Entrada/sortida

José Ramón Herrero Zaragoza
Teodor Jové Lagunas
Enric Morancho Llena

PID_00169370

UOC
Universitat Oberta de Catalunya

Índex

Introducció

En aquest mòdul didàctic analitzem un dels elements necessaris en l'execució d'un programa: **l'entrada/sortida**. Aquesta permet que un procés pugui rebre o enviar informació des de l'exterior i cap a l'exterior.

Ens centrarem en l'anàlisi dels diversos tipus de dispositius, el tractament que en fa el sistema operatiu i la percepció que en té l'usuari mitjançant aquest tractament. Veurem que la varietat de dispositius fa desitjable independitzar els programes respecte als dispositius, és a dir, que un mateix codi pugui funcionar amb diferents dispositius sense ser modificat. Veurem com això es pot aconseguir mitjançant una interfície de sistema operatiu amb operacions uniformes que treballen en dispositius virtuals. Per tal de consolidar aquests coneixements mostrarem sistemes reals. Farem servir el sistema operatiu UNIX i l'analitzarem tant des del punt de vista de l'intèrpret de comandes com des del punt de vista del programador. Comentarem també el cas del Windows.

Objectius

En els materials didàctics facilitats en aquest mòdul trobareu les eines necessàries per a assolir els objectius següents:

1. Entendre el concepte d'*entrada/sortida* (E/S) d'un procés.

2. Conèixer la diversitat de dispositius que existeixen i els seus modes de treball.

3. Entendre la necessitat d'independitzar els programes dels dispositius amb els quals treballen.

4. Aprendre el concepte de dispositiu virtual.

5. Conèixer el concepte d'*independència* dels dispositius basats en els dispositius virtuals i en la uniformitat de les operacions.

6. Distingir els dispositius físics dels dispositius lògics i dels dispositius virtuals.

7. Aprendre el significat de *sessió de treball amb un dispositiu* i conèixer les operacions bàsiques que s'efectuen durant la sessió de treball.

1. El concepte d'*entrada/sortida* (E/S)

Un procés, com a contenidor dels recursos necessaris per a executar un programa, disposa d'un espai lògic d'adreces on guardar el codi, les dades i la pila[1]. Quan l'usuari demana executar un programa, el sistema operatiu crea un procés i, entre altres coses, li reserva memòria, hi carrega aquest programa i les seves dades i el deixa llest per a executar-se. Amb això, el procés pot realitzar operacions basades en tot el que té emmagatzemat a memòria. Amb això, però, el programa sempre treballaria amb les mateixes dades, que serien les especificades pel programador en fer el programa. Per qüestions de seguretat, el sistema operatiu s'encarrega d'aïllar la memòria de cada procés en execució de forma que cap d'ells no pugui interferir, ja sigui per error o per mala intenció, en l'execució d'altres processos o del sistema operatiu mateix. Per tant, un procés no pot, en principi, anar a buscar dades a altres zones de memòria externes al seu espai lògic.

(1)Recordeu que la pila serveix per a emmagatzemar les variables locals de les subrutines, així com els paràmetres passats en invocar-les (a menys que es passin a través de registres).

Quelcom similar passa a una altra escala amb un sistema computador, format a un nivell elemental pel processador i la memòria. Si només poguéssim usar el computador per a treballar amb el que té inicialment a la memòria, l'ús que en podríem fer seria molt limitat.

Existeixen dispositius que permeten **enviar informació des de l'exterior cap a la memòria** i altres que permeten **enviar informació des de la memòria cap a l'exterior** del computador. El primer cas s'anomena **entrada** d'informació, i els dispositius que permeten fer-ho els anomenem dispositius d'entrada. El segon s'anomena **sortida**, i es pot fer amb dispositius de sortida. Parlem, per tant, de fer **entrada/sortida**.

Estudiarem els dispositius d'entrada/sortida en l'apartat següent.

Un cop introduït el concepte d'*entrada/sortida al sistema computador*, podem tornar a l'escala d'un procés. Cal tenir mecanismes que ens permetin entrar informació cap a la memòria d'un procés des de l'exterior i mecanismes per a poder enviar informació des de la seva memòria cap a l'exterior. Això ens pot permetre, per exemple, introduir dades en un procés des d'un teclat. Amb aquestes dades el procés pot fer càlculs i els resultats d'aquests càlculs es poden mostrar, per exemple, per pantalla.

2. El concepte de *dispositiu d'entrada/sortida*

El concepte de *dispositiu d'entrada/sortida* està molt relacionat amb el de *perifèric*. Entenem per perifèrics elements com ara:

- El teclat, el ratolí o una targeta digitalitzadora, que permeten introduir informació en el sistema perquè pugui ser manipulada pels processos.
- Les pantalles, que serveixen per a mostrar a l'exterior la informació que els processos elaboren.
- Els discos, que permeten emmagatzemar informació.
- Els mòdems o les xarxes, destinats a transferir informació entre processos.

Figura 1
Amb el teclat introduïm informació en el sistema.

Malgrat això, i tal com veurem al llarg d'aquest mòdul, un dispositiu d'entrada/sortida no ha d'estar per força associat a un perifèric, sinó que pot ser qualsevol objecte de maquinari (*hardware*) o programari (*software*) gestionat pel sistema que sigui capaç de portar a terme el tipus d'accions que hem descrit.

> Un **dispositiu d'entrada/sortida** és un objecte gestionat pel sistema operatiu amb el qual els processos poden fer operacions de lectura/escriptura amb la finalitat d'obtenir informació, emmagatzemar-la, mostrar-la o transferir-la.

El sistema operatiu és l'encarregat de gestionar els dispositius proporcionant les operacions necessàries per tal que els processos hi puguin accedir. Tal com veurem, aquesta gestió ha d'oferir els elements que indiquem a continuació:

- Una **interfície d'accés al procés/dispositiu** que sigui **independent** de les particularitats de cadascun dels dispositius, però que alhora permeti, si és necessari, explotar-ne totes les característiques.
- Un **entorn portable** que permeti a les aplicacions treballar amb dispositius diferents sense haver de modificar-les.

3. Les característiques dels dispositius d'entrada/sortida

Malgrat que la funció dels diferents dispositius és genèricament la mateixa (permetre l'entrada o la sortida de dades), l'ús que se'n fa i el resultat final que proporcionen poden ser ben diferents segons el tipus i el model de dispositiu que utilitzem. Així doncs, els dispositius tenen característiques singulars que el sistema ha de gestionar de maneres diferents: les impressores bolquen la informació en un suport material com el paper, els teclats permeten entrar dades, les pantalles visualitzen les sortides, etc.

Exemple

No és el mateix tenir una impressora d'injecció de tinta que tenir-ne una de làser, ni tampoc és el mateix tenir la impressora connectada a l'ordinador mitjançant un port paral·lel que tenir-la connectada mitjançant un port USB.

Podem fer una **classificació dels dispositius** segons les seves característiques a fi d'analitzar les diferències que presenten. Atès el seu gran nombre i varietat, les hem agrupat en **característiques físiques**, **d'accés** i **de control**. De totes les característiques que anomenarem a continuació, només comentarem amb una certa profunditat les que afecten directament els usuaris. Hem de fer notar que aquesta classificació és una de les moltes que podríem fer basant-nos en les característiques dels dispositius i que, per tant, s'ha d'entendre com un marc orientatiu per a aconseguir una comprensió òptima de la diversitat dels dispositius.

3.1. Característiques físiques

Les característiques físiques fan referència a propietats intrínseques dels dispositius. Són les següents:

1) Els dispositius poden ser extraïbles o fixos: hi ha dispositius que són fixos, com un disc dur intern, i d'altres que són extraïbles, com un disc extern, un llapis de memòria (*pen drive*) o un reproductor de música en format MP3, que es poden connectar a un ordinador a través d'un port USB per tal de transferir informació entre l'ordinador i el dispositiu extern. De vegades el dispositiu està format per una unitat d'accés que permet accedir a contenidors o volums d'informació extraïbles. Un exemple el trobem en la unitat de lectura o escriptura de discos compactes i els discos compactes (CD, DVD, etc.).

Els dispositius extraïbles necessiten un tractament addicional; cal proporcionar els mecanismes necessaris per a poder inserir o extreure dinàmicament aquests dispositius sense necessitat d'aturar el sistema. Per qüestions d'eficiència, el sistema pot guardar a la memòria informació sobre els canvis que es volen realitzar en un dispositiu i retardar l'escriptura en aquest. Cal, per tant, materialitzar els canvis en el dispositiu en qüestió abans de procedir a la seva extracció, ja que en cas de no fer-ho aquests canvis es perdrien[2].

[(2)]Aquestes optimitzacions s'empren a la majoria de sistemes de gestió de fitxers. Per això és important aturar el sistema de manera correcta demanant-li que s'aturi (*shutdown*). D'aquesta manera ens assegurem que el sistema sincronitza el contingut de la memòria i el dels dispositius abans d'aturar-se i no es perden les actualitzacions fetes.

2) La capacitat d'emmagatzematge dels dispositius: els dispositius poden servir per a visualitzar informació, recollir-la o emmagatzemar-la. Els dispositius d'emmagatzematge tenen una capacitat, o un espai, concreta, que pot variar entre els diferents tipus de dispositius. El sistema operatiu és l'encarregat de gestionar-la. Les característiques d'aquests dispositius varien en funció dels aspectes següents:

a) La geometria: els dispositius d'emmagatzematge poden tenir l'espai organitzat de maneres diverses, seguint geometries diferents. Per exemple, un disc organitza l'espai en pistes, cares i sectors; una cinta, en blocs; etc.

b) El tipus d'emmagatzematge, permanent o temporal: els dispositius d'emmagatzematge poden contenir la informació de manera permanent, fins que s'esborra explícitament, o bé poden guardar-la temporalment. En aquest últim cas, la informació s'esborra automàticament quan es compleix una condició concreta. Per exemple, en un fitxer temporal la informació emmagatzemada es destrueix en tancar-lo.

