

U1

Programació

Multiprocés

Programació de serveis i processos DAM

Continguts

1. Conceptes bàsics
2. Programació concurrent
3. Programació paral·lela
4. Programació distribuïda
5. Avantatges i inconvenients del multiprocés
6. Classificació de processos
7. Estats d'un procés
8. Fils
9. Comunicació entre processos
10. Gestió de processos
11. Sincronització entre processos
12. Programació d'aplicacions multiprocés

1. Conceptes bàsics

Programa

Un programa és un element **estàtic**, un conjunt d'instruccions, unes línies de **codi** escrites en un llenguatge de programació, que descriuen el tractament que cal donar a unes dades inicials (d'entrada) per aconseguir el resultat esperat (una sortida concreta).

Procés

Un procés és **dinàmic**, és una instància d'un programa en execució, que realitza els canvis indicats pel programa a les dades inicials i obté una sortida concreta. El procés, a més de les instruccions, requerirà també de recursos específics per a l'execució com ara el comptador d'instruccions del programa, el contingut dels registres o les dades.

1. Conceptes bàsics

Executable

És un fitxer que conté la informació necessària per crear un procés a partir de les dades emmagatzemades d'un programa.

Anomenarem “**executable**” al fitxer que permet posar el programa en execució com a procés.

1. Conceptes bàsics

Sistema operatiu

El sistema operatiu és l'encarregat de la **gestió de processos**. Els crea, els elimina i els proveeix dels recursos necessaris que en permetin l'execució i també la comunicació entre ells.

Quan s'executa un programa, el sistema operatiu **crea una instància del programa**: el procés. Si el programa es torna a executar es crearia una nova instància totalment independent a l'anterior (un nou procés) amb les seves pròpies variables, piles i registres.

El SO també gestiona la memòria, el sistema de fitxers, l'entrada i sortida, la seguretat del sistema...

1. Conceptes bàsics

Quan un procés es troba en execució es troba completament en memòria i té assignats els recursos que necessita.

Un procés no pot escriure en zones de memòria assignada a altres processos, **la memòria no és compartida entre processos**. Cada procés té una estructura de dades en la qual es guarda la informació associada a l'execució del procés. Aquesta zona s'anomena **Bloc de Control de Procés (BCP)**.

Els processos competeixen amb altres processos executats a la vegada pels recursos del sistema.

Opcionalment el sistema operatiu permet també que els processos puguin comunicar-se i col·laborar entre ells.

Process-Id
Process state
Process Priority
Accounting Information
Program Counter
CPU Register
PCB Pointers
.....

Process Control Block

1. Conceptes bàsics

Processador

Un processador és el component de maquinari d'un sistema informàtic encarregat d'executar les instruccions.

- Un **sistema monoprocessador** és aquell que està format únicament per un processador.

P.e. Windows 95

- Un **sistema multiprocessador** està format per més d'un processador, per tant, es poden executar simultàniament diversos processos.

També es poden qualificar de multiprocessadors aquells dispositius el processador dels quals té més d'un nucli (multicore), és a dir, tenen més d'una CPU al mateix circuit integrat del processador.

2. Programació concurrent

L'**execució simultània de processos** s'anomena també **concurrència**. No vol dir que s'han d'executar exactament al mateix moment, l'intercalat de processos també es considera execució concurrent.

La majoria dels llenguatges de programació actuals poden fer ús d'aquest recurs i dissenyar aplicacions en les quals els processos s'executin de manera concurrent.

Actualment, **la majoria dels sistemes operatius aprofiten els temps de repòs dels processos, quan esperen**, per exemple, alguna dada de l'usuari o es troben pendents d'alguna operació d'entrada/sortida, per introduir al processador un altre procés, simulant així una execució paral·lela.

