





TEMA 4: ENTRADA I SORTIDA

 Assign	
 Status	

Definicions

(E/S serà ES d'ara en endavant)

L'entrada sortida és la transferència d'informació entre un procés i l'exterior. Moltes vegades, l'ES és l'única tasca del procés, com per exemple en un processador de texts.

S'ha de saber gestionar l'ES per un ús correcte, compartit i eficient dels recursos.

Dispositius

Tenim una gran quantitat de dispositius d'ES, que es poden classificar de la següent forma:

- Segons tipus: lògic, físic, etc.
- Segons velocitat: teclat, disc dur, etc.
- Segons flux d'accès: ratolí (byte a byte), DVD (bloc a bloc), etc.
- I molts més.

Independència de dispositius

L'objectiu principal es que un procés sigui independent del dispositiu amb que interaccioni.

Per a fer-ho, hem de tenir unes operacions d'ES uniformes. Contem amb un accés a tots els dispositius amb les mateixes crides a sistema.

Utilitzarem també dispositius virtuals, i posteriorment ja es farà la traducció al dispositiu adient.

Per últim, se'ns dona la possibilitat de canviar l'ES d'un procés:

```
./programa < disp1 > disp2
```

Habitualment, s'ha de dissenyar en tres nivells: virtual, lògic i físic.

La virtual ens aporta independència, el procés treballa amb dispositius virtuals sense saber quin és el que hi ha realment.

La lògica ens aporta compartició de dispositius, ja que es poden produir accessos concurrents al mateix dispositiu. No és fins al moment d'executar un programa que es determina sobre quins dispositius es treballa.

El nivell físic, separa les operacions en software de l'implementació. El codi físic sol ser gairebé sempre en ensamblador.

Dispositiu virtual

El nivell virtual aïlla a l'usuari de la complexitat de gestió del dispositius físics.

S'estableix una correspondència entre nom simbòlic i l'aplicació d'usuari, a través d'un dispositiu virtual.

Un nom simbòlic és la representació al sistema d'un dispositiu: `/dev/dispX` o bé `/dispX` un nom de fitxer

Un dispositiu virtual representa un dispositiu en ús d'un procés. El dispositiu virtual (nombre enter) és equivalent al canal, i alhora, al descriptor de fitxer. Els processos tenen 3 canals estàndards:

- Entrada estàndard canal 0 (`stdin`)
- Sortida estàndard canal 1 (`stdout`)
- Sortida error estàndard canal 2 (`stderr`)

Les crides a sistema de transferència de dades utilitzen com identificador el dispositiu virtual

Dispositiu lògic

Estableix correspondència entre dispositiu virtual i físic. Gestiona els dispositius tinguin o no representació física. Ofereix compartició als dispositius físics, és a dir, accés concurrent.

En nivell lògic, es tenen en compte els permissos i, a Linux, s'identifiquen amb un nom de fitxer.

Dispositiu físic

Implementa a baix nivell les operacions de nivell lògic. Es tradueixen els paràmetres de nivell lògic a paràmetres concrets. Inicialitzem el dispositiu comprovant si està lliure, en cas contrari, és posa a la cua. Realitza la programació de l'operació demanada, i espera, si és necessari, a que finalitzi la operació. Retorna els resultats o informa sobre algun eventual error.

A Linux, un dispositiu físic s'identifica amb tres paràmetres:

- Tipus: Block / Character
- Amb dos nombres:
 - Major: Indica el tipus de dispositiu
 - Minor: Instància concreta respecte al major.

Device Drives

Per oferir independència, es defineix el conjunt d'operacions que ofereixen els dispositius. És un superconjunt de totes les operacions que es poden oferir per accedir a un dispositiu físic. No tots els dispositius ofereixen totes les operacions. Quan es tradueix de dispositiu virtual a lògic i físic, d'aquesta manera està definit quines operacions corresponen.

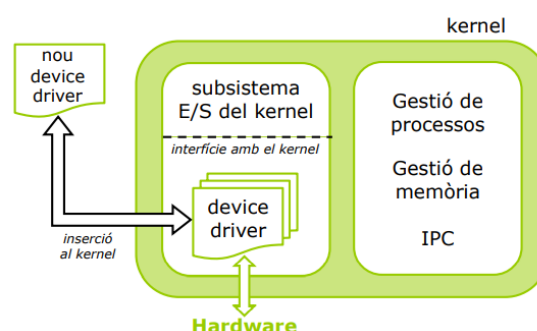
Els programadors de Sistemes Operatius no poden generar codis per tots els dispositius, models,... Cal que el fabricant proporcioni el conjunt de rutines de baix nivell que implementen el funcionament del dispositiu.