3) El tipus d'unitat de transferència: les unitats de transferència entre el sistema i els dispositius poden ser diverses. En general, en distingim els dos tipus següents:

a) Dispositius de caràcters: són dispositius que transfereixen **caràcters**. Per exemple, un terminal és un dispositiu de caràcters d'entrada/sortida format per una pantalla i un teclat, perquè cada cop que premem una tecla el caràcter s'envia al sistema.

b) Dispositius de blocs: són dispositius que transfereixen blocs de caràcters, entenent com a *bloc* un conjunt de caràcters. Per exemple, el disc és un dispositiu de blocs, ja que cada cop que es vol llegir o escriure en el disc s'ha de transferir com a mínim un bloc, per exemple un sector del disc amb 512 bytes.

4) La velocitat de transferència dels dispositius: la velocitat de transferència, o amplada de banda del dispositiu, és un altre dels factors que diferencien els dispositius. Es mesura en bits/segon o en bytes/segon. Per exemple, un cable USB 1.0 pot arribar a una velocitat de 12 Mbit/s, mentre que un USB 2.0 pot arribar als 480 Mbit/s i un USB 3.0 pot arribar a 4,8 Gbit/s.

La velocitat de transferència dels dispositius normalment és molt més petita que la velocitat del processador. Aquesta diferència de velocitats fa que el sistema, en gestionar els dispositius, hagi d'aplicar tècniques que permetin reduir els temps d'espera dels processos. Veurem algunes d'aquestes tècniques en la secció sobre optimització.

5) El tipus de codificació de la informació: la informació que es llegeix o s'escriu en un dispositiu ha de seguir unes normes de codificació que depenen de cada dispositiu en concret. Per exemple, un terminal orientat a caràcters

pot visualitzar la informació codificant-la de diferents maneres, que poden ser ASCII de 7 bits o de 8 bits, UTF-8, ISO-8859-1 o moltes altres. En aquest últim cas, els caràcters per sobre del codi ASCII 127 depenen de l'alfabet (català, castellà, etc.) i del fabricant. Per tant, en visualitzar-los el resultat que s'obté pot ser diferent del que s'esperava.

Quan intentem visualitzar el contingut d'un fitxer executable (codificat en binari) a la pantalla es produeix una situació sorprenent perquè molts caràcters són no imprimibles.

6) Condicions d'error: els tipus d'error possibles, la forma de notificar-los, les seves conseqüències i les possibles respostes difereixen d'un dispositiu a un altre. Per exemple, si intentem llegir d'una impressora hem de rebre un error, ja que aquesta és un dispositiu de sortida però no d'entrada. En canvi, amb un ratolí passa el contrari, ja que és un dispositiu d'entrada i ha de notificar un error si intentem escriure-hi.

3.2. Característiques d'accés

Les característiques d'accés fan referència a la manera en què el sistema operatiu i els processos accedeixen a la informació que contenen els dispositius. Són les següents:

1) Accés per entrada o sortida: els dispositius poden ser d'entrada, de sortida o dels dos tipus alhora. En funció d'aquesta característica, una operació d'accés pot tenir sentit o no. El sistema és l'encarregat de controlar quin tipus d'accés s'intenta fer i si és possible.

2) Accés compartit o exclusiu: a un dispositiu d'ús exclusiu només hi pot accedir un usuari alhora, al contrari del que succeeix amb els dispositius compartits. Un exemple de dispositiu exclusiu és la impressora, ja que no pot escriure dos documents alhora; si es barregessin les línies de dos o més documents en un sol full de paper, aquest resultaria inservible. En la secció d'optimitzacions veurem com es pot alleugerir la restricció de no poder treballar amb dispositius no compartibles.

3) Accés seqüencial, directe o indexat: l'accés a una informació concreta continguda en un dispositiu d'emmagatzematge es pot fer bàsicament de les tres maneres següents:

- **Seqüencialment**: s'accedeix a la informació que segueix a l'última informació a la qual s'ha accedit. Implica el manteniment d'aquesta posició, mitjançant el punter de lectura/escriptura, per part del sistema operatiu.
- **Directament**: s'accedeix directament a una posició concreta.
- **De manera indexada**: s'accedeix a la informació mitjançant una clau lògica.

Cada dispositiu permet, en funció de la seva estructura, un o més d'aquests mètodes d'accés. El sistema, mitjançant la seva gestió, pot oferir aquells mètodes que el dispositiu no accepta directament.

4) Accés síncron o asíncron: en el moment d'accedir a un dispositiu pot ser que la informació no estigui disponible. Llavors, segons el mode de funcionament emprat, tenim els casos següents:

- En **mode síncron**, si la informació no està disponible, el procés que efectua l'accés s'esperarà fins que ho estigui. El procés passarà a l'estat *Blocked*.

Vegeu també

Al mòdul 2 expliquem més àmpliament com funciona l'estat *Blocked*.

- En **mode asíncron**, el procés fa una petició al sistema per a realitzar una entrada/sortida, però no s'espera a saber-ne el resultat. Això permet que el procés se segueixi executant. Quan vulgui saber si l'entrada/sortida ha acabat, s'haurà de sincronitzar fent una petició al sistema operatiu. Això és necessari per a saber si ja disposem d'una dada que s'ha demanat abans d'operar amb ella. També pot ser necessari saber si s'ha escrit una dada que volíem escriure per tal de reutilitzar una posició de memòria o alliberar un recurs.

5) Altres característiques de l'accés als dispositius d'entrada: quan s'entren dades des d'alguns dispositius, com ara els terminals, és necessari veure el que s'entra i, en cas d'equivocació, s'ha de poder rectificar. Aquestes dues característiques dels terminals fan referència respectivament al que s'anomena *echo* (ressò) i al mode d'edició *cooked*/*raw* (cuinat/cru):

- **El fet d'activar o no el mode *echo***: no fem servir l'*echo* quan, per exemple, volem introduir una contrasenya.

Exemple

El caràcter ASCII 127 (*delete*) produeix l'eliminació d'un dels caràcters entrats.

- **El mode d'edició *cooked*** fa que alguns caràcters que componen l'entrada s'interpretin com a comandes que alteren la informació que s'ha introduït. Amb aquest mode activat, tots els caràcters que controlen l'edició s'interpretaran i, per tant, no es lliuraran al procés que efectua l'operació d'entrada. Així doncs, un mateix dispositiu es pot comportar de maneres molt diferents segons quins d'aquests dos modes estigui actiu. En general, el sistema és l'encarregat d'implementar-les per programari; tanmateix, alguns dispositius les poden tenir implementades directament per maquinari.

3.3. Característiques de control

Les característiques de control fan referència a la manera d'indicar als dispositius el que es vol que facin. Podem distingir-ne els dos tipus següents:

1) Les característiques del controlador: el sistema controla els perifèrics mitjançant l'accés a un conjunt de ports que pertanyen al maquinari anomenat controlador. Les característiques d'aquests controladors condicionen la manera en què el sistema gestiona les entrades/sortides, que pot ser per interrupcions, per enquesta, amb accés directe a la memòria (DMA[3]), etc. El sistema operatiu amaga als usuaris totes aquestes particularitats.

(3)DMA és l'acrònim corresponent a l'expressió anglesa *Direct Memory Access*.

2) Els llenguatges de comandes: alguns dispositius utilitzen llenguatges de control per a configurar el tipus d'entrada/sortida que han de fer. Per exemple, algunes impressores làser utilitzen el llenguatge *PostScript* per a especificar el format de la informació que han d'imprimir. Un altre exemple serien els caràcters de control que s'utilitzen en un terminal per a editar les entrades o per a indicar si la sortida ha de ser en negreta, subratllada, etc.

3.4. Conseqüències de la diversitat dels dispositius

Tal com es pot veure en la relació de característiques que acabem d'esmentar, els dispositius són molt diferents entre si. Això ocorre no només entre tipus diferents, sinó també dins dels del mateix tipus. Com a conseqüència d'aquestes diferències tenen lloc les situacions següents:

- S'accedeix als dispositius i es manipulen mitjançant **operacions diferents** i amb paràmetres diferents.
- Els dispositius poden produir **resultats diferents** com a resposta a operacions que en un principi podrien semblar iguals.
- Finalment, i com a resultat de les dues consideracions anteriors, els dispositius produeixen **errors diferents**, fins i tot en situacions anàlogues.

Aquesta diversitat fa que, com a usuaris, desitgem independitzar-nos d'aquesta multitud de casos diferents. El sistema operatiu és l'encarregat d'aconseguir aquest objectiu; mitjançant la seva gestió proporciona una màquina virtual que fa que tota aquesta diversitat sigui transparent per a l'usuari, i així li ofereix un grau més elevat d'independència dels dispositius. En els apartats següents veurem com es pot aconseguir.

Vegeu també

Recordeu que vam presentar el concepte de *màquina virtual* en l'apartat 1 del mòdul 2.

4. Els dispositius físics, els lògics i els virtuals

En aquest apartat classifiquem els dispositius en tres categories, cada cop més abstractes, orientades a permetre l'escriptura de programes independents dels dispositius. Es tracta dels dispositius:

- físics
- lògics
- virtuals

4.1. Els dispositius físics

Un dispositiu físic és un dispositiu que existeix físicament. El formen el perifèric i el seu maquinari de control (*device controller*), que constitueixen la part física, i el programari que els gestiona, que anomenem programa controlador (*driver*). El programa controlador es comunica directament amb el controlador del dispositiu a través de registres de control.

Vegeu també

Podeu veure un esquema dels dispositius físics, lògics i virtuals a la figura 2 en el subapartat 4.3.

Per norma general, són dispositius físics tots els que estan associats a un perifèric. Les impressores, els teclats, els ratolins, o els escàners són alguns exemples de dispositius físics.

4.2. Els dispositius lògics

Els dispositius físics són els dispositius més evidents en un sistema i ràpidament els identifiquem com a tals. No obstant això, en el sistema hi ha un altre tipus de dispositius que no estan necessàriament associats a un perifèric; són els dispositius lògics.

Exemple

Un disc simulat a la memòria és un dispositiu que no existeix físicament, sinó que és el resultat d'un algoritme que simula, en aquest cas, un dispositiu físic, un disc.

Un **dispositiu lògic** és el resultat d'un programari del sistema que defineix aquest dispositiu. Un dispositiu lògic pot tenir associat un dispositiu físic o no. Així doncs, els dispositius lògics, a diferència dels físics, poden estar formats únicament pel seu programa controlador.

