Linux Root FileSystem

Mg. Ing. Gonzalo E. Sanchez
MSE - 2022

Implementación de Sistemas Operativos II

Linux Root FileSystem

- Principios
- FileSystem mínimo

Principios

Principios

- Los sistemas de archivos se utilizan para organizar información en directorios y archivos.
- Se hace en dispositivos de almacenamiento o sobre una red.
- Son organizados en un esquema de jerarquías.
- En sistemas UNIX tanto las aplicaciones como los usuarios ven una única jerarquía global de archivos y directorios.
- Aunque sea única, puede ser compuesta de varios filesystems.

Principios

- Los filesystems se **montan** en un lugar específico en esta jerarquía de directorios.
- Un filesystem de un dispositivo de almacenamiento puede montarse en un directorio específico.
- El directorio se denomina **punto de montaje**.

Principios

- Esto tiene una ventaja: las aplicaciones pueden acceder fácilmente a directorios y archivos.
- No es necesario conocer la ubicación específica en el dispositivo de almacenamiento.
- Los contenidos de ese directorio reflejan el contenido del dispositivo de almacenamiento.
- Si el dispositivo se desmonta, el directorio vuelve a estar vacío.

Principios

- EJERCICIO:
  - Crear un punto de montaje (directorio): **mkdir /mnt/pendrive**
  - Chequear que está vacío: **ls /mnt/pendrive**
  - Montar un dispositivo de almacenamiento en este punto de montaje.
    mount -t vfat /dev/sda1 /mnt/pendrive
  - Chequear que se pueden acceder a los contenidos del pendrive
    ls /mnt/pendrive

Principios

- El comando **mount** permite montar filesystems.
- **mount** sin argumentos muestra los filesystems montados actualmente.
- Uso: **mount -t <type> <device> <mountpoint>**
- **type** marca el tipo de filesystem.
- **device** es el dispositivo de almacenamiento, o la ubicación de la red donde montar.

Principios

- **mountpoint** es el directorio donde los archivos o directorios serán accesibles.
- **umount** permite desmontar filesystems.
- Este comando es necesario antes de hacer un reboot o antes de desconectar un pen drive (por ejemplo).
- Esto es porque el kernel escribe un caché en memoria para incrementar el desempeño.
- **umount** se asegura que este caché sea volcado (escrito) en el dispositivo de almacenamiento.

Principios

- Existe un filesystem particular que se monta en la raíz de la jerarquía, y se identifica con **/**.
- Este sistema de archivos se denomina **root filesystem**.
- Tanto **mount** como **umount** son programas, son archivos dentro de un filesystem.
- Entonces, los programas para montar y desmontar están en un sistema que debe ser montado.
- La pregunta es: Y cómo se monta el **root filesystem**?

Principios

- Como el root filesystem es el primero en ser montado, claramente no puede ser montado con el comando **mount**.
- Se monta directamente por el kernel, según la opción **root=** que se le pasa como argumento.
- Cuando no hay ningún filesystem disponible para montar, el kernel entra en pánico (*kernel panic*).

```
Please append a correct "root=" boot option
Kernel panic - not syncing: VFS: Unable to
mount root fs on unknown block(0,0)
```

Principios

- Puede ser montado desde diferentes ubicaciones:
  - Una partición de un disco duro.
  - Una partición de un pendrive.
  - Una partición de una memoria SD.
  - Una partición de una memoria NAND.
  - Desde la red, utilizando el protocolo NFS.
- La lista no es exhaustiva, existen más ubicaciones posibles.
- Es tarea del diseñador del sistema seleccionar la configuración correcta mediante **root=**.

Principios

- Montando el rootfs desde un disco duro o desde un pendrive:
  - **root=/dev/sdXY** donde **X** determina el dispositivo e **Y** la partición.
  - Ejemplo: **root=/dev/sdb2** es la segunda partición del segundo disco (puede ser USB o un dispositivo ATA).
- Desde una memoria SD:
  - **root=/dev/mmcblkXpY**, donde **X** es un número indicando el dispositivo e **Y** un número indicando la partición.
  - **/dev/mmcblk0p2** es la segunda partición del primer dispositivo.

Principios

- Montando el rootfs desde un dispositivo Flash:
  - **root=/dev/mtdblockX**, donde **X** es el número de la partición.
  - **/dev/mtdblock3** es la cuarta partición de un chip NAND (esto es si hay un solo chip NAND presente - lo usual).

