T1S1. Programació multiprocés

1. Conceptes bàsics.
2. Programació concurrent.
3. Funcionament bàsic del sistema operatiu.
4. Processos.
5. Gestió de processos..

1. Conceptes bàsics

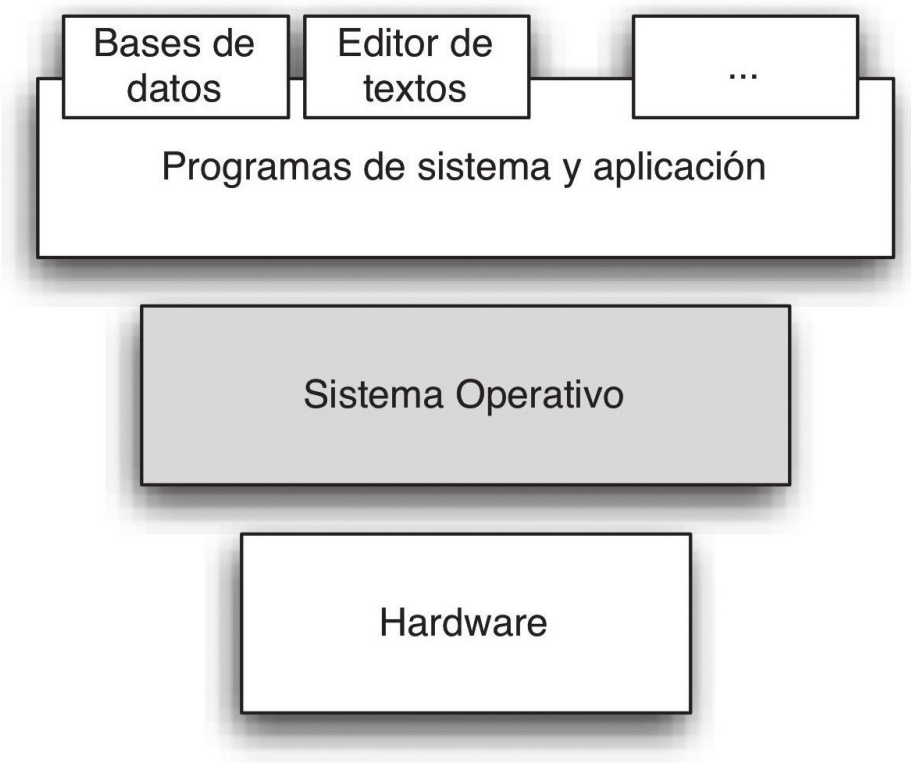
Per a poder començar a entendre com s'executen diversos programes alhora, és imprescindible adquirir uns certs conceptes.

- **Programa**: es pot considerar un programa a tota la informació (tant codi com dades) emmagatzemada en disc d'una aplicació que resol una necessitat concreta per als usuaris.
- **Procés**: quan un programa s'executa, podrem dir de manera molt simplificada que és un procés. En definitiva, pot definir-se procés com un programa en execució. Aquest concepte no es refereix únicament al codi i a les dades, sinó que inclou tot el necessari per a la seua execució. Això inclou tres coses:
    - Un comptador del programa: que indica per quina instrucció s'està executant.
    - Una imatge de memòria: és l'espai de memòria que el procés està utilitzant.
    - Estat del processador: es defineix com el valor dels registres del processador sobre els quals s'està executant.

On s'emmagatzema la informació mentre el programa s'està executant en la CPU? A la memòria principal o RAM.

És important destacar que els processos **són entitats independents, encara que executen el mateix programa.** De tal forma, poden coexistir dos processos que executen el mateix programa, però amb diferents dades (és a dir, amb diferents imatges de memòria) i en diferents moments de la seua execució (amb diferents comptadors de programa).

- **Executable**: un fitxer executable conté la informació necessària per a crear un procés a partir de les dades emmagatzemades d'un programa. És a dir, direm “executable” al fitxer que permet posar el programa en execució com a procés.
- **Sistema operatiu**: programa que fa d'intermediari entre l'usuari i les aplicacions que utilitza i el maquinari de l'ordinador..



- **Dimoni (daemon en Linux) o servei (Windows)**: procés no interactiu (no disposen d'interfície d'usuari) que està executant-se contínuament en segon pla, és a dir, és un procés controlat pel sistema sense cap intermediació de l'usuari. Solen proporcionar un servei bàsic per a la resta de processos.