Un exemple de dispositiu lògic és un fitxer. Probablement ens interessa desar el fitxer i l'emmagatzemem, per exemple, en un disc o en un llapis USB. Per tant, en aquest cas hi ha algun dispositiu físic associat al dispositiu lògic fitxer. En alguns casos, però, el dispositiu lògic no té associat un dispositiu físic d'entrada/sortida, sinó que és el sistema operatiu el que gestiona i emmagatzema la informació necessària a la seva memòria. Això passa, per exemple, amb

un dispositiu anomenat *pipe*, que permet comunicar informació entre processos a través seu i consisteix, bàsicament, en una zona de memòria (memòria intermèdia) gestionada pel sistema operatiu mateix.

El que el sistema operatiu ens ofereix són els dispositius lògics. Com a usuaris d'un sistema, podem indicar que volem usar un dispositiu lògic. De fet, no podrem accedir directament als dispositius físics. Això queda reservat al sistema operatiu. Sí que podrem, però, usar els dispositius físics indirectament a través dels dispositius lògics. Per exemple, no indicarem que volem llegir una cara, una pista i un sector d'un disc, sinó que indicarem que volem llegir un fitxer, i això pot provocar l'accés a unes posicions concretes d'un disc.

Altres **exemples de dispositius lògics** són els que presentem a continuació:

1) El **nul**, que és un dispositiu d'entrada/sortida en el qual podem escriure tot el que vulguem i que sempre és buit. En realitat, és un dispositiu que ignora qualsevol cosa que s'hi escrigui i que en lectura no retorna mai cap informació. S'utilitza per a deixar de banda resultats no desitjats. Aquest dispositiu es basa exclusivament en el programari que el defineix, sense cap dispositiu físic associat.

El nul s'utilitza per a deixar de banda els missatges sobre l'estat de l'evolució d'un procés que surten generalment en pantalla i que indiquen que el procés treballa.

2) La **finestra**, que és un dispositiu lògic d'entrada/sortida que combina quatre dispositius físics: una pantalla, la memòria, un teclat i un dispositiu apuntador, com pot ser un ratolí o un llapis òptic. La finestra és una àrea gràfica emmagatzemada a la memòria que es representa totalment o parcialment en una pantalla. Dels quatre dispositius esmentats, tenim que:

- La pantalla és el dispositiu on es reflecteixen les sortides.
- El teclat és el mitjà per a fer les entrades.
- El dispositiu apuntador ens permet portar a terme certes operacions de control a la finestra, com ara tancar-la, variar-ne l'àrea i la posició o seleccionar-ne una entre diverses per a poder-hi efectuar entrades des del teclat.
- La memòria emmagatzema tota la informació relacionada amb la finestra.

A diferència del dispositiu nul, el programari que configura les finestres es basa en la manipulació dels programes controladors dels dispositius físics que el componen.

3) La **cua de missatges**, que és un dispositiu d'entrada/sortida pensat per a comunicar processos i que pot tenir les dues resolucions següents:

- Una de totalment lògica, basada en una estructura de dades i unes operacions que la manipulen, que s'utilitza per a comunicar processos que s'executen en una mateixa màquina.

- Una altra basada en dispositius físics, com ara el mòdem o la placa d'accés a la xarxa, que s'utilitza per a comunicar processos que s'executen en màquines diferents.

4) L'**espai lògic**, que és una agrupació d'informació emmagatzemada a la memòria física mitjançant l'ús de la memòria virtual. L'espai lògic d'un procés també es pot veure com un dispositiu que es pot llegir o en el qual es pot escriure. Aquesta visió dels espais lògics no és lluny del concepte de fitxer i, per tant, pot rebre el mateix tractament.

Vegeu també

Consulteu l'apartat 3 del mòdul 3, dedicat a la memòria virtual.

L'espai lògic té utilitat per a aplicacions com ara els depuradors de programes (*debugger*), que necessiten accedir a l'espai lògic del procés que estan depurant.

4.3. Els dispositius virtuals

Els **dispositius virtuals**, també coneguts com a **canals virtuals** o simplement **canals**, representen el nivell més abstracte i independent del dispositiu. Un dispositiu virtual és un dispositiu que, *a priori*, no està associat a cap dispositiu concret.

El programa l'utilitza igual que qualsevol altre dispositiu, però sense saber d'entrada en quin dispositiu en concret s'efectuaran les operacions que s'hi especifiquen. Serà en temps d'execució del programa que el sistema associarà un dispositiu concret al dispositiu virtual mitjançant el codi del sistema operatiu, com mostrem a la figura 2. Podrem associar un dispositiu virtual a algun dels dispositius visibles com a usuaris del sistema operatiu, és a dir, a un dispositiu lògic.

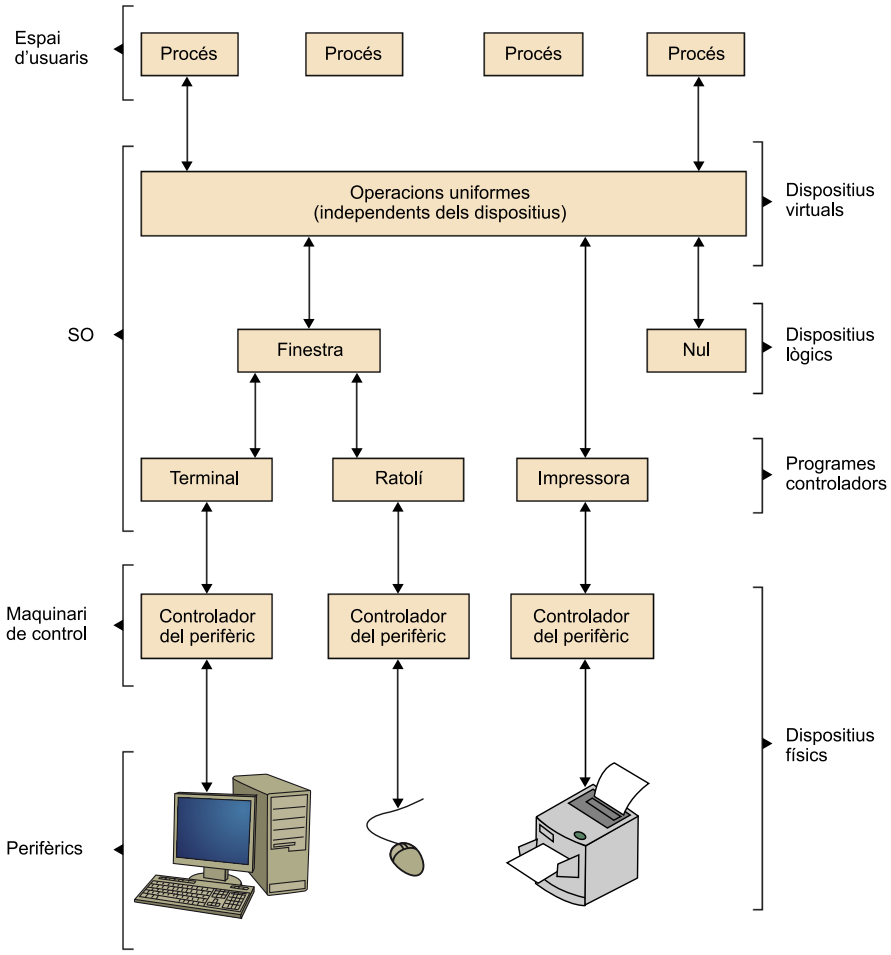

Figura 2. Dispositius físics, lògics i virtuals

5. La independència dels dispositius

Com a programadors ens interessa poder fer programes que siguin portables i que es pugin adaptar a situacions diferents. Aquesta necessitat inclou l'entorn d'entrada/sortida en el qual s'executaran. Hem vist que els dispositius tenen característiques que fan que la seva gestió i les operacions que s'hi poden portar a terme siguin diferents. L'objectiu del sistema operatiu és proporcionar una màquina virtual que uniformitzi els dispositius tot respectant-ne les particularitats per a facilitar així als usuaris l'ús que en faran. Això s'aconsegueix independitzant el codi dels programes respecte dels dispositius que utilitzen i de les operacions amb què s'hi adrecen.

Alguns casos, com ara un programa que copia fitxers, una senzilla màquina d'escriure, un editor de línies i un visualitzador de fitxers, es poden reduir a un sol programa que escriu en un dispositiu el que ha llegit en un altre. Malgrat es pugui fer aquesta simplificació, i atès que els dispositius amb els quals hauran de treballar els programes anteriors són tan diferents, difícilment podem pensar que un únic programa font pugui servir per a tots els dispositius. Al llarg d'aquest apartat veurem com, gràcies a la gestió del sistema operatiu, és possible efectuar totes aquestes operacions amb un únic codi font.

Vegeu també

Consulteu l'exemple comodí del subapartat 7.1.2 d'aquest mòdul didàctic.

5.1. Utilització dels dispositius virtuals

El primer pas per a independitzar els programes de l'entorn d'entrada/sortida amb el qual treballaran es basa en la utilització dels dispositius virtuals. En l'apartat anterior hem dit que podem associar un dispositiu virtual a algun dels dispositius visibles com a usuaris del sistema operatiu, és a dir, a un dispositiu lògic. Aquesta **associació entre un dispositiu lògic i un dispositiu virtual** es pot materialitzar de les dues maneres següents:

a) En **l'associació implícita**, també anomenada **redireccionament de les entrades/sortides**, el programa ja troba l'associació feta en l'instant en què inicia la seva execució. El sistema i el procés que han iniciat l'execució del programa són els encarregats de fer l'associació. Els dispositius virtuals associats de manera implícita s'anomenen **dispositius estàndard i**, en general, en tenim tres, que són l'estàndard d'entrada (*standard input*), el de sortida (*standard output*) i el d'error (*standard error*).

b) L'associació explícita entre un dispositiu lògic i un de virtual feta pel programa mateix durant l'execució. Per a efectuar-la necessita una operació específica que associï un dispositiu virtual a un dispositiu lògic. A partir del moment en què s'ha fet l'associació, el programa especificarà totes les operacions d'entrada/sortida del dispositiu mitjançant el dispositiu virtual.

Vegeu també

Vegeu els dispositius estàndard en l'apartat 7.1. d'aquest mòdul didàctic.

En l'apartat 4 del mòdul 5, veurem la manera en què les operacions emprades en l'associació explícita s'utilitzen per a dur a terme verificacions de protecció.

