2. CONCEPTE SPECIFICE SO

2.1 INTRERUPERE

Un proces poate fi intrerupt in timpul executiei din diverse cauze.

Intreruperea este un mecanism prin care UC ia cunostinta de ***aparitia unui eveniment*.**

O intrerupere se produce:
1. la terminarea unei operatii de intrare/iesire;
2. la epuizarea intervalului de timp UC acordat unui proces;
3. la tentativa de executie a unei instr. inexistente sau interzise;
4. la impartirea la zero sau la depasire;
5. la tentiva de accesare a unei zone de memorie interzise procesului sau in cazul unei adrese invalide de memorie;
6. la executia instructiunii de lansare a unui apel sistem.

Clasificarea evenimentelor care genereaza o intrerupere:

Eveniment	Cauza	Provenienta
Intrerupere externa	Exterioara executiei procesului urent	Aparitia unui eveniment extern asincron
Exceptie (trap, deruta)	Legata de executia procesului curent	Aparitia unei erori sau a unei situatii exceptionale in derularea procesului
Apel sistem	idem (generare explicita)	Cererea unui serviciu de la sistemul de operare

Observatie: termenul generic de INTRERUPERE este adesea folosit pentru aceste trei tipuri de evenimente.

Intreruperile se clasifica in:
1. *intreruperi hardware*, generate din cauze externe executiei proceselor;
2. *intreruperi software*, generate din cauze interne executiei proceselor.

Intreruperile hardware sunt *asincrone* fata de executia proceselor (se produc la momente de timp nepredictibile) iar cele software sunt *sincrone* (se produc intotdeuna la executia unei anumite instructiuni apartinind codului proceselor).

Intreruperile software pot sa fie generate de UMM (intreruperea de violare de zona de memorie) sau chiar de UC (in cazul unei instructiuni interzise sau inexistente, sau a instructiunii de lansare a unui apel sistem- *sys-call*).
Observatie: UMM face parte din microprocesor sau este un circuit separat. Rolul UMM este urmatorul: in cazul oricarui acces la memorie, este indicata o adresa, ce este verificata de UMM si transformata in dupa un set de reguli care este sub controlul UC. Rezultatul este o adresa fizica valida sau o eroare (exceptie).

Exemplu : Modul tipic de desfasurare al unei intreruperi hardware este urmatorul:
1. dupa executia fiecarei instructiuni UC testeaza un indicator special de prezenta intrerupere;
2. prin mecanisme hardware (pornind dela numarul, sau codul intreruperii) este regasita si apoi plasata pe magistrala de adrese, adresa rutinei de tratare intrerupere;
3. prin mecanisme hardware (nu programat!) sunt salvate in stiva PC si PSW ale procesului curent si adresa rutinei de tratare intrerupere este incarcata in PC.;
4. se executa in continuare codul acestei rutine, care are la inceput o secventa de salvare in TP a informatiilor referitoare la procesul intrerupt.

Pentru fiecare tip de intrerupere:
- exista o rutina de tratare a intreruperii (*interrupt handler*);
- adresele rutinelor de tratare a intreruperii sunt mentinute intr-un *vector de intreruperi*, MI, incepind de la adresa fizica 0;
- vectorul de intreruperi face parte din nucleul SO.

La aparitia unei intreruperi, indiferent de tipul ei, are loc *comutarea de context a UC*, adica salvarea informatiilor de stare a procesului curent (in executie) si incarcarea informatiilor necesare executiei unui nou proces, rutina de tratare a intreruperii. Contextul UC se refera la:
- continutul RG; continutul PSW (starea de executie: in asteptare/activ; modul de lucru: master/slave; masti pentru intreruperi);
- continutul PC; informatii despre zonele de mem. accesibile si drepturile asociate.

Comutarea de context este o *operatie indivizibila*, neintreruptibila, executata prin mecanisme hardware si software.