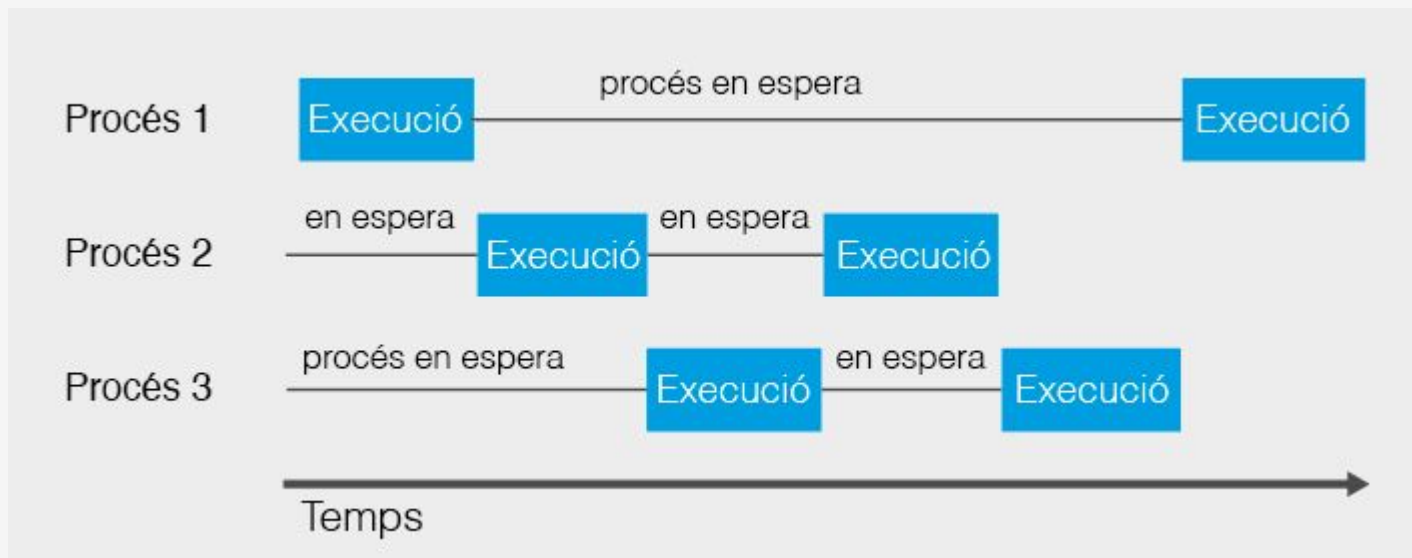
2. Programació concurrent

Processos concurrents: processos que s'executen a la vegada. Aquest fet fa augmentar el rendiment del sistema informàtic ja que aprofita més el temps del processador.

Multiprogramació: el sistema operatiu gestiona l'execució de processos concurrents a un sistema monoprocessador.

2. Programació concurrent

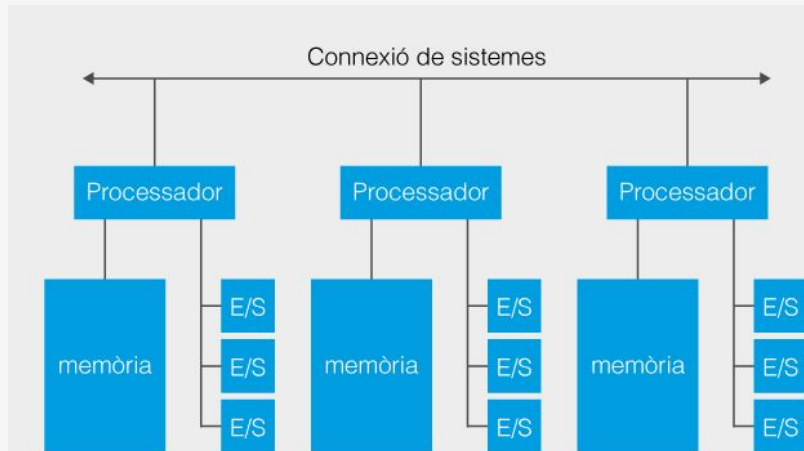
Execució de processos concurrents



2. Programació concurrent

Els **Sistemes multiprocessadors** poden ser:

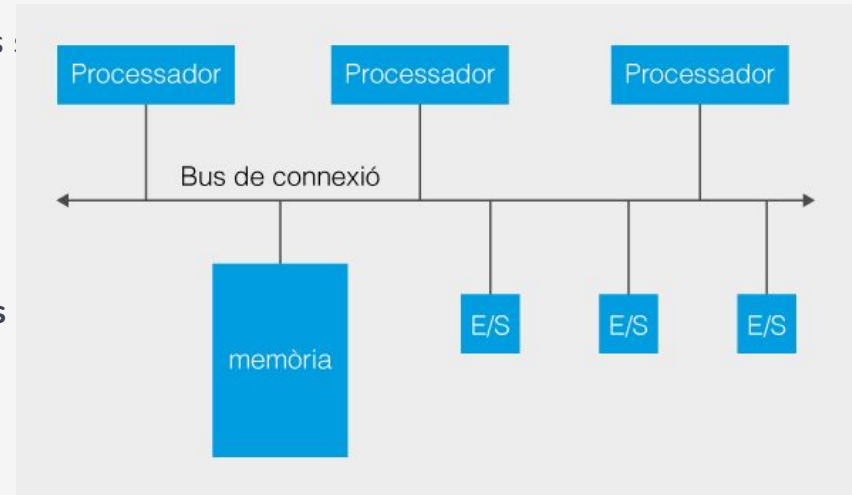
- **Debilment acoblats:**
 - Cada processador té una memòria associada.
 - A les execucions que necessiten col·laboració entre processos **cal l'intercanvi de missatges a través d'enllaços de comunicacions**, per habilitar la comunicació entre processos.
 - Un tipus d'aquests sistemes poc acoblats són els **sistemes distribuïts**, en els quals cada dispositiu monoprocessador (o multiprocessador) podria estar situat a llocs físicament distants. Una xarxa d'ordinadors o Internet podria ser un exemple d'un sistema dèbilment acoblat distribuït.



