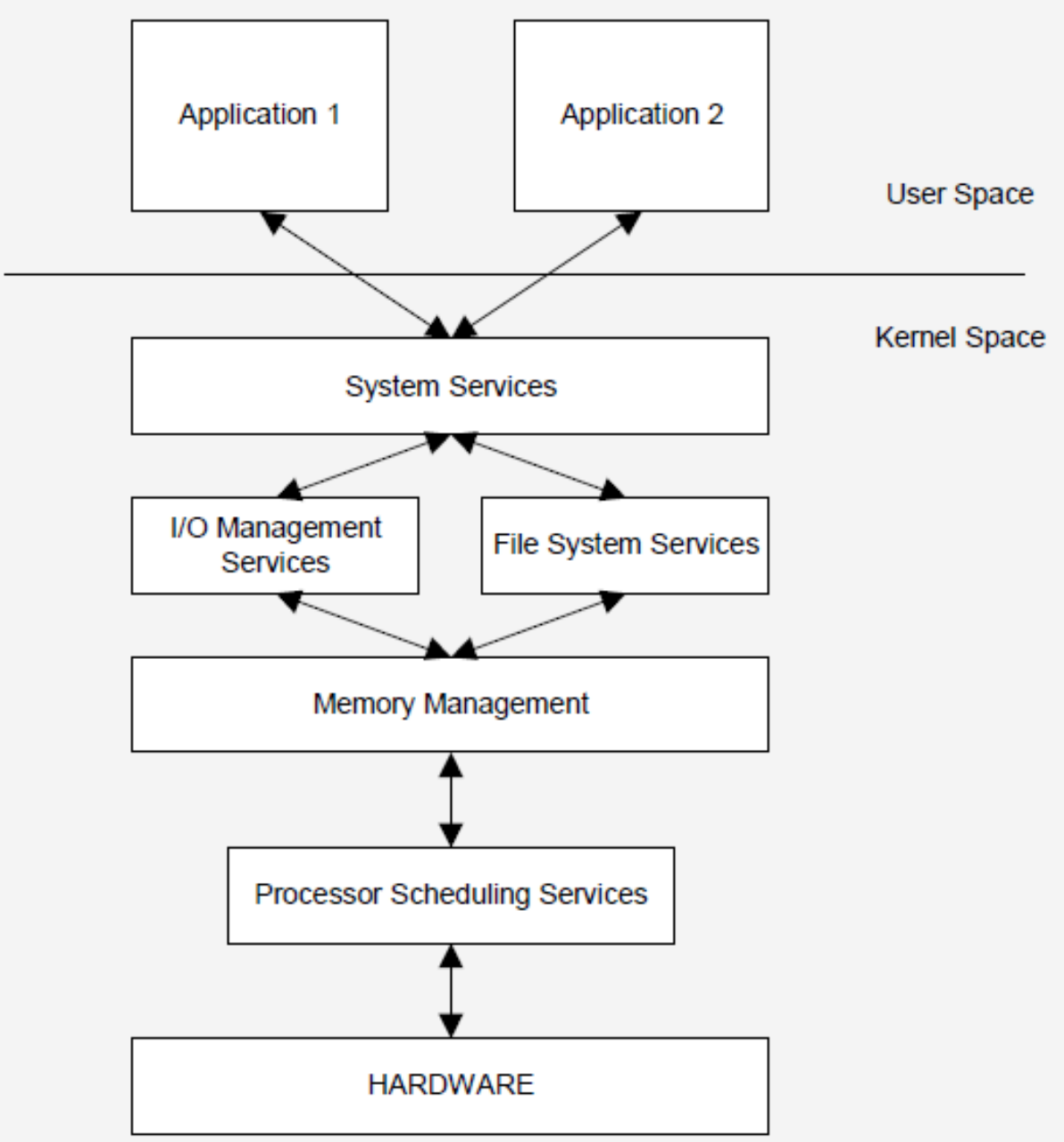
- Se enfoca en brindar una gran **flexibilidad** y modularidad.

- El núcleo esta rodeado por una serie de subsistemas compactos de forma que facilita la tarea de implementar un SO en una variedad de plataformas.

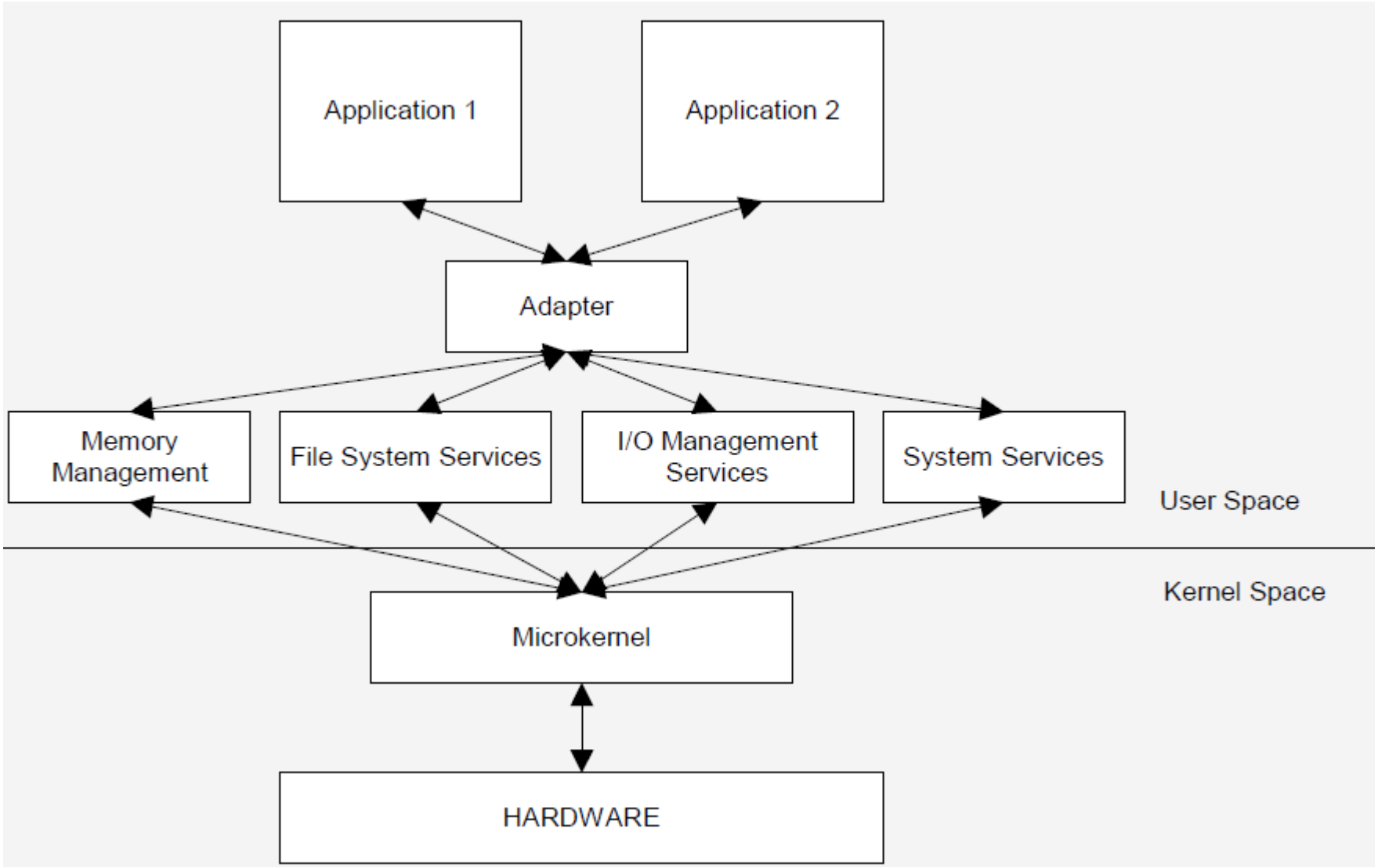
Filosofía

- Solo las funciones **esenciales** del sistema operativo deben permanecer en el núcleo.
- Servicios que tradicionalmente han formado parte del SO ahora son **subsistemas externos** que interactúan con el núcleo.
- Reemplaza la estratificación tradicional de capas verticales de un SO en horizontal.
- Los componentes externos al núcleo interactúan entre si mediante mensajes distribuido a través del núcleo.