5.2. Definició d'operacions uniformes

Per a aconseguir la independència dels programes respecte de les entrades/sortides no n'hi ha prou d'utilitzar dispositius virtuals. No serveix de res fer que els programes treballin en dispositius virtuals si aquests han d'utilitzar operacions específiques dels dispositius concrets amb els quals estan associats. Per a evitar que els programes hagin de conèixer les particularitats dels dispositius reals, el sistema ha d'uniformitzar totes les operacions que es poden fer en els dispositius.

Per a fer-ho, cal conèixer quines són les operacions més utilitzades pels programes i quines són les més comunes a la majoria dels dispositius. Aquestes operacions s'han de definir en un conjunt nou d'operacions independents de les característiques dels dispositius, el qual ha de ser ofert pel sistema. A fi de proporcionar aquesta nova funcionalitat, el sistema afegeix una capa de programari per sobre dels programes controladors (codi específic de cada dispositiu), que són les operacions independents dels dispositius. El programador usarà aquesta interfície uniforme per a indicar les operacions que cal fer en els dispositius virtuals. Internament, el sistema operatiu utilitzarà les rutines específiques per al dispositiu concret associat al dispositiu virtual. Aquesta informació es determina en temps d'execució gràcies a la informació emmagatzemada internament en el moment d'establir l'associació entre el dispositiu virtual i un dispositiu lògic.

Si analitzem les operacions d'entrada/sortida podem veure que hi ha operacions bàsiques d'accés i de control, orientades a donar un accés uniforme, i operacions específiques de control, orientades a poder gestionar les particularitats de cada dispositiu. A continuació estudiarem cadascuna d'aquestes operacions.

5.2.1. Operacions bàsiques d'accés

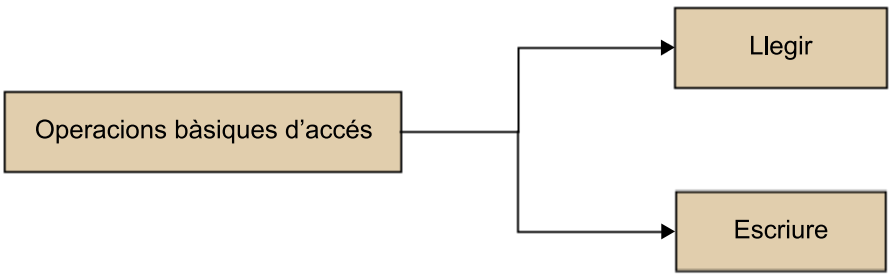

Figura 3

Les operacions bàsiques d'accés són dues, **llegir** i **escriure**, i ens han de permetre accedir a la informació de manera independent del dispositiu. La seva funcionalitat ha de ser compatible amb la dinàmica de funcionament de tots els dispositius. Aquesta dinàmica inclou el tipus d'accés (seqüencial o directe), el tipus de les dades que s'han de transferir i la seva estructura.

El tipus d'accés més habitual és el seqüencial. Això no obstant, s'ha de donar opció al fet que les aplicacions puguin fer accessos directes a la informació. Per a permetre-ho, és necessari proporcionar l'operació *posicionar*, la qual s'analitza juntament amb les operacions bàsiques de control.

Vegeu també

Trobareu informació sobre l'accés seqüencial en el subapartat 3.2 d'aquest mòdul didàctic.

Per a especificar quina informació s'ha de transferir n'hi ha prou amb dos paràmetres:

- *buf*: Especifica la zona de memòria des de la qual es trauran les dades amb l'operació o a la qual s'entraran.
- *cont:* Indica el nombre de dades màxim que s'han de transferir.

Si no hi introduïm més paràmetres, hem de suposar que accedim sempre al mateix tipus de dades. Si no fos així, caldria incloure-hi un altre paràmetre que especifiqués amb quin dels tipus de dades reconeguts pel sistema es vol fer l'operació d'entrada/sortida.

Ara bé, la majoria de sistemes redueixen tots els tipus de dades possibles a un de sol, el més elemental, que és el byte. El motiu d'aquesta decisió és senzill: no se'n poden reconèixer tots els tipus possibles; per més tipus que reconegués el sistema, sempre hi hauria aplicacions amb tipus nous. D'altra banda, qualsevol tractament específic es pot fer a la biblioteca o al programa. Aquesta idea encaixa perfectament amb un dels principis dels sistemes operatius, que és el d'oferir eines perquè els usuaris del sistema o el seu administrador puguin fer polítiques que s'adaptin a les seves necessitats particulars.

A fi de completar la llista de paràmetres, s'ha d'indicar el dispositiu virtual en el qual es vol treballar (*disp*) i, finalment, un cop executades aquestes operacions, el sistema ha de retornar informació sobre l'estat en què han finalitzat.

Així doncs, les operacions d'accés mínimes queden tal com s'indica a continuació:

- estat = llegir (*disp, buff, cont*)
- estat = escriure (*disp, buff, cont*)

5.2.2. Operacions bàsiques de control

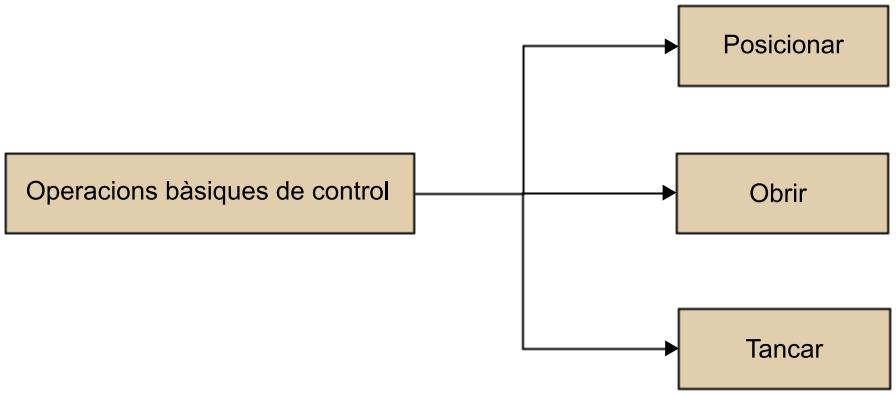

Figura 4

Hi ha tres operacions bàsiques de control, que són **posicionar**, **obrir** i **tancar**, les quals permeten portar a terme les operacions de control més comunes que s'executen en els dispositius.

Tal com hem vist en el cas de les operacions d'accés, és necessari proporcionar un mecanisme per a poder fer accessos directes. L'operació *posicionar* ens ha de permetre accedir directament a qualsevol posició dins de la informació dels dispositius que ho permetin. Els paràmetres necessaris per a executar aquesta operació són els dos següents:

- El dispositiu virtual en el qual es vol efectuar l'operació (*disp*)
- La posició en la qual, o des de la qual, es vol portar a terme l'operació d'accés següent (*pos*)

L'inici i el final d'una sessió d'accés a un dispositiu es defineixen amb les operacions *obrir* i *tancar*. Les aplicacions no solen accedir als dispositius de manera aïllada, sinó que normalment fan una sèrie d'accessos que segueixen una mateixa pauta; o bé s'escriu tot un fitxer de manera seqüencial, o bé es modifica parcialment mitjançant un accés directe, o bé es llegeix seqüencialment, etc. En cadascuna d'aquestes sessions, els accessos que la componen tenen necessitats comunes. Podem resumir aquestes necessitats en tres idees bàsiques:

1) Associar un dispositiu lògic a un dispositiu virtual de manera explícita.

2) Inicialitzar les estructures de dades internes del sistema operatiu necessàries per a fer els accessos (seqüencial i directe).

3) Verificar els drets d'accés que té el procés sobre el dispositiu.

Si no existissin les operacions *obrir* i *tancar* que emmarquen tots els accessos d'una sessió, caldria cobrir les necessitats comunes per a cada accés i, en conseqüència, es multiplicaria la feina que ha de fer el sistema.

En obrir **una sessió de treball amb un dispositiu**, mitjançant l'operació *obrir*, cal indicar les dades següents:

- El nom del dispositiu lògic (*nom*).
- El tipus d'accés[4] que es vol efectuar (*op*).

(4)Els accessos poden ser de lectura, d'escriptura o de lectura i escriptura simultàniament.

Amb aquests paràmetres, el sistema ha de verificar l'existència del dispositiu i els drets d'accés del procés a aquest dispositiu. Si el dispositiu existeix i es té dret a accedir-hi, el sistema generarà un dispositiu virtual que l'associarà, per a aquesta sessió de treball, al dispositiu lògic. Al mateix temps inicialitzarà, amb el valor adequat, el punter de lectura/escriptura necessari per a realitzar un accés seqüencial; aquest valor serà l'inici de la informació del dispositiu en cas de lectura o modificació o bé el final de la informació en cas d'afegir (*append*). Això es farà per als dispositius en què tingui sentit, com per exemple en el cas dels fitxers. En **tancar la sessió de treball amb el dispositiu**, és a dir, en executar l'operació *tancar*, el sistema alliberarà les estructures de dades necessàries per a l'accés i desfarà l'associació entre el dispositiu virtual i el lògic.

Vegeu també

Trobareu més informació sobre el control d'accessos als dispositius en el subapartat 4.3. del mòdul didàctic 5.

Les operacions *obrir* i *tancar* s'utilitzen per a portar a terme de manera explícita totes les operacions. El sistema, però, les pot efectuar de manera implícita en iniciar o finalitzar un procés, tal com hem comentat en el cas de l'assignació de dispositius lògics a virtuals.

Així, doncs, les operacions bàsiques de control podrien ser les següents:

- estat = posicionar(*disp, pos*)
- disp = obrir(*nom, op*)
- estat = tancar(*disp*)

5.2.3. Operacions específiques de control

Finalment, el sistema permet, mitjançant l'operació *control*, que els usuaris que ho necessitin puguin gestionar les particularitats dels dispositius.

Segons el sistema, podem trobar l'operació de control de les dues maneres següents:

a) Com una operació única amb paràmetres variables que depenen de la funció que es vol efectuar i del dispositiu en el qual s'efectua.

b) Com tantes operacions diferents com funcions s'han de realitzar.

6. Optimització de l'entrada/sortida

És interessant conèixer algunes de les optimitzacions que fa el sistema pel que fa a la gestió de l'entrada/sortida, com ara l'emmagatzematge a la memòria intermèdia (*buffering*) i la gestió de les cues (*spooling*). Tot seguit veurem aquestes dues tècniques amb més detall.