Al codi i dades per accedir a un dispositiu se'l coneix com Device driver. Seguint l'especificació de la interfície d'accés a les operacions d'ES definida pel SO

Per tal d'afegir un nou dispositiu podem, o recompilar el Kernel, o afegir nous dispositius sense recompilar el Kernel, això si, cal que el SO ofereixi un mecanisme per afegir dinàmicament codi i dades al Kernel.

Identifiquem unes operacions comunes i les diferències específiques s'encapsulen en mòduls del SO: device driver. S'aïlla, la resta del kernel, de complexitat de la gestió dels dispositius. A més, protegeix el Kernel en front d'un codi escrit per altres.



El controlador de un dispositiu genèric és el conjunt de rutines que gestionen un dispositiu, i que permeten a un programa interactuar amb aquest dispositiu:

- Respecten la interfície definida pel SO → Cada SO té definida la seva pròpia interfície
- Implementen tasques dependents del dispositiu → Cada dispositiu realitza tasques específiques
- Contenen habitualment codi de baix nivell → Accés als ports d'ES, gestió d'interrupcions...
- Queden encapsulades en un arxiu binari.

Inserció dinàmica de codi

Els kernels actuals ofereixen mecanismes per afegir codis i dades al kernel sense haver de recompilar-lo tot, ja que això trigaria hores.

L'inserció es fa de forma dinàmica, és a dir, en temps d'execució.

Gestió de l'E/S a Linux

Tipus de fitxers a Linux

A linux, tots els dispositius s'identifiquen amb un fitxer, de diferents tipus:

- Block device
- Character device
- Directory
- FIFO
- Pipe
- Symmlink
- Regular File
- Socket

Creació de fitxers a Linux

La majoria de fitxers es poden crear mitjançant `mknod` , excepte directoris i softlinks.

Es pot utilitzar la comanda `mknod nom_fitxer "tipus" major minor` :

- Major i minor
- "tipus" = c → Dispositiu de caràcters
- "tipus" = b → Dispositiu de blocs
- "tipus" = p → Pipe

Estructures de dades del Kernel

L'inodo és una estructura de dades que conté tota la informació relativa a un fitxer (mida, tipus, proteccions, propietari...) Està tota la informació d'un fitxer excepte el nom, que està en un altre lloc. S'emmagatzema a disc, però hi ha una còpia a memòria per optimitzar el temps d'accés.

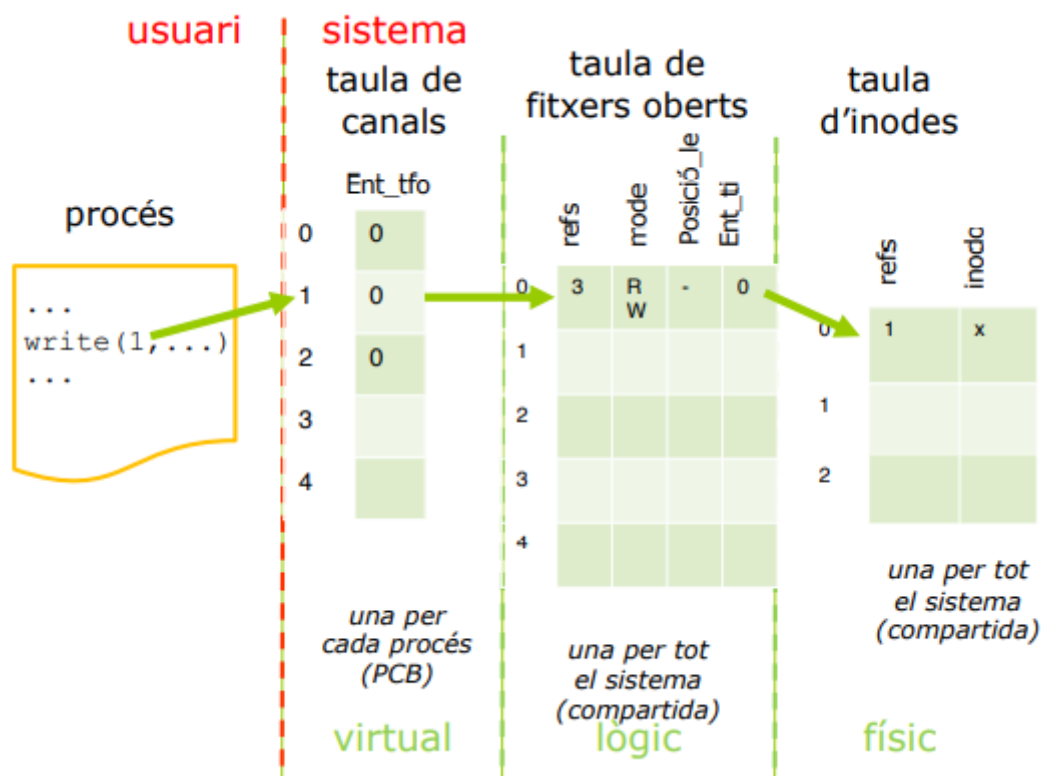
Si fem referència a les estructures de dades globals trobem:

- Taula de fitxers oberts (TFO)
 - Gestiona els fitxers en ús en tot el sistema.
 - Poden haver entrades compartides entre més d'un procés i per varies entrades del mateix procés.
 - Una entrada de la TFO referencia a una entrada de la TI.
 - Camps que assumirem: `num_referencies, mode, punter_le, num_entrada_ti`
- Taula d'inodes (TI)
 - Conté informació sobre cada objecte físic obert, incloent les rutines del DD

- És una còpia en memòria de les dades que tenim al disc
- Camps que assumirem: `num_referencies, dades_inode`

En canvi, si fem referència a les estructures per un procés:

- Taula de canals (TC): per cada procés (es guarda a la PCB)
- Indica a quins fitxer estem accedint
- S'accedeix mitjançant el canal, que és un índex a la TC
- El canal és el dispositiu virtual
- Un canal referència a una entrada de la taula de fitxers oberts (TFO)
- Camps que assumirem: `num_entrada_TFO`



Crides a sistema bàsiques

Operacions bloquejants i no bloquejants

Un procés que consumeix CPU sense estar fent res, serà bloquejat pel SO, i passarà d'estat RUN a BLOCKED.

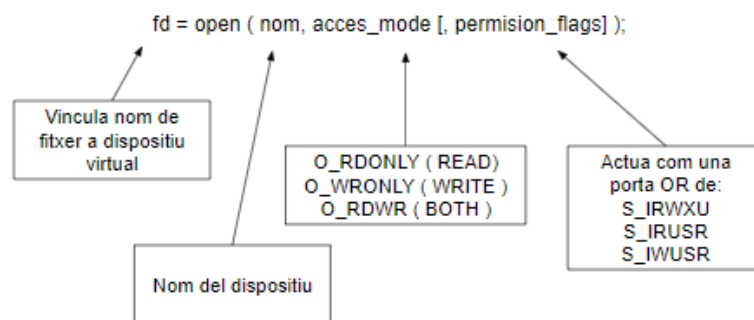
Operació bloquejant: Un procés realitza una transferència de N bytes i espera a que acabi. Retorna el nombre de bytes que s'han transferit:

- Si les dades es poden disposar al moment, es realitza la transferència i el procés retorna al moment, sense perdre temps.
- Si les dades no estan disponibles, el procés es bloqueja:
 1. Canvia de RUN a BLOCKED
 2. Es posa en execució el procés que toca (Segons la política de planificació)
 3. Quan arriben les dades es produeix una execució per a despertar el procés
 4. El procés es posarà de nou en execució quan la política de planificació li ho permeti.

Operació no bloquejant: El procés sol·licita una transferència, disposant de les dades que disposi, i retorna immediatament aquelles dades, siguin totes les necessàries o no.

Operacions bàsiques d'ES

Open



En el cas que vulguéssim truncar en contingut d'un fitxer afegirem el flag `O_TRUNC` a l' `acces_mode` . Dos exemples de l'ús d'aquesta instrucció serien:

`open ("X", O_RDWR|O_CREAT, S_IRUSR|S_IWUSR)` → Si el fitxer no existia es crea, si existeix no té cap efecte.

`open ("X", O_RDWR|O_CREAT|O_TRUNC, S_IRWXU)` → Si el fitxer no existia el crea, si existia s'alliberen les seves dades i es posa el tamany a 0 bytes.

La instrucció `open` fa efecte sobre les estructures de dades internes del sistema. Ocupa un canal lliure de la TC, sempre serà el primer disponible. Ocupa una nova

entra de la TFO i, s'associa aquestes estructures al DD corresponent.

Read

`n = read (fd, buffer, count);`

Demana la lectura de count caràcters del dispositiu associat al canal fd. Si el número de caràcters és superior a count, llegira els primers count caràcters. Contrariament, si n'hi ha menys, llegira els caràcters disponibles. I per últim, si no hi ha caràcters actuarà d'una forma, o d'una altre depenent del dispositiu, com per exemple:

- Es pot bloquejar, esperant a que n'hi hagi
- Pot retornar amb cap caràcter llegit.

