Entrada i Sortida

Lluís Garrido

lluis.garrido@ub.edu

Grau d'Enginyeria Informàtica

Introducció

Què és entrada i sortida ?

- El cor d'un ordinador és la CPU, que s'encarrega de processar les dades i les instruccions.
- La CPU no serveix de res si no hi ha dispositius que emmagatzemin les dades i permetin interactuar amb el usuaris.

Gran part del codi d'un sistema operatiu està destinat a la gestió dels dispositiu d'entrada i sortida.

Introducció

Un sistema operatiu

- S'encarrega de gestionar el gran ventall de dispositiu d'entrada i sortida disponibles avui en dia: discos durs, unitats de memòria flash, teclats, pantalla, ratolí, la connexió via xarxa, etc.
- Ofereix als programadors una API senzilla d'utilitzar i que és el més independent possible del dispositiu.

Introducció

Els dispositiu d'entrada i sortida es poden agrupar en tres grups

1. Dispositius d'interfície d'usuari: permeten la comunicació entre l'usuari i l'ordinador (ratolí, teclat, impressora, pantalla, etc.)
2. Dispositius d'emmagatzemament: permeten desar dades de forma no volàtil (discos, unitats flash, etc.)
3. Dispositius de comunicacions: permeten la comunicació entre ordinadors (targes de xarxa, etc.)

Introducció

La gestió de l'entrada i sortida és complexa

- En general tots els dispositius d'entrada i sortida són lents comparat amb la velocitat de processament de la CPU. Accedir a la memòria RAM requereix nanosegons, mentre que accedir a un disc dur requereix milisegons.
- Els dispositius d'entrada i sortida són doncs el coll d'ampolla en un ordinador. Es dediquen molts esforços a optimitzar les mecanismes d'entrada i sortida a un sistema operatiu.

Introducció

Device	Data rate
Keyboard	10 bytes/sec
Mouse	100 bytes/sec
56K modem	7 KB/sec
Telephone channel	8 KB/sec
Dual ISDN lines	16 KB/sec
Laser printer	100 KB/sec
Scanner	400 KB/sec
Classic Ethernet	1.25 MB/sec
USB (Universal Serial Bus)	1.5 MB/sec
Digital camcorder	4 MB/sec
IDE disk	5 MB/sec
40x CD-ROM	6 MB/sec
Fast Ethernet	12.5 MB/sec
ISA bus	16.7 MB/sec
EIDE (ATA-2) disk	16.7 MB/sec
FireWire (IEEE 1394)	50 MB/sec
XGA Monitor	60 MB/sec
SONET OC-12 network	78 MB/sec
SCSI Ultra 2 disk	80 MB/sec
Gigabit Ethernet	125 MB/sec
Ultrium tape	320 MB/sec
PCI bus	528 MB/sec
Sun Gigaplane XB backplane	20 GB/sec

- Les velocitats dels dispositius és molt variada (taula extreta del llibre de Tanenbaum).

Tipus de dispositius

Els dispositius d'entrada i sortida es poden dividir en dues categories

- Dispositius de bloc: emmagatzema la informació en blocs de mida fixa; els blocs són adreçables i es poden llegir i escriure de forma independent de la resta. Exemple: els discos (en totes les seves formes).
- Dispositius de caràcter: entrega o accepta un flux de bytes sense cap mena d'estructura de bloc; no són adreçables. Exemple: teclat, ratolí, la impressora, el dispositiu d'accés a la xarxa.

Aquest esquema no és perfecte (p.ex. rellotge, pantalla). Però l'esquema és prou general com a base pel disseny del programari associat.

Al llarg d'aquest tema utilitzarem el disc (el clàssic, no pas els d'estat sòlid) com a eix central per al dispositiu d'entrada i sortida.

El controlador de dispositiu

Un dispositiu es d'entrada i sortida composa (generalment) de

- Una part mecànica. En un disc dur: els discos, el braç mecànic i un capçal per llegir o escriure a pocs nanòmetres de la superfície.
- Un component electrònic – anomenat controlador de dispositiu – que permet controlar la part mecànica. En un disc, el controlador de dispositiu s'encarrega (en llegir) de rebre el flux de dades del disc, comprovar si hi ha hagut errors i desar-les en un *buffer* intern del controlador per copiar-les després a memòria RAM.

El controlador de dispositiu

Comunicació el sistema operatiu amb el controlador

- Cada controlador té alguns registres que són utilitzats per comunicar-se amb el sistema operatiu. El sistema operatiu pot ordenar, a través dels registres, que entregui dades, que les accepti, que s'apagui, que s'engegui; o saber si el dispositiu està preparat per acceptar una nova comanda.
- Molts controladors tenen a més un *buffer* de dades en què el sistema operatiu hi pot escriure o llegir. Un disc dur o la tarja gràfica tenen un *buffer* de dades.

El controlador de dispositiu

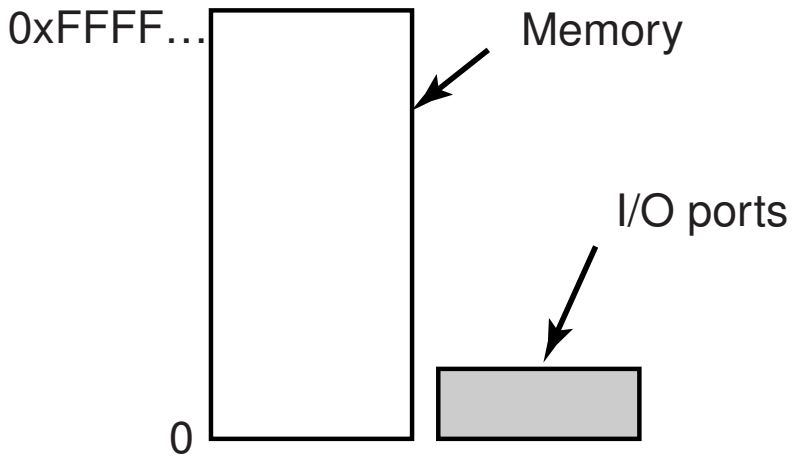
Com es comunica el sistema operatiu amb el controlador ?

- Fent servir ports d'entrada i sortida
- Fent servir entrada i sortida mapada a memòria

El controlador de dispositiu

Comunicació amb ports d'entrada i sortida

- Els registres de control es mapen en un espai de memòria separat del de la memòria RAM.
- Calen instruccions màquina específiques per accedir-hi.