2. Programació concurrent

- Fortament acoblats

- Els processadors comparteixen una mateixa memòria i estan interconnectats a ella a través d'un bus. També poden tenir una petita memòria cau a cada processador.
- Hi ha col·laboració entre processadors compartint variables a la memòria comuna a mode de resultats parcials o també per comunicar-se entre ells.
- Actualment, la majoria de sistemes operatius són multiprocessador.
- Es poden dividir en **sistemes multiprocés simètrics**, en els quals els processadors del sistema són de característiques similars i competeixen entre iguals per executar processos, i **sistemes multiprocés asimètrics** en els quals un dels processadors del sistema, el màster, controla la resta de processadors. Aquest sistema també s'anomena master-slave.



3. Programació paral·lela

La programació paral·lela és **la programació concurrent quan es dona en un sistema multiprocessador**.

Recordem que parlem de programació concurrent quan l'execució en un dispositiu informàtic de forma simultània diferents tasques (processos). La programació concurrent es dona en un computador amb un únic processador

En els sistemes multiprocessador la concurrència és real, per tant el temps d'execució acostuma a ser menor que en dispositius monoprocessadors.



3. Programació paral·lela

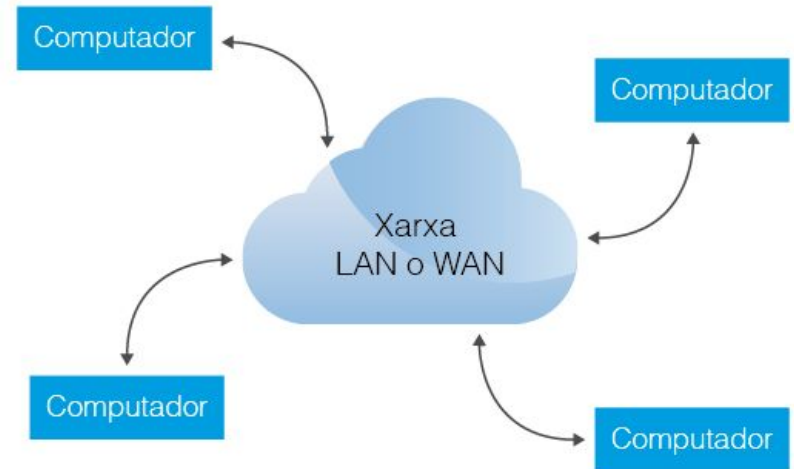
El principal **inconvenient** de la programació paral·lela són els controls que hem d'afegir per tal que la comunicació i sincronització dels processos que s'executen concurrentment siguin correctes.

Dos processos s'han de sincronitzar o comunicar quan un procés necessita alguna dada que està processant l'altre o bé un d'ells ha d'esperar la finalització de l'altre per poder continuar la seva execució.

4. Programació distribuïda

La **programació distribuïda** és un tipus de programació concurrent en la qual els processos són executats en una xarxa de processadors autònoms o en un sistema distribuït. És un sistema de computadors independents que des del punt de vista de l'usuari del sistema es veu com una sola computadora.

En els sistemes distribuïts, a diferència dels sistemes paral·lels, **no existeix memòria compartida**, per tant si necessiten la utilització de variables compartides es crearan rèpliques d'aquestes variables als diferents dispositius de la xarxa i caldrà controlar la coherència de les dades.



4. Programació distribuïda

La comunicació entre processos per intercanviar dades o sincronitzar-se entre ells es fa amb missatges que s'envien a través de la xarxa de comunicacions que comparteixen.

Avantatge:

- Es tracta de sistemes altament escalables i per tant reconfigurables fàcilment i d'alta disponibilitat.

Inconvenients:

- Són sistemes heterogenis, **complexos de sincronitzar**.
- Les **comunicacions es fan a través de missatges que utilitzen la xarxa de comunicació compartida** i això pot provocar-ne la **saturació**.

5. Advantages i desavantatges del multiprocés

Avantatges:

- **Increment de la la potència de càlcul i del rendiment.** Quan s'executen diversos processos a l'hora, la velocitat d'execució global es pot incrementar.
- **Flexibilitat.** Si augmenten el processos que s'estan executant és capaç de distribuir la càrrega de treball dels processadors, i pot també reassignar dinàmicament els recursos de memòria i els dispositius per ser més eficients.
- **Fàcil creixement.** Si el sistema ho permet, es poden afegir nous processadors de forma senzilla i així augmenten la seva potència.
- **Tolerància a fallades.** Una fallada d'un processador no fa que el sistema s'aturi.
- **Especialització.** Els sistemes multiprocés ens permeten diferenciar processos per la seva especialització i, per tant, **reservar processadors per operacions complexes**, aprofitar-ne d'altres per processaments paral·lels i per avançar l'execució.

