

El nucli del sistema operatiu

Sistemes Operatius 1

Oliver Díaz / Lluís Garrido

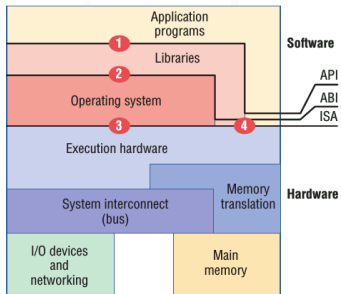
Grau d'Enginyeria Informàtica

El nucli: és el cor del sistema operatiu, és qui es “comunica” amb el maquinari per aconseguir fer d'àrbitre, ser un il·lusionista i oferir una interfície comuna a les aplicacions. Com funciona tot plegat?

Organització de les transparències

- ➊ **Introducció**
- ➋ El concepte de procés
- ➌ Mode dual d'operació
- ➍ Transferència de control de seguretat entre usuari i nucli
- ➎ Annex: estructura d'un sistema operatiu

Què és un sistema operatiu?



Un sistema operatiu es pot veure com una màquina estesa:

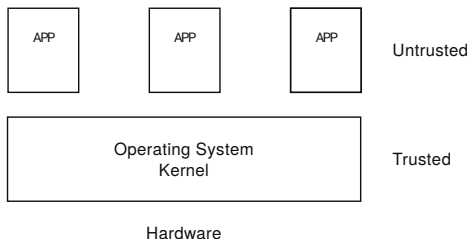
- Oculta a l'usuari de tots els detalls escabrosos que han de ser realitzats per accedir als dispositius.
- Ofereix a l'usuari una **màquina virtual** i una **interfície** (les crides a sistema), molt més senzilla d'utilitzar.

El **nucli** (en anglès, **kernel**) del sistema operatiu té la tasca de **protegir**:

- **Fiabilitat**: un error de programació o malícia en una aplicació no hauria d'afectar a la resta d'aplicacions. El sistema operatiu ha de funcionar correctament independentment del que faci l'aplicació d'usuari.
- **Seguretat**: cal limitar les accions que les aplicacions individuals poden realitzar. Pex, cal impedir que una aplicació pugui escriure a disc i modificar el codi del sistema operatiu.
- **Privacitat**: en un sistema multiusuari, cada usuari (i les aplicacions associades) només hauria de poder accedir a les dades que té permès accedir.
- **Eficiència**: les aplicacions no haurien de poder consumir tots els recursos (CPU, memòria, ...) en detriment d'altres aplicacions. El sistema operatiu ha de repartir els recursos entre les aplicacions.

El nucli del sistema operatiu s'ha d'encarregar de la **protecció**

- El nucli és una peça de programari de confiança per fer qualsevol cosa amb el maquinari; amb **accés total a les capacitats del maquinari**.
- Tota la resta, les aplicacions, que no són de confiança, executen en un entorn restringit, amb un **accés restringit** a la capacitat real de la màquina.



Com s'aconsegueix això? Veiem-ho...

Situacions similars esdevenen en aplicacions reals. Per exemple, moltes vegades les aplicacions d'usuari han d'executar codi no fiable de tercers

- Els navegadors web pot executar codi JavaScript. Si el navegador no proveeix de protecció un codi JavaScript maliciós podria prendre control del navegador. Les contrasenyes, etc. podrien ser enviades a l'atacant.
- Altres aplicacions acostumen a poder executar codi de tercers. Això és beneficiós per a l'aplicació ja que permet ampliar la funcionalitat de l'aplicació en sí; però planteja la qüestió de protegir-se de codi de tercers.

En un sistema operatiu s'utilitza el concepte de **procés**: és una aplicació d'usuari que s'executa amb drets restringits¹.

- El nucli s'executa directament al processador amb drets il·limitats. Els processos s'executen al processador amb limitacions sobre operacions potencialment perilloses. Això s'aconsegueix gràcies a la col·laboració del maquinari.
- La gestió de l'accés d'un procés al maquinari està controlat pel nucli. Un procés necessita autorització del nucli per accedir a una determinada posició de memòria, per escriure a disc, per modificar la configuració del maquinari, etc.

¹A sistemes operatius s'utilitza el concepte de “procés” en comptes de parlar del terme “aplicació”

La protecció implica diverses qüestions que cal respondre

- 1 El **concepte de procés**: què és un procés i en què es diferencia d'un programa?
- 2 **Mode dual d'operació**: com implementa un sistema operatiu un procés? Quin maquinari fa falta per executar de forma eficient un procés en mode restringit?
- 3 **Transferència del control de seguretat**: com es pot passar la frontera entre els processos, no fiables, i el nucli, fiable?²

²De moment no diferenciarem entre el concepte de “nucli” i “sistema operatiu”.