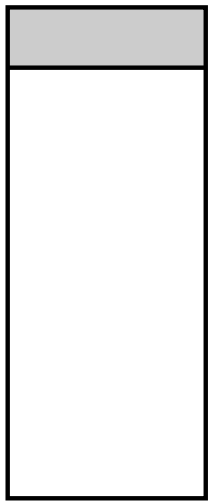
Podem veure la llista de ports registrats amb

```
cat /proc/ioports
```

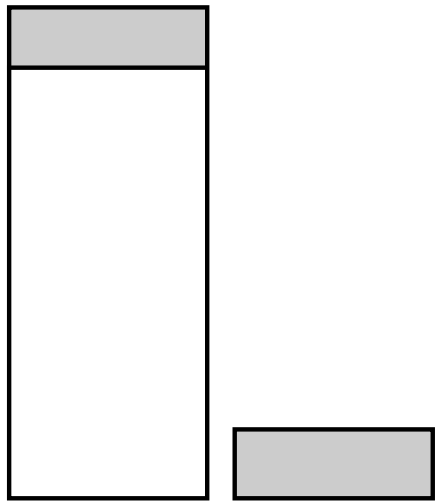
El controlador de dispositiu

Comunicació amb entrada i sortida mapada a memòria

- Els registres de control es mapen (virtualment) a l'espai de memòria RAM.
- Es poden utilitzar instruccions C per accedir als dispositius. El mecanisme de protecció de dispositius el proveeix la memòria virtual.



Mapat a memòria



Sistema híbrid

El controlador de dispositiu

El sistema operatiu, a més de controlar el dispositiu amb els registres, necessita intercanviar dades amb aquest.

- La CPU pot intercanviar dades byte a byte. És suficient amb dispositius lents.
- Per aprofitar més la CPU s'utilitza sovint un mètode anomenat DMA – Direct Memory Access –, independentment de si els registres es mapen a memòria o als ports d'entrada i sortida.

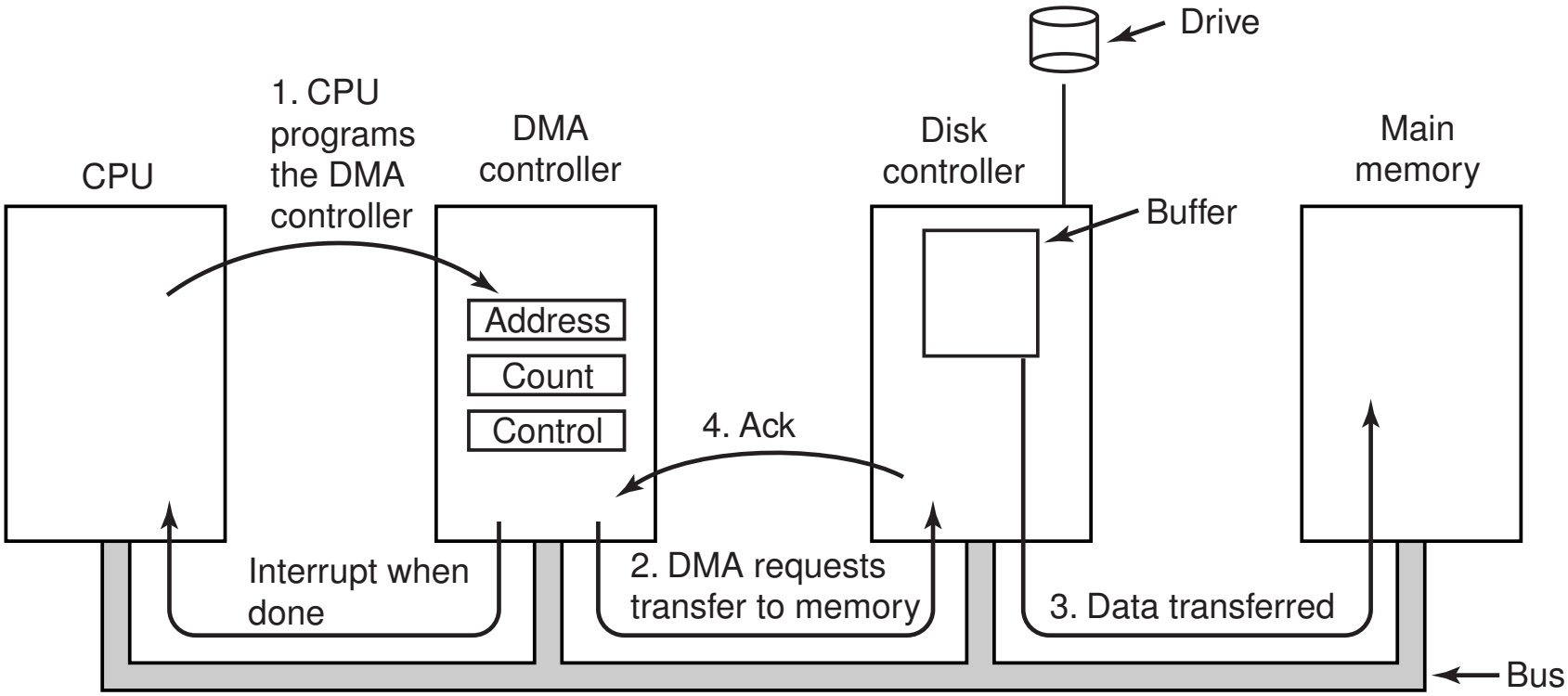
Direct Memory Access

Què passa si llegim un bloc de disc sense DMA ?

1. El controlador de disc llegeix els sectors del disc, bit a bit, i ho desa al buffer intern del controlador.
2. En acabar el controlador de disc realitza una interrupció a la CPU.
3. El sistema operatiu pot accedir aleshores al buffer intern i transfereix byte a byte les dades a la memòria RAM.

Direct Memory Access

Què passa si llegim un bloc de disc amb DMA ?



Direct Memory Access

Què passa si llegim un bloc de disc amb DMA ?

1. El sistema operatiu programa el controlador de DMA per indicar-li què ha de transferir i a on.
2. El sistema operatiu programa el controlador de disc perquè llegeixi les dades de disc.
3. Així que hi hagi dades vàlides al buffer intern del controlador de disc, el controlador DMA realitza la transferència de les dades a memòria RAM.
4. En finalitzar la transferència el controlador de DMA fa una interrupció a la CPU.

Direct Memory Access

La transferència de dades pot ser

- Per paraula: el controlador DMA espera que el bus estigui lliure. Aleshores el controlador DMA realitza una transferència d'una paraula i allibera el bus, igual que ho faria la CPU.
- Per bloc: el controlador DMA espera que el bus estigui lliure. Aleshores el controlador DMA realitza unes quantes transferències i allibera el bus.