5. **Advantages i inconvenients del multiprocés**

Inconvenients:

- **Incrementa la complexitat de gestió.** El control que s'ha de realitzar quan hi ha diversos processos en execució i han de compartir informació o comunicar-se. Això incrementa la complexitat de la programació i penalitza el temps d'execució.
- **Possible coll d'ampolla.** En un entorn multiprocessador paral·lel o distribuït, el tràfic dels busos de comunicació s'incrementa paral·lelament al nombre de processadors o computadors que incorporem al sistema, i això pot arribar a ser un crític coll d'ampolla.

6. Classificació de processos

Depenent de **com s'estan executant**, els processos es classificaran com a processos en:

- **Primer pla (foreground)**: Els processos en primer pla mantenen una comunicació amb l'usuari, ja sigui informant de les accions realitzades o esperant les seves peticions per mitjà d'alguna interfície d'usuari.
- **Segon pla (background)**: No es mostren explícitament a l'usuari, bé perquè no els cal la seva intervenció o bé perquè les dades requerides no s'incorporen a través d'una interfície d'usuari.
 - Els processos en background també són anomenats processos **dimoni**. Aquesta nomenclatura prové del nom **daemon** que prové de les sigles angleses DAEMON (Disk And Execution Monitor).

6. Classificació de processos - Servei

Un **servei** és un tipus de procés que no té interfície d'usuari.

- Poden ser, depenent de la seva configuració, inicialitzats pel sistema de forma automàtica, en el qual van realitzant les seves funcions sense que l'usuari se n'assabenti o bé es poden mantenir a l'espera que algú els faci una petició per fer una tasca en concret.

Per exemple, servidor MySQL, antivirus, firewall, connexió wi-fi...

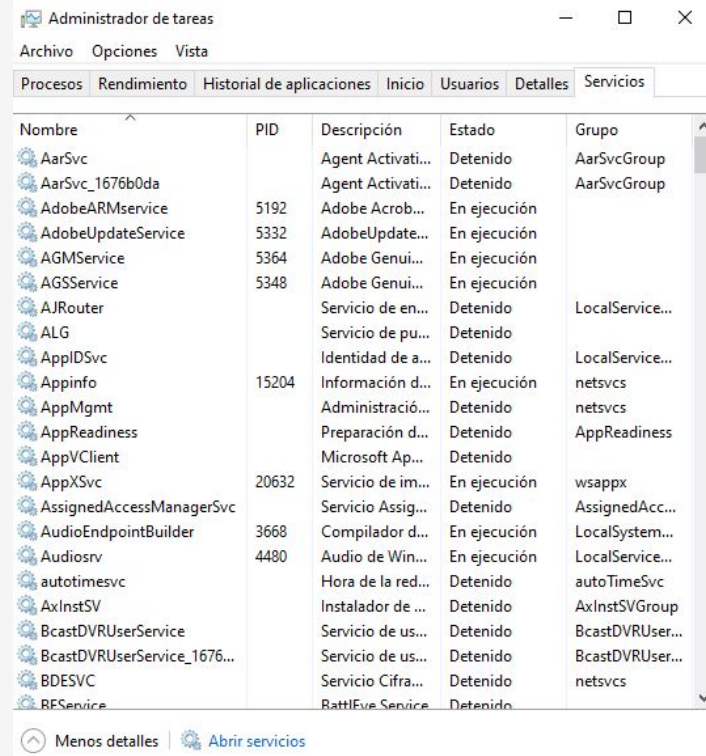
També hi ha crides a serveis distribuïts. Per exemple, serveis que ens ofereixen servidors que són a Internet: HTTP, DNS, FTP.

6. Classificació de processos - Servei

Els serveis són processos executats en segon pla.

“Servei” és una nomenclatura utilitzada en Windows. En sistemes Linux s'anomenen també processos daemon.

Com que els serveis no disposen d'interfície d'usuari directa, emmagatzemen la informació generada o les possibles errades que vagin produint, en fitxers de registre habitualment coneguts com a logs.



