

# ENTRADA I SORTIDA

## Dispositius (Tipus)

### Dispositiu Virtual (fd)

→ Fitxer Obert // Dispositiu E/S

Número que identifica a un dispositiu dins un procés.

Per defecte tots els processos tenen:

0 - STDIN read(0, ...)  
1 - STDOUT write(1, ...)  
2 - STDERR

El fa servir usuari.

### Dispositiu Lògic

Representació Virtual gestionada pel SO que simula el funcionament disp. físic.

# Pot ser que si que relacioni amb un disp. físic.

Permet una interfaz al nivell físic. ∇ Linux els identifica amb "file name" <sup>/dev/null</sup> per exemple

### Dispositiu Físic

Dispositiu físic (Real) que són gestionats pel Device Driver.

També apareixen en /dev (igual que Disp. Lògic).

Identificat: • Block / Character: Identifica el tipus que treballa.

• Major: Informa al Kernel quin és el seu Device Driver. (DD)

• Minor: Identificador dins dels disp. que gestiona mateixa DD.

Pot ser assignat automàticament o forçat el identificador

Dispositiu Virtual	File Descriptor (fd)
Dispositiu Lògic	File Name
Dispositiu Físic	B/C + major + minor

## Device Driver

Codi + Data que permet interactuar amb el disp. (HW). # cat /proc/devices per veure.

Ha d'operar en "Kernel Mode" pq necessita accedir a coses físiques del ordinador.

// Per exemple: Tradueix un read() a com ho ha de moure l'agulla del disc.

Per afegir un nou:

• Recompilar el Kernel: Fer que el DD estigui en el Core del Kernel. (Carregat es durant arranque)

• Dynamic Kernel Module: Permet estendre mòduls del Kernel sense recompilar # Plug & Play

∇ Ha de seguir les especificacions d'interfaz del SO ⇒ ∇ Depenent del SO i HW

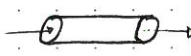
Dins un DD hi ha:

- Info. general del DD: Nom, autor, ...
- Implementacions de funcions genèriques: read, write, open, ... .read = min - read
- Init Function: Executat quan s'installa; Registra Major; Associa funcions a les del DD.
- Exit Function: Unregistra les funcions associades a el Major.

## Procediment

1. Compilar el DD en format \*.ko. #ID: B/C aquí
2. Instal·lar les rutines del driver. "insmod driver.ko" #LKM
3. Crear Disp. Lògic i vincular-lo a un disp. físic "mknod /dev/my\_driver c/b major minor"
4. Crear Disp. Virtual on ja depèn del procés. "open("/dev/my\_driver", ...);"

## Exemples de Dispositius Lògics

- Terminal: Dispositiu que representa conj. Teclat + Pantalla.
- Fitxer de dades: Representen informació emmagatzemada a disc. Interpretat seg. Bytes.
- Pipe: Implementa Buffer temporal (FIFO). Interacciona info processador. 
  - Sense Nom: Nomen processador que siguin parents. (Herència). #xd no té lògica de l'vide real.
  - Amb Nom: Qui tingui permís.
- Socket: Interacciona info entre ordinadors connectats per xarxa. No fa falta mateixa

## Estructures de Dades

### Inode

Tots els vists.

Estructura de dades que guarda informació relativa a un fitxer (Camp depèn tipus).

NO guardi el nom del fitxer. Guardat en disc, però en RAM si en ús.

Major i minor en aquest fitxer.

El camp "pointer to data" guarda punten a blocks on hi ha el contingut.



## File Descriptor Table (FDT) [Procés]

Taula que conté quin fd pot usar el procés. Sempre té 0, 1, 2.

Guardat en la PCB del procés (s'hereden).

En la taula, l'index es refereix al fd:  $FDT[i]$  és un punter a OFT.

## Open File Table (OFT) [S.O.]

Taula que guarda quins disp. virtuals estan en ús. Guardat en RAM.

Cada entrada correspon a un disp. virtual diferent i guarda un ptr a

l'entrada de la Taula de Inodes corresponent.

ptr-le apunta pos. dins  
fitxer que toca llegir/escriure

Diversos processos poden apuntar a la mateixa entrada.