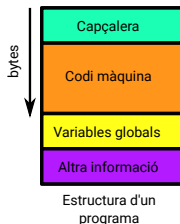
Esquema “El nucli del SO”

- ➊ Introducció
- ➋ **El concepte de procés**
- ➌ Mode dual d'operació
- ➍ Transferència de control de seguretat entre usuari i nucli
- ➎ Annex: estructura d'un sistema operatiu

El concepte de procés

Aquest és el model de programació al qual estem acostumats

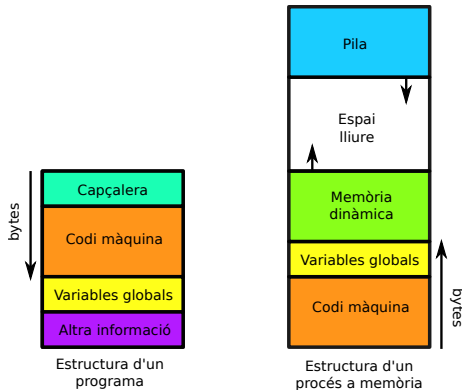
- 1 Es realitza un programa en un llenguatge d'alt nivell (per exemple, C).
- 2 S'utilitza un compilador per convertir el codi en instruccions màquina.
- 3 El compilador genera un **codi executable del programa**. El codi executable té una “estructura interna” coneguda pel sistema operatiu.



El concepte de procés

En executar un programa

- El sistema operatiu copia les instruccions màquina de disc a memòria física, “crea” la pila i la zona de memòria dinàmica.
- Per executar el procés només cal que el sistema operatiu cridi a la funció “main”. Ja tenim un **procés** executant-se!

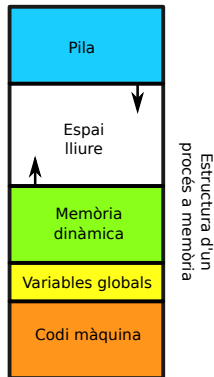


El concepte de procés

On s'emmagatzema cada variable en aquest exemple ?

- Regió global: variables globals
- Regió de pila: variables locals a la funció i l'històric de les crides a funcions.
- Regió de memòria dinàmica: hi apunta la variable “vector”.

```
int global;  
  
int func(int a)  
{  
    int b;  
    // Processament  
}  
  
void main(void)  
{  
    char str[10];  
    int local, *vector;  
  
    global = local = 0;  
  
    // Processament  
    func(2);  
    // Reservar memoria  
    vector = malloc(1000);  
}
```



Un **procés és una instància d'un programa**, de la mateixa forma que un objecte és una instància d'una classe a programació orientada a objectes.

- La “creació” de, per exemple, la pila i la zona de memòria dinàmica implica assignar els permisos necessaris a aquella zona perquè el procés hi pugui llegir i/o escriure.
- Podem executar múltiples instàncies d'un mateix programa. El sistema operatiu s'encarrega de carregar a memòria múltiples còpies de les instruccions màquina del programa, així com crear múltiples còpies de la pila i la zona de memòria dinàmica.
- El compilador és un programa i en executar-lo el sistema operatiu crea el procés corresponent. El mateix passa amb l'interpret de Python, Java, un navegador, ...

- ➊ Introducció
- ➋ El concepte de procés
- ➌ **Mode dual d'operació**
- ➍ Transferència de control de seguretat entre usuari i nucli
- ➎ Annex: estructura d'un sistema operatiu

Suposem que el sistema operatiu ha copiat les instruccions màquina del programa a memòria i ara...

- ❶ Comença a executar el procés. Per això salta a la funció principal d'entrada (el “main” al C).
- ❷ La CPU comença a executar les instruccions que hi troba: multiplicacions, escriptura i lectura de posicions de memòria, etc.

Com es pot assegurar que les instruccions que executa un procés no “fan mal” a altres processos o el nucli del sistema operatiu? El sistema operatiu ha d'assegurar **protecció** entre processos!

Mode dual d'operació

Com es pot assegurar que les instruccions que executa un procés no “fan mal” a altres processos o el nucli del sistema operatiu mateix?

- La solució adoptada actualment es basa en què el maquinari pot executar en dos modes. És l'anomenat **mode dual** d'operació: en **mode nucli** (o kernel) el processador pot executar qualsevol instrucció; en **mode usuari** el processador “comprova” cada instrucció abans d'executar-la.
- El nucli s'executa en mode nucli mentre que els processos en mode usuari. Si el maquinari detecta que el procés fa una operació invàlida “avisa” al nucli d'aquesta infracció.

Quin maquinari és necessari perquè el nucli pugui protegir els processos entre sí? Com a mínim

- 1 **Instruccions privilegiades:** Totes les instruccions potencialment perilloses estan prohibides en mode usuari.
- 2 **Protecció de memòria:** Tots els accessos a memòria física d'un procés fora de l'espai que se li ha estat assignat estan prohibits en mode usuari.
- 3 **Interrupcions de temporitzador:** Independentment del que faci un procés, el nucli del procés ha de prendre el control de forma periòdica de la màquina per fer “gestions”.

A més, el maquinari ha de proveir d'un mecanisme per passar de forma segura de mode nucli a mode usuari i a la inversa.

Mode dual d'operació: instruccions privilegiades

Els **processos d'usuari** poden utilitzar només un **subconjunt de totes les instruccions disponibles** (mode d'usuari d'execució), mentre que **el nucli** té a la seva disposició **tota la potència de la màquina** (mode nucli d'execució).