Administrador de tareas

Archivo Opciones Vista

Procesos Rendimiento Historial de aplicaciones Inicio Usuarios Detalles **Servicios**

Nombre	PID	Descripción	Estado	Grupo
AarSvc		Agent Activati...	Detenido	AarSvcGroup
AarSvc_1676b0da		Agent Activati...	Detenido	AarSvcGroup
AdobeARMservice	5192	Adobe Acrob...	En ejecución	
AdobeUpdateService	5332	Adobe Update...	En ejecución	
AGMSvc	5364	Adobe Genui...	En ejecución	
AGSSvc	5348	Adobe Genui...	En ejecución	
AJRouter		Servicio de en...	Detenido	LocalService...
ALG		Servicio de pu...	Detenido	
AppIDSvc		Identidad de a...	Detenido	LocalService...
Appinfo	15204	Información d...	En ejecución	netsvcs
AppMgmt		Administració...	Detenido	netsvcs
AppReadiness		Preparación d...	Detenido	AppReadiness
AppVClient		Microsoft Ap...	Detenido	
AppXSvc	20632	Servicio de im...	En ejecución	wsappx
AssignedAccessManagerSvc		Servicio Assig...	Detenido	AssignedAcc...
AudioEndpointBuilder	3668	Compilador d...	En ejecución	LocalSystem...
AudioSrv	4480	Audio de Win...	En ejecución	LocalService...
autotimesvc		Hora de la red...	Detenido	autoTimeSvc
AxInstSV		Instalador de ...	Detenido	AxInstSVGroup
BcastDVRUserService		Servicio de us...	Detenido	BcastDVRUser...
BcastDVRUserService_1676...		Servicio de us...	Detenido	BcastDVRUser...
BDESVC		Servicio Cifra...	Detenido	netsvcs
BESvc		BattlEye Service	Detenido	

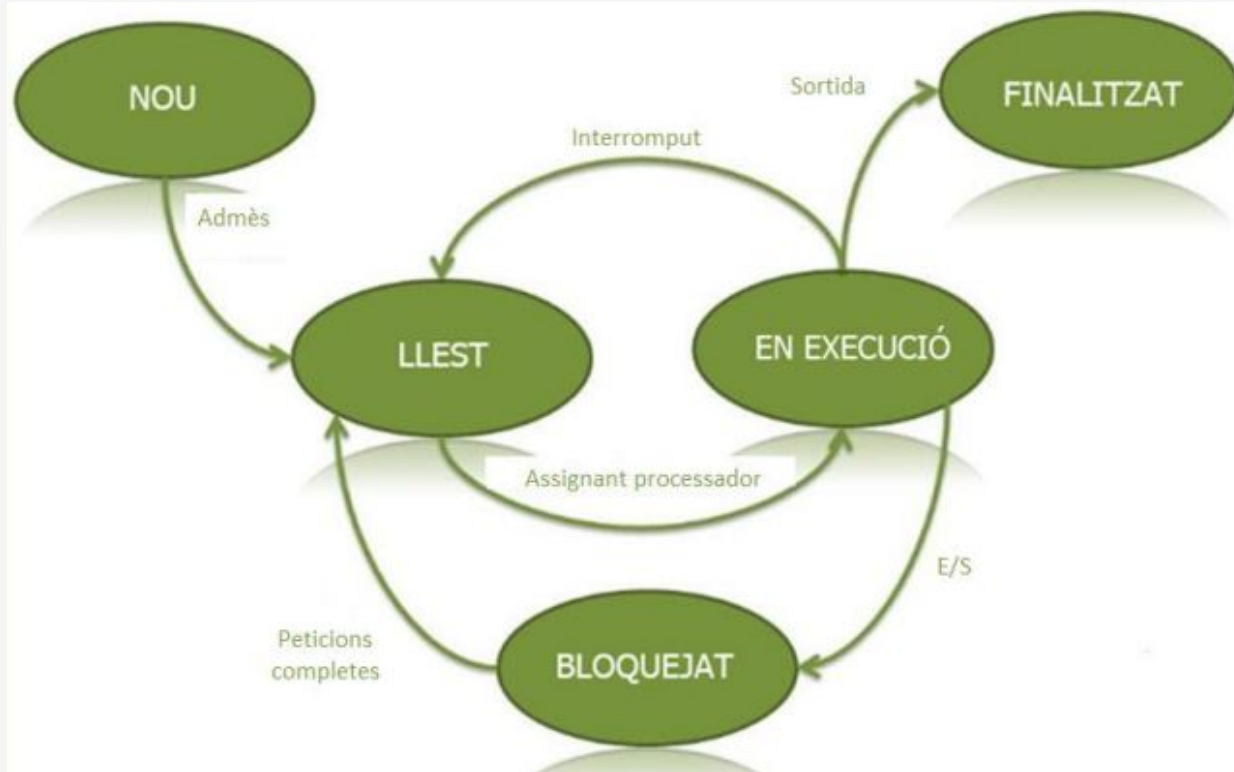
Menos detalles Abrir servicios

6. Classificació de processos - Servei

Alguns serveis poden estar a l'espera de ser cridats per realitzar les seves funcions.

Per exemple, un servidor web té un servei actiu. A Linux és un `httpd` el que està escoltant un port de la xarxa i quan l'usuari fa una petició a través de la xarxa per obtenir una plana web, és el procés dimoni el que s'encarrega de processar la petició i enviar la plana de retorn, si té accés autoritzat i la plana existeix, o bé fer arribar a través de la xarxa el missatge informatiu indicant la impossibilitat del lliurament.