2. Programació Concurrent.

La **computació concurrent** permet la possibilitat de tindre en execució al mateix temps múltiples tasques interactives. És a dir, permet realitzar diverses coses al mateix temps, com escoltar música, visualitzar la pantalla de l'ordinador, imprimir documents, etc. Penseu en tot el temps que perdríem si totes aqueixes tasques s'hagueren de realitzar l'una després de l'altra. Aquestes tasques es poden executar en:

- Un **únic processador (multiprogramació)**. En aquest cas, encara que per a l'usuari parega que diversos processos s'executen al mateix temps, si solament existeix un únic processador, solament un procés pot estar en un moment determinat en execució. Per a poder anar canviant entre els diferents processos, el sistema operatiu s'encarrega de canviar el procés en execució després d'un període curt de temps (de l'ordre de mil·lisegons). Això permet que en un segon s'executen múltiples processos, creant en l'usuari la percepció que múltiples programes s'estan executant al mateix temps .
  Aquest concepte es denomina **multiprogramació**. La multiprogramació no millora el temps d'execució global dels programes ja que s'executen intercanviant els uns pels altres en el processador, al contrari. No obstant això, permet que parega que diversos programes s'executen al mateix temps.

- Un **sistema multiprocessador (programació paral·lela):** existeixen dos o més processadors, 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. El sistema operatiu, igual que per a un únic processador, s'ha d'encarregar de planificar els treballs que s'executen en cada processador i canviar els uns pels altres. En aquest cas tots els processadors comparteixen la mateixa memòria per la qual cosa és possible utilitzar-los de manera coordinada mitjançant el que es coneix per **programació paral·lela**.
  La programació paral·lela permet millorar el rendiment d'un programa si aquest s'executa de manera paral·lela en diferents processadors ja que permet que s'executen diverses instruccions alhora. Cada execució en cada procesador serà un procés o una tasca del mateix programa podent cooperar entre si.

- **Diversos ordinadors distribuïts en xarxa**. Cadascun dels ordinadors tindrà els seus propis processadors i la seua pròpia memòria. La gestió dels mateixos forma part del que es denomina **programació distribuïda.**
  La programació distribuïda possibilita la utilització d'un gran nombre de dispositius (ordinadors) de manera paral·lela, la qual cosa permet aconseguir elevades millores en el rendiment de l'execució de programes distribuïts.

No obstant això, com cada ordinador posseeix la seua pròpia memòria, impossibilita que els processos puguen comunicar-se compartint memòria, havent d'utilitzar altres esquemes de comunicació més complexos i costosos a través de la xarxa que els interconnecte.

3. Funcionament bàsic del sistema operatiu.

La part central que realitza la funcionalitat bàsica del sistema operatiu es denomina **kernel.** És una part xicoteta del programari del sistema operatiu, si la comparem amb el necessari per a implementar la seua interfície (i més hui dia, que és molt visual). A tota la resta del sistema se'l denomina programes del sistema. El kernel és el responsable de gestionar els recursos de l'ordinador, permetent el seu ús a través de crides al sistema.

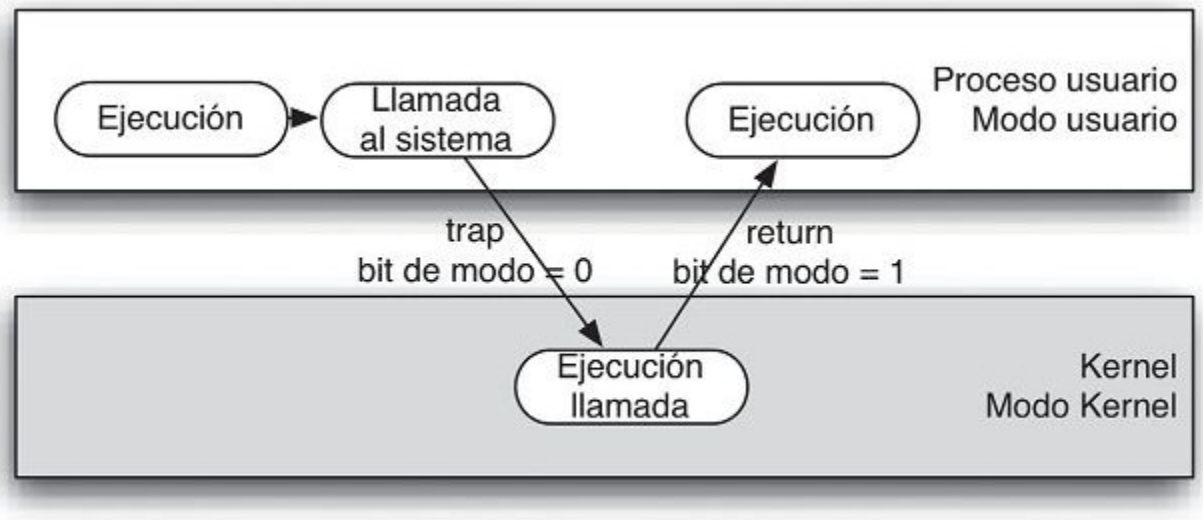
En general, el **kernel del sistema funciona sobre la base d'interrupcions.** Una interrupció és una suspensió **temporal de l'execució d'un procés**, per a passar a executar una rutina que tracte aquesta interrupció. Aquesta rutina serà dependent del sistema operatiu. És important destacar que mentre s'està atenent una interrupció, es deshabilita l'arribada de noves interrupcions. Quan finalitza la rutina, es reprén l'execució del procés en el mateix lloc on es va quedar quan va ser interromput.