Kernel tradicional Monolítico



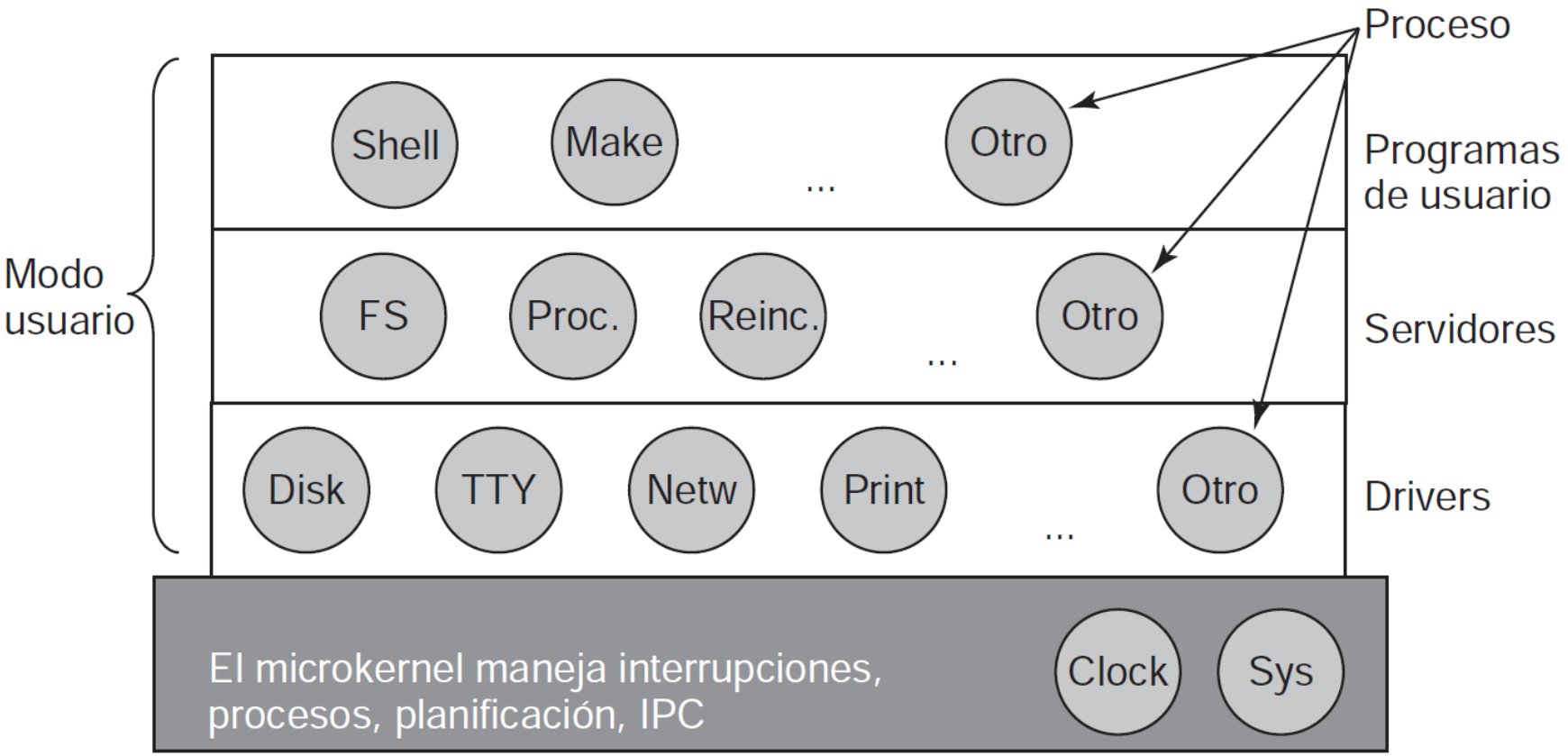
Configuración Microkernel



Prestaciones

- Flexibilidad (puede reiniciar módulos sin reiniciar el SO)
- Baja demanda de memoria: el L4 (Mach) solo necesita aproximadamente 32 Kilobytes de momoria
- Sin embargo un microkernel + un OS regular probablemente utilizará mas memoria que un OS con kernel monolítico.
- Despachos SMP fáciles.

Estructura típica de un microkernel



Aplicaciones

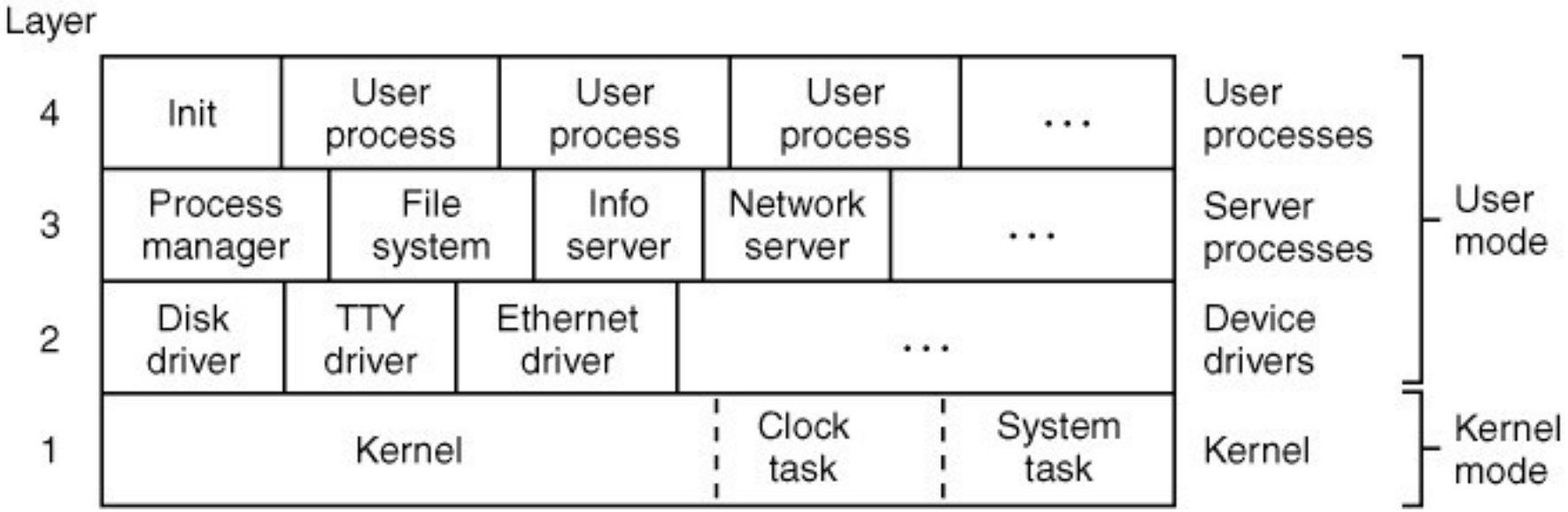
- Aplicaciones de tiempo real, industriales, aeronáuticas y militares (requerimientos de confiabilidad altos).

- Algunos microkernels:
  - Integrity, K 42, L4, PikeOS, QNX, Symbian y MINIX3

MINIX 3

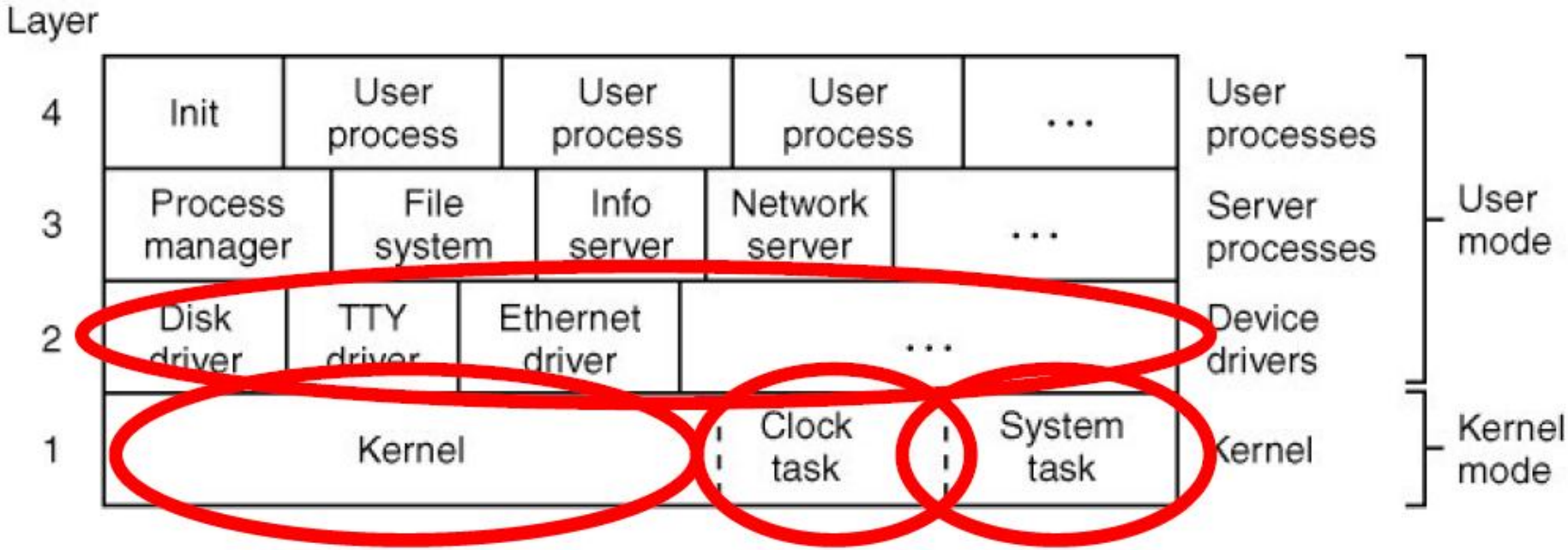
- MINIX 3 es un sistema de código abierto en conformidad con POSIX, disponible en www.minix3.org
- El microkernel tiene:
  - 3200 lineas de C:
    - Administra y planifica los procesos
    - Comunicación entre procesos
    - Ofrece aprox. 35 llamadas al kernel
  - 800 lineas de assembler
    - Funciones de bajo nivel
    - Atrapar interrupciones
    - Conmutar procesos