6.1. L'emmagatzematge a la memòria intermèdia

De vegades, la velocitat dels dispositius no s'adapta a la velocitat amb què el procés s'està executant, ja sigui perquè els dispositius són més lents o bé perquè el procés treballa a ratxes. Per tal de solucionar aquest problema i aconseguir que ni els dispositius ni els processos s'hagin d'esperar mútuament, el sistema operatiu utilitza la memòria intermèdia (*buffer*). Així, aquesta memòria té la funció d'esmorteir les diferències de velocitat.

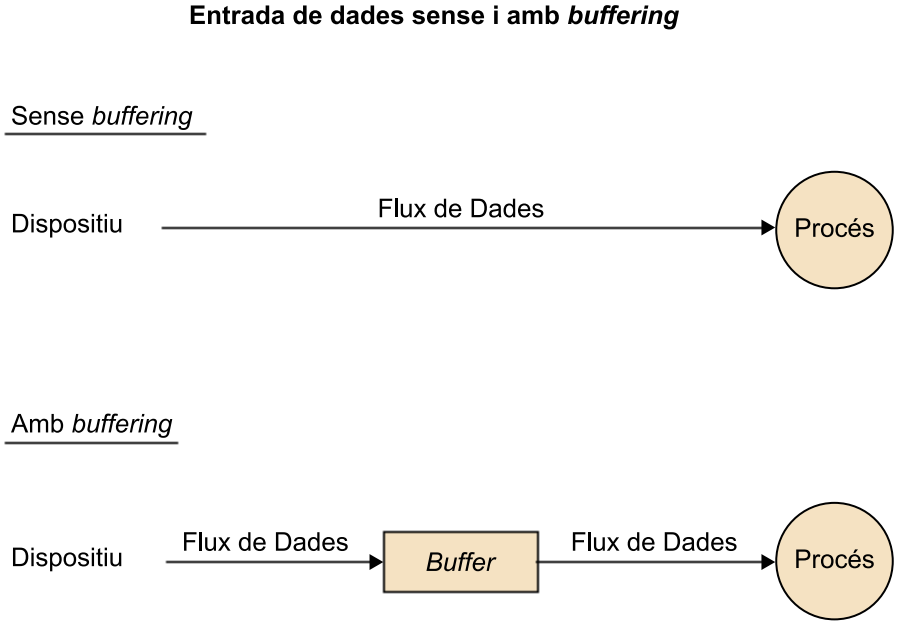

Figura 5. L'emmagatzematge en la memòria intermèdia

En l'esquema amb la memòria intermèdia, el dispositiu produeix informació i el procés la consumeix. El dispositiu ha de passar la informació a la memòria intermèdia i només sorgeixen problemes si aquesta és plena. Per altra banda, el procés consumeix únicament informació de la memòria intermèdia i només s'ha d'esperar si aquesta és buida.

6.2. La gestió de les cues

La tècnica de la gestió de cues o SPOOL (*Simultaneous Peripherical Operation On-Line*) es basa en el fet que les entrades/sortides d'un procés s'implementin amb un pas intermedi emprant dispositius compartibles de gran capacitat

d'emmagatzematge i ràpids (discos, memòria, etc.). Els processos d'usuari no usen directament el dispositiu final, sinó que encuen la seva petició en una memòria intermèdia predeterminada (per exemple, un fitxer en un disc o en una zona de memòria) i poden continuar la seva feina sense esperar que finalitzi l'entrada/sortida. El sistema operatiu, a través d'un procés encarregat d'aquesta feina, anomenat gestor de cues, anirà tractant les peticions encuades a la memòria intermèdia a mesura que el dispositiu final estigui disponible. Això permet millorar el rendiment en un sistema multiprogramat quan s'utilitzen perifèrics més lents que el processador, ja que els processos no han d'esperar que el recurs final estigui disponible ni que es completi l'entrada/sortida. Permet també que diversos processos puguin avançar concurrentment tot i intentar utilitzar un dispositiu no compartible, com per exemple una impressora.

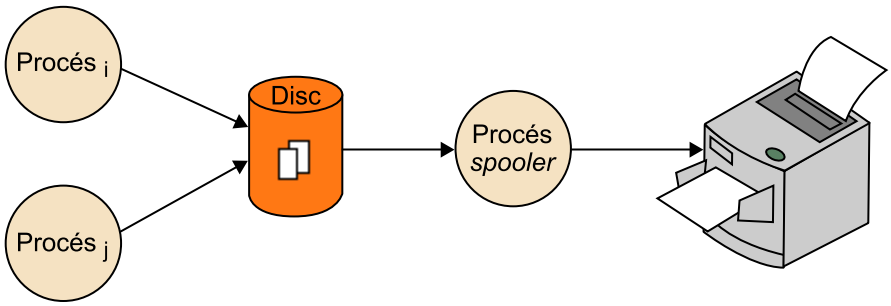

Figura 6. Impressió d'un document

Un cas molt corrent és el de treballar amb un processador de textos i fer una impressió d'un document relativament llarg. Si no s'utilitza la tècnica de la gestió de cues (*spool*), el computador, i per tant tampoc l'usuari, no podrà fer res més fins que no acabi la impressió. Un efecte addicional de la utilització de la tècnica de la gestió de cues és la sensació que tenen els usuaris d'estar utilitzant tots alhora la impressora, tot i que és un dispositiu no compartible. Això és possible gràcies al fet que el dispositiu intermedi (en aquest cas el disc) és compartible.

Vegeu també

Podeu consultar l'execució concurrent de processos en el subapartat 2.3 del mòdul 1.

Vegeu també

Vegeu els dispositius compartibles i no compartibles en el subapartat 3.2 d'aquest mòdul.

7. Exemples

7.1. Entrada/sortida a l'UNIX

L'UNIX, com tots els sistemes operatius, ofereix una visió dels dispositius que fa que les característiques pròpies de cadascun siguin transparents per a l'usuari. En l'UNIX, un dispositiu és qualsevol objecte en el qual es poden fer operacions de lectura o escriptura.

El punt de vista de l'usuari de l'UNIX és el d'un dispositiu en el qual es pot escriure o llegir seqüències de bytes fent servir un entorn únic i ben definit.

El sistema operatiu UNIX reconeix els dos tipus bàsics de dispositius que esmentem a continuació:

1) Els **dispositius de blocs**, com ara els discos, les cintes i els fitxers. Permeten l'accés directe, ja que la unitat d'adreçament és un bloc de grandària fixada pel sistema.

Per a aquests tipus de dispositius, l'UNIX utilitza una memòria cau (també anomenada memòria *cache*) de blocs amb la finalitat de millorar el rendiment en els seus accessos, ja que el temps d'accés a la memòria és molt menor que el temps d'accés a alguns dispositius de blocs. Bàsicament, consisteix a emprar una zona de memòria, composada per diverses memòries intermèdies i que gestiona el sistema operatiu per tal d'evitar accessos al disc, reutilitzant informació prèviament llegida o endarrerint l'escriptura d'informació nova de forma que les actualitzacions parcials es facin a la memòria[5].

[(5)]Cal, però, sincronitzar de tant en tant la informació que hi ha a la memòria amb la que hi ha al dispositiu, per tal d'evitar inconsistències en cas de patir una caiguda del sistema. També cal fer-ho en el moment en què es tanca el sistema (*shutdown*) com es va avançar en l'apartat 3.1 d'aquest mòdul didàctic.

Figura 7

2) Els **dispositius de caràcters**, com ara els terminals, les impressores, els ratolins, etc. En general, tots aquells dispositius que no utilitzen la memòria cau de blocs. Solen ser dispositius en els quals s'han de fer els accessos de manera seqüencial.

Internament, els diferents tipus de dispositius s'identifiquen segons si són de blocs o de caràcters i per un número anomenat *major*. Els diferents dispositius d'un mateix tipus s'identifiquen mitjançant un altre número, anomenat *minor*.

Així doncs, internament, un dispositiu s'identifica completament mitjançant el trio tipus bàsic, *major*, *minor*.

El sistema gestiona els dispositius mitjançant programes controladors associats a cada tipus de dispositiu, tal com mostra la figura següent:

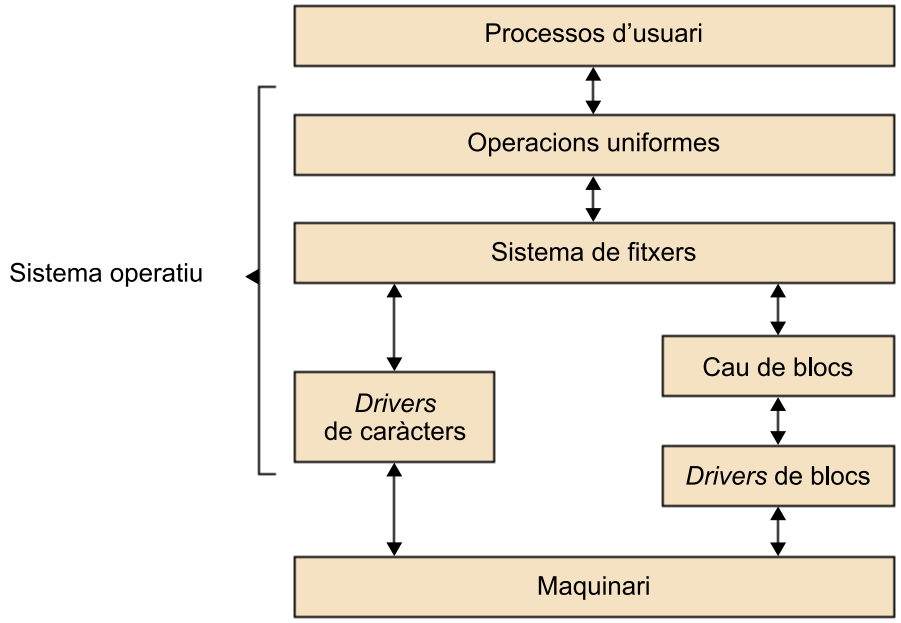

Figura 8. Esquema de les entrades/sortides a UNIX

L'usuari que vol adreçar-se als dispositius lògics de l'UNIX ho ha de fer a través del sistema de fitxers, mitjançant el qual l'UNIX anomena **fitxers especials**.

Exemple

En l'UNIX, una finestra té el nom `/dev/pts/0`, el disc es diu `/dev/sda`, etc.