Principios

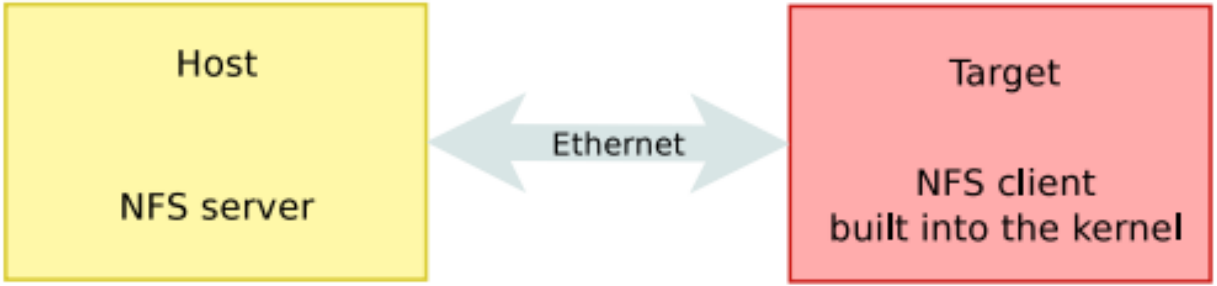
- Una vez que la red funciona, el punto de montaje puede ser un directorio en el HOST.
- Debe ser exportado por NFS, lo cual es muy conveniente para desarrollar un sistema.
- Hace que actualizar archivos en el root filesystem sea muy sencillo y rapido, sin hacer un reboot.
- Mucho más rápido que a través del puerto serial.

Principios

- Da la posibilidad de tener un gran filesystem cuando todavía no se tiene soporte de dispositivos de almacenamiento.
- El root filesystem puede ser tan grande como para tener herramientas de compilación nativa.
- Esto implica la posibilidad de compilar todos los componentes en el target directamente.
- **NO** se recomienda, por el poco poder de procesamiento disponible (mejor cross-compilar desde el HOST).

Principios

- Desde el lado del HOST workstation, es necesario un servidor NFS.



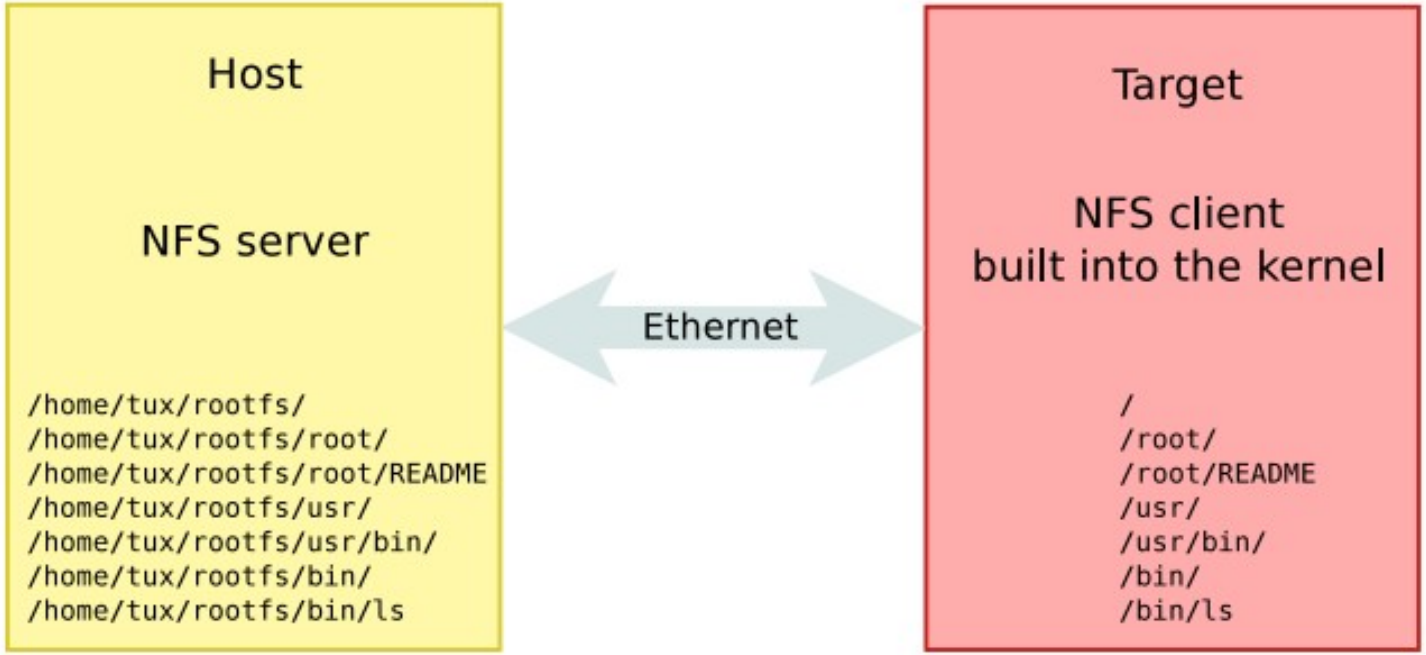
- Es necesario instalar el servidor, exponer el directorio a servir mediante el archivo **/etc/exports** e iniciar el servicio