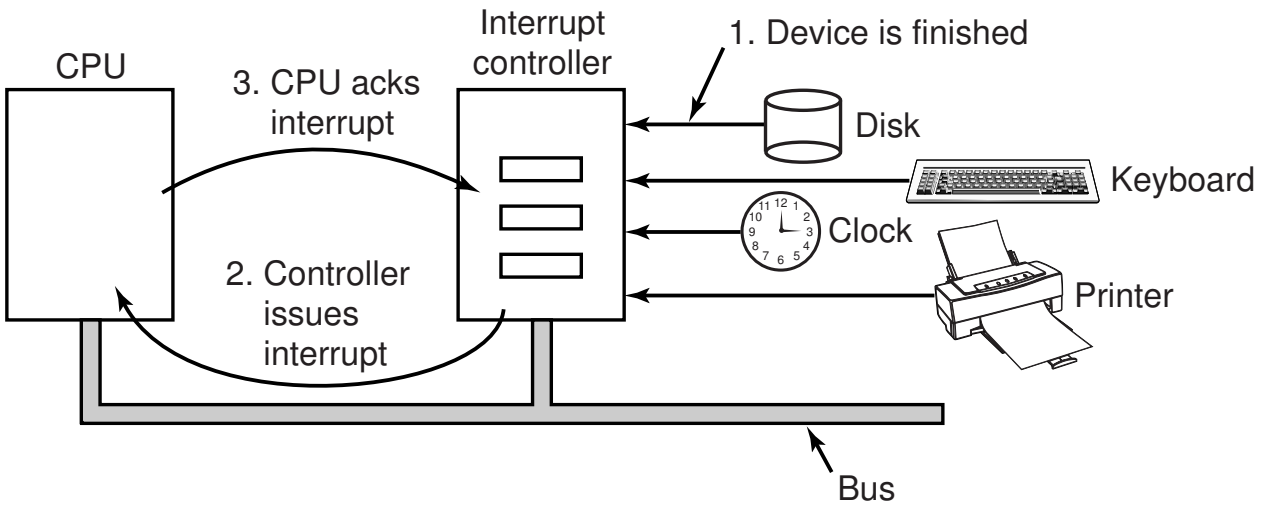
Tingueu en compte que

- La segona forma és més eficient que la primera però pot bloquejar la CPU (i altres dispositius) durant un temps substancial.
- No sempre és útil utilitzar DMA: la CPU acostuma a ser més ràpida que el controlador DMA.

Interrupcions

Esquema de funcionament d'una interrupció

- El controlador de dispositiu es programa perquè produeixi una interrupció en finalitzar la tasca a realitzar. Això deixa lliure a la CPU perquè realitzi altres tasques.



Podem veure les interrupcions realitzades amb

```
cat /proc/interrupts
```

Interrupcions

Esquema de funcionament d'una interrupció

1. En finalitzar la tasca, el controlador de dispositiu realitza una interrupció. La interrupció es capturada pel controlador d'interrupcions.
2. El controlador d'interrupcions decideix si s'accepta depenent de la prioritat de la interrupció.
3. El controlador d'interrupcions posa un valor al bus indicant el dispositiu que ha interromput i ho senyalitza a la CPU.
4. La CPU deixa de fer el que estava fent, passa a mode kernel, i salta a una funció del sist. op. (que depèn del valor al bus).
5. A la funció se serveix el dispositiu i es notifica al controlador d'interrupcions.
6. En finalitzar l'execució de la funció es retorna al punt en què s'ha produït la interrupció i es passa a mode usuari.

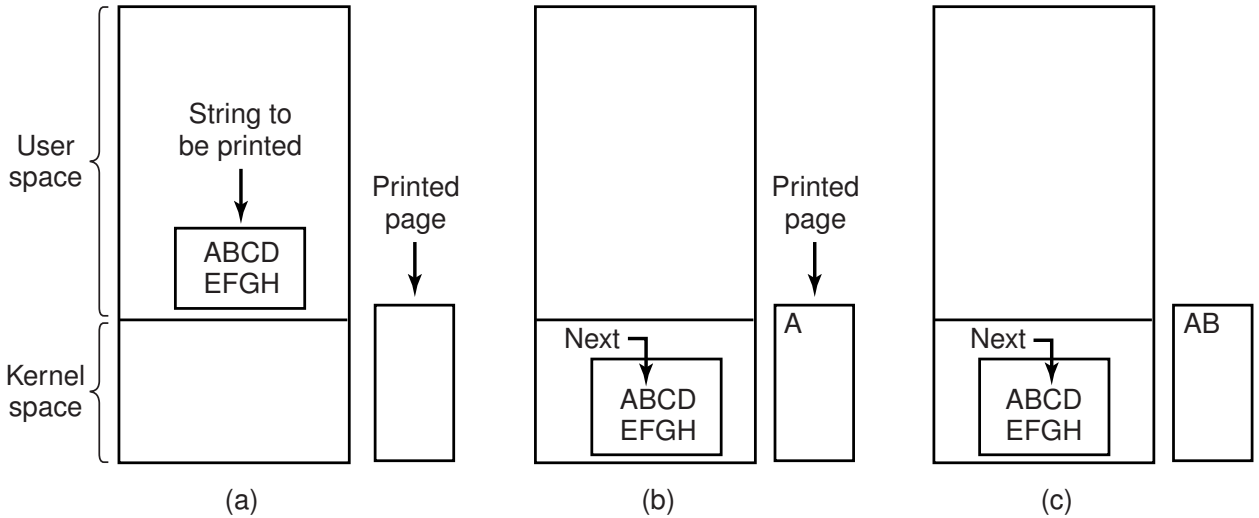
Programació de dispositius d'entrada i sortida

Hi ha fonamentalment tres formes diferents per realitzar entrada i sortida

- Entrada i sortida programada
- Entrada i sortida per interrupcions
- Entrada i sortida per DMA

Entrada i sortida programada

- Suposem que des d'un procés d'usuari es vol imprimir una cadena a la impressora



Programació de dispositius d'entrada i sortida

Entrada i sortida programada

1. L'usuari crida a una funció del sistema operatiu per imprimir la cadena.
2. El sistema operatiu copia la cadena a imprimir a un *buffer* intern de l'espai del sistema operatiu.
3. Realitza el següent bucle fins imprimir tota la cadena.
   - Esperar que el dispositiu estigui preparat per acceptar noves dades.
   - Envia un caràcter a la impressora.

Aquest mètode s'anomena polling o busy waiting.

Programació de dispositius d'entrada i sortida

Entrada i sortida programada

- La CPU està ocupada fins que s'envia tota la cadena de caràcters a la impressora.
- El sistema operatiu no podrà fer una altra tasca (p.ex. canvi de context a un altre procés) mentre estigui fent aquesta tasca.
- Aquesta tècnica s'utilitza encara avui en dia. Evita que arribin moltes interrupcions a la CPU.

Programació de dispositius d'entrada i sortida

Entrada i sortida per interrupcions

- Es programa el dispositiu – la impressora — perquè produeixi una interrupció quan estigui preparat.
- Cada cop que es produeix una interrupció del dispositiu, la CPU hi envia un nou caràcter.

Observar

- La idea és la mateixa que abans amb l'avantatge que ens estalviem haver d'esperar que el dispositiu estigui llest.
- Un cop enviat el caràcter, el sistema operatiu pot realitzar altres tasques (fins que es produeix la interrupció).