És a dir, el sistema operatiu no és un procés dimoni pròpiament dit que proporcione funcionalitat a la resta de processos, sinó que ell només s'executa responent a interrupcions. Quan salta una interrupció es transfereix el control a la rutina de tractament de la interrupció. Així, les rutines de tractament d'interrupció poden ser vistes com el codi pròpiament dit del kernel.

Les **crides al sistema** són la interfície que proporciona el kernel perquè els programes d'usuari puguen fer ús de manera segura de determinades parts del sistema.

Els errors d'un programa podrien afectar altres programes o al propi sistema operatiu, per la qual cosa per a assegurar la seua execució de la forma correcta, el sistema implementa una interfície de crides per a evitar que unes certes instruccions perilloses siguen executades directament per programes d'usuari.
El **mode dual** és una característica del maquinari que permet al sistema operatiu protegir-se. El processador té dues maneres de funcionament indicats mitjançant un bit:

- Mode usuari (1). Utilitzat per a la execució de programes d'usuari..
- Mode kernel(0), també anomenat “mode supervisor” o “mode privilegiat”. Les instruccions del processador més delicades solo es poden executar si el processador està en mode kernel.


Ejecución
Llamada al sistema
Ejecución
Proceso usuario
Modo usuario
trap
bit de modo = 0
return
bit de modo = 1
Ejecución llamada
Kernel
Modo Kernel

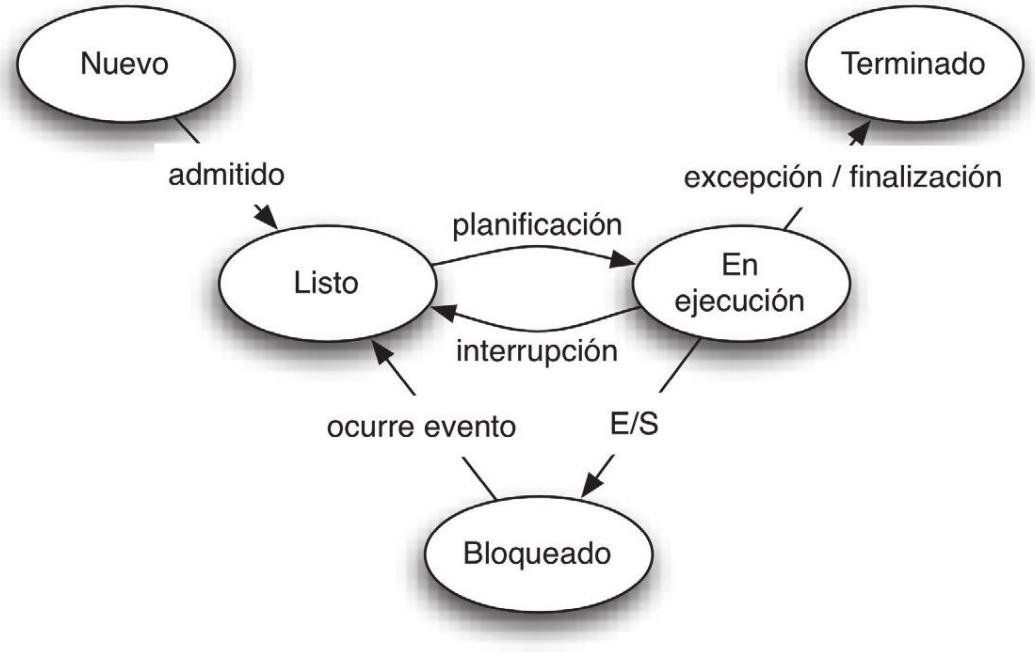
4. Processos.