L'accés bàsic als dispositius que ofereix l'UNIX als processos és l'accés seqüencial. No obstant això, també proporciona crides perquè es puguin fer accessos directes.

Una altra característica important és el control dels accessos simultanis a un mateix dispositiu. Per defecte, l'UNIX permet que més d'un procés accedeixi concurrentment al mateix dispositiu, tot i que proporciona les eines necessàries, mitjançant les crides de control, perquè els usuaris puguin treballar amb models d'exclusió mútua.

Els processos utilitzen els *file descriptors*[6] per a accedir als dispositius un cop ja s'han obert. Des del punt de vista de l'usuari, un ***file descriptor*** és un identificador del dispositiu virtual que forma part de l'entorn d'execució del procés.

(6)Els *file descriptors* són els dispositius virtuals, o canals, de l'UNIX.

En l'UNIX hi ha tres *file descriptors*, que normalment estan assignats de manera implícita i que per conveni tenen un significat concret, que són el 0 o canal d'entrada estàndard (*standard input*), l'1 o canal de sortida estàndard (*standard output*) i el 2 o canal d'error estàndard (*standard error*). Els dispositius d'entrada i de sortida estàndard fan referència a l'entrada/sortida per defecte. El canal d'error estàndard s'utilitza per a comunicar les situacions anòmales o de control en què es troba el procés durant la seva execució.

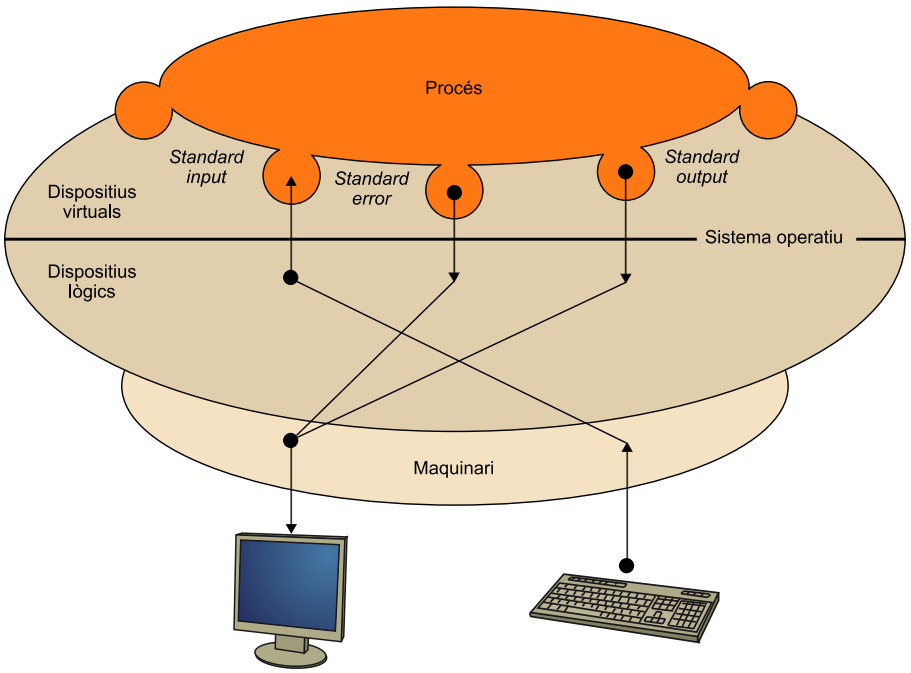

Figura 9. Entrada/sortida estàndard dels processos

Exemple

Un compilador pot llegir el fitxer font per la seva entrada estàndard, escriure l'executable per la seva sortida estàndard i indicar els errors o fer advertències sobre l'estil del programa pel seu canal d'error estàndard.

7.1.1. Les *pipes*, un cas especial de dispositiu

Una ***pipe*** és un dispositiu lògic destinat a comunicar processos. El seu funcionament és el d'una cua de caràcters amb una longitud fixa en què els processos poden llegir i/o escriure.

Quan un procés vol fer servir les *pipes* es pot trobar amb els dos problemes següents:

- Si intenta llegir una *pipe* buida, quedarà bloquejat fins que algun altre procés hi escrigui algun caràcter per tal que el procés bloquejat en pugui efectuar la lectura. El lector d'una *pipe* buida només es bloquejarà si hi ha escriptors potencials, és a dir, processos que tinguin la *pipe* oberta per a escriure-hi.

- Si intenta escriure en una *pipe* plena, quedarà bloquejat fins que algun altre procés llegeixi prou caràcters de la *pipe* perquè el procés bloquejat pugui efectuar-hi l'escriptura.

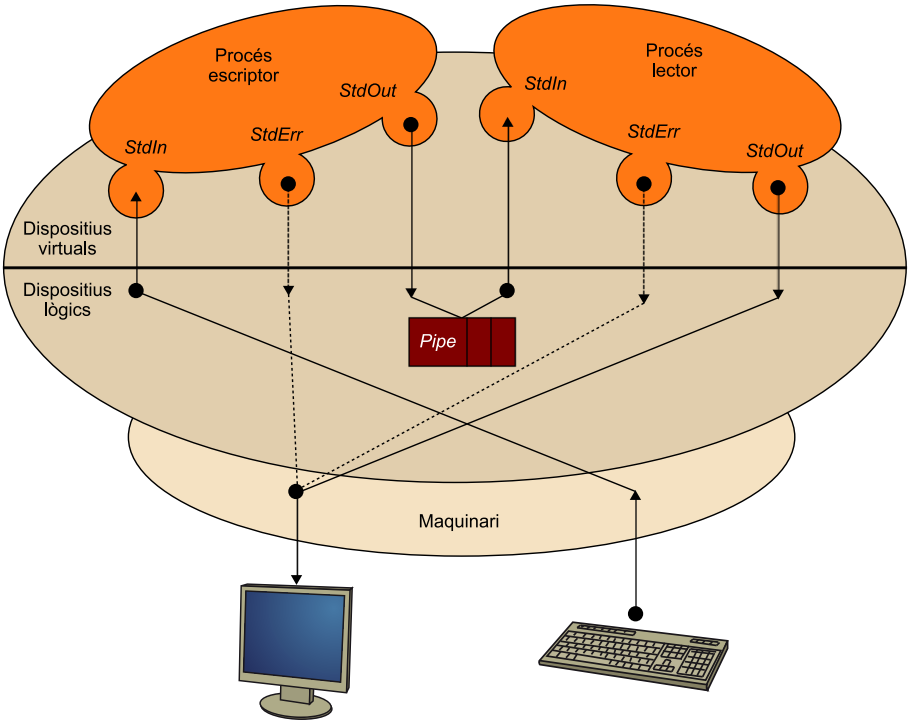

Figura 10. Comunicació de dos processos d'UNIX mitjançant una *pipe*

Hi ha dos tipus de *pipes*, que són les anomenades simplement *pipes* i les *named pipes*.

- Una ***pipe*** és un dispositiu que es crea en el moment en què un procés l'obre i que es destrueix quan l'últim procés que el té obert el tanca. És un dispositiu que no es pot obrir de manera explícita mitjançant l'operació *obrir* (*open*). Un cop creat, els diferents processos que volen utilitzar-lo l'han d'associar a un dispositiu virtual[7] de manera implícita. A causa d'aquesta particularitat, les *pipes* no hi tenen associat cap nom que aparegui en el sistema de fitxers.

[(7)] En l'UNIX, un dispositiu virtual és un *file descriptor*.

- Una ***named pipe*** té un comportament similar al d'una *pipe* però, a diferència d'aquesta, apareix al sistema de fitxers. Per tant, els processos la poden obrir com si fos un fitxer, sempre que tinguin drets per a fer-ho.

7.1.2. El punt de vista des de les crides al sistema

L'objectiu d'aquest subapartat és fer un repàs breu de les principals crides al sistema relacionades amb l'entrada/sortida. En cap cas no pretenem fer una descripció exhaustiva de totes les seves possibilitats ni de la seva sintaxi.

En general, el sistema operatiu UNIX té una estructura de crides al sistema idèntica a la que hem descrit al llarg d'aquest mòdul. Les correspondències entre les crides de l'UNIX i les que hem vist en les operacions uniformes són les que podeu veure a la taula següent:

Operacions uniformes	
Crides al sistema	Crides d'UNIX
Llegir	`read`
Escriure	`write`
Posicionar	`lseek`
Obrir	`open`
Tancar	`close`
Control	`ioctl`
	`fcntl`

Operacions de control

`ioctl` controla el dispositiu lògic associat a un *file descriptor*.
`fcntl` controla les característiques del *file descriptor* i de la sessió d'entrada/sortida que representa.
La interfície de les crides al sistema i els exemples d'utilització estan disponibles en els recursos de l'aula.

El procés comodí

Un exemple d'utilització de les crides d'entrada/sortida de l'UNIX és un procés que escriu a la sortida estàndard el que ha llegit a l'entrada estàndard.

```
/* PROCÉS COMODÍ */
main()
{
       char buff;
       int fi_in,fi_out;

       fi_out = 1;
       fi_in = read(0,&buff,1);

       /*mentre l'entrada no trobi fi de fitxer,*/
       /*i no es produeixin errors d'accés*/
       while((fi_in>0)&&(fi_out>0))
       {
              fi_out = write(1,&buff,1);
              fi_in = read(0,&buff,1);
       }
}
```

En el cas de l'UNIX, els fitxers no tenen una marca de final de fitxer. Si estem llegint, sabrem que hem arribat al final del fitxer quan la crida `read` retorni un 0.

Per simplicitat, es mostra el codi amb les crides bàsiques, però seria recomanable fer un control d'errors.

Un cas especial són les crides `pipe` i `dup`:

- La crida al sistema `pipe` crea una *pipe* i dos *file descriptors*, un per a lectura i l'altre per a escriptura. Un cop creada, s'accedeix a la *pipe* com si fos un dispositiu més.

- La crida al sistema `dup` associa un *file descriptor* nou al dispositiu associat al *file descriptor* que se li passa com a paràmetre.

Vegeu també

Vegeu com es pot utilitzar la crida `dup` per a redireccionar i enllaçar els *file descriptors* de diferents processos en l'apartat 4.1.2 del mòdul 6.