## Inode Table (IT) [S.O.]

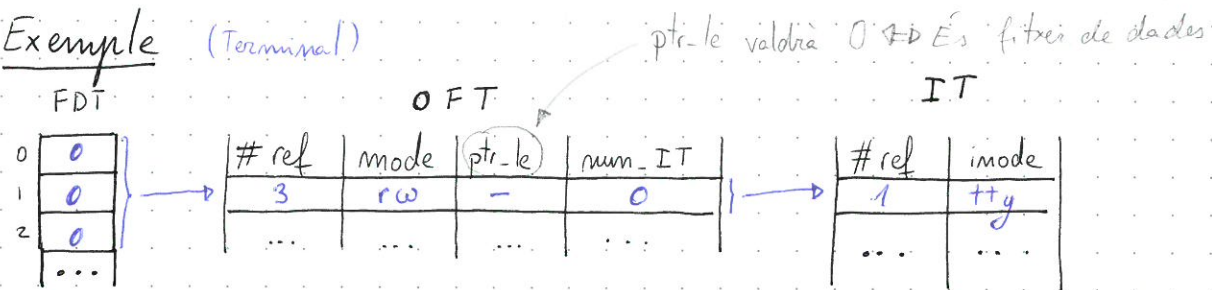
/etc/passwd i /dev/null

Taula que guarda el inode corresponent a cada fitxer.

Guardat → RAM.  
→ Disc.

⚠ SEMPRE hi haurà una entrada en aquesta taula per cada fitxer (Tots tipus).

## Exemple (Terminal)



## Operacions E/S Bloquejants

Si crida a sistema requereix de N Bytes i aquests no estan disponibles ⇒ BLOCK.

Quan estan les dades disponibles, SO reb interrupció i torna a ficar ⇒ READY.

`open()`, `read()`, `write()` són bloquejants.

## Operacions E/S Bàsiques

### open()

"Belatona" fd amb disp. lògic (file name).

→ Pot ser → Absoluta  
→ Relativa

flags: mode: Permisos, Pot ser numèric ougo

• `O_RDONLY` "`O_RDONLY | O_CREAT | O_EXCL`" Crea si no existeix. Altrament "-1".

• `O_WRONLY`

"`O_RDONLY | O_CREAT | O_TRUNC`" Borra contingut si existeix.

• `O_RDWR`

SO-4-T-2

## Read

m-bytes-lligits = read (fd, @logi-on-dexar, m-bytes-lligit.);

A l'hora de llegir pot passar que:

- Retorni si no hi ha dades a llegir. { Depèn del dispositiu.
- Blocn fins que hi hagi dades a llegir.

Pot retornar:

0 si he arribat el EOF. End Of File.

quant-lligit que pot ser igual o inferior al que volíem llegir.

ptr-le avança automàtic (Kernel) en OFT. ptr-le += quant-lligit.

## Write

m-bytes-escrits = write (fd, @logi-on-lligit, m-bytes-escriure);

Escriu el que hi hagi espai que còpiem. Si  $\begin{cases} \text{Disc} \Rightarrow \text{BLOCK} \\ \text{Pantalla} \Rightarrow \text{NO BLOCK} \end{cases}$

Si no hi ha suficient pot:

- Blocn fins que hi hagi espai dispo. { Depèn del dispositiu.
- Retorni 0 immediatament.

ptr-le avança automàticament (Kernel) en OFT. ptr-le += quant-escrit.

## Dup

newfd = dup (fd);

Duplica el fd amb el índex més petit.

# Ens pot servir per no perdre prèvi.

## Dup2

newfd = dup2 (fd, newfd);

Ara no és el menor sino l'especificat.

Si ja estava en ús, tanca el que hi havia.

## pipe

• Amb nom: mkfifo (/path/to/pipe-nom);

• Sense nom: pipe (rfd [2]); on es guarda

Així crea 2 fd en FDT i un inode temporal.

ÉS BLOCKEJAN → És lectura i no hi ha dades en pipe.

→ És escritura i la pipe està plena.

IMPO: TANCAR EXTREM QUE NO USEM.

En terminal:

Si al llegir li falten bytes per processar, queden guardats en un buffer intermig i el pròxim read() ho agafa.

```
char buff[1024];
int m = read (fd, buff, (buff));
❌ strlen pq llegira 0 donat que buff buit.
✅ sizeof pq llegira tot el que còpiem.
```



Si escribim a ningú esolta SIGPIPE Així mata per defecte (sense nom)



## lseek

Mouve ptr. le i permetre accés a posicions concretes.

$\text{new\_pos} = \text{lseek}(\text{fd}, \text{offset}, \text{relative\_a})$  <sup>Mag. del desplaçament</sup>

- SEEK\_SET: Desde inici + offset } NO pot ser negatiu offset
  - SEEK\_CUR: Desde actual + offset
  - SEEK\_END: Desde final + offset
- } Pot ser negatiu el offset

## Sistema de Fitxers

- Definir espai de direccions / esquema directoris.
- Accesos a fitxers. (Permisos, propietaris, ...)
- Assignar / Alliberar espai de fitxers.
- Trobar / Emmagatzemar fitxers.

Directoris: Fitxer especial que

associa Fitxer  $\sim$  N.º inode.

Fitxer	Inode
hola.txt	2

Amb "stat file" podem veure els atributs de l'anode.