Estructura de MINIX



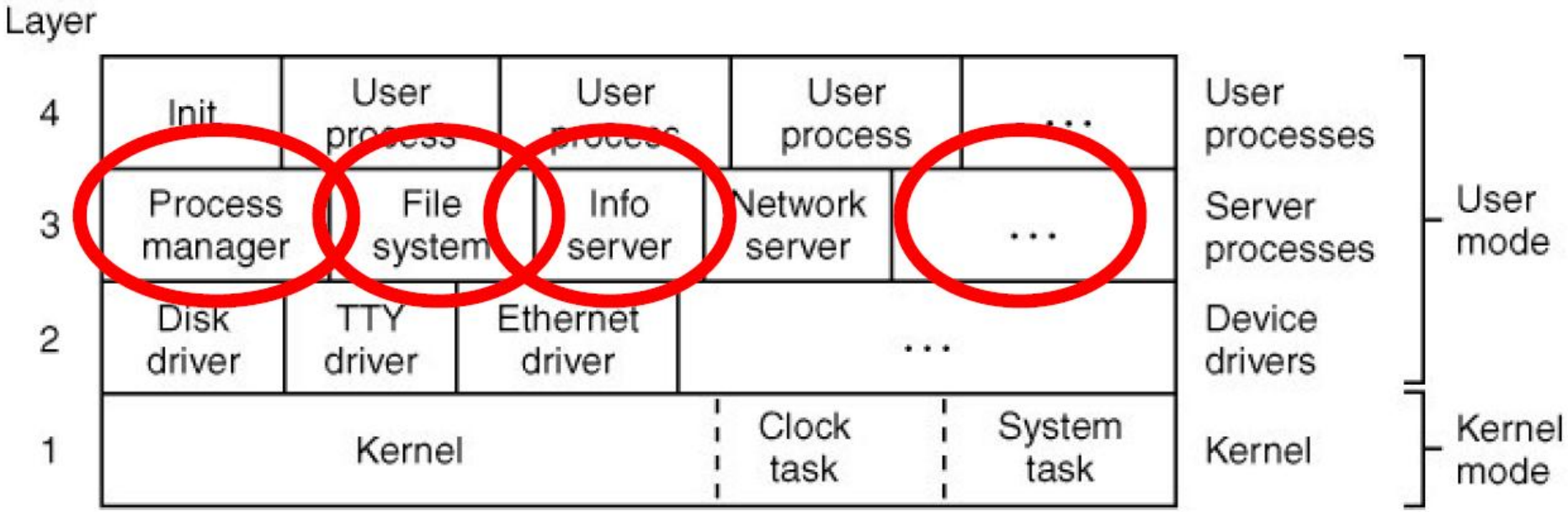
- Las dos primeras capas están compiladas y enlazadas en un solo ejecutable llamado kernel
- Cada capa tiene menos privilegios que la anterior: el núcleo puede ejecutar cualquier instrucción

Estructura interna



- **Kernel:** Planificación de procesos, comunicación entre procesos, operaciones con puertos de E/S, interrupciones.
- **Tarea de reloj:** Suministra servicios de temporización.
- **Tarea del sistema:** Suministra llamadas al núcleo (kernel), solo para drivers y servidores.
- **Drivers:** Suministran servicios relacionados con algún dispositivo periférico.

Estructura interna



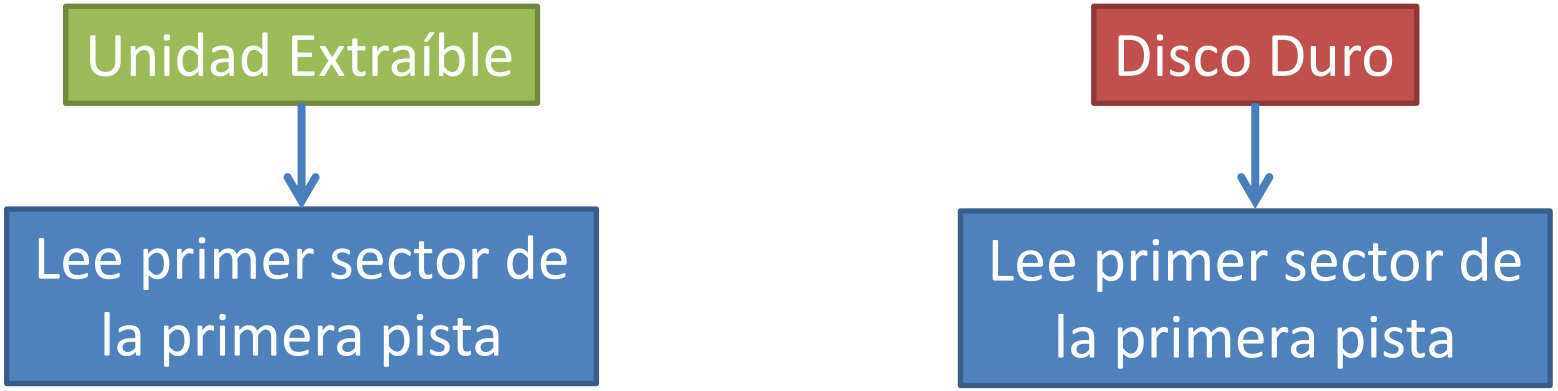
- **Manejador de procesos:** Suministra llamadas al sistema relacionadas con la ejecución de procesos y señales.
- **Sistema de ficheros:** Suministra llamadas al sistema relacionadas con ficheros.
- **Servidor de información:** Suministra servicios para depuración y obtención del estado del sistema.
- **Servidor de reencarnación:** Arranca (y rearranca si es necesario) drivers que no se cargan al mismo tiempo que el núcleo.