Programació de dispositius d'entrada i sortida

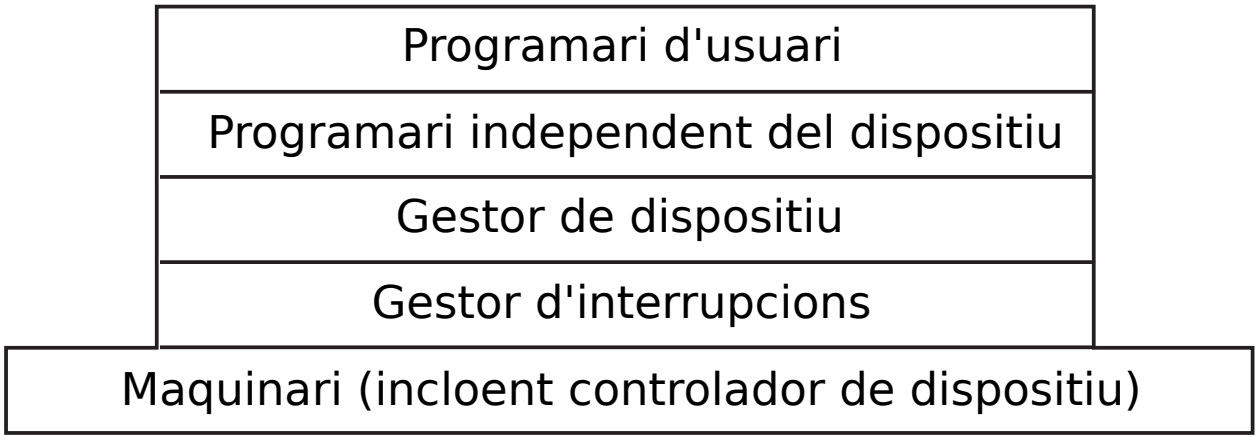
Entrada i sortida per DMA

- La feina de transferir les dades al dispositiu es realitza pel controlador de DMA, i no la CPU (com abans).
- El controlador DMA és lent comparat amb la CPU; en determinats casos la programació d'entrada i sortida programada o per interrupcions pot ser millor.

Programari d'entrada i sortida

El programari d'entrada i sortida a un sistema operatiu acostuma a estar organitzat en 4 capes, tal com es mostra aquí.

Cada capa té assignada una determinada tasca a realitzar i ofereix una interfície a les capes adjacents.



Programari d'entrada i sortida

Capa de gestor de dispositiu (*device driver*)

- Recordar que tot maquinari d'entrada i sortida té un controlador de dispositiu amb uns registres per controlar-lo.
  - El controlador de ratolí indica als seus registres cap a on s'ha mogut i quins botons hi ha polsats.
  - El controlador de disc permet situar el braç en un sector determinat i llegir o escriure-hi dades.
- El gestor de dispositiu és un programari específic que permet controlar el controlador de dispositiu associat.
  - Típicament l'escriu el propi fabricant del dispositiu.
  - Cada gestor de dispositiu gestiona un tipus de dispositiu o, a tot estirar, un conjunt de dispositius relacionats.

Programari d'entrada i sortida

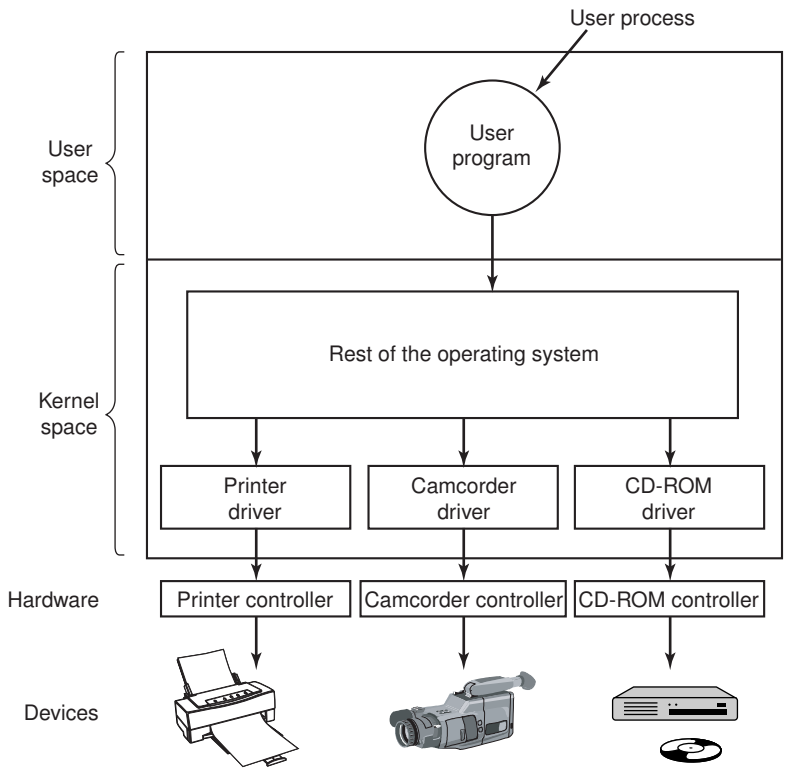
Capa de gestor de dispositiu (*device driver*)

- A la majoria dels sistemes operatius actuals (Windows, Unix, Linux, MacOs) els gestors de dispositius resideixen al mateix sistema operatiu, executant en mode *kernel*. Una errada de programació al gestor pot fer que tot el sistema operatiu quedi penjat.
- Idealment els gestors s'haurien d'executar fora del sistema operatiu, en mode usuari, amb crides a sistema per accedir als registres del controlador (Mach, GNU Hurd). Una errada de programació al gestor no afecta al sistema operatiu.

Programari d'entrada i sortida

Capa de gestor de dispositiu (*device driver*)

- Els dissenyadors d'un sistema operatiu defineixen una interfície estandarditzada que els gestors de dispositiu han d'implementar (una pels dispositius de bloc i una altra pels dispositius de caràcter).



Programari d'entrada i sortida

Capa de gestor de dispositiu (*device driver*)

El gestor de dispositiu

- Rep instruccions del sistema operatiu per realitzar una determinada tasca.
- Introdueix les peticions d'accés al dispositiu en una llista (p.ex. FIFO, per prioritat, etc.).
- Executa cada petició comunicant-se amb el controlador associat.
  - Si el dispositiu és lent (p.ex. disc), el gestor demana al controlador que produeixi una interrupció en acabar. Seguidament el gestor s'adorm (p.ex. fent servir un semàfor).
  - Si el dispositiu és ràpid (p.ex. pantalla, disc RAM, etc.) no cal fer-ho.
- En acabar cada petició es retorna al sistema operatiu el resultat. Si hi ha operacions pendents, el gestor de dispositiu atén les següents peticions.

Programari d'entrada i sortida

Capa de gestor de dispositiu (*device driver*)

- El gestor de xarxa introdueix els bytes a enviar en una cua FIFO (no s'han de reordenar les dades).
- El gestor de discos ha de planificar adequadament les peticions d'accés a disc. Això és per causa que un disc té un braç mecànic que cal moure d'una posició a l'altre. Cal realitzar una bona planificació que minimitzi el moviment dels capçals per obtenir un bon rendiment.