## Linuks

- Hard-Link: Fitxer HL referència directament al n.º inode.

Fitxer -

NO permeten cicle. Per cada "/PATH/TO/file" diferent  $\Rightarrow$  incrementa n.º ref. inode.

⚠️⚠️ **IMPO**: Aquest n.º de referències NO és el de la IT.

# Quan n.º ref. inodes = 0  $\Rightarrow$  s'elimina.

- Soft-Links: Fitxer SL que apunta al fitxer que apunta al n.º inode.

Fitxer l

Es com un accés directe. Com que no apunta  $\Rightarrow$  no incrementa ref inodes.

SL permeten cicle.

Obs: En "/" el "." i ".." apunten a ell mateix. (Mateix n.º inode).

Obs: Per una banda hi ha referències inode desde directoris (El que si decreix a 0  $\Rightarrow$  elimina) i referències a inode respecte fitxers oberts. SÓN DIFERENTS.

Cridar a Sistema = 3 `link`, `unlink`, `symlink`, `chown`, `chmod`, `chgrp`, `stat`, `lstat` &

## Unitat de Treball Disc

Unitat del SO = Bloque. # Gestionat pel SO

Unitat divisió dispositiu = Sector. # Gestionat pel HW

Conjunt de sectors consecutius = Partició / Sistema fitxers. Tenen "id" = Dispo. Lògic. # SO <sup>Gestió</sup>

# "C:", "D:" en Windows & "/dev/hda1" en UNIX.

SO assigna blocs de dispo a fitxers dins una partició.

Són independents, cada partició pot fer servir sistema fitxers diferents. ↳ Ext4  
↳ XFS

## Gestió Emmmagatzematge

### Inode

Estructura de dades que conté metadades d'un fitxer.

NO conté <sup>name</sup> PATH

Conté també una taula d'índexs que gestiona blocs assignats al fitxer. ↳ De disc

Els índexs poden ser: → Directes: Apunten directament al bloc en disc.

→ Indirectes: Apunten a altres taules per crear referència.

### Llista Reunions Dispo.

El S.O. gestiona llistes per assignar n° inodes i blocs de disc dispo.

Assigna el primer n° inode dispo.

Assigna els blocs pertinents al fitxer.

### Estructures Especials

- Superbloc: Blocs de la partició on es guarden metadades (info) del sistema fitxers.

Es guarda en memòria per ser més ràpid. <sup>Multiple copies repartides pel disc</sup> PQ NO ES POT PERDRE

- Buffer Cache: Conté últims blocs llegits en la memòria per agilitzar.

⚠ Per a buscar un fitxer (qualsevol) és necessari examinar a tots els previs.




## Relació entre syscalls i estructures de dades

### Gen

Objectiu: Buscar el inode del fitxer amb el que es vol treballar i deixar-ho en mem.

Procediment:

1. Llegir directori del fitxer 
  - Està en memòria i auedim.
  - Està en disc i gràcies a b.info dins inode llegim.
2. Quan busquem l'inode podem passar dues coses: Trobem, No Trobem.

→ Trobem Inode:

- Comprovem que permissos OK sino ERROR.
- Si és un Soft-Link hem de carregar fitxer que apunta (Repetir process).
- Modificar FDT, OFT, IT

⚠⚠ Si PATH ocupa més que l'espai assignat, es guarda en un bloc i hauran d'auedir a aquest.

→ No Trobem Inode:

- Ús flag creació: <sup>Act. Superbloc</sup> Reservem i inicialitzem un inode. Al principi inode.OB = No blocs dades. Actualitzar llista inodes al directori, última mod, mida.
- Sense flag creació: Retornar error.

⚠⚠ MAI auedim bloc dades del fitxer objectiu. # Fixet que en el cas del PATH auedim al del previ, no objectiu real.

### read

Partim de:

- Tindre en IT el inode del fitxer i quin bloc té.
- Tindre en OFT una entrada del fitxer amb valor de ptr.

Hauran de: Calcular quants blocs serà necessari llegir.

- Comprovem que no estiguem al final del fitxer.
- Calculem en quin bloc partim:  $ptr / mida\_bloc$ .

# Si blocs l'hem fet servir prev. i està en mem, no fa falta anar a disc.

### write

Primer hem de comprovar permissos d'escriptura.

Si  $ptr == size(file) \Rightarrow$  Mod. inodes lliures del superbloc. + Act. llista blocs inode file.

# També actualitzar el camp "mida" en inode.

## close

Tasques a Realitzar :

- Totes les dades que estaven a buffers de memòria que no s'havien escrit en blocs del fitxer s'escrivem. # Optimització que es fa per no anar tant a disc.
- Actualitzar inode en memòria amb noves metadades. # Mínim Date Last Mod
- Copiar inode memòria al inode del disc pq canvis siguin persistents.