7. Estats d'un procés



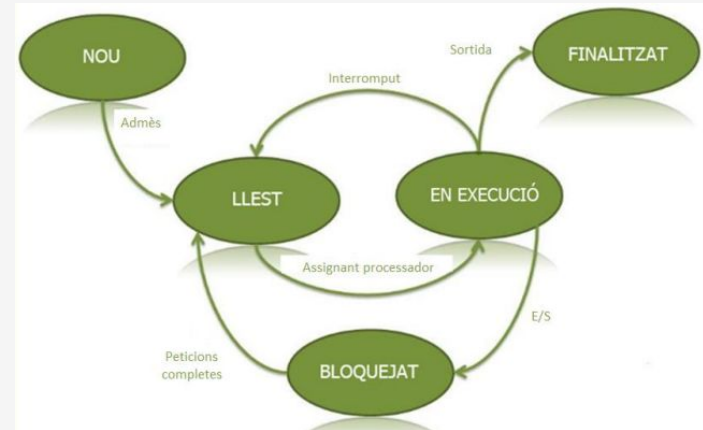
7. Estats d'un procés

Llest

- El procés està esperant ser assignat al processador per a la seva execució
- Una CPU clàssica (amb un sol nucli) només es pot dedicar a cada moment a un procés. Els processos que estan preparats per executar romanen en estat llest fins que se'ls concedeix la CPU.
- Llavors passen a l'estat "En execució".

En execució

- El procés té la CPU i aquesta executa les seves instruccions



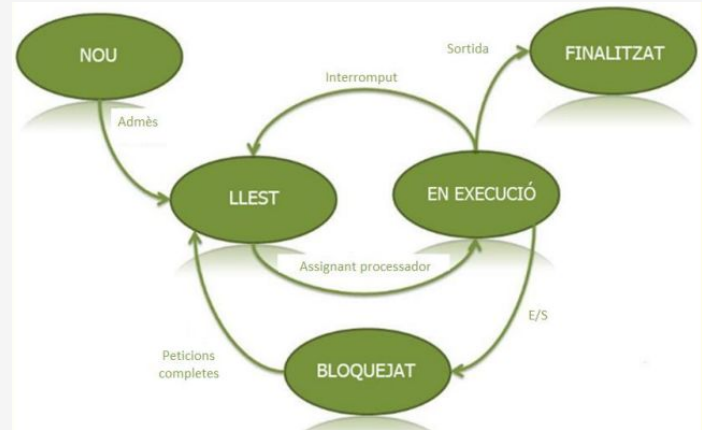
7. Estats d'un procés

Bloquejat

El procés està esperant que passi algun succés, com per exemple que acabi una operació E/S.

Finalitzat

El procés ha estat tret del grup de processos executables pel sistema operatiu alliberant els recursos utilitzats per aquest procés, per exemple, la memòria.



8. Fils

- Dins de cada procés poden executar-se diversos fils. D'aquesta manera els blocs d'instruccions que presentin certa independència puguin executar-se a la vegada.
- Un fil és la unitat de processament més petita que pot ser planificada per un sistema operatiu.
- Un fil és simplement una tasca que pot ser executada al mateix temps amb una altra tasca.
- Els fils comparteixen els recursos del procés (dades, codi, memòria, etc). Per tant, si un fil modifica una variable del procés, la resta de fils podran veure la modificació quan accedeixin a la variable.