Booteo del SO

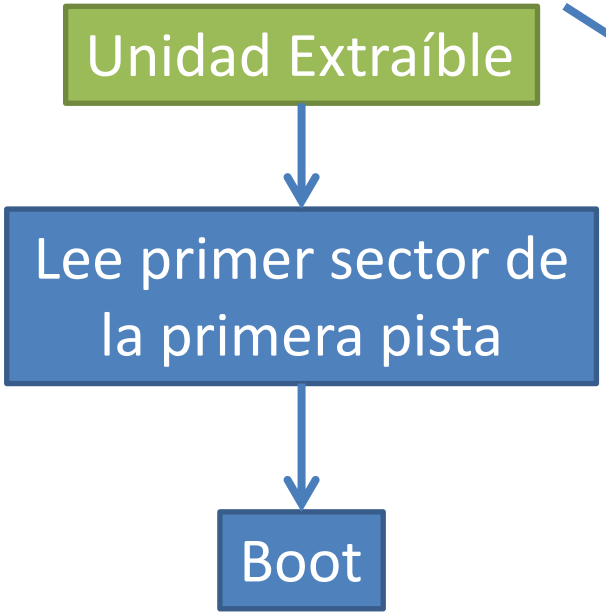
Unidad Extraíble

Disco Duro

Booteo del SO

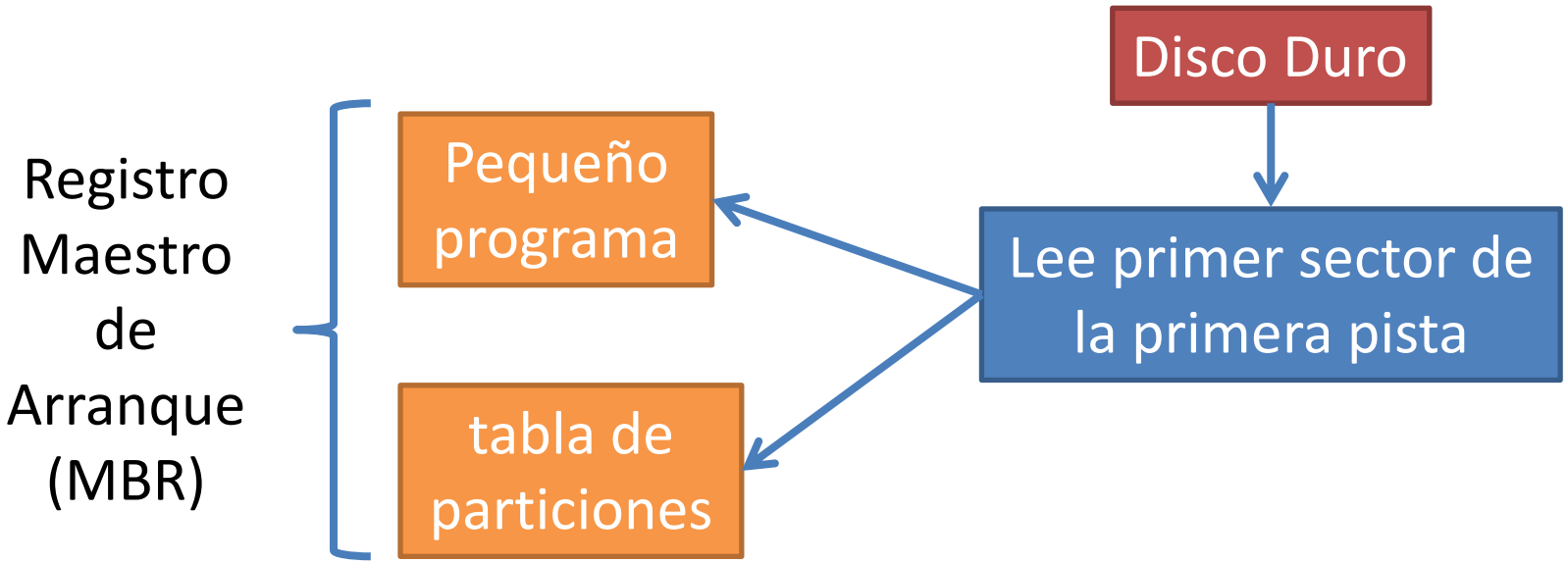


Booteo del SO

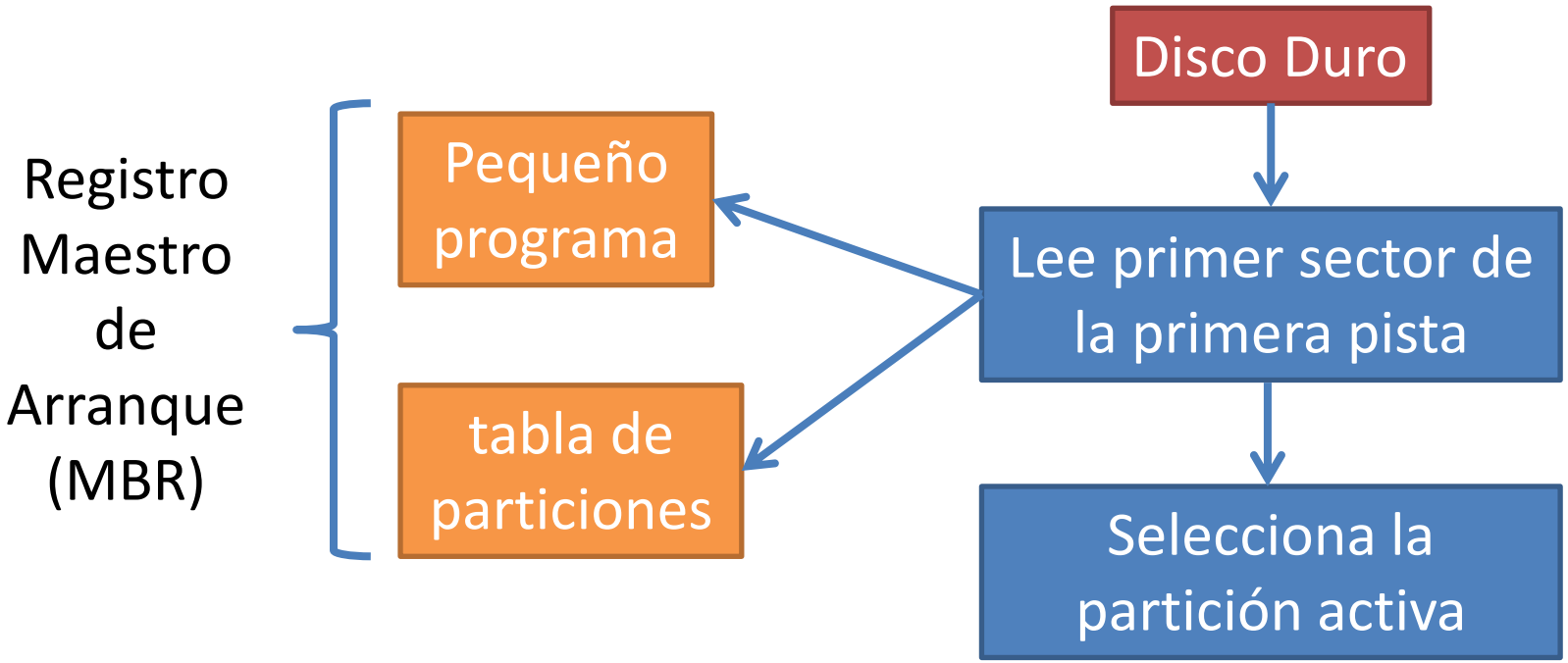


- Lee el primer sector y lo coloca en memoria
- Ejecuta el código (bootstrap)
- Lanza un programa mas grande: "boot" que luego carga el SO