Principios

- Es necesario que el kernel este compilado con las opciones correspondientes.
  - CONFIG_NFS_FS=y (NFS support)
  - CONFIG_IP_PNP=y (configure IP at boot time)
  - CONFIG_ROOT_NFS=y (support for NFS as rootfs)
- Al bootear deben estar presentes las opciones:
  - root=/dev/nfs (we want rootfs over NFS)
  - ip=<IP_address> (target IP address)
  - nfsroot=<IP_address>:<directorio_a_servir> (NFS server details)

Principios

- EJEMPLO:



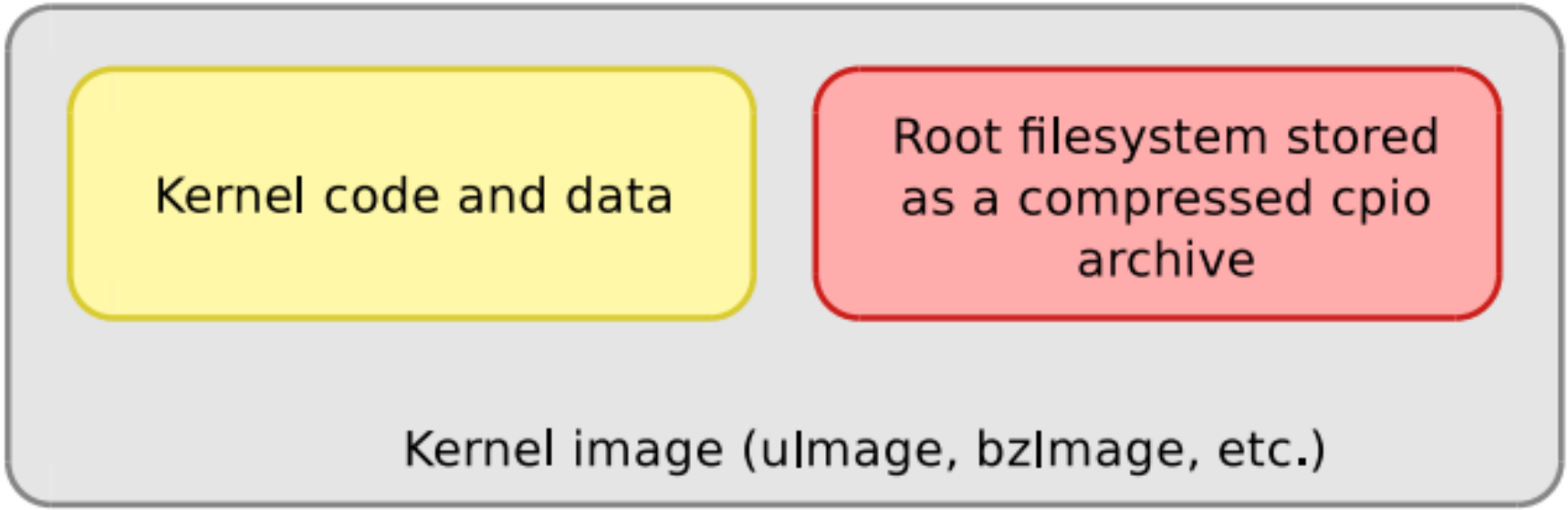
Principios

- Es posible también tener el fs integrado al kernel.
- Por lo tanto, se carga el fs en memoria junto con el kernel.
- Este mecanismo es llamado **initramfs**
- Este fs está comprimido e incluido dentro del kernel (parte del binario).
- Existen casos donde se carga por separado mediante el bootloader.

Principios

- **initramfs** es útil en dos casos:
  - Booteo rápido de filesystems muy pequeños.
  - Como paso intermedio antes de pasar a un root fs “real”.
- El segundo caso tiene sentido cuando los drivers del dispositivo de almacenamiento no son parte del kernel.
- Esto es utilizado siempre en PCs de escritorio y servidores para mantener el tamaño de la imagen de kernel razonable.