Capa de gestor de dispositiu (device driver)

- Política de planificació en discos: suposem que el disc és a la posició 6 i volem accedir a les posicions 20, 2, 56 i 7 del disc. Quina política d'accés és la millor ?
- Política FCFS (First Come First Served): l'ordre en què se serveixen les peticion és

$$6 \quad \rightarrow \quad 20 \quad \rightarrow \quad 2 \quad \rightarrow \quad 56 \quad \rightarrow \quad 7$$

El desplaçament total és

$$|6-20|+|20-2|+|2-56|+|56-7| = 135$$

Programari d'entrada i sortida

Capa de gestor de dispositiu (*device driver*)

- Política de planificació en discos: suposem que el disc és a la posició 6 i volem accedir a les posicions 20, 2, 56 i 7 del disc. Quina política d'accés és la millor ?
- Política SSF (Shortest Seek First): se serveixen les peticions que està més a prop del capçal. L'ordre és

$$6 \quad \rightarrow \quad 7 \quad \rightarrow \quad 2 \quad \rightarrow \quad 20 \quad \rightarrow \quad 56$$

El desplaçament total és

$$|6-7|+|7-2|+|2-20|+|20-56| = 60$$

Aquest esquema però pot fer que s'endarrereixin de forma indefinida determinades peticions.

Programari d'entrada i sortida

Capa de gestor de dispositiu (*device driver*)

- Política de planificació en discos: suposem que el disc és a la posició 6 i volem accedir a les posicions 20, 2, 56 i 7 del disc. Quina política d'accés és la millor ?
- Política SCAN: es mouen els capaçals d'un extrem a l'altre del disc, servint les peticions que vagin en aquest sentit. Suposem que som a la posició 6 i el moviment és ascendent

$$6 \rightarrow 7 \rightarrow 20 \rightarrow 56 \rightarrow \text{canvi de sentit} \rightarrow 2$$

El desplaçament total és

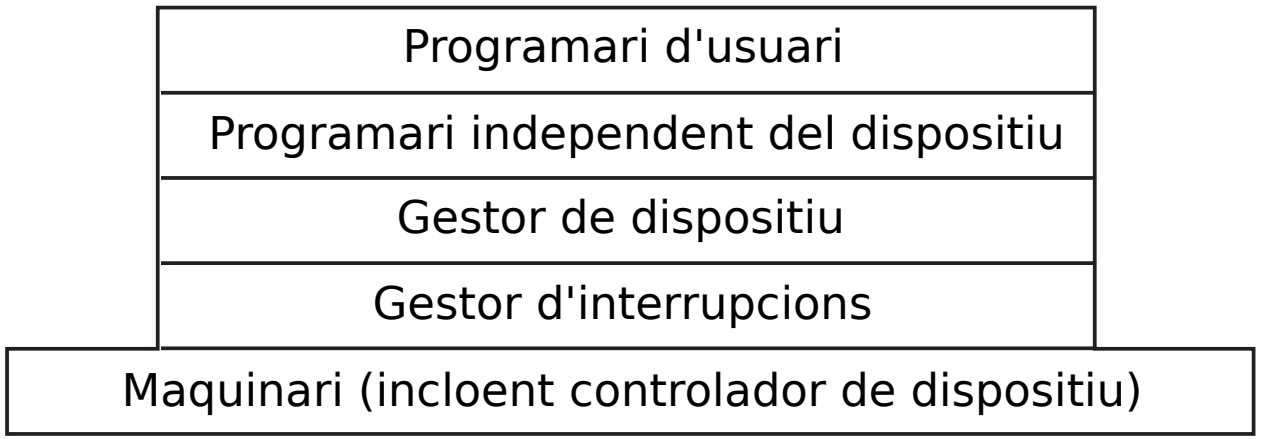
$$|6-7|+|7-20|+|20-56|+|56-2| = 104$$

Aquest tipus d'esquema és l'utilitzat als sistemes operatius.

Programari d'entrada i sortida

El programari d'entrada i sortida a un sistema operatiu acostuma a estar organitzat en 4 capes, tal com es mostra aquí.

Hem vist la capa del gestor de dispositiu. Centrem-nos a la resta de capes.



Programari d'entrada i sortida

Capa de gestor d'interrupcions

- Ocasionalment l'entrada i sortida programada és útil. La majoria de les vegades, però, caldrà utilitzar interrupcions.
- En realitzar-se una operació d'entrada i sortida bloquejant el gestor de dispositiu realitza aquestes tasques:
  1. Notifica al dispositiu que produeixi una interrupció en finalitzar.
  2. El gestor de dispositiu s'adorm, per exemple, a un semàfor[1].
- En produir-se una interrupció és el gestor d'interrupcions qui la gestiona
  1. Identifica quin dispositiu ha realitzat la interrupció
  2. Desperta el gestor de dispositiu corresponent. El gestor de dispositiu pot continuar l'execució.

[1] Veurem els semàfors a la 2a part de l'assignatura!

Capa independent del dispositiu

Les tasques realitzades a la capa independent són, generalment,

- Oferir una interfície uniforme de les funcions dels gestors a la resta del sistema operatiu.
- Donar un nom al dispositiu. A Linux cada dispositiu està identificat per dos nombres sencers (major i minor device).
- Oferir protecció perquè només els usuaris admesos hi puguin accedir.
- Fer el *buffering* de les dades.
- Gestionar els errors d'entrada i sortida.

Anem a veure amb més detall algun d'aquests punts.

Programari d'entrada i sortida

Capa independent del dispositiu

Cada dispositiu s'identifica mitjançant dos nombres sencers únics

- Aquests són els gestors de dispositius instal·lats i el seus corresponent major device

  ```
  cat /proc/devices
  ```

- Cada gestor de dispositiu pot gestionar diversos dispositius (identificats amb el minor device)

  ```
  ls -l /dev
  ```

  al costat de cada dispositiu apareix el major i minor device. Els permisos permeten controlar qui té dret per accedir-hi.

Programari d'entrada i sortida

Capa independent del dispositiu

Cada dispositiu s'identifica mitjançant dos nombres sencers únics

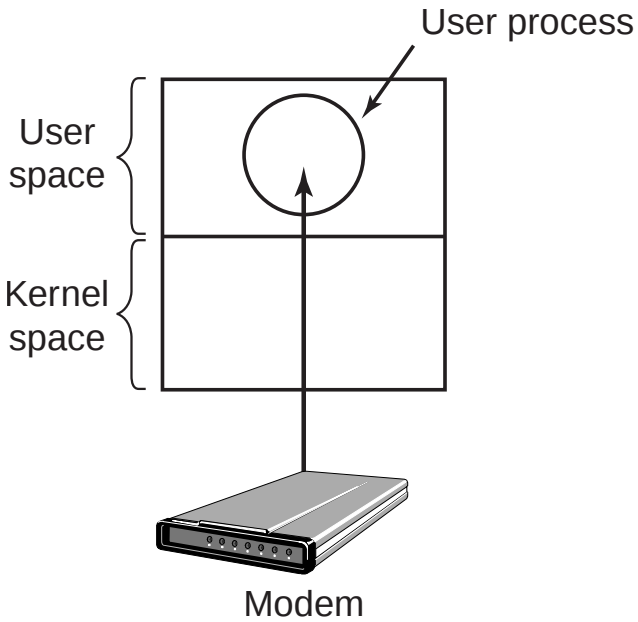
- Executar el codi `exemple1.c` i fer

  `cat /proc/<pid>/maps`

  la sortida ens indica, per a cada zona del mapa de memòria del procés, amb quin dispositiu (major, minor) i fitxer es fa la memòria virtual.