Exista doua scheme de baza pentru tratarea intreruperilor:

1. ***un indicator unic*** este utilizat toate intreruperile;
   - o rutina generala de tratare intrerupere este activata automat la comutarea de context;
   - o codul intreruperii, este furnizat prin mecanisme hard in PSW sau intr-o locatie bine stabilita din memorie;
   - in functie de valoarea acestui cod, rutina de tratare apeleaza procedura de tratare intrerupere corespunzatoare;;

2. o valoare distincta, ***nivelul intreruperii***, este livrata in functie de tipul intreruperii:
   - pentru fiecare nivel exista o rutina diferita de tratare intrerupere, activata automat la comutarea de context in cazul aparitiei intreruperii respective;
   - exista o *ordine de priorit. prestabilita* a nivelelor
   - pe parcursul executiei rutinei de tratate a unei intreruperi de nivel superior este posibil ca:
     - sa intirzie livrarea intreruperilor de nivel inferior (*mascare* sau *inhibare* a nivelelor inferioare);
     - sa se suprime livrarea celor de nivel inferior (*dezarmarea* nivelelor inferioare).

La aparitia unei intreruperi, UC este trecuta automat in modul de lucru *master*, ceea ce permite executarea tuturor instructiunilor nepermise in modul *slave* (instructiuni de I/E, de schimbare a modului de lucru, de control al intreruperilor).

2.2 PROCES

Procesul este un program in executie. Programul este un obiect static, imuabil, procesul este un obiect dinamic, a carui stare se schimba in permanenta. La rulari diferite acelasi program poate sa parcurga faze diferite, in functie de evenimentele externe care se pot apare, sau in functie de interactiunea cu utilizatorul.

Observatie: un program poate fi comparat cu un drum trasat pe o harta, procesul este o "excursie" reala pe acel traseu.

La un moment dat un proces este caracterizat printr-o stare. Starea procesului arata in ce faza de executie a programului s-a ajuns in acel moment. ***Starea unui proces*** la un moment dat este descrisa de urmatoarele elemente:
1. ***codul programului*** care este executat;
2. valorile tuturor ***variabilelor*** folosite in program;
3. ***istoricul apelurilor de functii*** (ce functii au inceput sa se execute si nu s-au incheiat inca);
4. valoarea ***contorului de instructiuni*** si a ***registrului de stare program***;
5. informatiile despre ***resursele folosite*** in acel moment.

Aceste informatii sunt retinute astfel:

Informatia	**Locul unde este stocata**
Codul programului	in memorie, in segmentul de cod
Variabilele	in memorie, in segmentul de date
Functiile in curs de executie	in memorie, in segmentul de stiva
Contorul de instructiuni si registrul de stare program	in registrii PC si PSW ai unitatii centrale
Informatii despre resurse	in memorie, in structurile nucleului SO

Informatiile din tabela sunt necesare si suficiente pentru descrierea starii unui proces.
Practic pentru a transforma un program intr-un proces, SO trebuie sa aloce cite o zona de memorie pentru fiecare din aceste elemente sa plaseze in PC adresa primei instructiuni din codul programului si in PSW informatiile privind modul de executie al programului.

Avind in vedere ca din toate aceste elemente, singurul care ramine neschimbat este codul programului, in cazul in care este necesar sa fie executat simultan de mai multe ori acelasi program, acelasi cod poate fi partajat de mai multe procese (executat cu contoare de instructiuni diferite). Practic se face o multiplicare a unui proces in memorie, fara multiplicarea codului sau.

Folosirea in comun a codului unui program de catre mai multe procese se numeste **reentranta**.

Modulele nucleului sunt reentrante la majoritatea SO multiprogramate (***time-sharing***).

2.2.1 GESTIUNEA PROCESELOR

Primele sisteme de calcul permiteau executia unui singur program al un moment dat; programul avea controlul complet al sistemului si acces la toate resursele echipamentului. Calculatoarele moderne permit incarcarea in memorie a mai multor programe simultan, in scopul executiei lor concurente. Aceasta evolutie a necesitat un control mai puternic, o mai mare separatie intre diversele programe si a dus in cele din urma la aparitia conceptului de proces.
Procesul este o abstractizare a notiunii de program.