Quines instruccions s'haurien de restringir?

- Un procés d'usuari no ha de ser capaç de modificar, entre altres coses, el seu nivell de privilegi d'execució. Un procés d'usuari no ha de poder modificar les interrupcions del maquinari, ni canviar la regió de memòria a la qual pot accedir, etc.
- Un procés pot obtenir privilegis de forma indirecta mitjançant una **crida a sistema**: una crida que transfereix el control al sistema operatiu per fer l'operació demanada pel procés (i.e. escriure a disc).

Mode dual d'operació: instruccions privilegiades

Què passa si un procés d'usuari executa una instrucció no autoritzada?

- Aquest esdeveniment provoca una **excepció de processador**. L'excepció provoca que el maquinari transfereixi de forma immediata el control a una determinada funció del nucli, que pot prendre les accions necessàries³.
- Típicament, en produir-se una excepció per una instrucció invàlida, aquesta funció decideix aturar el procés que ha provocat l'excepció (es "mata").

³Ara veurem com sap el maquinari a quina funció del nucli s'ha de transferir el control.

Mode dual d'operació: protecció de memòria

Per executar un procés (d'usuari), cal que

- El procés resideixi a memòria física perquè pugui ser executat.
- El nucli del sistema operatiu ha de residir també a memòria perquè pugui gestionar les crides a sistema del procés, interrupcions de temporitzador així com les possibles excepcions.

Típicament hi pot haver múltiples (desenes!) de processos executant

- El sistema operatiu ha de configurar el maquinari perquè cada procés només pugui accedir a la seva parcel·la de memòria.
- Si un procés intenta accedir (per lectura o escriptura) a una posició de memòria no permesa, es produeix també **una excepció**.

Com pot evitar un sistema operatiu (en particular, el nucli) que un procés accedeixi a parts no permeses a memòria física? De nou, es fa gràcies al maquinari. Al llarg de la història hi ha hagut diversos formes d'enfocar el problema.

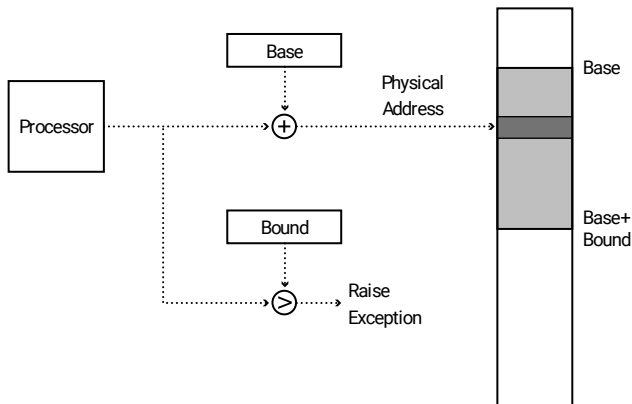
- Un esquema senzill fa servir dos registres maquinari: **base i limit**.
- Avui en dia els ordinadors de sobretaula fan servir **memòria virtual**.

Anem a veure'ls breument, ho veurem amb més detall al capítol de memòria virtual.

Mode dual d'operació: protecció de memòria

Esquema amb dos registres: **base i limit**. Aquests registres només poden ser modificats per instruccions privilegiades.

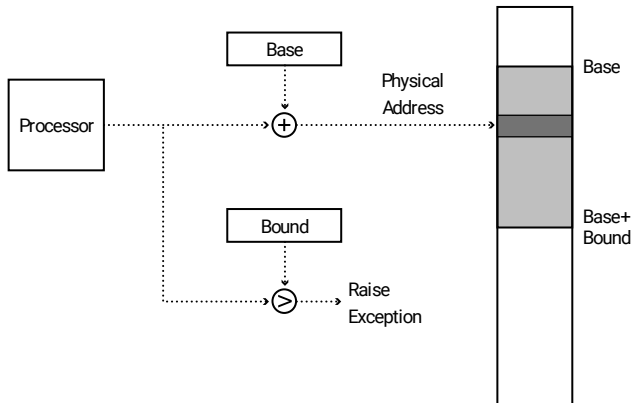
- Cada cop que s'accedeix a memòria, se suma la base a l'adreça. Si se supera el límit, es produeix una excepció.



Mode dual d'operació: protecció de memòria

Observar que, amb aquest esquema

- Cal realitzar una comparació i una suma cada cop que es vol accedir a una posició de memòria. Però el cost addicional val la pena per tal d'implementar la protecció.
- El sistema operatiu opera directament sobre memòria real.



Esquema amb dos registres: base i limit. Té importants defectes

- Pila i memòria dinàmica expansible? Amb només dos registres cal preveure la memòria màxima que ocuparà el procés en el moment de carregar-lo de disc a memòria. Aquesta previsió és difícil de fer.
- Compartició de memòria? Amb aquest esquema no es pot aconseguir que diversos processos “comparteixin” una zona de memòria (per exemple, la de les instruccions màquina).
- Fragmentació de memòria? Amb aquest esquema tot el procés ocupa un espai “continu” que no es pot fragmentar en trossos.