8. Fils

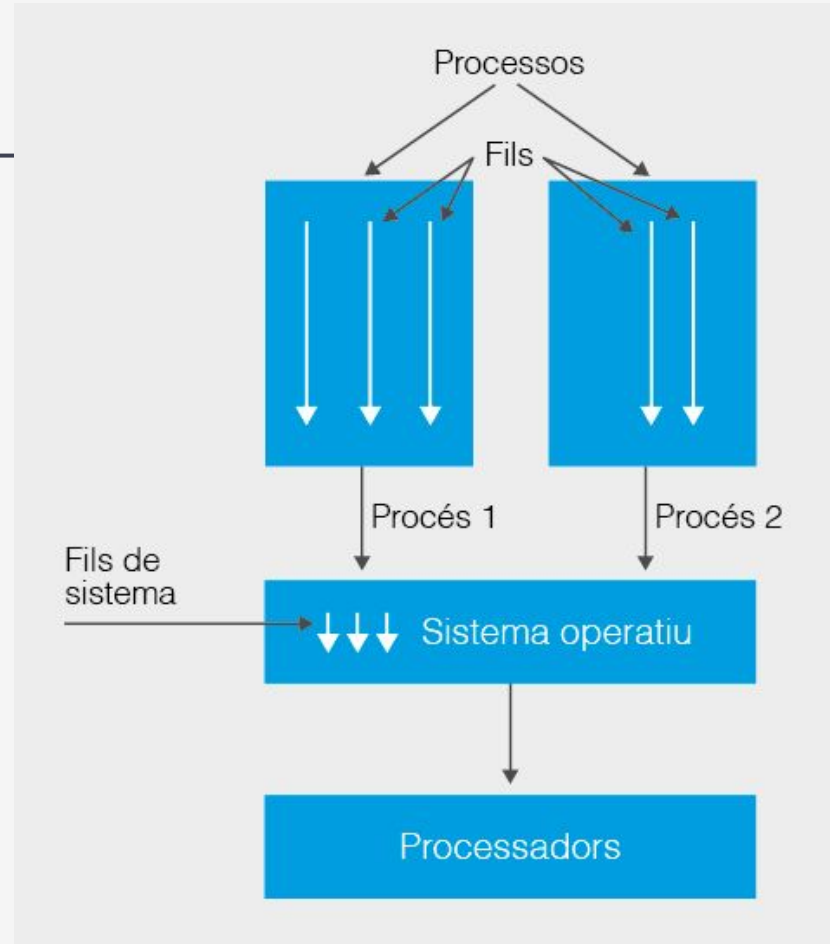
- Per exemple, un fil podria encarregar-se de la interfície gràfica (icones, botons, finestres), mentre que un altre fa una llarga operació internament. D'aquesta manera el programa respon més àgilment a la interacció amb l'usuari.
- En sistemes operatius que proveeixen facilitats pels fils, és més ràpid canviar d'un fil a un altre dins del mateix procés, que canviar d'un procés a un altre.
- És possible que els fils requereixin d'operacions atòmiques per impedir que les dades comunes siguin canviades o llegides mentre s'estiguin modificant. El descuit d'això pot generar estancament.

8. Fils

- Els **processos** són anomenats **entitats pesades** perquè estan a espais de direccionament de memòria independents, de creació i de comunicació entre processos, cosa que consumeix molts recursos de processador.
- Els **fil**s són anomenats **entitats lleugeres**. Per què ni la creació de fils ni la comunicació consumeixen gaire temps de processador.
- Un procés estarà en execució mentre algun dels seus fils estigui actiu. Tot i així, també és cert que si finalitzem un procés de forma forçada, els seus fils també acabaran l'execució.

8. Fils

- Dos processos amb diferents fils en execució. Cada fil s'executa de forma independent.
- Cada fil es processa de forma independent malgrat que pertanyi o no a un mateix procés.
- L'única diferència entre el tractament de fils i processos la trobem a nivell de memòria.



9. Comunicació entre processos

Un procés és un programa en execució i, com qualsevol programa, rep informació, la transforma i produeix resultats. Aquesta acció es gestiona a través de:

- **L'entrada estàndard** (stdin): lloc d'on el procés llegeix les dades d'entrada que requereix per a la seva execució.
 - Normalment sol ser el teclat, però podria rebre'ls d'un fitxer, de la targeta de xarxa o fins d'un altre procés, entre d'altres llocs.
 - La lectura de dades al llarg d'un programa (per exemple mitjançant `scanf` en C) llegirà les dades de la seva entrada estàndard.
- La **sortida estàndard** (stdout): lloc on el procés escriu els resultats que obté.
 - Normalment és la pantalla, tot i que podria ser, entre altres, la impressora o fins a un altre procés que necessiti aquestes dades com a entrada.
 - L'escriptura de dades que es realitzi en un programa (per exemple mitjançant `printf` en C o `System.out.println` en Java) es produeix per la sortida estàndard.

9. Comunicació entre processos

- La **sortida d'error** (stderr): lloc on el procés envia els missatges d'error.
 - Habitualment és el mateix que la sortida estàndard, però pot especificar que sigui un altre lloc, per exemple un fitxer per identificar més fàcilment els errors que tenen lloc durant l'execució.
 - La utilització de System.out i System.err en Java es pot veure com un exemple d'utilització d'aquestes sortides

En la majoria dels sistemes operatius, aquestes entrades i sortides en procés fill són una còpia de les mateixes entrades i sortides que tingués el seu pare. De tal manera que si es crida a l'operació create dins d'un procés que llegeix d'un fitxer i mostra la sortida estàndard per pantalla, el seu fill corresponent llegirà del mateix fitxer i escriurà a la pantalla.

9. Comunicació entre processos

- En Java, en canvi, el procés fill creat de la classe Process no té la seva pròpia interfície de comunicació, per la qual cosa l'usuari no pot comunicar-se amb ell directament.
- Totes les seves sortides i entrades d'informació (stdin, stdout i stderr) es redirigeixen al procés pare a través dels següents fluxos de dades o streams:
 - OutputStream: flux de sortida del procés fill. El stream està connectat per un pipe a l'entrada estàndard (Stdin) del procés fill.
 - InputStream: flux d'entrada del procés fill. El stream està connectat per un pipe a la sortida estàndard (Stdout) del procés fill.
 - ErrorStream: flux d'error del procés fill. El stream està connectat per un pipe a la sortida estàndard (stderr) del procés fill. Per defecte, per a la JVM, stderr està connectat al mateix lloc que stdout.