Principios


Kernel code and data
Root filesystem stored as a compressed cpio archive
Kernel image (uImage, bzImage, etc.)

Principios

- El contenido de initramfs se define a nivel de configuración de kernel (**CONFIG_INITRAMFS_SOURCE**).
- Esta opción puede ser el path a un directorio que contenga el root filesystem.
- Puede ser el path a un archivo **cpio**.
- Puede ser un archivo de texto describiendo los contenidos de **initramfs** (describe la estructura de directorios y archivos).

Principios

- En el proceso de compilación de kernel se toman de manera automática los contenidos de **CONFIG_INITRAMFS_SOURCE.**

- Algunos detalles aquí.

Principios

- La organización de un root filesystem en términos de directorios está bien definido.
- El estándar está en **File Hierarchy Standard** (link).
- La mayoría de los sistemas linux cumplen con esta estructura.
- Las aplicaciones esperan esta estructura.
- Tener una estructura similar en todos los sistemas hace más fácil los desarrollos.

Principios

- Directorios importantes:
  - **/bin** -> Programas básicos
  - **/boot** -> Kernel (solo cuando el kernel se carga desde un filesystem. Esto no es común en arquitecturas x86)
  - **/dev** -> Devices Files.
  - **/etc** -> Configuraciones de sistema.
  - **/home** -> Directorio home de los usuarios.
  - **/lib** -> Bibliotecas básicas.
  - **/media** -> Puntos de montaje para dispositivos removibles.

Principios

- Directorios importantes:
  - **/mnt** -> Puntos de montaje para medios estáticos.
  - **/proc** -> Punto de montaje para el filesystem virtual proc.
  - **/root** -> Directorio home para el usuario root.
  - **/sbin** -> Programas básicos del sistema.
  - **/sys** -> Punto de montaje para el filesystem virtual sysfs
  - **/usr/bin** -> Programas que no son básicos.
  - **/usr/lib** -> Bibliotecas que no son básicas.

Principios

- Directorios importantes:
  - **/usr/sbin** -> Programas de sistema no básicos.
  - **/var** -> Archivos de datos variables. Se incluyen datos administrativos, datos de logs, archivos temporales y transitorios.

Principios

- Los programas básicos se instalan en **/bin** y **/sbin.**
- Todos los demás programas se instalan en **/usr/bin** y **/usr/sbin**
- Las bibliotecas básicas se instalan en **/lib** y las demás en **/usr/lib.**
- **/bin** y **/sbin** contienen programas como **ls**, **ifconfig**, **cp**, **bash**, etc.
- **/lib** contiene la biblioteca C y a veces algunas otras bibliotecas básicas.

Principios

- Uno de los roles importantes del kernel es permitir a las aplicaciones acceder a los dispositivos de hardware.
- Para el kernel, la mayoría de los dispositivos se presentan a espacio usuario mediante dos abstracciones:
  - Character devices.
  - Block devices.
- Internamente el kernel identifica cada dispositivo mediante **Type** (char o block) y los números **Major** y **Minor**.

Principios

- Los dispositivos de tipo **block** se componen de dispositivos de bloques de tamaño fijo que pueden leerse y escribirse.
- Usualmente utilizados para discos duros, pen drives, memorias SD.
- Los dispositivos **char** originalmente eran un stream de bytes infinito, sin inicio o final y sin tamaño.
- Ejemplo típico: puerto serie.
- La mayoría de los dispositivos que no son del tipo **block** se representan como un **char** device.

Principios

- Una decisión importante de diseño en UNIX fue representar la mayoría de los objetos de sistema como archivos.
- Esto permite a las aplicaciones a manipular los objetos de sistema con la API de acceso a archivos.
- Así, los dispositivos tenían que ser representados como archivos a las aplicaciones.
- Esto se logra mediante los **device files**.

Principios

- En sistemas linux básicos los device files deben ser creados de manera manual a traves de **mknod.**
- Para lograr esto se deben tener permisos de superusuario.
- Implica que la coherencia entre device files y los dispositivos manejados por el kernel librada al desarrollador del sistema.
- En sistemas más complicados existen mecanismos para agregar/quitar estos automáticamente.