Memòria virtual: utilitzat a la majoria de processadors moderns d'avui en dia.

- El processador genera “adreces virtuals” que són traduïdes per un maquinari específic a una adreça física.
- Aquesta és la sortida per pantalla del codi `exemple1.c`. Totes les adreces que es mostren a continuació són virtuals, és a dir, no són pas direccions físiques reals.

Funcio main: 4195741

Variable global: 6295628

Variable local: 140733565474012

Variable vector: 140733565474000

Vector apunta a: 38584336

Mode dual d'operació: protecció de memòria

Exemple de mapat d'una adreça virtual a una adreça física (arquitectura de 16 bits)

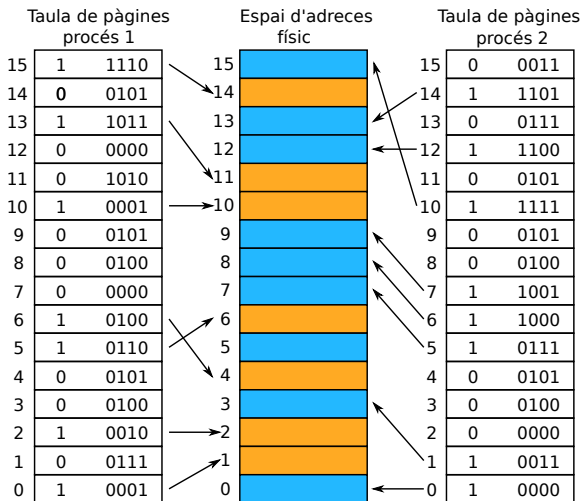


Característiques de la memòria virtual

- Els processos “viuen” a l'espai virtual. Cada procés creu té disponible tot l'espai de memòria possible tot i que no existeixi físicament.
- El sistema operatiu (nucli) s'encarrega de gestionar les taules dels processos.
- El mapat d'una adreça virtual a adreça física es realitza a nivell de maquinari.
- Posicions contigües a memòria virtual no tenen perquè ser contigües a l'espai físic.

Mode dual d'operació: protecció de memòria

Exemple amb dos processos: la memòria virtual permet gestionar la protecció de memòria entre aquests.



Recordem quin maquinari és necessari perquè el nucli pugui protegir els processos i usuaris entre sí. Hem vist fins ara els primers dos punts:

- 1 **Instruccions privilegiades:** Totes les instruccions potencialment perilloses estan prohibides en mode usuari.
- 2 **Protecció de memòria:** Tots els accessos a memòria física d'un procés fora de l'espai que se li ha estat assignat estan prohibits en mode usuari.
- 3 **Interrupcions de temporitzador:** Independentment del que faci un procés, el nucli del procés ha de prendre el control de forma periòdica el control de la màquina per fer "gestions".

Mode dual d'operació: interrupcions de temporitzador

En el moment en què els sistema operatiu comença a executar un procés, aquest és lliure d'executar qualsevol instrucció no privilegiada, cridar a qualsevol funció, fer un bucle, etc.

- Cada procés creu que té disponible tota la CPU per a ell sol. Per donar aquesta il·lusió el sistema operatiu ha de poder indicar, periòdicament, quin procés s'executa a la CPU per assegurar que tots els processos hi obtenen temps d'execució.
- El procés es pot penjar en un bucle infinit. Com es pot aturar? El sistema operatiu ha de poder comprovar, periòdicament, si l'usuari vol aturar el procés pel motiu que sigui.
- Per fer totes aquestes gestions el **sistema operatiu ha de poder prendre, periòdicament, el control de la CPU.**

Mode dual d'operació: interrupcions de temporitzador

S'utilitza un **temporitzador** (a nivell de maquinari) que **interromp l'execució del procés** cada determinat temps (per exemple, 10 mil·lisegons).

- En arribar el temporitzador a zero es transfereix el control a una determinada funció sistema operatiu i es passa a mode nucli d'execució. El sistema operatiu pot aleshores decidir quin procés s'executa a continuació a la CPU, es poden planificar accessos a disc, a xarxa, etc.
- Es poden produir altres interrupcions de l'execució d'un procés: per exemple, un disc avisa al sistema operatiu que ha llegit les dades demanades, el ratolí avisa que l'usuari l'ha mogut, etc.

Quina funció crida el maquinari en produir-se una interrupció? Ho veurem en un moment.

Sobre el mode dual d'operació

Hem parlat sobre el mode dual d'operació (mode nucli i d'usuari).

- El resum: **el maquinari ofereix una sèrie de serveis perquè el sistema operatiu (nucli) pugui gestionar l'execució de processos**. Hi ha una estreta col·laboració entre maquinari i nucli.
- El sistema x86 suporta més modes d'operació (en concret, 4). La idea al darrera és permetre que el sistema operatiu pugui executar fent servir múltiples nivells de privilegi per assegurar que un error en una determinada part no provoqui una fallada del sistema. Actualment ni Linux, ni MacOS, ni Windows fan servir els 4 nivells.
- A l'actualitat, a alguns sistemes operatius, el nucli del sistema operatiu executa en mode nucli mentre que una altra porció del sistema operatiu en mode usuari (per seguretat). Això depèn de l'arquitectura del sistema operatiu (veure Annex).