Capa independent del dispositiu

La capa independent també s'encarrega de fer el *buffering* de les dades. Per què cal fer-ho ?

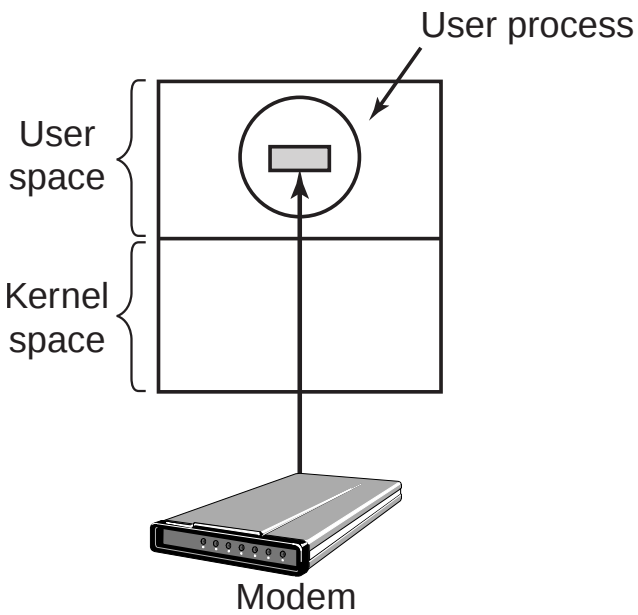


- L'usuari demana un caràcter al dispositiu.
- En arribar el caràcter el sistema operatiu copia el caràcter rebut a l'espai de l'usuari.
- L'usuari demana el següent caràcter, i així successivament.

- Aquesta solució és ineficient, no és una bona forma de procedir.

Capa independent del dispositiu

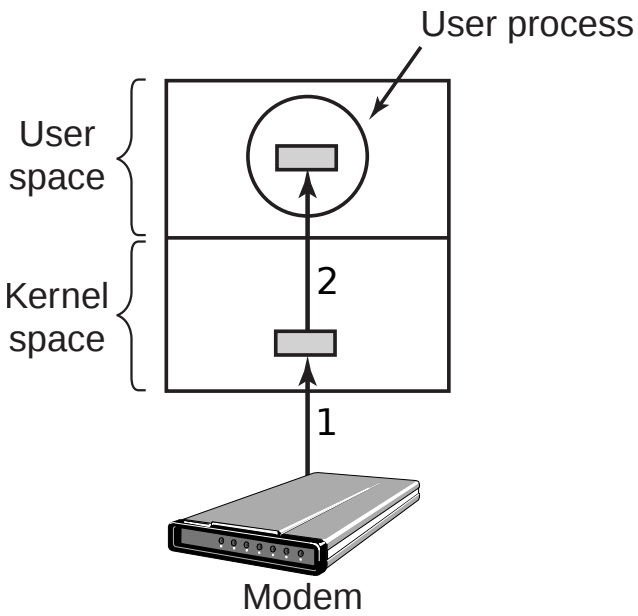
La capa independent també s'encarrega de fer el *buffering* de les dades. Per què cal fer-ho ?



- L'usuari proveeix un *buffer* de n caràcters i demana n caràcters al dispositiu.
- El sistema operatiu copia els caràcters rebuts a l'espai de l'usuari.
- Aquesta solució és més eficient que l'anterior. Què passa però si la pàgina de l'usuari no es troba a la RAM ?

Capa independent del dispositiu

La capa independent també s'encarrega de fer el *buffering* de les dades. Com ho fa ?

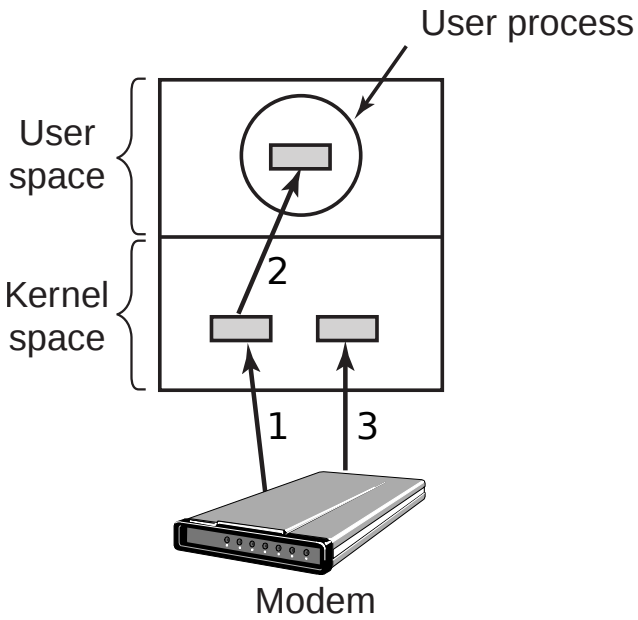


- El sistema operatiu assigna, per a l'operació, un *buffer* a l'interior. Les dades rebudes es copien en aquest *buffer*.
- En tenir totes les dades al *buffer* es copien a l'espai de l'usuari (si cal es porta la pàgina a RAM).
- Es complica la lògica del sistema operatiu.

- Què passa si mentre es fa la copia arriben noves dades ?

Capa independent del dispositiu

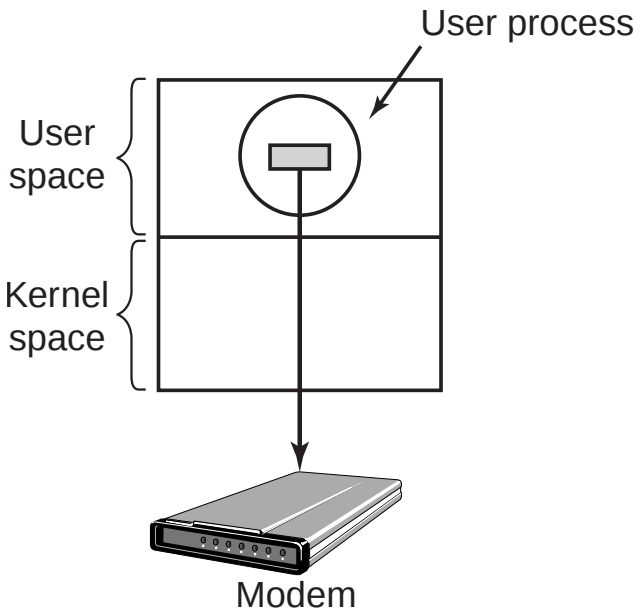
La capa independent també s'encarrega de fer el *buffering* de les dades. Com ho fa ?