Principios

- Se utiliza **devtmpfs** (filesystem virtual) desde kernel version 2.6.32
- **udev** daemon, que se utiliza tanto en servidores como en PC de escritorio.
- El programa **mdev**, que es una solución más liviana que **udev**.
- Los filesystems virtuales son también llamados **Pseudo Filesystem.**

Principios

- Uno de los pseudo filesystem más importantes es **proc** y existe desde los principios de Linux.
- Permite al kernel exponer estadísticas de procesos corriendo en el sistema.
- Permite al usuario ajustar en tiempo de ejecución parámetros de sistema sobre el gerenciamiento de procesos, etc.
- Se hace mediante el comando **sysctl** (artículo sobre esto).
- Información relacionada a la línea de comandos: **/proc/cmdline**

Principios

- El filesystem **proc** es utilizado por muchas aplicaciones estándar en espacio usuario.
- Todas estas aplicaciones esperan que este pseudo filesystem esté montado en **/proc**.
- EJEMPLO: aplicaciones como **ps** o **top** no funcionan sin **proc** montado.
- Se utiliza el mismo comando para montar cualquier filesystem: **mount**
- **`mount -t proc nodev /proc`**

Principios

- **proc** representa como un directorio cada proceso que esta corriendo en el sistema.
- Directorios que contienen información general relacionada a dispositivos:
  - /proc/interrupts
  - /proc/devices
  - /proc/iomem
  - /proc/ioports

Principios

- Otro pseudo filesystem importante es **sysfs** que fue integrado a partir del kernel 2.6.
- Permite representar en espacio usuario la visión que tiene el kernel de los buses, dispositivos y drivers.
- Útil para varias aplicaciones espacio usuario que necesitan listar el hardware y hacer consultas (**udev**, **mdev**).
- Como en el caso de **proc**, las aplicaciones que utilizan **sysfs** esperan que este montado en **/sys**.

Principios

- Mismo comando para montar cualquier otro **fs**:
  - **`mount -t sysfs nodev /sys`**

FileSystem Mínimo

FS mínimo

- Para que un sistema linux funcione, se necesita algunas pocas aplicaciones (como mínimo).
- Se necesita una aplicación **init** la cual es la primer aplicación en espacio usuario en ejecutarse.
- Se lanza inmediatamente luego de montar el root filesystem.
- El sistema trata de lanzar **/sbin/init**, **/bin/init**, **/etc/init** y **/bin/sh**