Fa pocs anys els sistemes operatius no eren tan fiables i segurs com avui en dia

- L'antic MS-DOS, de Microsoft, no proveïa de protecció de memòria. Tot i que era acceptable en aquella època, errors a les aplicacions feien petar tot el sistema. I el sistema era molt vulnerable a virus.
- Fins el 2002 els sistemes Mac OS no proveïen de la capacitat d'interrupcions de temporitzador. El procés havia de cridar manualment i de forma periòdica al sistema operatiu perquè aquest pogués comprovar si hi havia alguna cosa a fer. Però si el procés es penjava en un bucle infinit l'ordinador es penjava. I aleshores calia reiniciar l'ordinador!

Penseu-hi:

- Per a cadascun dels mecanismes necessaris per suportar mode dual – instruccions privilegiades, protecció de memòria i interrupcions de temporitzador – explicar què pot anar malament si en falla un d'ells, suposant que disposem dels altres dos.

Aquest és l'esquema de la presentació vist al principi

- 1 Introducció
- 2 El concepte de procés
- 3 Mode dual d'operació
- 4 **Transferència de control de seguretat entre usuari i nucli**
- 5 Annex: Estructura d'un sistema operatiu

Hem vist els primers tres punts. Com es pot passar la frontera entre els processos, no fiables, i el nucli, fiable?

Tot procés s'executa en mode usuari amb la protecció necessària. Com passem del mode usuari a mode nucli? I a la inversa, de mode nucli a mode usuari? Aquestes transicions són habituals i, per tant, ha de ser ràpides i fiables. Ara veurem:

- Quan es realitza la transferència de **mode usuari a mode nucli**?
- Quan es realitza la transferència de **mode nucli a mode usuari**?
- Com es realitza, tècnicament, aquesta transferència?

Ens concentrem primer en la transferència de **mode usuari a mode nucli**; la transició en sentit invers es realitza “desfent” la transferència de mode usuari a mode nucli.

Hi ha tres raons per les quals es pot realitzar la transferència de mode usuari a mode nucli

- **Excepcions**
- **Interrupcions**
- **Crides a sistema**

Les excepcions i les crides a sistema són **síncrones** (es produeixen per l'execució del procés), mentre que la interrupció és **asíncrona**.

Mode usuari a mode nucli: excepcions

Una **excepció és una condició inesperada de l'execució del procés.**
Per exemple,

- El procés intenta executar una instrucció privilegiada.
- El procés intenta accedir a una posició de memòria fora de l'espai permès.
- Accés per escriptura a una posició de memòria de només lectura del mateix procés.
- Divisió per zero.
- “Breakpoint” d'un debugger, etc.

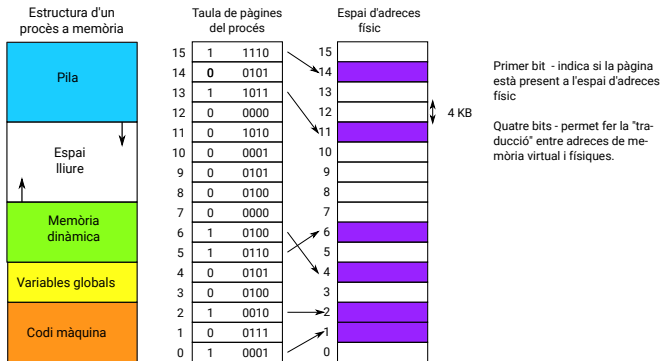
En produir-se una excepció, el maquinari atura l'execució del procés actual i passa a executar, en mode nucli, una determinada funció del nucli del sistema operatiu. Quina funció? Ho veurem...

Les excepcions són particularment útils a la **virtualització**: emular maquinari que realment no existeix.

- Per exemple, els processadors de baix consum poden no suportar determinades operacions en coma flotant. El sistema operatiu pot “simular” aquestes operacions: quan un procés realitza una operació d'aquest tipus es produeix una excepció; el sistema operatiu emula la instrucció; i es retorna el valor al procés i es continua l'execució normalment.
- Quan un sistema operatiu s'executa en mode usuari a una màquina virtual (pex, VirtualBox) aquest intentarà accedir al maquinari com si tingués privilegis totals. El sistema operatiu amfitrió (el que executa realment a la màquina) capturarà aquest intent i realitzarà l'operació demanada.

Mode usuari a mode nucli: excepcions

Les excepcions són també l'eina principal per a la gestió de la memòria virtual. Si un procés accedeix a una posició no permesa del procés o accedeix a una posició vàlida però que en aquell moment no és a la RAM, es produeix una excepció.



Ho veurem amb detall al capítol de memòria virtual.

Una **interrupció és un senyal asíncron de maquinari produït per un esdeveniment** (no produït per l'execució del procés).