7.1.3. El punt de vista des de l'intèrpret de comandes

En aquest subapartat, igual que en l'anterior, introduïm les característiques principals dels intèrprets de comandes o *shells* de l'UNIX, sense pretendre ser exhaustius.

El tema més interessant de l'UNIX, des del punt de vista de les entrades/sortides, és la facilitat amb què es poden redireccionar els *file descriptors* estàndards dels processos des del *shell*, i el potencial que això representa.

Per a entendre aquest potencial cal analitzar els dos factors que esmentem a continuació:

1) El primer concepte que s'ha d'analitzar és el de **filtre**. Un filtre és un programa que llegeix l'entrada estàndard, manipula la informació i escriu el resultat d'aquesta manipulació a la sortida estàndard. Aquesta operació la repeteix mentre no detecta un final de fitxer.

Exemple de filtre

La comanda `tail` de l'UNIX és un filtre que treu per la sortida estàndard les últimes deu línies que ha llegit per l'entrada estàndard.

Marca de fi de fitxer

El sistema avisa els processos mitjançant el senyal de fi de fitxer que llegeix d'un dispositiu en el qual ja no es pot aconseguir més informació. Per exemple, si s'accedeix seqüencialment a un fitxer i s'intenta llegir informació més enllà de l'últim caràcter emmagatzemat, o bé en el cas d'accedir a una *pipe* si ja s'ha llegit tota la informació que contenia i no queden més processos que la tinguin oberta per a escriure-hi. A l'UNIX aquest senyal es tradueix en el fet que l'operació `read` retorna un número positiu menor que el nombre de caràcters que s'ha sol·licitat llegir.

2) El **redireccionament** és el segon concepte. Mitjançant la sintaxi del *shell* es poden redireccionar els *file descriptors* estàndard d'un procés des de la comanda que els invoca.

El símbol < s'utilitza per a redireccionar l'estàndard d'entrada, i el símbol > per a l'estàndard de sortida.

	Exemples de redireccionament
`$ ps`	`ps` mostra l'estat dels processos per la sortida estàndard actual.
`$ ps > nom`	Guarda l'estat dels processos en un fitxer `nom`
`$ tail`	`tail` mostra a la sortida estàndard les últimes deu línies que entren per l'entrada estàndard.
`$ tail < nom`	Mostra les deu últimes línies del fitxer `nom` per la sortida estàndard actual.

3) En tercer lloc, des d'una comanda podem **encadenar l'execució de més d'un programa** mitjançant *pipes*. Això s'aconsegueix redireccionant la sortida estàndard d'un procés i l'entrada estàndard d'un altre a una *pipe*.

Per a indicar que es connecten dos processos mitjançant una *pipe* s'utilitza el símbol |.

Exemples de redireccionament amb el procés comodí

Copia el contingut del fitxer origen al fitxer destí.

```
$ comodi <origen >desti
```

Actua com una màquina d'escriure i envia tot el que escrivim en el terminal cap a la impressora.

```
$ comodi </dev/tty >/dev/lp0
```

Edita el fitxer destí. Actua com un editor de línies senzill.

```
$ comodi </dev/tty >desti
```

Mostra el contingut del fitxer origen.

```
$ comodi <origen >/dev/tty
```

Exemple d'encadenament

Podem combinar la comanda `ps` amb el filtre `sort` (ordena el que llegeix per l'entrada estàndard i ho escriu a la sortida estàndard) i seleccionar les primeres línies amb `head` per a aconseguir els dos processos en execució que han utilitzat durant més temps la UCP.

```
$ ps -ef | sort -r -k 7 | head -3
UID     PID   PPID  C STIME   TTY    TIME        CMD
albert  2973  1     6 12:16   ?      00:38:33    /usr/lib/firefox-3.5.8/firefox
root    1180  1172  1 10:40   tty7   00:07:39    /usr/bin/X
```

En un sistema UNIX podem trobar informació sobre cada comanda i les seves opcions amb `man nom_comanda`.

Com podeu veure, si tenim un conjunt de filtres que realitzen funcions concretes podem, mitjançant les *pipes* i les redireccions, efectuar operacions realment complexes sense haver d'escriure ni una sola línia de codi.

7.2. Entrada/sortida al Windows

Molts dels conceptes que hem aprés en aquest mòdul són aplicables als sistemes Windows. En aquest sistema podem emprar la **Windows API**, també anomenada **WinAPI**. El programa següent il·lustra l'ús d'algunes crides relacionades amb el sistema d'entrada/sortida. Les operacions de lectura i escriptura treballen amb estructures de tipus `HANDLE` que corresponen als dispositius virtuals. Podem observar que ens podem referir a l'entrada estàndard amb `STD_INPUT_HANDLE` i a la sortida estàndard com `STD_OUTPUT_HANDLE`. Què creieu que fa el següent programa?

```
#include <windows.h>
#include <stdio.h>

#define MAX_SIZE 64

void panic(char *miss)
{
    fprintf(stderr, "Error in %s\n", miss);
    ExitProcess(1);
}

int main(int argc, char *argv[])
{
    HANDLE h_in, h_out;

    DWORD num_read, num_written;
    BOOL ret;
    char buf[MAX_SIZE];

    if (argc == 1) {
        h_in = GetStdHandle(STD_INPUT_HANDLE);
        if (h_in == INVALID_HANDLE_VALUE) panic("GetStdHandle");
    }
    else {
        h_in = CreateFile(argv[1], GENERIC_READ, 0, NULL, OPEN_EXISTING, 0, NULL);
        if (h_in == INVALID_HANDLE_VALUE) panic("CreateFile");
    }

    h_out = GetStdHandle(STD_OUTPUT_HANDLE);
    if (h_out == INVALID_HANDLE_VALUE) panic("GetStdHandle");

    while (1) {
        ret = ReadFile(h_in, (void *)buf, MAX_SIZE, &num_read, NULL);
        if (ret) {
            if (num_read > 0) {
                ret = WriteFile(h_out, (void*)buf, num_read, &num_written, NULL);
```

```
                if (!ret) panic("WriteFile");
                if (num_read != num_written) panic("Missing chars");
            }
            else break;
        }
        else panic("ReadFile");
    }
}
```

Resum

En aquest mòdul hem estudiat el concepte **d'entrada/sortida** bàsicament des del punt de vista de l'usuari del sistema. Hem vist que hi ha diferents **tipus de dispositius (físics i lògics)** amb característiques i particularitats molt diferents. Davant d'aquesta realitat, l'objectiu del sistema operatiu consisteix a oferir una visió dels dispositius que, en principi, faci transparent per als usuaris totes aquestes particularitats a fi que les aplicacions siguin el màxim d'independents dels dispositius que utilitzen. Ara bé, el sistema operatiu ha de permetre també que aquelles aplicacions que ho necessitin puguin manipular les característiques i extreure'n tot el rendiment que aquestes ofereixen. Per a aconseguir-ho, la gestió dels dispositius que fa el sistema operatiu ofereix una interfície d'accés formada pels elements següents:

- **Operacions uniformes** independents de les característiques concretes dels dispositius.
- **Dispositius virtuals** en els quals treballen aquestes operacions independents.

Com a cas concret, hem analitzat el **sistema d'entrada/sortida de l'UNIX**, en el qual podem destacar:

- El **redireccionament** dels dispositius estàndard d'entrada/sortida des de l'intèrpret de comandes.
- Les ***pipes*** com un cas de dispositiu lògic que permet comunicar processos entre si.
- Els **programes filtre**, que permeten efectuar operacions complexes a partir de l'encadenament de petits programes mitjançant *pipes*.

Complementàriament, hem introduït la manera en què es pot programar la part d'entrada/sortida d'un procés en un sistema Windows emprant la **Windows API**.

Activitats

1. Estudieu en el manual de l'UNIX quins redireccionaments es poden fer des dels intèrprets de comandes Bash.

2. Analitzeu les comandes de l'UNIX i familiaritzeu-vos especialment amb aquelles que es puguin considerar filtres, com per exemple `sort`, `wc`, `tail`, `head`, `cut`, `grep`, `uniq`, `cat`, `tee`...

3. Confronteu la vostra experiència com a usuaris de Linux, Windows o Mac amb el que hem estudiat en aquest mòdul didàctic.

4. Busqueu informació sobre les crides al sistema de l'UNIX i localitzeu les crides relacionades amb l'entrada/sortida. Podeu trobar-ne informació en els recursos de l'aula i en els manuals en línia d'un sistema UNIX mitjançant la comanda `man`.

5. Busqueu informació sobre la Windows API i localitzeu les crides relacionades amb l'entrada/sortida.

Exercicis d'autoavaluació

1. Quins dels fitxers següents es poden imprimir?

a) un fitxer amb codi font.

b) un fitxer objecte.

c) un fitxer amb assemblador.

d) un fitxer executable.

Justifiqueu la resposta.

2. Creieu que el rellotge del sistema es pot considerar un dispositiu? Justifiqueu la resposta.

3. Digueu avantatges i inconvenients del fet que el sistema operatiu reconegui l'estructura de la informació. Justifiqueu la resposta.

4. Quins d'aquests dispositius utilitzen la tècnica del *buffering*:

a) la pantalla.

b) el ratolí.

c) el teclat.

d) el disc.

Justifiqueu la resposta.

5. En quin cas els processos que utilitzen finestres poden rebre un senyal de fi de fitxer?

6. Quins paràmetres hauria de tenir una operació de lectura o escriptura que donés accés seqüencial a la informació si no existissin les operacions obrir i tancar? I si l'accés fos directe? Justifiqueu les respostes.

7. Els dispositius virtuals es poden assignar a dispositius lògics de manera implícita o explícita. Quines conseqüències pot tenir el fet de disposar només de l'assignació explícita? I si només tenim la implícita? Justifiqueu les respostes.

8. Quines conseqüències té el fet de redireccionar l'entrada i la sortida estàndard del procés comodí cap un terminal? Com es podria solucionar? Justifiqueu les respostes.

9. Què fa el programa d'exemple d'ús del Windows API de l'apartat 7.2?

Solucionari

Exercicis d'autoavaluació

1. Es poden imprimir **a**, **c** i **d** perquè la impressora és un dispositiu orientat a caràcters i, per tant, es poden imprimir tots els fitxers de text. Hem de tenir clar que pot haver-hi fitxers executables que siguin de text, com ara els *scripts*. Els fitxers objecte o binaris no són imprimibles, ja que la seva codificació no es basa en caràcters. En cas que s'imprimeixin, el resultat serà inintel·ligible.