Se poate spune despre sistemul de operare ca este un ansamblu de procese: cele ale utilizatorilor, care executa codul programelor utilizator si cele proprii, care executa codul din modulele (programele) apartinid sistemului de operare. Toate aceste procese se executa in concurenta, multiplexind unitatea centrala intre ele. Comutarea unitatii centrale se face prin mecanisme implementate prin sistemul de operare. La un moment dat unitatea centrala executa instructiuni apartinind unui singur program; dupa un interval foarte scurt de timp trece la urmatorul program s.a.m.d. Astfel, pe durata unei secunde, UC executa portiuni din mai multe programe, dind senzatia pentru de paralelism in executie.

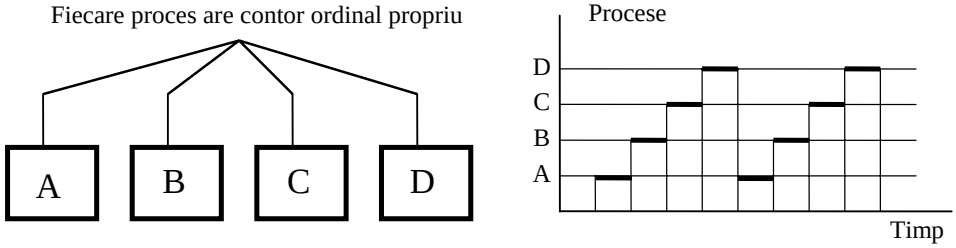
Comutare repetata a unitatii centrale intre procese este un **psedoparalelism** in executia programelor. El nu trebuie confundat cu **paralelismul real** in executie, care se produce cind echipamentul de calcul dispune de mai multe procesoare care se executa simultan procese diferite. De asemenea nu trebuie confundat cu **paralelismul in lucru**, care semnifica functionarea simultana a unitatii centrale si dispozitivelor de intrare/iesire. Creatorii sistemelor de operare moderne au incercat de-a lungul anilor sa imbunatateasca modelul pseudoparalelismului in executia programelor si au reusit sa faca aceasta activitate in acelasi timp invizibila pentru utilizatori si extrem de eficienta.

2.2.2 MODELUL DE PROCES

In cadrul acestui model sistemul de operare este un ansamblu de procese secventiale. Un proces este mai mult decit codul unui program (cod numit uneori **sectiune de text** a procesului); el inglobeaza si informatia cuprinsa intr-un registru special al unitatii centrale, numit **contor de instructiuni** (sau **contor ordinal**). Contorul ordinal retine adresa instructiunii urmatoare care se va executa, deci practic tine evidenta desfasurarii procesului. Se considera ca notiunea de proces include si informatia din registrele generale ale unitatii centrale (inclusiv pointerul de stiva) precum si datele si variabilele programului; acestea din urma formeaza **sectiunea de date** a procesului.

Un program nu este deci identic cu procesul care se creeza prin lansarea sa in executie; programul este o **entitate pasiva** (ca si continutul oricarui fisier de pe disc de exemplu) pe cind procesul este o **entitate activa**, cu un contor ordinal care specifica instructiunea urmatoare care se va executa si cu un ansamblu de resurse asociate.

Procesele se desfasoara ca si cum fiecare ar avea propria sa unitate centrala virtuala, dar asa cum s-a aratat anterior, exista o singura unitate centrala care este comutata pe rind intre procese, la intervale scurte de timp. Important este ca desfasurarea procesului, in sensul atingerii unui anumit scop dorit, nu este afectata de executia pseudoparalela. Fenomenul de comutare rapida se numeste (asa cum s-a spus in capitolul anterior) multiprogramarea unitatii centrale.



Cele patru programe A,B,C,D devin in momentul executiei patru procese paralele si independente A,B,C,D, fiecare avind un flux propriu de desfasurare. Se observa pe graficul din figura 2.1 ca dupa un anumit timp toate procesele au progresat, dar ca la un moment dat exista un singur proces activ.