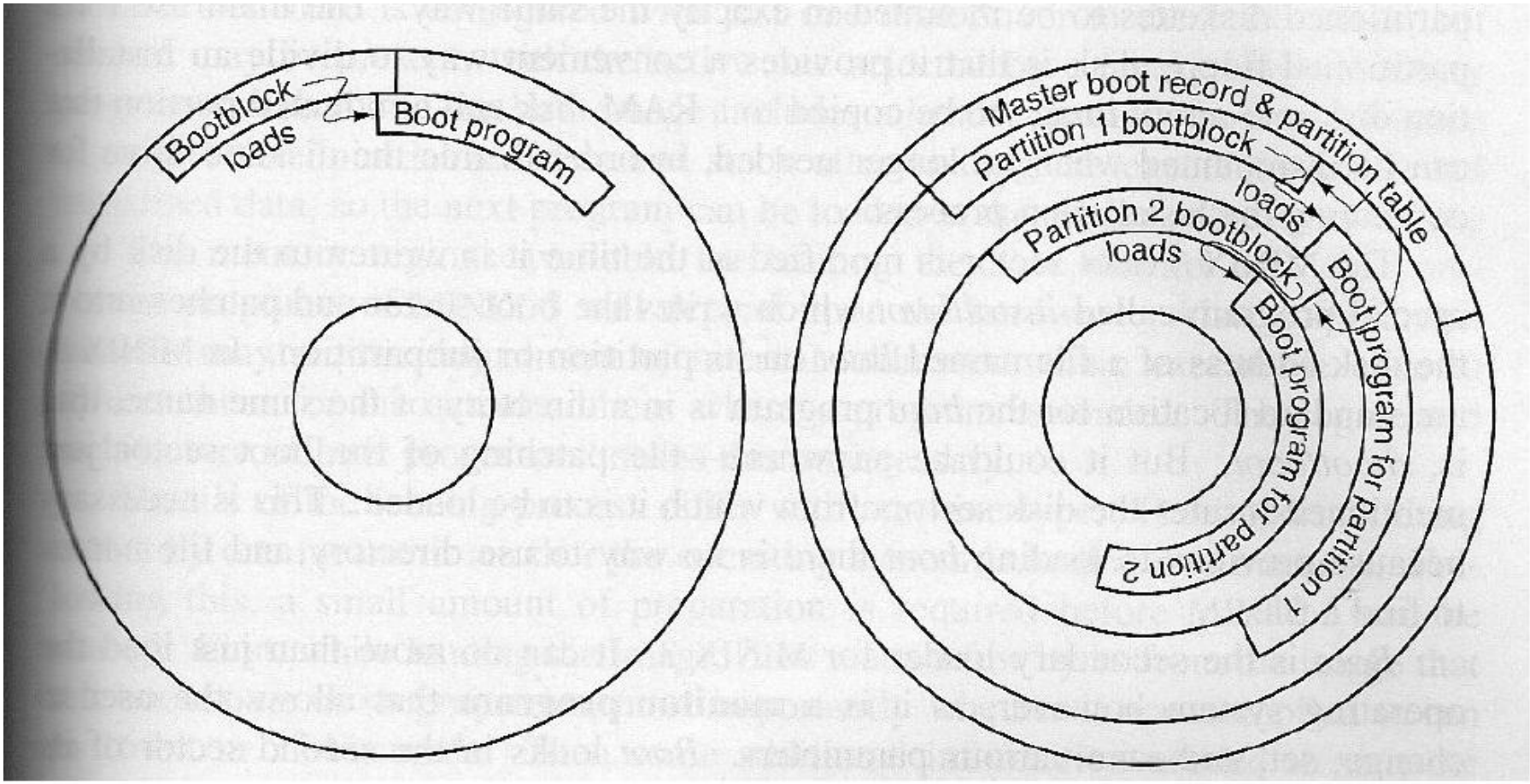
Booteo del SO



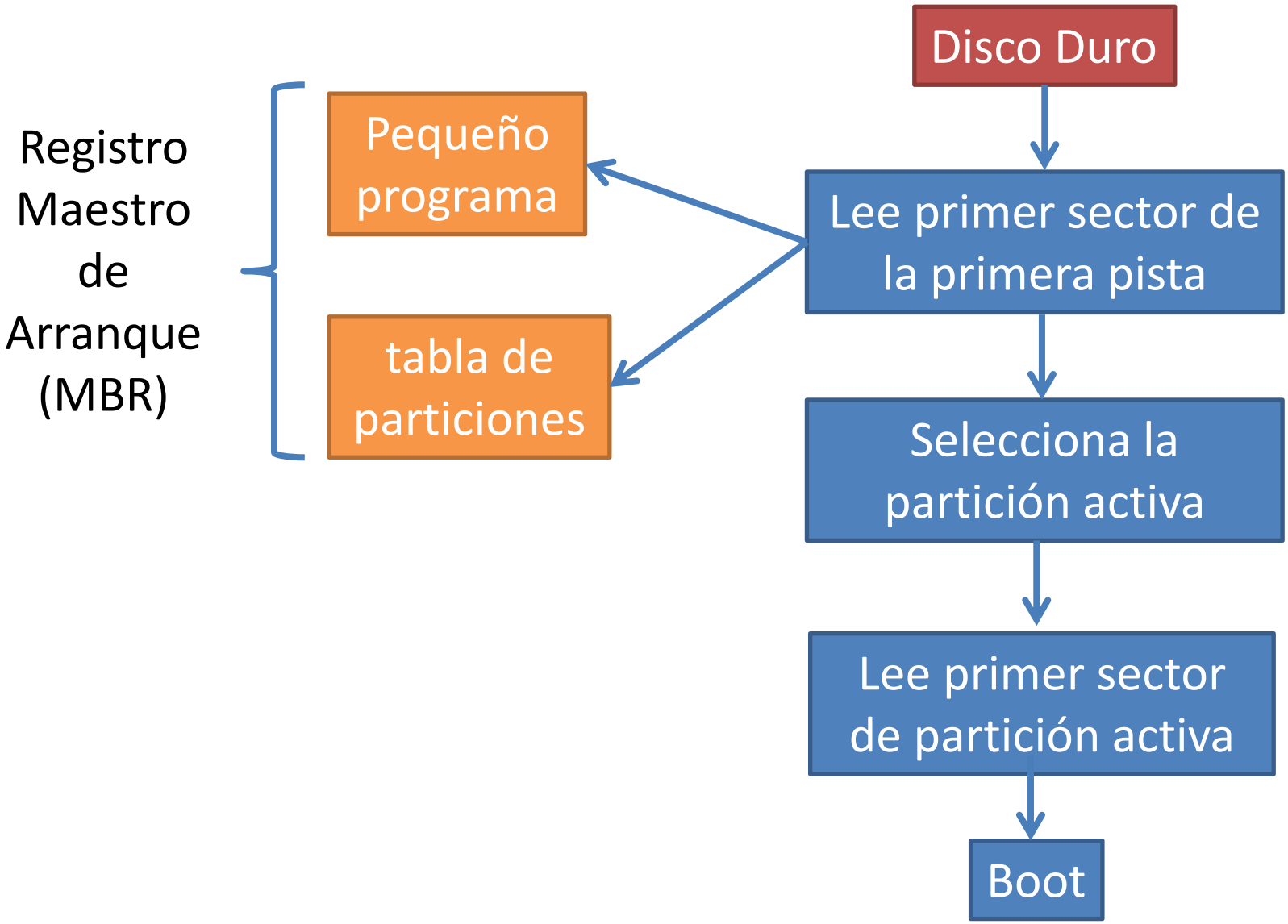
Booteo del SO



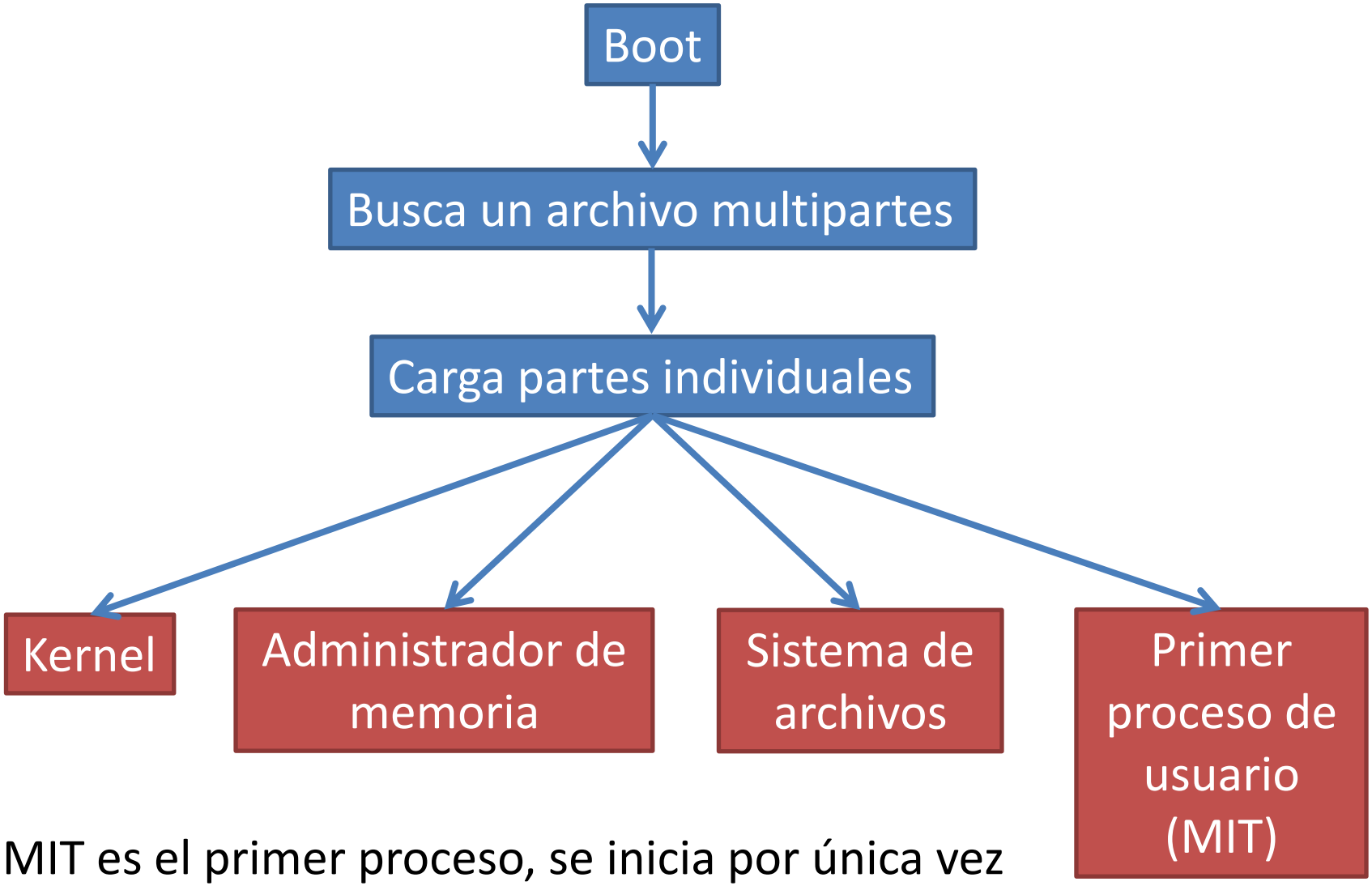
Booteo del SO



Booteo del SO

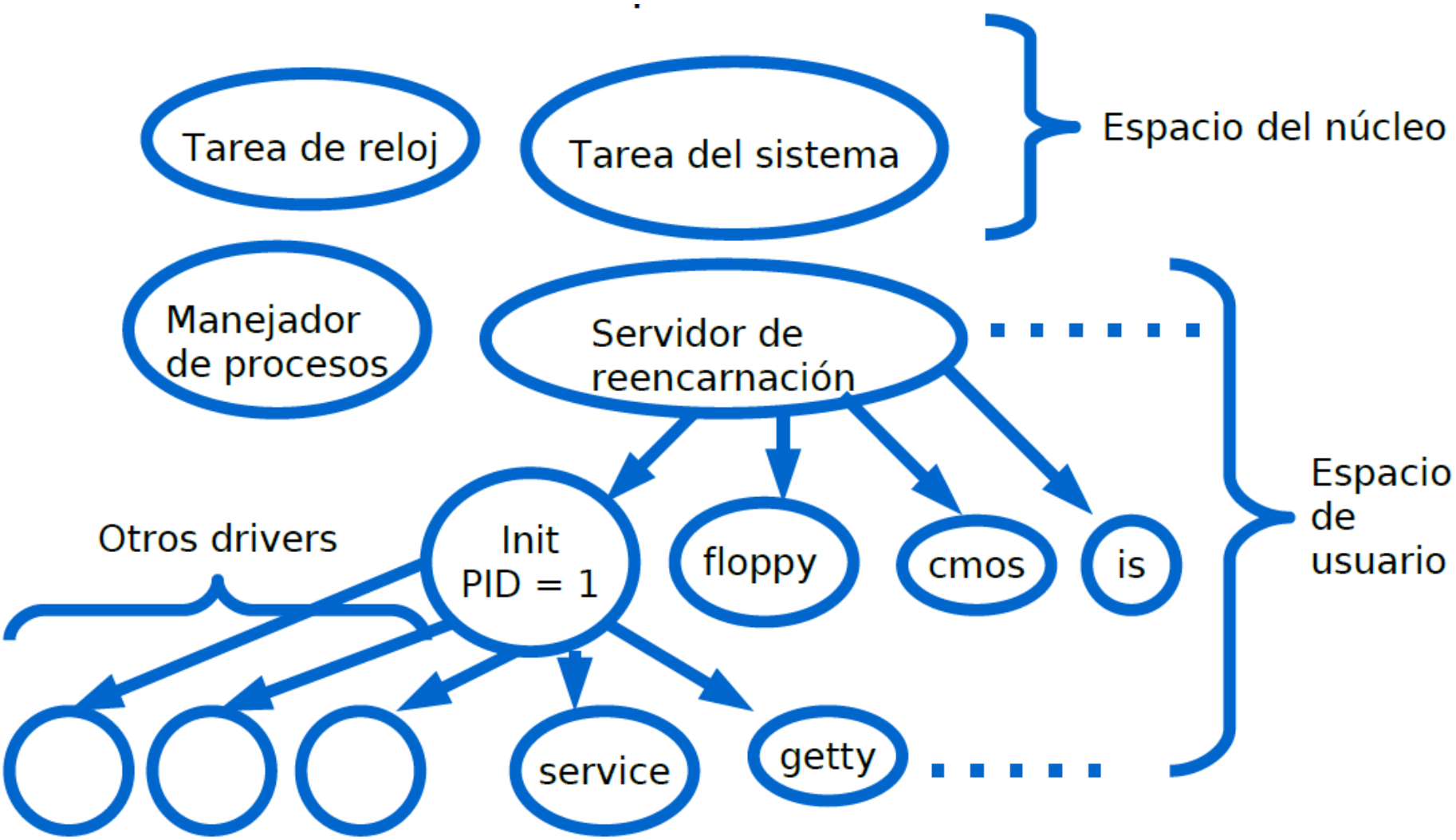


Booteo del SO



MIT es el primer proceso, se inicia por única vez

Inicialización del árbol de procesos



Organización del código fuente de MINIX

- Organización para IBM-PC compatible con procesador Intel 386 o posterior (palabras de 32 bits).

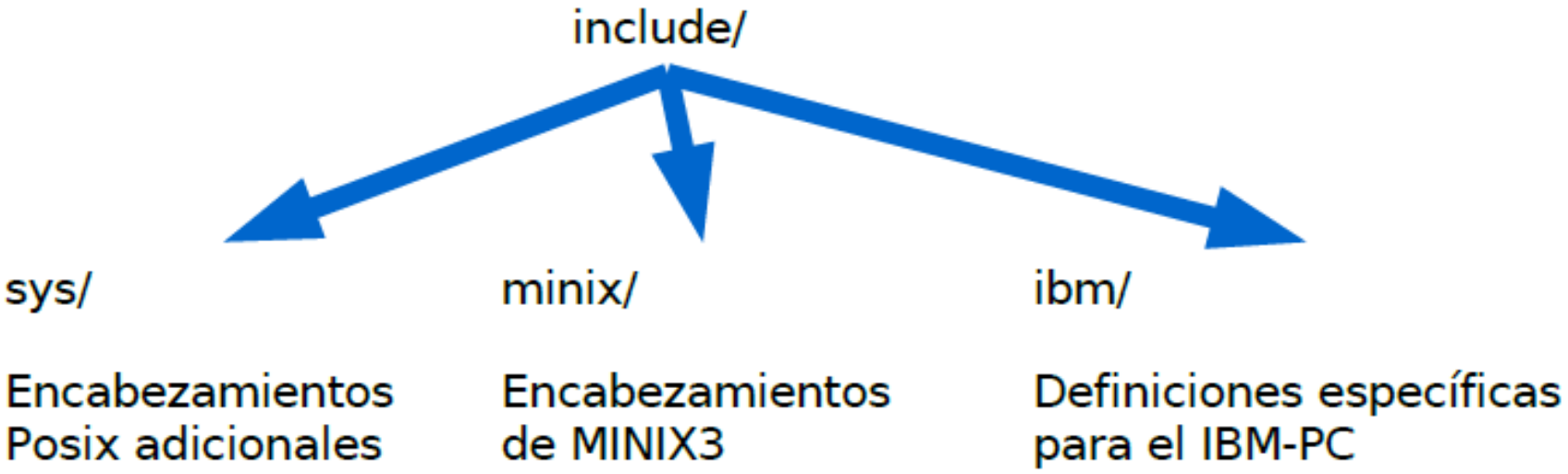
Ubicaciones

- **Código fuente**: /usr/src (src/ en adelante).
- **Ficheros de encabezamiento**: /usr/src/include (include/ en adelante).
- En cada directorio del árbol de ficheros fuentes hay un fichero **Makefile.**
- El directorio src/ puede ser relocalizado ya que Makefile utiliza caminos de referencia locales.
- Los ficheros de encabezamiento son un caso especial.

  ATENCIÓN AL MODIFICAR LOS FICHEROS DE ENCABEZAMIENTO!!
- Cada fichero Makefile espera encontrar los encabezamientos en /usr/include. Sin embargo, src/tools/Makefile espera los encabezamientos en /usr/src/include (borra /usr/include y lo rellena con /usr/src/include).