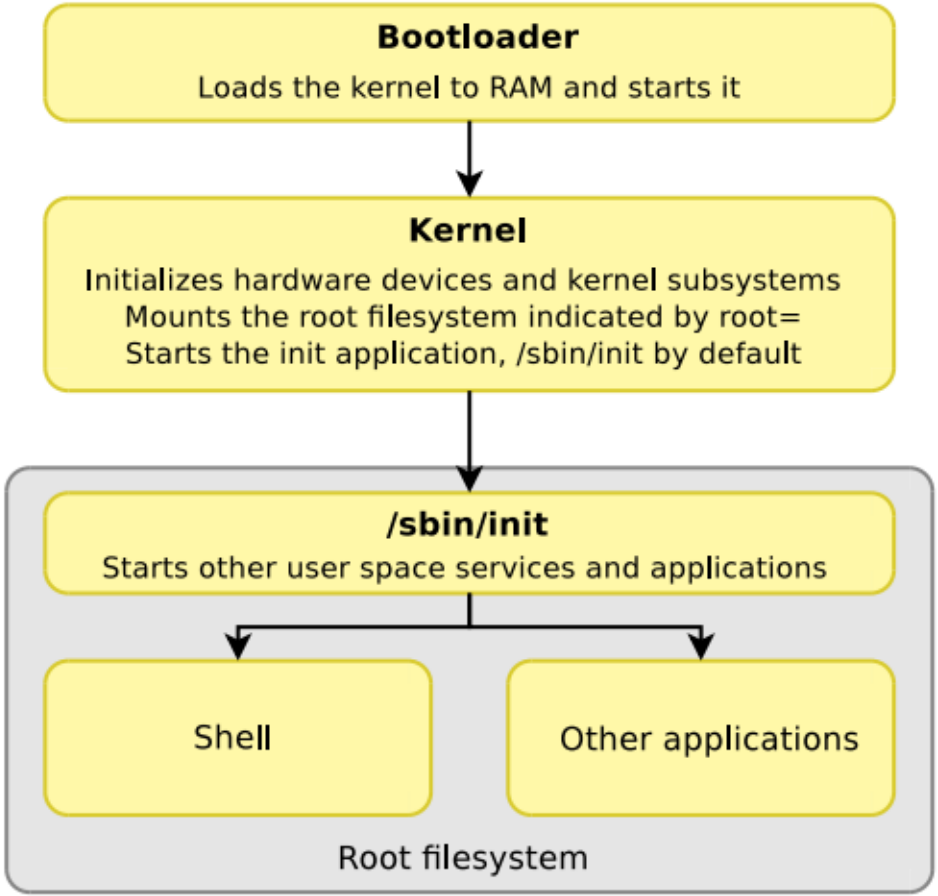
FS mínimo

- En el caso de que se utilice **initramfs** solo se busca **/init**.
- Puede especificarse otro path en **rdinit** (argumento de kernel).
- Esta aplicación **init** es la encargada de iniciar todas las demás aplicaciones y servicios.

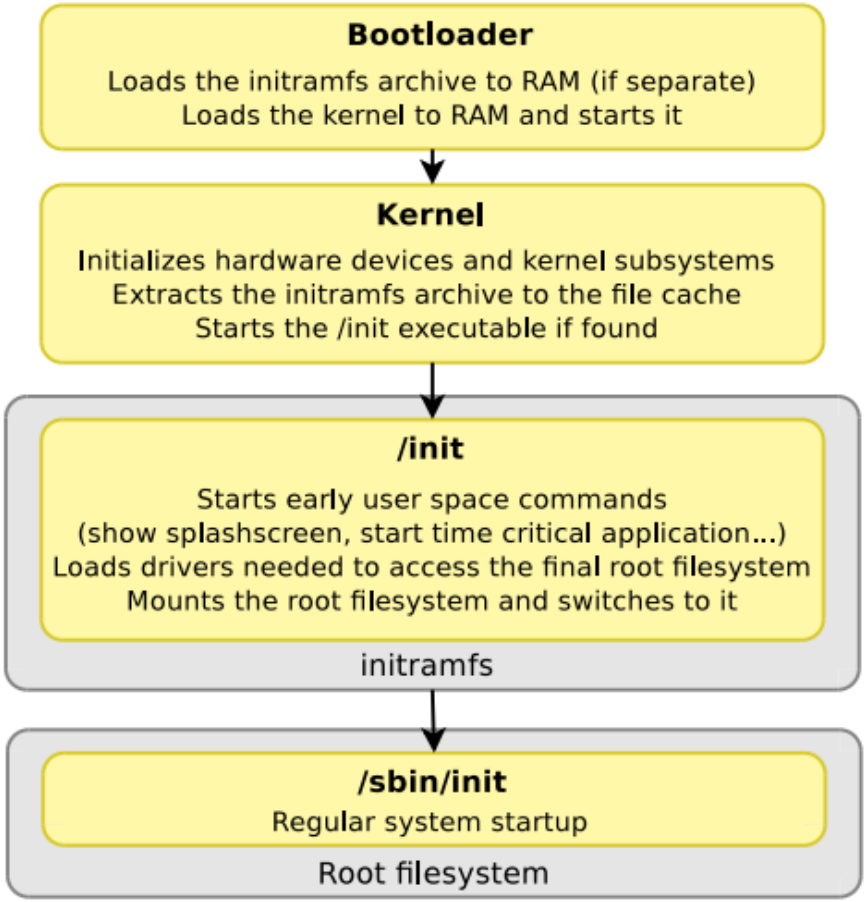
FS mínimo

- Se necesita usualmente un **shell** para permitir al usuario interactuar con el sistema.
- Aplicaciones básicas para copiar, mover y listar archivos (**mv**, **cp**, **mkdir**, **cat**).
- Todos estos componentes básicos deben estar integrados en el root filesystem para poder ser utilizadas.

FS mínimo


Bootloader
Loads the kernel to RAM and starts it
Kernel
Initializes hardware devices and kernel subsystems
Mounts the root filesystem indicated by root=
Starts the init application, /sbin/init by default
/sbin/init
Starts other user space services and applications
Shell
Other applications
Root filesystem

FS mínimo


Bootloader
Loads the initramfs archive to RAM (if separate)
Loads the kernel to RAM and starts it
Kernel
Initializes hardware devices and kernel subsystems
Extracts the initramfs archive to the file cache
Starts the /init executable if found
/init
Starts early user space commands
(show splashscreen, start time critical application...)
Loads drivers needed to access the final root filesystem
Mounts the root filesystem and switches to it
initramfs
/sbin/init
Regular system startup
Root filesystem

FS mínimo

HANDS ON

1. Generar un filesystem mínimo.
2. Comprobar el funcionamiento.

Gracias.