- Funciona de forma similar a una excepció: el processador atura l'execució del procés actual i es passa a executar una determinada funció del sistema operatiu.
- Cada tipus d'interrupció té una funció gestor associada: interrupcions de temporitzador; interrupcions sobre operacions d'entrada-sortida: avisar quan el ratolí es mou, quan es finalitza una operació a disc, quan arriben dades per la xarxa, etc.

Mode usuari a mode nucli: crides a sistema

Una **crida a sistema** és qualsevol funció proveïda pel sistema operatiu i que pot ser cridada per l'usuari.

- És la interfície que ofereix el sistema operatiu als processos per accedir als dispositius o realitzar una altra operació a què només té dret d'accés el nucli.
- A nivell de programació s'assembla a una crida a una funció: la crida a sistema té paràmetres i valors de retorn.

Les crides a sistema s'implementen gràcies a una instrucció màquina específica ("trap instruction") que fa que:

- 1 La CPU passi de mode usuari a mode nucli i es cridi una determinada funció determinada del sistema operatiu.
- 2 El sistema operatiu realitza aleshores l'operació demanada i, un cop finalitzat, torna a mode usuari continuant l'execució just després de la "trap instruction".

El sistema operatiu ofereix un gran nombre de crides a sistema. A Linux són més de 250! Les crides a sistema inclouen operacions com:

- Control de processos (creació, execució, finalització, ...)
- Gestió de fitxers (creació, obrir, llegir, escriure, tancar, esborrar, ...)
- Gestió de dispositius (connexió, desconnexió, escriure, llegir, ...)
- Comunicació entre processos (escriure i enviar missatges, ...)
- Informació de manteniment (establir hora i data, saber quins processos s'executen, ...)

Mode usuari a mode nucli: crides a sistema

Veiem un exemple de crida a sistema. Les següents funcions permeten escriure dades a un dispositiu

- Podem fer servir una crida al sistema operatiu

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

La funció `write`, disponible a una llibreria d'usuari, 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 “trap instruction”).

- La funció `printf` és una crida a una llibreria de l'espai d'usuari

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

Aquesta instrucció genera, a l'espai d'usuari, la cadena a imprimir. Un cop generada es fa la crida al sistema operatiu amb un `write`.

Mode usuari a mode nucli: crides a sistema

Anem a rastrejar quines crides a sistema realitza una aplicació qualsevol. La utilitat `strace` ens ho permet fer: des de la línia de comandes executar

```
strace ./exemple2
```

Podeu rastrejar les crides a sistema d'una altra aplicació, la que vulgueu

```
strace kwrite
```

Mode nucli a mode usuari

Causes que fan que es passi de **mode nucli a mode usuari**

- **Restablir l'execució** després d'una excepció, interrupció o crida a sistema: quan el nucli finalitza de gestionar l'operació, restableix l'execució passant a mode usuari.
- **Canviar l'execució a un altre procés**: el sistema operatiu pot decidir canviar l'execució a un altre procés. Per fer-ho el sistema operatiu ha de recuperar la informació del procés "antic" i restablir l'execució en el punt en què es va aturar.
- **Executar un nou procés**: per a crear un procés el sistema operatiu carrega en memòria les instruccions a executar, configura la pila, estableix la primera instrucció a executar del procés, i passa a mode usuari.
- **Upcall** (crida del nucli als processos): mecanisme perquè els processos puguin rebre notificacions asíncrones d'esdeveniments; són similars a interrupcions però a nivell d'usuari.

Per pensar-hi...

- Què passaria si, en produir-se una excepció, el maquinari ho notifiqués al sistema operatiu però aquest no prengués cap acció cap acció? Hi hauria fallades de seguretat? Altres coses?
- Què passaria si, en produir-se una interrupció de temporitzador, el maquinari no fes cap notificació al sistema operatiu? (La resta d'interrupcions funcionen)

Tot procés s'executa en mode usuari amb la protecció necessària. Com passem del mode usuari a mode nucli? I a la inversa, de mode nucli a mode usuari? Aquestes transicions són habituals i, per tant, ha de ser ràpides i fiables.

- Quan es realitza la transferència de mode usuari a mode nucli?
- Quan es realitza la transferència de mode nucli a mode usuari?
- Com es realitza, tècnicament, aquesta transferència?

Hem vist els dos primers punts. El darrer punt ja l'heu vist a Estructura de Computadors.... repassem de forma beu.