El sistema operatiu és l'encarregat de posar en execució i gestionar els processos. Per al seu correcte funcionament, al llarg del seu cicle de vida, els processos poden canviar d'estat. És a dir, a mesura que s'executa un procés, aquest procés passarà per diversos estats. El canvi d'estat també es produirà per la intervenció del sistema operatiu.

4.1 Estats d'un procés.

Els estats d'un procés són:

- **Nou.** El procés està sent creat a partir del fitxer executable.
- **Llest**: el procés no es troba en execució encara que està preparat per a fer-ho. El sistema operatiu no li ha assignat encara un processador per a executar-se. El planificador del sistema operatiu és el responsable de seleccionar que procés està en execució, per la qual cosa és el que indica quan el procés passa a execució..
- **En execució:** el procés s'està executant. El sistema operatiu utilitza el mecanisme d'interrupcions per a controlar la seua execució. Si el procés necessita un recurs, incloent la realització d'operacions d'entrada eixida (E/S), cridarà a la crida del sistema corresponent. Si un procés en execució s'executa durant el temps màxim permés per la política del sistema, salta un temporitzador que llança una interrupció. En aquest últim cas, si el sistema és de temps compartit, el per a i ho passa a l'estat Llest, seleccionant un altre procés perquè continue la seua execució.
- **Bloquejat:** el procés està bloquejat esperant que ocórrega algun succés (esperant per una operació de E/S, bloquejat per a sincronitzar-se amb altres processos, etc.). Quan ocorre l'esdeveniment que el desbloqueja, el procés no passa directament a execució sinó que ha de ser planificat de nou pel sistema.
- **Acabat:** el procés ha finalitzat la seua execució i allibera la seua imatge de memòria. Per a acabar un procés, el mateix ha de cridar al sistema per a indicar-li'l o pot ser el propi sistema el que finalitze el procés mitjançant una excepció (una interrupció especial).



4.2 Cues de processos.

Un dels objectius del sistema operatiu és la multiprogramació, és a dir, admetre diversos processos en memòria per a maximitzar l'ús del processador. Això funciona ja que els processos s'aniran intercanviant l'ús del processador per a la seua execució de manera concurrent. Per a això, el sistema operatiu organitza els processos en diverses cues, migrant-los d'unes cues a unes altres:

- Una **cua de processos** que conté tots els processos del sistema.
- Una **cua de processos preparats** que contenen tots els processos llestos esperant per a executar-se.
- **Diverses cues de dispositiu** que conté els processos que estan a l'espera d'alguna operació de E/S..

4.3 Planificació de processos.

Per a gestionar les cues de processos, és necessari un planificador de processos. El planificador és l'encarregat de seleccionar els moviments de processos entre les diferents cues. Existeixen dos tipus de planificació:

- **A curt termini** : selecciona quin procés de la cua de processos preparats passarà a execució. S'invoca molt sovint.
    - Planificació sense desallotjament. Únicament es canvia el procés en execució si aquest procés es bloqueja o acaba.
    - Planificació *apropiativa. S'executa el procés amb major prioritat.
    - Temps compartit: cada cert temps (anomenat quant) es desallotja el procés que estava en execució i se selecciona un altre procés per a executar-se. En aquest cas, totes les prioritats dels fils es consideren iguals.
- **A llarg termini:** selecciona quins processos nous han de passar a la cua de processos reparats. S'invoca amb poca freqüència, per la qual cosa pot prendre's més temps a prendre la decisió. Controla el grau de multiprogramació (nombre de processos en memòria)

4.4 Canvis de context.

Quan el processador passa a executar un altre procés, la qual cosa ocorre molt sovint, el sistema operatiu ha de guardar el context del procés actual i restaurar el context del procés que el planificador a curt termini ha triat executar. La salvaguarda de la informació del procés en execució es produeix quan hi ha una interrupció.

Es coneix com a context a:

- Estat del procés.
- Estat del processador: valors dels diferents registres del processador.
- Informació de gestió de memòria: espai de memòria reservada per al procés.