Quan arribem al EOF (end of file), l'operació retorna sense cap caràcter llegit. La posició de l/e s'avança automàticament tantes posicions com bytes llegits.

Write

Demana al escriptura de count caràcters del dispositiu associat al canal fd. Si l'espai disponible és superior a count, escriura count caràcters. Contrariament, si n'hi ha menys, escriura els caràcters disponibles. I per últim, si no hi ha espai actuarà d'una forma, o d'una altre depenent del dispositiu, com per exemple:

- Es pot bloquejar, esperant a que n'hi hagi
- Pot retornar amb cap caràcter llegit.

La posició de l/e s'avança automàticament tantes posicions com bytes llegits.

Dup/dup2/close

`newfd = dup (fd)` → Duplica un canal. Ocupa el primer canal lliure, que ara contindrà una còpia de la informació del canal original fd. Retorna l'identificador del nou canal.

`newfd = dup2(fd, newfd)` → Igual que dup, però el canal duplicat és forçosament newfd. Si newfd era un canal vàlid, prèviament es tanca.

`close (fd)` → Allibera el canal fd i les estructures associades dels nivells inferiors. Cal tenir en compte que diferents canals poden apuntar a la mateixa entrada de la TFO, per tant, tancar un canal significaria decrementar el contador de referències a aquella entrada. El mateix passa amb apuntadors a la llista de inodes.

Pipe


```
pipe(fd_vector);
```

- Crea una pipe sense nom. Retorna 2 canals, un de lectura i un d'escriptura.
- No utilitza cap nom del VFS (pipe sense nom) i, per tant, no executa la crida open.
- Només pot ser utilitzada per comunicar el procés que crea la pipe i qualsevol descendent d'aquest.

Per crear una pipe amb nom caldrà fer: `mknod + open`

Internament, també crea 2 entrades a la taula de fitxers oberts, una de lectura i una d'escriptura.

Una pipe s'utilitza per comunicar dos processos, i tot i que es bidireccional, s'ha de usar per cada un dels dos processos en una única direcció.

Si estem llegint, es bloqueja el procés fins que la pipe tingui dades, i si estem escrivint, es bloqueja el procés si la pipe esta plena fins que es buidi.

Es bloqueja també el procés si no es tanquen els canals que no es fan servir, ja que estariem esperant llegir o escriure per canals que no rebràn dades.

Iseek

La posició de lectura i, escriptura es modifica manualment per l'usuari. Permet fer accessos directes a posicions concretes de fitxers de dades. S'inicialitza a 0 l'open, i s'incrementa automàticament al fer lectura i escriptura. El podem moure manualment amb aquesta crida.



Pa los amigos: Reposiciona el punter de lectura / escriptura de un fitxer.

Gestió de característiques dels dispositius




Tot i que les crides a sistema son uniformes, els dispositius no.

Existeixen crides que modifiquen característiques de dispositius lògics i virtuals.

- `ioctl (fd, cmd [, ptr])` per a dispositius lògics
- `fcntl (fd, cmd, [, args])` per a dispositius virtuals

Terminal

La terminal té un buffer intern on guarda els caràcters que es teclejen. Per a modificarlos abans de ser tractats, tenim certes funcionalitats:

- : Esborra un caràcter.
- : Esborra una línia.
- : End of File / Final de l'entrada de dades.


Es pot tenir un buffer d'escriptura per implementar més funcionalitats, tals com "pujar N línies" o "avançar N paraules a la dreta".

Una terminal pot ser Canònica o No canònica. Es determina per si envia les dades després de processar-les o abans, respectivament.

Terminal Canònica

Funcionament per defecte.

En lectura, el buffer guarda els caràcters fins que s'envia l'acció. Si hi ha un procés bloquejat esperant caràcters, li en dona.

 provoca una interrupció a la lectura actual, encara que no s'hagi llegit cap caràcter.



Bastant segur de que també pot ser ^C, perquè ^D tanca la meua terminal.

En escriptura, s'escriu un bloc de caràcters i el procés no es bloqueja.

No obstant, els comportaments bloquejants poden ser modificats amb les crides a sistema adequades.

Pipes

En la lectura, s'agafen les dades que es necessitin, i en cas de no haver-hi, es bloqueja esperant a que n'arribin.

Si la pipe es buida i tots els canals d'escriptura estan tancats, es rep un EOF.

En escriptura, si hi ha espai a la pipe, s'utilitza, i si esta plena el procés es bloqueja esperant a que es buidi.