- S'utilitza un *buffer* doble (per a cada operació d'entrada i sortida). Les dades rebudes es copien en el primer *buffer*.
- En copiar les dades rebudes a l'espai de l'usuari es poden inserir noves dades al segon buffer.

Capa independent del dispositiu

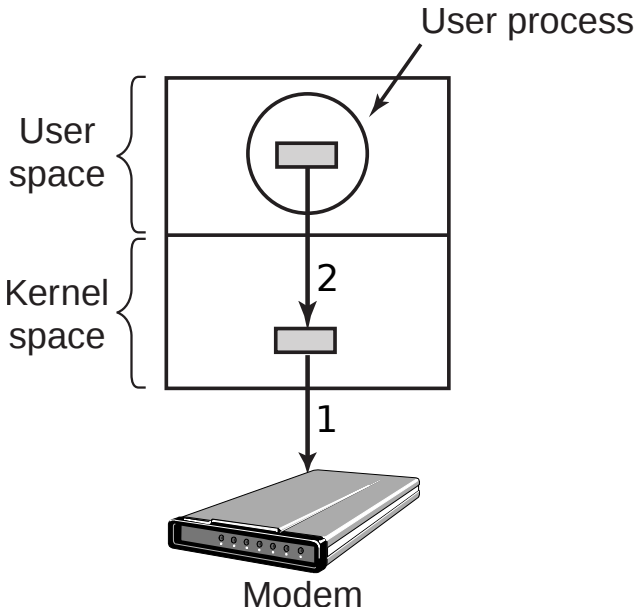
El *buffering* també és molt important en escriure dades al dispositiu. Suposem que l'usuari vol escriure un *buffer* de n caràcters al dispositiu. Per què cal fer-ho ?



- L'usuari fa la crida al sistema operatiu per escriure les dades.
- El sistema operatiu bloqueja el procés que fa la crida fins que les dades han estat escrites.
- Aquesta solució és poc eficient.

Capa independent del dispositiu

El *buffering* també és molt important en escriure dades al dispositiu. Suposem que l'usuari vol escriure un *buffer* de n caràcters al dispositiu. Com ho fem ?

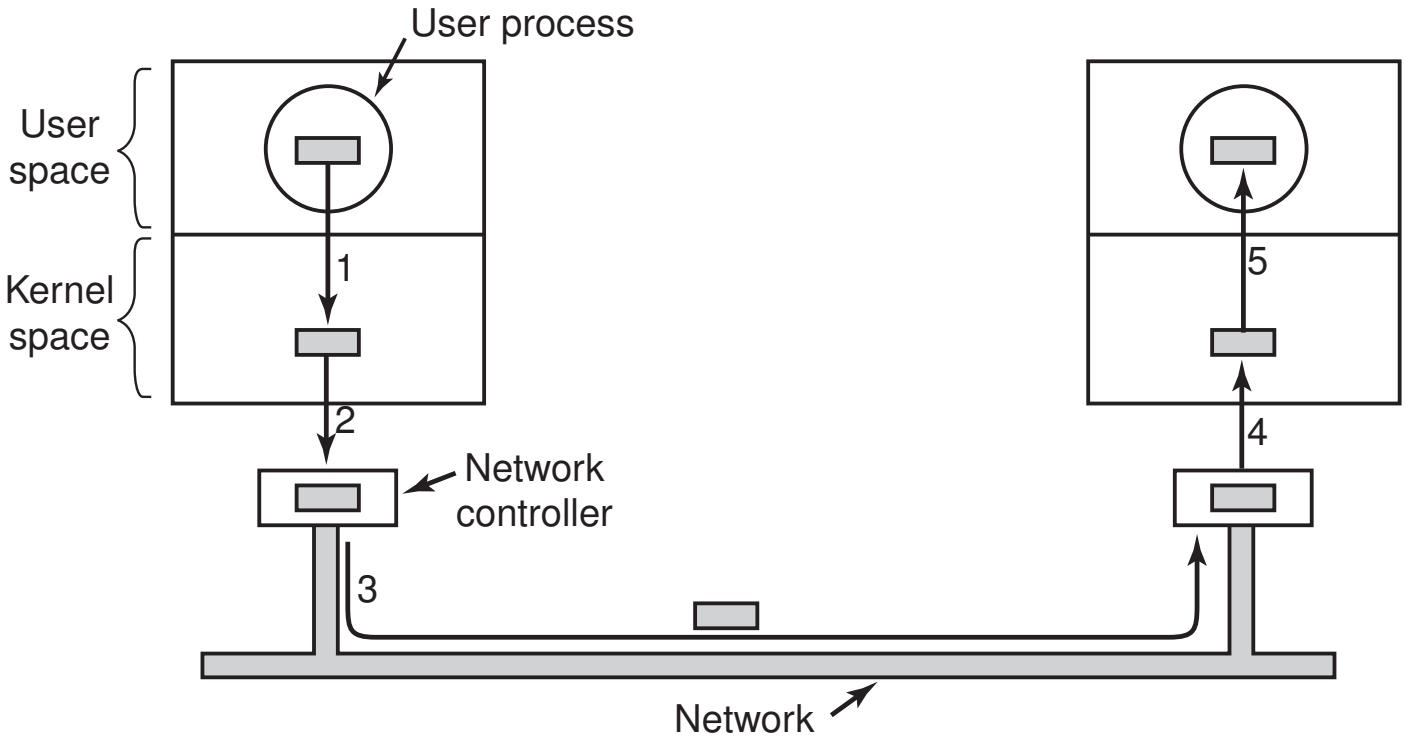


Segona solució

- L'usuari fa la crida al sistema operatiu per escriure les dades.
- El sistema operatiu copia les dades a un *buffer* intern i pot retornar de seguida. Es fa la petició per escriure les dades al dispositiu.
- L'operació d'escriptura real al dispositiu es realitza més endavant.
- Aquesta solució és eficient.

Capa independent del dispositiu

El *buffering* és una tècnica molt utilitzada, però pot afectar el rendiment si les dades s'han de copiar masses vegades.



Es copien les dades al controlador de xarxa que assegura uniformitat en la velocitat d'enviament de dades.