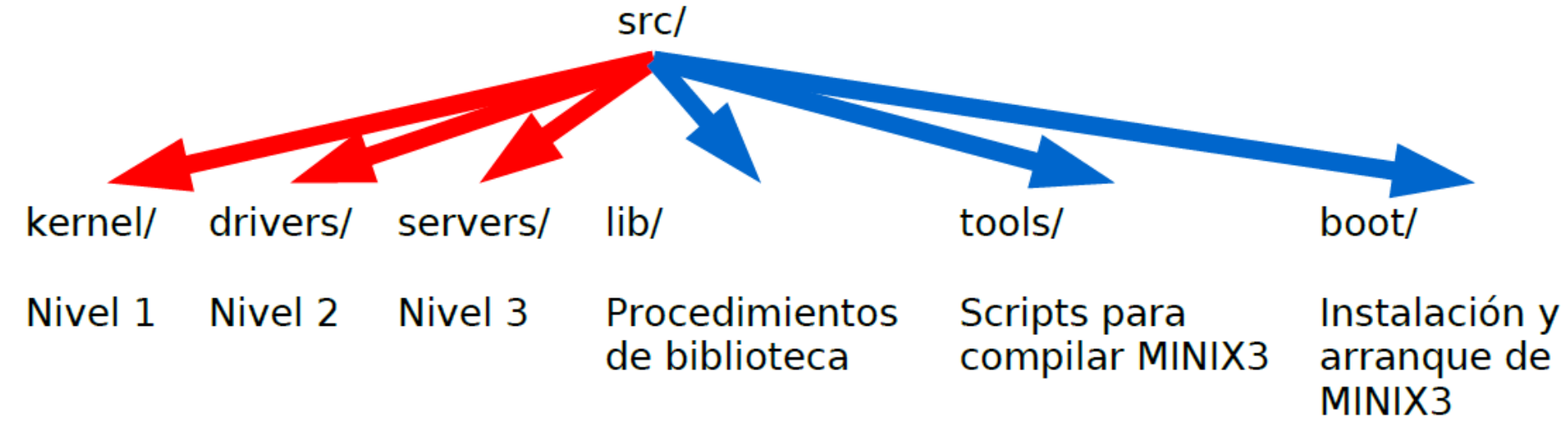
Organización del código en MINIX

Subdirectorios adicionales

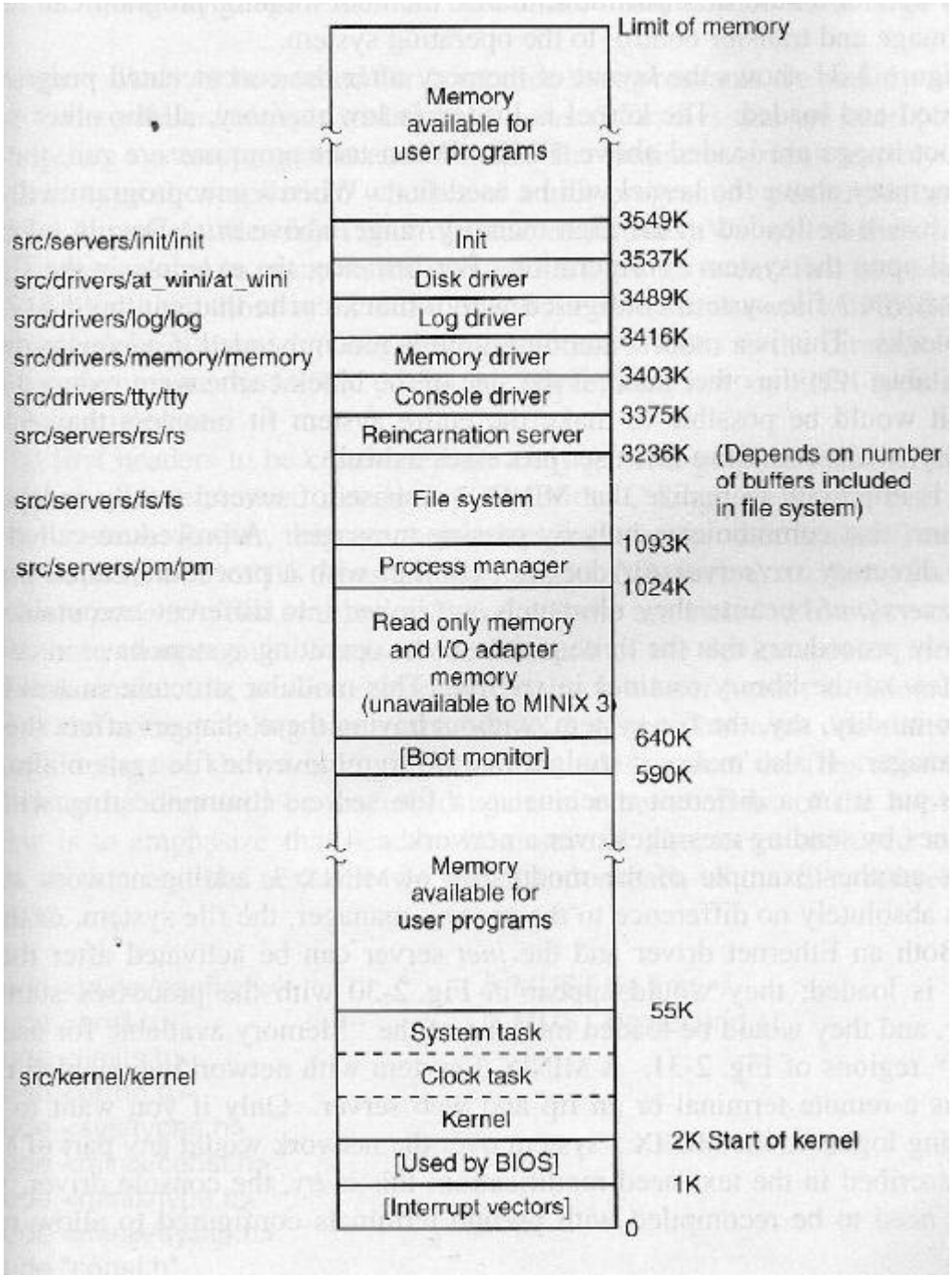


Organización del código MINIX

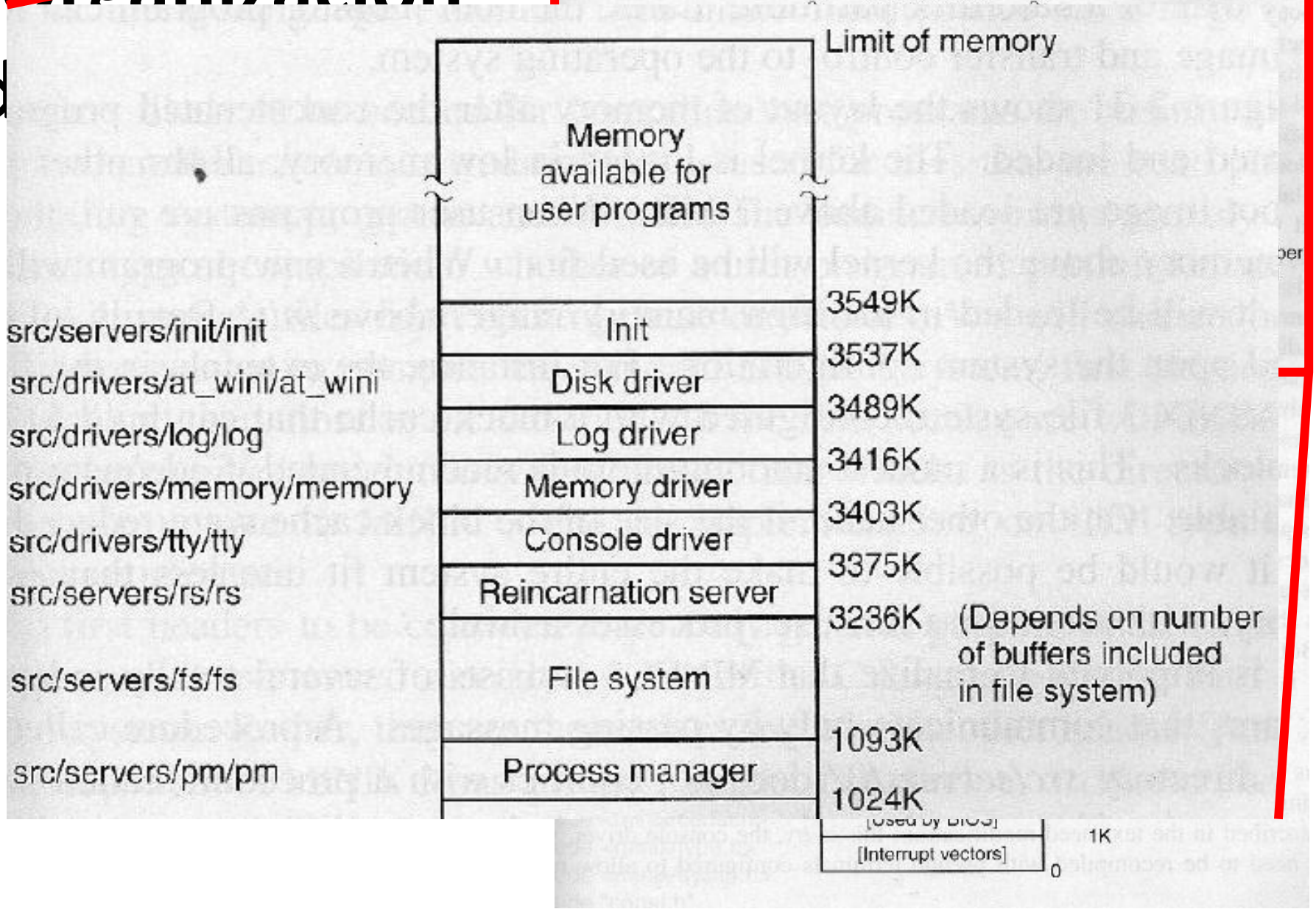
- Subdirectorios adicionales



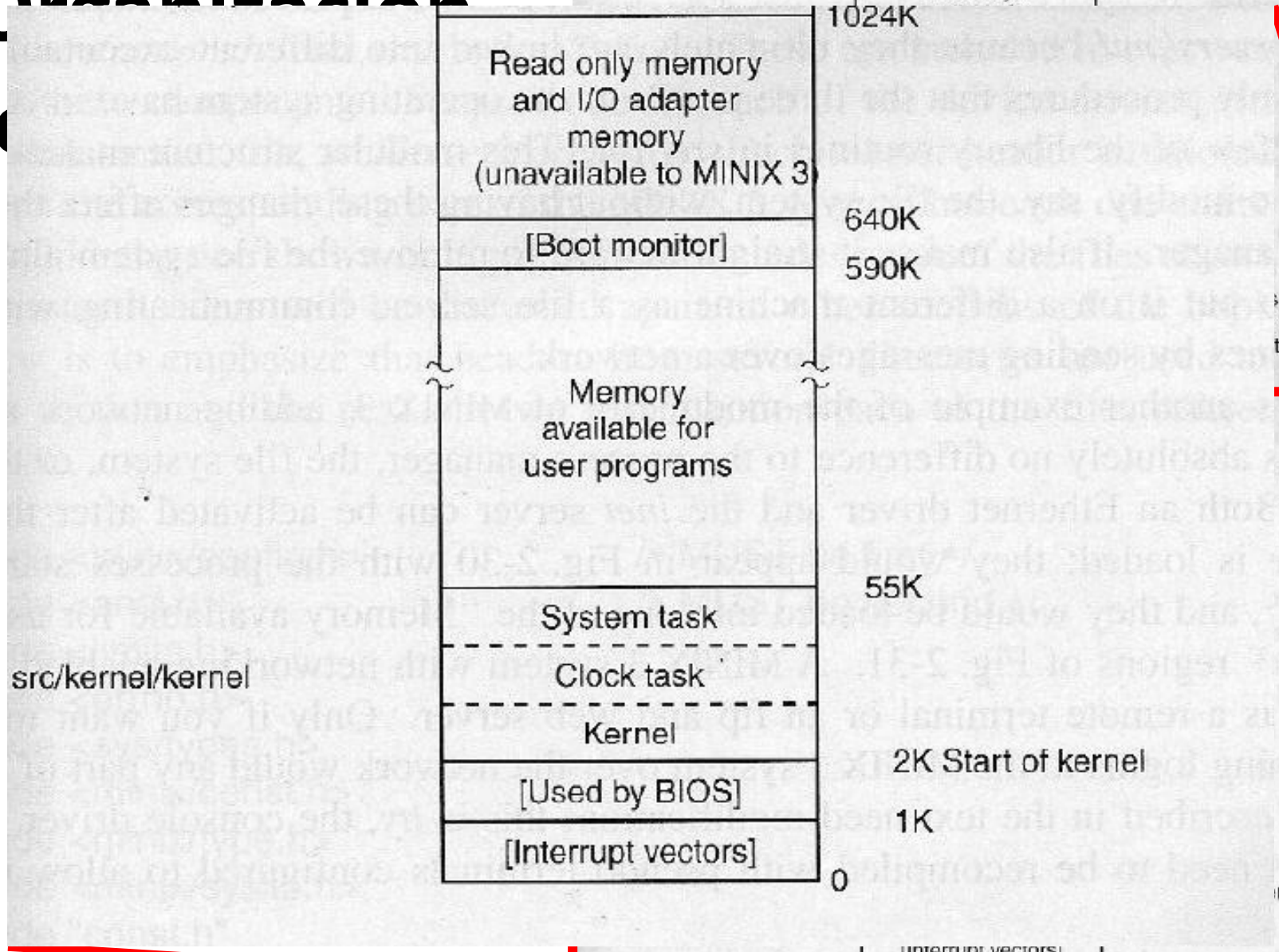
Organización de la memoria



Organización d



Organización de



Consideraciones de C en MINIX

include/ansi.h determina si el compilador utilizado cumple los requerimientos de C standard.

En MINIX se definen muchos **tipos de datos** (ej. **include/sys/types.h**), pero el compilador funciona correctamente si sustituimos los mismos por los tipos subyacentes. Todos los tipos declarados acaban en **_t**

Es necesario destacar que en ***const.h*** se utiliza la macro **EXTERN.** Permite utilizar variables globales sin caer en múltiples redefiniciones de las mismas.

Consideraciones de C en MINIX

En C, cualquier variable o procedimiento es accesibles fuera del fichero fuente en el que se declara, salvo que se marque como ***static***.

Para ganar claridad se define **PRIVATE** como sinónimo de *s**tatic*** y **PUBLIC** como ***string*** *nulo*.

Ejemplos:

PUBLIC void lock_dequeue(rp)

Esta declaración implica que *lock_dequeue* puede ser invocado desde fuera del fichero fuente en el que se declara

PRIVATE void dequeue(rp)

Esta función solo puede ser invocada desde dentro del mismo fichero fuente en que se declara.

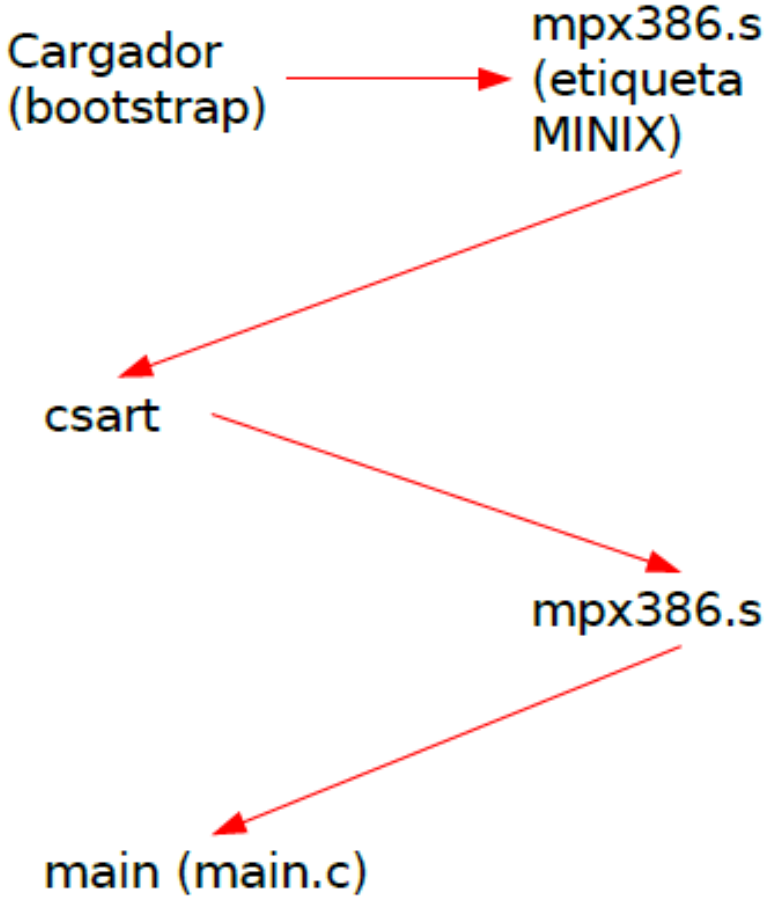
Arranque de MINIX

- El programa de arranque **boot** (ubicado en **/boot/boot**) es un monitor que carga y arranca el sistema operativo.
- MINIX reserva 1024 bytes como bloque de arranque, pero solamente se cargan los primeros 512 bytes al arrancar. Los otros 512 bytes se utilizan para almacenar información de interés para el proceso de inicialización.
- Boot busca el nombre especificado en el parámetro image= xxxx. que por defecto es **/boot/image**.
- La **imagen** de MINIX es la concatenación de varios ficheros ejecutables (núcleo, manejador de procesos, sistema de ficheros, servidor de reencarnación, manejadores de dispositivos y por último init).