10. Gestió de processos

- El sistema operatiu és l'encarregat de crear els nous processos seguint les directrius de l'usuari.
- Quan un usuari vol obrir un programa, el sistema operatiu és el responsable de crear i posar en execució el procés corresponent que l'executarà.
- La posada en execució d'un nou procés es produeix a causa de que hi ha un procés en concret que està demanant la seva creació.
- Qualsevol procés en execució sempre depèn del procés que el va crear, establint un vincle entre tots dos. Al seu torn, el nou procés pot crear nous processos, formant-se el que s'anomena un arbre de processos.

10. Gestió de processos

- Per identificar els processos, els sistemes operatius solen utilitzar un identificador de procés (process identifier [PID]) unívoc per a cada procés. La utilització del PID és bàsica a l'hora de gestionar processos, ja que és la forma que té el sistema de referir-se als processos que gestiona.

PID	Process Name
0	kernel_task
1 ▼	launchd
113	mds
158	WindowServer
138	coreservicesd
48054	diskimages-helper
115	loginwindow
45649	fseventsd
301	coreaudiod
55644	cupsd
15 ▼	configd
49045	eapolclient

11. Sincronització de processos

- Els mètodes de comunicació de processos es poden considerar com a mètodes de sincronització ja que permeten al procés pare portar el ritme d'enviament i recepció de missatges.
- Concretament, els enviaments i recepcions pels fluxos de dades permeten a un procés fill comunicar-se amb el seu pare a través d'un canal de comunicació unidireccional bloquejant.
- **Espera de processos** (operació wait):
 - A més de la utilització dels fluxos de dades es pot esperar per la finalització del procés fill i obtenir la seva informació de finalització mitjançant l'operació wait.
 - L'esmentada operació bloqueja al procés pare fins que el fill finalitza la seva execució mitjançant exit. Com a resultat es rep la informació de finalització del procés fill.

11. Sincronització de processos

- **Espera de processos** (operació wait):
 - Aquest valor de retorn s'especifica mitjançant un nombre sencer. El valor de retorn indica el resultat de l'execució. No té res a veure amb els missatges que es passen entre pare i fill a través dels streams. Per convenció s'utilitza 0 per indicar que el fill ha acabat de forma correcta.
 - Mitjançant `waitFor()` de la classe `Process` el pare espera bloquejat fins que el fill finalitza la seva execució, tornant immediatament si el fill ha finalitzat amb anterioritat o si algú l'interromp (en aquest cas es llança la interrupció *InterruptedException*).
 - A més es pot utilitzar `exitValue()` per obtenir el valor de retorn que va tornar un procés fill. El procés fill ha d'haver finalitzat, si no, es llança l'excepció *IllegalThreadStateException*.

12. Programació d'aplicacions multiprocés

- La programació concurrent és una forma eficaç de processar la informació al permetre que diferents successos o processos es vagin alternant en la CPU per proporcionar multiprogramació.
- La multiprogramació pot produir-se entre processos totalment independents, com podrien ser els corresponents al processador de textos, navegador, reproductor de música, etc., o entre processos que poden cooperar entre si per realitzar una tasca comuna.
- El sistema operatiu s'encarrega de proporcionar multiprogramació entre tots els processos del sistema, ocultant aquesta complexitat tant als usuaris com als desenvolupadors. No obstant això, si es pretén realitzar processos que cooperin entre si, ha de ser el propi desenvolupador qui ho implementi utilitzant la comunicació i sincronització de processos vistes fins ara.

12. Programació d'aplicacions multiprocés

- A l'hora de realitzar un programa multiprocés cooperatiu, s'han de seguir les següents fases:
 - **Descomposició funcional:** Cal identificar prèviament les diferents funcions que ha de realitzar la aplicació i les relacions existents entre elles.
 - **Partició:** Distribució de les diferents funcions en processos establint l'esquema de comunicació entre els mateixos. En ser processos cooperatius necessitaran informació els uns dels altres pel que han de comunicar-se.
 - Cal tenir en compte que la comunicació entre processos requereix una pèrdua de temps tant de comunicació com de sincronització. Per tant, l'objectiu ha de ser maximitzar la independència entre els processos minimitzant la comunicació entre ells.
 - **Implementació:** Es realitza un cop s'ha duit a terme la descomposició i la partició utilitzant les eines disponibles per la plataforma a utilitzar. En aquest cas, Java permet únicament mètodes senzills de comunicació i sincronització de processos per a realitzar la cooperació