Programari d'entrada i sortida

Capa independent del dispositiu

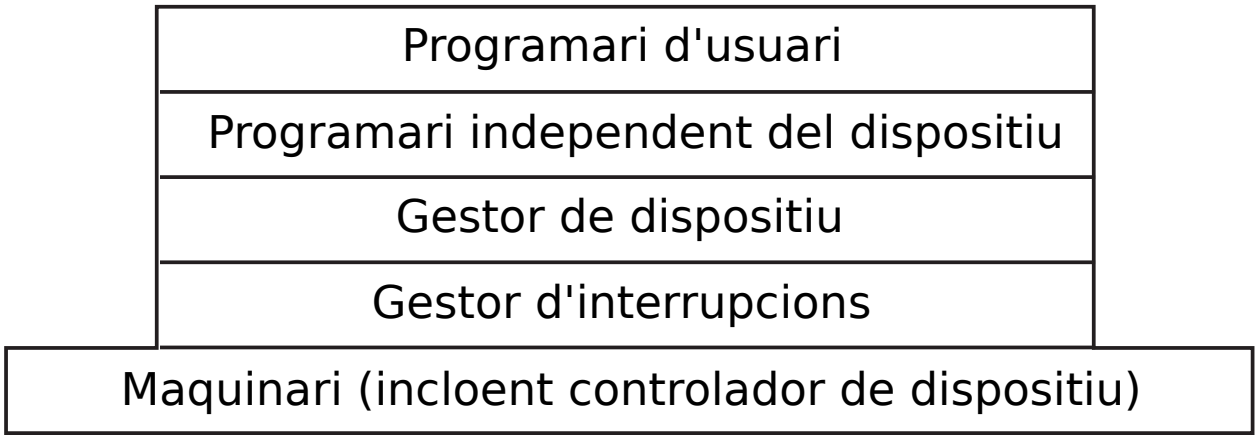
La capa independent també s'encarrega de la gestió d'errors, per exemple

- En intentar escriure a un dispositiu de només entrada (p.ex. teclat)
- En intentar llegir o escriure a sectors de disc defectuosos, o llegir d'una càmera apagada.
- En haver tret un dispositiu USB sense desmuntar-lo correctament (el gestor de dispositius haurà de descartar les peticions pendents).
- Etc...

Programari d'entrada i sortida

El programari d'entrada i sortida a un sistema operatiu acostuma a estar organitzat en 4 capes, tal com es mostra aquí.

Ens falta la capa d'usuari.



Programari d'entrada i sortida

Capa de programari d'usuari

L'usuari disposa de funcions del sistema operatiu per escriure i llegir dades

- L'operació d'escriptura és, en general, no bloquejant. En realitzar una escriptura el sistema operatiu copia les dades a un *buffer* intern i retorna de seguida. L'escriptura al dispositiu es fa més endavant.
- L'operació de lectura és, en general, bloquejant. En retornar de l'operació de lectura hi haurà al vector d'usuari les dades demanades.
- El sistema operatiu ofereix també operacions (de lectura) no bloquejants. En cridar a una operació de lectura no bloquejant es fa la petició de llegir i la funció retorna de seguida. Més endavant l'usuari tindrà les dades llegides al vector.

Programari d'entrada i sortida

Capa de programari d'usuari

- La següent funció és una crida a la llibreria estàndard d'entrada i sortida

  ```
  count = write(fd, vector, nbytes);
  ```

  La funció `write` fa poc més que posar els paràmetres en el lloc apropiat per a la crida a sistema i realitzar la crida en sí (amb una interrupció de software).

- La següent funció imprimeix una cadena per pantalla

  ```
  printf("El valor es %d\n", i);
  ```

  La funció `printf` genera, a l'espai d'usuari, la cadena a imprimir. Un cop generada es fa la crida al sistema operatiu (amb un `write`).

Programari d'entrada i sortida

Capa de programari d'usuari

- La funció per reservar memòria

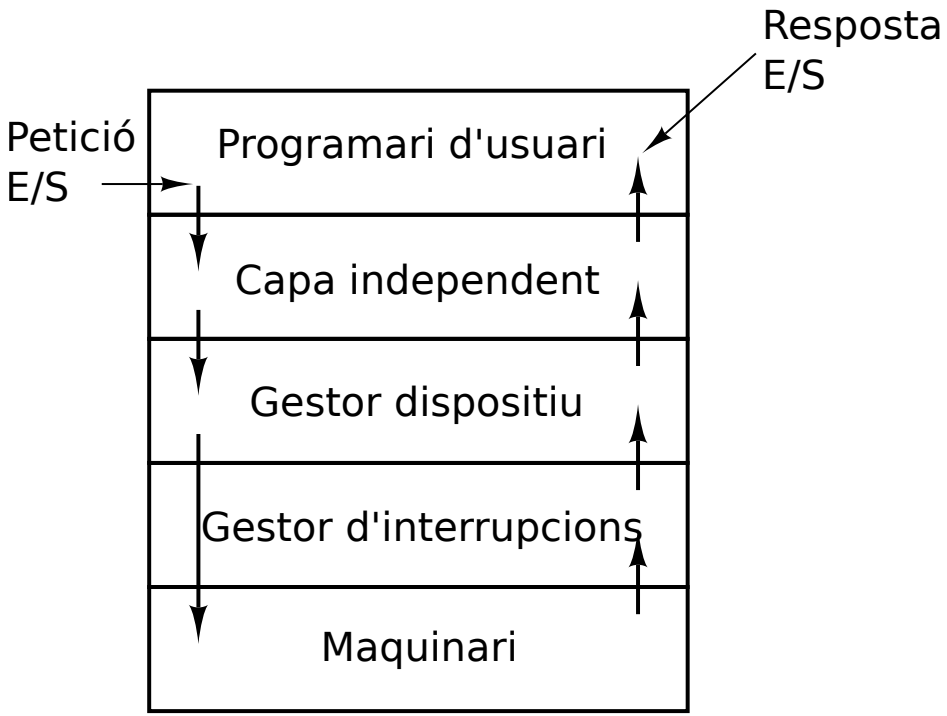
  ```
  v = malloc(sizeof(int) * 10);
  ```

  no és realment una crida a sistema. La funció `malloc`
  - Utilitza la crida a sistema `sbrk` per demanar memòria al sistema operatiu (en blocs de p.ex. 128 KBytes).
  - La funció `malloc`, que executa en mode usuari, utilitza llistes per tal de gestionar la memòria assignada.

Programari d'entrada i sortida

Capa de programari d'usuari

Aquest és el flux d'execució en realitzar un usuari una operació d'entrada i sortida.



Programari d'entrada i sortida

Capa de programari d'usuari

Flux d'execució per a una instrucció de lectura.

1. L'usuari crida a l'operació de lectura.
2. La capa independent comprova si el que es demana es troba ja el buffer intern. En cas afirmatiu, retorna les dades demanades de seguida. En cas negatiu, fa la petició al gestor.
3. El gestor programa el dispositiu per realitzar la lectura i s'adorm.
4. En acabar el dispositiu, interromp i el gestor d'interrupcions desperta al gestor.
5. El gestor extreu les dades del dispositiu.
6. La capa independent copia les dades al *buffer* d'intern i d'allà al *buffer* d'usuari.
7. El procés d'usuari es desperta i pot continuar l'execució.

Programari d'entrada i sortida

Capa de programari d'usuari

Flux d'execució per a una instrucció de d'escriptura.

1. L'usuari crida a l'operació d'escriptura.
2. La capa independent copia les dades al *buffer* intern i permet que la funció retorni de seguida. Es fa la petició al gestor de dispositiu.
3. El gestor de dispositiu inclou la petició a la seva llista de peticions.
4. El gestor extreu una petició de la llista i programa el dispositiu.
5. En acabar el dispositiu, interromp i el gestor d'interrupcions desperta al gestor.
6. El gestor torna a passar al pas 4.

Observar: en escriure el procés que fa la crida no s'adorm!