Arranque de MINIX

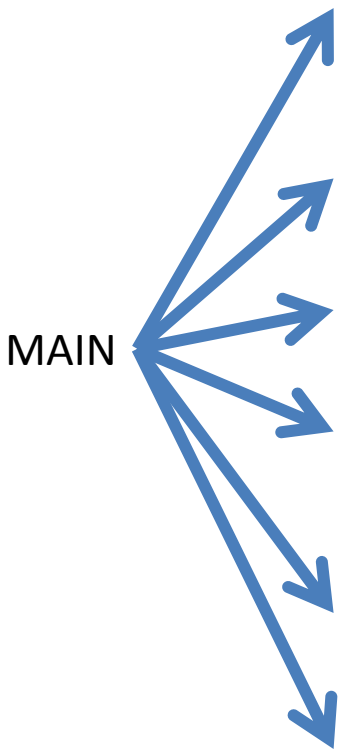
- Una vez cargado el sistema operativo en memoria se realizan algunas actividades preparatorias:
  - “Boot” al cargar la imagen lee de ella algunos bytes de información que le indica algunas de sus **propiedades**.
  - Se averigua información adicional para pasarla al núcleo. Información de los componentes de MINIX en la imagen es colocada en un vector en la memoria del cargador (boot) y la dirección base de dicho vector se suministra al kernel.
  - Se suministra al núcleo también la dirección del monitor (boot) a la que debe volverse al terminar de ejecutarse el sistema operativo.
- La información comunicada al núcleo se transfiere por medio de la pila.

Inicialización de MINIX



- El cargador ha introducido en la pila datos necesarios y de interés para el núcleo.
- mpx386 prepara el marco de pila para poder llamar a cstart (start.c).
- Inicializa GDT e IDT y copia los parámetros de la pila (pasados por el cargador) a memoria del núcleo.
- Hace efectivas GDT e IDT con lgdt y lidt. mpx386.s salta (jmp) a main.

Inicialización de MINIX



1. Inicializa como libres todos los elementos de la tabla de procesos y elimina los privilegios de la tabla de privilegios.
2. Inicializan los elementos correspondientes a los procesos cargados en la imagen de arranque.
3. Inicializa el puntero de pila de las tareas (salvo para el núcleo, denominado en ocasiones como HARDWARE).
4. Se actualizan los mapas de memoria (segmento de texto y datos) de los procesos a partir de la información que leyó el monitor de MINIX al arrancar en su memoria.
5. Se actualiza el contador de programa y el registro de estado de los procesos.
6. Se introducen en sus correspondientes colas de prioridad a los procesos.

Continuará….

Bibliografía

- Cap 1, Tanembaum "Sistemas Operativos modernos"
- Cap 3, William Stallings "Sistemas Operativos"
- Cap 2, Tanembaum "Sistemas Operativos, Diseño e implementación"