Sistemes de fitxers

Tasques del sistema de fitxers

El sistema de fitxers es responsable de organitzar els fitxers del sistema, garantir l'accés correcte als fitxers, gestionar espai lliure i ocupat pels fitxers de dades i, trobar o emmagatzemar les dades dels fitxers de dades.

En qualsevol cas, tots els fitxers tenen un nom que s'ha de emmagatzemar i gestionar. Els noms de fitxers s'organitzen en directoris.

Espai de noms: Directoris

Els directoris són les estructures lògiques que organitzen els fitxers. És un fitxer especial gestionat pel sistema operatiu, que permet enllaçar els noms dels fitxers amb els seus atributs.

Visió d'usuari

Els directoris s'organitzen de forma jeràrquica. Permeten que l'usuari classifiqui les seves dades. Els sistemes de fitxers organitzen els dispositius emmagatzemant, en un espai de noms global amb un únic punt d'entrada. Un directori sempre té, com a mínim, dos fitxers especials (referència al director actual, i al directori pare).

Noms de fitxers

Com que el nom del fitxer està separat de la informació (inode), a Linux es permet que un inode sigui referenciat per més d'un nom de fitxer. Hi ha dos tipus de vincles entre nom de fitxer i inode:

- Hard-link: És el vincle per defecte, fa una còpia del fitxer destí.
- Soft-link: És un fitxer especial, es crida de forma exclusiva, i crea un punter a la direcció del fitxer, no una còpia.



Si eliminem el fitxer vinculat, el Hard-Link no presentarà canvis ja que és una còpia, mentre que el Soft-Link retornarà error per fitxer desconegut.

La existència dels dos tipus d'enllaços influeix en la estructura del directori. No es permeten cicles amb hard-links, en canvi, sí que es permeten cicles amb soft-links.

Permisos d'accés a fitxers

El sistema de fitxers ens permet assignar diferents permisos als fitxers, es defineixen nivells d'accés i operacions que es poden fer.

Unitats de treball del disc

Un dispositiu d'emmagatzematge està dividit en sectors. La unitat d'assignament del SO és un bloc (1 bloc equival a 1 o més sectors).

Cada partició té la seva pròpia estructura de directoris i de fitxers independent de la resta de particions.

Gestió de l'espai ocupat

Per cada fitxer, el S.O. ha de saber quins són els seus blocs, ho aconsegueix mitjançant:

- Taula d'índexos: blocs assignats a un fitxer.
- Inodes del fitxer.

Gestió de l'espai lliure

- Llista de blocs lliures i llista d'inodes ocupats.
- Si fa falta un nou inode o bloc, s'agafa de la llista de lliures.

Metadades

Metadades persistents guardades a disc:

- Inodes i llista de blocs d'un fitxer
- Directoris
- Llista de blocs lliures
- Llista d'inodes lliures
- Altra informació necessària per al sistema de fitxers.

Entenem per superbloc la partició que conté les metadades del sistema de fitxers. Utilitzem memòria per a guardar aquestes metadades mentre estan en ús.

Relació entre crides a sistema i estructures de dades.

Open

Localitza l'inode del fitxer amb el que es vol treballar i el deixa a memòria. Si el directori està a la cache de directoris, simplement s'accedeix. En canvi, si el directori està a disc, s'ha de llegir i guardar a memòria.

Si localitzem l'inode cal comprovar si l'accés es correcte (no retorna erro). Si tenim un soft-link es llegeix el path del fitxer al que apunta i localitza l'inode corresponent. I per últim, cal modificar la taula de canals, la taula de fitxers oberts i la taula d'inodes.

Si no localitzem l'inode, salta un error per accedir a un fitxer que no existeix. No accedit a cap bloc de dades del fitxer objectiu.

Si utilitzem l'open per a crear un nou fitxer, s'ha de reservar i inicialitzar un nou inode. Actualitzar llista d'inodes lliures al superbloc, i actualitzar directori a on es crea el fitxer.

Read

Tenim a la taula d'inodes l'inode corresponent. Tenim a la TFO el valor del punter de lectura/escriptura.

Cal calcular els blocs involucrats, per a fer-ho, mirem si estem al final del fitxer, i altrament calculem els blocs.

- Dividim el valor del punter per la mida d'un bloc per saber el primer bloc.
- Amb el paràmetre de mida de syscall, podem calcular els blocs que queden per llegir.

Write

La diferencia entre write i read es que si l'escriptura es fa al final del fitxer, poder el fitxer necessita més blocs. Per tant, caldrà accedir a la llista de blocs lliures.

Close

Provoca l'actualització de l'inode amb els canvis que s'hagin fet. Com a mínim s'actualitzarà la data d'últim accés.