El canvi de context és temps perdut, ja que el processador no fa treball útil durant aqueix temps. Únicament és temps necessari per a permetre la multiprogramació i la seua duració depén de l'arquitectura en concret del processador

5. Gestió de processos..

5.1 Arbre de processos

El sistema operatiu és l'encarregat de crear i gestionar els nous processos seguint les directrius de l'usuari. Així, 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. Encara que el responsable del procés de creació és el sistema operatiu, ja que és l'únic que pot accedir als recursos de l'ordinador, el nou procés es crea sempre per petició d'un altre procés. La posada en execució d'un nou procés es produeix pel fet que hi ha un procés en concret que està demanant la seua creació en el seu nom o en nom de l'usuari.

En aquest sentit, qualsevol procés en execució sempre depén del procés que el va crear, establint-se un vincle entre tots dos. Al seu torn, el nou procés pot crear nous processos, formant-se el que es denomina un arbre de processos. Quan s'arranca l'ordinador, i es carrega en memòria el *kernel del sistema a partir de la seua imatge en disc, es crea el procés inicial del sistema. A partir d'aquest procés, es crea la resta de processos de manera jeràrquica, establint pares, fills, avis, etc.

Per a identificar als 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.

En Windows Utilitza l'Administrador de tasques per a obtindre els processos del sistema.

5.2. Operacions bàsiques amb processos.

Seguint el vincle entre processos establit en l'arbre de processos, el procés creador es denomina pare i el procés creat es denomina fill. Al seu torn, els fills poden crear nous fills. A la operació de creació d'un nou procés la denominarem create.

Quan es crea un nou procés hem de saber que pare i fill s'executen concurrentment. Tots dos processos comparteixen la CPU i s'aniran intercanviant seguint la política de planificació del sistema operatiu per a proporcionar multiprogramació. Si el procés pare necessita esperar fins que el fill acabe la seua execució per a poder continuar la seua amb els resultats obtinguts pel fill, pot fer-lo mitjançant l'operació wait.

Com s'ha vist a l'inici del capítol, els processos són independents i tenen el seu propi espai de

memòria assignat, anomenat **imatge de memòria.** Pares i fills són processos i, encara que tinguen un vincle especial, mantenen aquesta restricció. Tots dos usen espais de memòria independents. En general , sembla que el fill executa un programa diferent al pare,
però en alguns sistemes operatius això no té perquè ser així. Per exemple , mentre que en sistemes tipus Windows existeix una funció createProcess() que crea un nou procés a partir de un programa diferent al que està en execució, en sistemes tipus UNIX, l'operació a utilitzar és fork(), que crea un procés fill amb un duplicat de l'espai de direccions del pare, és a dir, un duplicat del programa que s'executa des de la mateixa posició. No obstant això , en tots dos casos, els pares i fills (encara que siguen un duplicat en el moment de la creació en sistemes tipus UNIX) són independents i les modificacions que un faça en el seu espai de memòria, com a escriptura de variables, no afectaran l'altre..

Com a pare i fill tenen espais de memòria independents, poden compartir recursos per a intercanviar-se informació. Aquests recursos poden anar des de fitxers oberts fins a zones de memòria compartida. La **memòria compartida** és una regió de memòria a la qual poden accedir diversos processos cooperatius per a compartir informació. Els processos es comuniquen escrivint i llegint dades en aquesta regió. El sistema operatiu solament intervé a l'hora de crear i establir els permisos de quins processos poden accedir a aquesta zona. Els processos són els responsables del format de les dades compartides i de la seua ubicació

En acabar l'execució d'un procés, és necessari avisar al sistema operatiu de la seua terminació perquè d'aquesta manera el sistema allibere si és possible els recursos que tinga assignats. En general, és el propi procés el que li indica al sistema operatiu mitjançant una operació denominada exit que vol acabar, podent aprofitar per a manar informació respecte a la seua finalització al procés pare en aqueix moment.

El procés fill depén tant del sistema operatiu com del procés pare que el va crear. Així, el pare pot acabar l'execució d'un procés fill quan crega convenient. Entre aquests motius podria donar-se que el fill excedira l'ús d'alguns recursos o que la funcionalitat assignada al fill ja no siga necessària per algun motiu.

Per a això pot utilitzar l'operació destroy. Aquesta relació de dependència entre pare i fill, porta a casos com que si el pare acaba, en alguns sistemes operatius no es permeta que els seus fills continuen l'execució, produint-se el que es denomina “terminació en cascada”.