Com s'implementa, tècnicament, la transferència de mode usuari a mode nucli? I de mode nucli a mode usuari? Aquesta transferència es realitza gràcies a la col·laboració entre maquinari i programari (el nucli del sistema operatiu). Anem a veure com s'implementen

- ❶ La gestió de les interrupcions
- ❷ La gestió de les excepcions
- ❸ Les crides a sistema

Els mecanismes anteriors s'utilitzen per implementar també

- ❶ Executar nous processos
- ❷ Notificacions asíncrones d'esdeveniments a nivell d'usuari (upcall)

La gestió de les interrupcions, excepcions i crides a sistema

El vector d'interrupcions

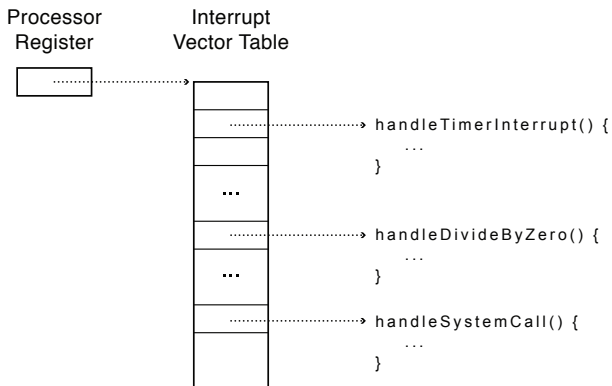
En produir-se una interrupció, excepció o crida a sistema, es passa a mode nucli. Però com sap el maquinari quina és la funció del sistema operatiu que ha de cridar?

- El maquinari identifica primer si un dispositiu ha realitzat una interrupció, si s'ha produït una excepció o si s'ha executat la "trap instruction".
- Cada cas s'identifica, a nivell de maquinari, per un **número sencer**.

La gestió de les interrupcions, excepcions i crides a sistema

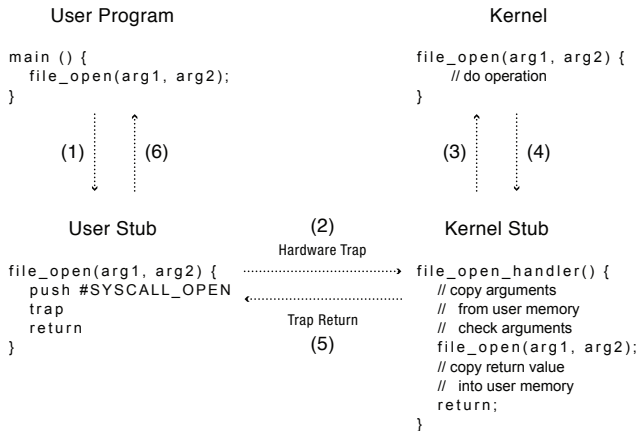
El vector d'interrupcions

El número sencer està associat a una entrada en un **vector d'interrupcions**. Aquesta entrada apunta a la funció que gestionarà l'esdeveniment (interrupció, excepció o crida a sistema).



Crides a sistema

Detalls tècnics de la **crida a sistema**: és, a nivell de programació de l'usuari, igual que una crida a qualsevol altre funció. Però es transfereix el control al sistema operatiu.



Els passos que es realitzen per fer una **crida a sistema** són:

- 1 L'usuari fa la crida (pex write). Aquesta funció emmagatzema, en mode usuari, els paràmetres de la funció a la pila d'usuari o en registres.
- 2 S'executa la "trap instruction". A sistemes x86 també s'anomena sovint "interrupció de programari". Es passa a mode nucli i s'executa una determinada funció del sistema operatiu.
- 3 Dintre del sistema operatiu cada crida a sistema s'implementa en funcions diferents. Es copien els arguments a la pila del nucli i es comprova que tinguin valors correctes.
- 4 S'executa la crida a sistema i, en acabar, se segueix el camí invers per continuar l'execució en el punt en què s'ha realitzat la crida. Es torna a passar a mode usuari.

El maquinari ofereix una sèrie de serveis perquè el sistema operatiu (el nucli) pugui gestionar l'execució de processos. Hi ha una estreta col·laboració entre maquinari i el sistema operatiu.

- El maquinari ofereix instruccions privilegiades, eines de protecció de memòria i interrupcions de temporitzador.
- Es pot passar de mode usuari a nucli amb les excepcions, interrupcions i crides a sistema.

Gràcies a aquestes eines oferides pel maquinari el sistema operatiu pot fer d'àrbitre, d'il·lusionista i oferir una interfície comuna a les aplicacions (processos).

Les tendències futures són, entre altres:

- **Suport per a ajust fi a la protecció:** una aplicació executada per un usuari té els permisos d'aquell usuari. Aquesta aplicació pot “robar” informació de l'usuari d'altres processos sense comprometre el sistema operatiu. Actualment alguns sistemes operatius (pex smartphones) comencen a incorporar sistemes per prevenir a les aplicacions l'accés a informació sensitiva.
- **Suport maquinari per a virtualització:** actualment el sistema operatiu amfitrió gestiona les interrupcions, excepcions i crides a sistema que realitza el sistema operatiu convidat (virtual). Atès l'ús cada cop més extens de màquines virtuals, les arquitectures dels processadors s'estan redissenyant perquè el sistema operatiu convidat pugui gestionar directament les interrupcions, excepcions o crides a sistema.