2. Sí, perquè per definició un dispositiu d'entrada/sortida és un objecte gestionat pel sistema operatiu en el qual els processos poden fer una operació de lectura o escriptura amb la finalitat d'obtenir informació, emmagatzemar-la, mostrar-la o transferir-la. Aquesta definició es pot aplicar al rellotge del sistema, ja que s'hi poden fer tant operacions de lectura com d'escriptura (consultar l'hora actual i posar el rellotge a l'hora, per exemple) amb la finalitat de mostrar-lo per pantalla, sincronitzar processos, sincronitzar màquines, etc.

3. El principal avantatge del fet que el sistema operatiu reconegui l'estructura de la informació és que es pot controlar que les manipulacions que es fan de la informació siguin correctes. Per exemple, que només es puguin editar fitxers de text i no de dispositiu, que els directoris no es puguin crear a mà com ho faríem amb un fitxer de text (això dóna més seguretat), etc. Com a principal desavantatge hi ha el fet que el sistema no pot preveure tots els tipus possibles de fitxers i, per tant, pot ser que el fet d'haver de conèixer totes les estructures possibles dels fitxers es converteixi en una barrera que impedeixi l'adaptació del sistema a situacions noves. Un altre desavantatge és el fet que ens dóna menys flexibilitat a l'hora de fer operacions fora del que és habitual, com per exemple editar un fitxer objecte per veure el seu contingut.

4. La utilitzen **b** i **c**. Tots dos són dispositius d'entrada i tenen una velocitat molt diferent de la del processador. A fi de reduir aquesta diferència s'utilitza l'emmagatzematge en la memòria intermèdia. Per a veure-ho més clar, només ens hem de fixar que quan l'intèrpret de comandes està executant una comanda ja es pot escriure la següent. Aquesta segona comanda es guarda en la memòria intermèdia fins que l'intèrpret la reclama. D'aquesta manera s'agilita tot el procés d'entrada/sortida.

També la **d**: el disc és un dispositiu que també treballa a una velocitat significativament més baixa que la del processador. A diferència del teclat i del ratolí, el disc és un dispositiu tant d'entrada com de sortida, i en ambdós casos es pot utilitzar la tècnica de l'emmagatzematge en la memòria intermèdia per a reduir l'impacte d'aquesta diferència de velocitat. En el cas de l'entrada, permet llegir blocs (unitat de transferència del disc) i emmagatzemar-los en la memòria intermèdia fins que els processos necessiten la informació. En el cas de la sortida, els processos deixen la informació a la memòria intermèdia i continua la seva execució sense esperar que el sistema hagi finalitzat l'escriptura en el disc.

La pantalla és un dispositiu que no té el problema de les diferències de velocitat i, per tant, no necessita utilitzar la tècnica de l'emmagatzematge en la memòria intermèdia.

5. Un senyal de fi de fitxer avisa els processos que estan llegint d'un recurs que ja no en podran extreure més informació. Es podria deduir que un procés que utilitza una finestra rebrà un senyal de fi de fitxer quan s'hagi tancat la finestra.

6. Si no existissin les operacions obrir i tancar, el sistema no tindria coneixement de les sessions de treball amb els fitxers i no podria distingir quines operacions es fan associades a una sessió o a una altra. Com a conseqüència d'això, no es podria guardar informació de l'estat de cadascuna de les sessions ni de la seva evolució.

Tot això es tradueix en el següent:

a) Amb els accessos seqüencials, el sistema no tindria constància de quina era l'última posició a què s'ha accedit. Aquesta informació s'hauria de gestionar des dels programes i passar-la com a paràmetre al sistema. En conseqüència, les operacions d'accés seqüencial no existirien i s'haurien de construir per programa mitjançant operacions d'accés directe.

b) Les operacions d'accés haurien de fer servir dispositius físics o lògics, però no virtuals, ja que és durant l'operació obrir que es fa l'associació en aquests dispositius.

c) Finalment, encara que això no afecti els paràmetres de les operacions d'accés, s'ha de fer notar que cada operació d'accés hauria de verificar el drets d'accés al dispositiu.

Així doncs, les operacions d'accés serien totes directes i tindrien el paràmetres següents:

Estat = accés(*nom*,*pos*,*buff*,*cont*)

On *nom* és el nom del dispositiu lògic, *pos* és la posició del dispositiu a què es vol accedir, *buff* és la variable que defineix on s'efectuarà l'accés o bé o des d'on es farà i *cont* és la quantitat d'informació que s'ha de transferir.

7. Si només es disposés de l'assignació explícita, no es podrien fer redireccionaments de l'entrada/sortida dels programes i estaríem limitats a les entrades/sortides definides en el codi.

Si només es disposés de l'assignació implícita, els programes no podrien especificar per a quins dispositius voldrien fer les entrades/sortides. Estarien obligats a utilitzar els dispositius estàndard que hagués previst el sistema. El fet de poder fer redireccionaments o no estaria condicionat pel tipus de mecanisme que proporcionés el sistema.

8. La conseqüència principal és que tot el que s'escriu es veu dos cops, un per efecte del mode *echo* del terminal com a dispositiu d'entrada i un altre com a resultat de la sortida que efectua el programa comodí. Per a evitar-ho i veure únicament un cop el que s'escriu, s'hauria de desactivar el mode *echo* del terminal.

9. El programa es comportarà de forma similar a la comanda `cat` de l'UNIX, és a dir, copiarà a la sortida estàndard el contingut del fitxer el nom del qual es passa com a paràmetre. En cas que es cridi el programa sense paràmetres, llavors copiarà el que arribi per l'entrada estàndard.

Glossari

***cooked/raw* ('cuinat/cru')** *m* Modes d'edició associats als dispositius d'entrada, com ara els terminals. El mode *cooked* fa que alguns caràcters que componen l'entrada s'interpretin com a comandes que alteren la informació que s'ha introduït. El mode *raw* no interpreta cap caràcter i, per tant, tots els caràcters es lliuren al procés que fa l'operació d'entrada.

dispositiu d'entrada/sortida *m* Objecte gestionat pel sistema operatiu en el qual els processos poden fer operacions de lectura/escriptura amb la finalitat d'obtenir informació, emmagatzemar-la, mostrar-la o transferir-la.

dispositiu físic *m* Dispositiu que existeix físicament. Està format pel perifèric i pel seu maquinari de control, que constitueixen la part física, i pel programari que els gestiona, anomenat programa de control.

dispositiu lògic *m* Dispositiu que no existeix físicament. És el resultat d'un programari del sistema que el crea.

dispositiu nul *m* Dispositiu d'entrada/sortida en el qual es pot escriure tot el que es vol i sempre continua buit. Ignora qualsevol cosa que s'hi escrigui i en lectura no retorna mai cap informació. Es basa exclusivament en el programari que el defineix.

dispositiu virtual *m* Dispositiu que no està associat *a priori* a cap dispositiu concret. El programa l'utilitza igual que ho faria amb qualsevol dispositiu, però sense conèixer *a priori* en quin dispositiu en concret s'efectuaran les operacions que s'hi especifiquen. Serà en temps d'execució del programa que el sistema associarà un dispositiu lògic al dispositiu virtual mitjançant el codi del sistema operatiu.

***echo* (ressò)** *m* Mode dels dispositius d'entrada/sortida, com ara els terminals, que provoca que es realitzi una sortida de tota la informació que entra pel dispositiu amb l'objectiu de verificar i, si és necessari, corregir la informació que s'està entrant (vegeu *cooked/raw*).

emmagatzematge en la memòria principal *m* Tècnica d'entrada/sortida que utilitza una memòria intermèdia anomenada *buffer* a fi d'esmorteir les diferències de velocitat dels dispositius respecte de la velocitat d'execució dels processos.
en *buffering*

file descriptors *m* Nom que reben el dispositius virtuals, o canals, en l'UNIX.

gestió de cues *f* Tècnica que fan servir les SPOOL (*Simultaneous Peripherical Operation On-Line*) que permet que les entrades/sortides d'un procés tinguin un pas intermedi per dispositius de gran capacitat d'emmagatzematge. Fan que el computador treballi amb un procés en concret mentre, concurrentment, els dispositius finals van traient o incorporant informació de manera més lenta.
en *spooling*

independència dels dispositius *f* Objectiu del sistema operatiu que consisteix a proporcionar una màquina virtual que uniformitzi els dispositius tot respectant les seves particularitats. Això s'aconsegueix independitzant el codi dels programes respecte dels dispositius que utilitzen i de les operacions amb les quals s'hi adrecen mitjançant la utilització de dispositius virtuals i operacions independents.

major* i *minor *m* Parella de nombres que identifiquen internament els dispositius de l'UNIX. El *major* identifica el tipus de dispositiu i el *minor* identifica un dispositiu concret del tipus *major*.

named pipe *m* Dispositiu lògic de l'UNIX que es comporta com una *pipe*, amb la diferència que té un nom que consta en l'espai de noms del sistema de fitxers.

operacions uniformes *m* Uniformització de les operacions que proporciona el sistema operatiu a fi d'aconseguir la independència dels programes respecte de les entrades/sortides. Per tal de proporcionar aquesta nova funció, el sistema afegeix una capa de programari per sobre dels programes controladors que depenen dels dispositius, que són les operacions uniformes d'entrada/sortida (són independents dels dispositius).

pipe *m* Dispositiu lògic de l'UNIX destinat a comunicar processos. El seu funcionament és el d'una cua de caràcters, amb una longitud fixa, en la qual els processos poden escriure i llegir. És un dispositiu que no figura en el sistema de fitxers, es crea quan l'obre un procés i es destrueix quan l'últim procés que el té obert es tanca.

programa controlador *m* Programari que gestiona un dispositiu d'entrada/sortida.

en *driver*

Bibliografia

Bibliografia bàsica

Silberschatz, A.; Galvin; Gagne, G. (2008, 8a. ed.). *Operating Systems Concepts*. John Wiley & Sons.

Tanembaum, A. (2009). *Modern Operating Systems*. Prentice-Hall.

Bibliografia complementària

Robbins, K.; Robbins, S. (2000). *UNIX: programación práctica*. Prentice Hall.

Recursos de l'aula. Manual de crides al sistema UNIX i programes d'exemple.