Aquest és l'esquema de la presentació vist al principi

- 1 Introducció
- 2 El concepte de procés
- 3 Mode dual d'operació
- 4 Transferència de control de seguretat entre usuari i nucli
- 5 Annex: Estructura d'un sistema operatiu

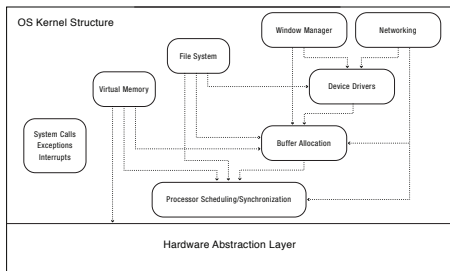
Hem vist que el nucli realitza les tasques bàsiques de protecció entre processos. Però, com està organitzat la resta del sistema operatiu? Hi ha molts mòduls en un sistema operatiu i aquests acostumen a interaccionar entre ells. Els dissenyadors dels sistemes operatius s'enfronten a un dilema fonamental:

- **Nuclis monolítics.** Si es centralitza la funcionalitat de (tot) el sistema operatiu al voltant del nucli, s'augmenta el rendiment i es facilita la integració total de mòduls. Però hi ha menys flexibilitat per fer-hi modificacions.
- **Microkernels.** Si es descentralitza la funcionalitat, es pot reduir el rendiment però es facilita la modificació o ampliació del sistema.

Nuclis monolítics

Utilitzat avui en dia àmpliament als sistemes operatius comercials: Windows, MacOS, Linux.

- La majoria de la funcionalitat del sistema operatiu està implementada al nucli⁴. Els mòduls del nucli criden directament a altres mòduls per fer les tasques necessàries.
- La interfície gràfica, llibreries o la línia de comandes s'executen en mode usuari. No formen part del sistema operatiu.



⁴Per això sovint no diferenciarem entre "sistema operatiu" i "nucli".

El dissenyador del sistema operatiu és lliure per desenvolupar la API que cregui necessària entre els mòduls. Hi ha, però, dos punts que acostumen a sorgir en els sistemes monolítics per assegurar la portabilitat

- 1 Hardware Abstraction Layer (HAL): la HAL és una interfície portable que utilitza el nucli per accedir a la operacions específiques del dispositiu.
- 2 Gestors de dispositiu (device driver) dinàmics

Nuclis monolítics: gestors de dispositiu dinàmics

És important que un sistema operatiu pugui gestionar múltiples dispositius entrada-sortida.

- Actualment hi ha una gran varietat de dispositius d'entrada-sortida. Hi ha una gran diversitat d'interfícies maquinari per gestionar els dispositius. A Linux, per exemple, un 70% del codi del sistema operatiu és programari específic per accedir als dispositius.
- És interessant desacoblar el sistema operatiu dels detalls específics de cada dispositiu. La solució, àmpliament adoptada avui en dia, són els gestors de dispositius dinàmics. Es tracta d'un programari específic, programat típicament pel fabricant, que s'executa un cop el nucli ha començat a executar i que s'encarrega de gestionar els dispositius connectats a la màquina.

El gestors dinàmics tenen, però, un defecte important

- S'executen en mode nucli i, per tant, poden corrompre el nucli del sistema operatiu i les seves estructures. Un programador maliciós pot utilitzar els gestors de dispositiu per introduir-hi un virus.
- Estudis recents han demostrat que al voltant del 90% de les “penjades” del sistema es deuen a errors als gestors de dispositiu i no pas al sistema operatiu en sí.

Com es pot abordar aquest problema?

Com es pot abordar el problema anterior? Entre altres,

- **Gestors de dispositiu executats en mode usuari:** Microsoft i Apple recomanen que els gestors s'executin en mode usuari i que utilitzin les crides a sistema per accedir al dispositiu. Tot i que a l'actualitat aquests tipus de gestors comencen a ser comuns, molts codis de gestors encara fan ús de les estructures internes del sistema operatiu, cosa que dificulta la seva “traducció” a mode usuari.
- **Gestor de dispositiu en entorn restringit (sandbox):** es proposa que el gestor s'executi en un entorn restringit (amb privilegis limitats) dintre del sistema operatiu. També poden executar-se en una màquina virtual, però el rendiment és més baix.

L'alternativa al sistema monolític es executar el codi del sistema operatiu, tant com sigui possible, en mode usuari: el **microkernel**!

- La idea d'un microkernel és similar a la del gestor de finestres X11 del Unix: el gestor de finestres s'executa en mode servidor i les aplicacions no el criden directament, sinó que criden a funcions d'una llibreria que envia les peticions al servidor (com si fos una connexió de xarxa) per dibuixar coses a pantalla.
- Windows NT, precursor de Windows 8 i Windows 10, va ser dissenyat inicialment com a microkernel. Gran part de la seva funcionalitat va ser migrada un altre cop al nucli per motius de rendiment. Minix, precursor de Linux, té un disseny de microkernel. Però Linux no...

Respecte el microkernel

- Per al programador no hi ha diferència a l'hora de programar per un sistema operatiu monolític o microkernel.
- El microkernel facilita la modularització, fiabilitat i seguiment d'errors de programació. Però es redueix el rendiment ja que no es fan crides directes (com al monolític).
- A l'actualitat s'intenta s'adopta un model híbrid en què part dels serveis s'executen en mode usuari i part en model nucli, tot dependent del compromís entre complexitat i rendiment.