Trebuie subliniat ca intr-un mediu multiprogramat progresul proceselor nu este uniform, tocmai datorita fenomenului de comutare al unit. centrale; de asemenea nu se poate aprecia cu precizie viteza de executie a unui proces. Cind un proces are nevoie sa starteze o operatie la un moment precis, sau trebuie sa astepte durate de timp bine stabilite, el apeleaza la mecanisme speciale puse la dispozitie de sistemul de operare. In nici un caz prelucrarile legate de o masurare precisa a timpului nu se fac pe baza stabilirii vitezei de desfasurare a procesului, deoarece viteza procesului nu poate fi apreciata in mod corect intr-un mediu multiprogramat. Inafara de aceste situatii in care este necesara evaluarea cu precizie a timpului, procesele nu sunt afectate de multiprogramare si de faptul ca se ele se desfasoara cu viteze diferite.

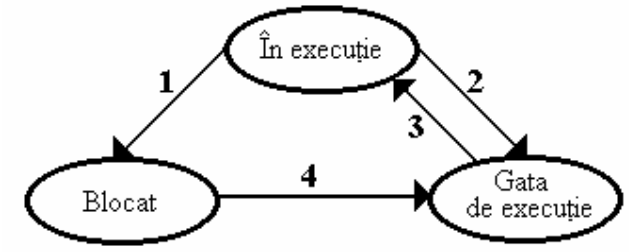
Ideea ce trebuie retinuta este ca un proces este o activitate din cadrul sistemului de operare, activitate care se face pornind de la un program executabil si care poseda niste date.
Procesul este caracterizat in orice moment prin niste informatii de stare. Un procesor unic poate fi partajat de mai multe procese pe durata unui interval de timp, comutarea unitatii centrale intre procese facindu-se pe baza unui **algoritm de planificare** aplicat de sistemul de operare. Pe baza acestui algoritm se determina cind un proces trebuie suspendat, si carui proces ii va fi acordata unitatea centrala in momentul urmator.

2.2.3 STARILE UNUI PROCES

In cadrul sistemului de operare procesele se pot crea sau se pot distruge; modul in care se fac aceste operatii depinde de la un tip de sistem la altul.
Exemplu : in cadrul UNIX procesele se creeza cu ajutorul apelului sistem FORK, apel care face o copie a procesului apelant. Procesul apelant se va numi proces tata, iar copia sa proces fiu. Procesul tata si cel fiu isi vor continua executia in continuare in mod paralel. De cele mai multe ori in cadrul procesului fiu se incarca si se lanseaza in executie prin apelul EXEC un alt program, ceea ce va duce la inlocuirea procesului fiu cu un alt proces, complet diferit. Legatura de filiatie se mentine pina la terminarea procesului lansat prin EXEC.

Un proces se poate bloca la un moment dat pe parcursul existentei sale in sistem. Acest fenomen are loc cind dintr-un un motiv oarecare, el nu isi poate continua executia.
In cele mai multe cazuri blocarea unui proces se produce pentru a astepta aparitia unui eveniment extern, cum ar fi de exemplu terminarea unei operatii de intrare/iesire. Producerea evenimentului extern este de obicei strict necesara pentru a continua prelucrarile din cadrul procesului.
Pe de alta parte, un proces ales (in curs de executie) poate fi stopat si el la un moment dat, desi el are toate conditiile de a se desfasura in continuare. Aceasta stopare este efectuata de sistemul de operare cind este necesara cedarea unitatii centrale unui alt proces, ce trebuie la rindul sau sa progreseze. In primul caz blocarea procesului este inerenta problemei, in al doilea caz suspendarea este legata de o caracteristica tehnica a sistemului (nu exista suficient de multe procesoare pentru a acorda cite unul fiecarui proces !).
Pornind de la observatiile de de mai sus se poate spune ca starile posibile ale unui proces din sistem sunt:

1. - **in executie**;
2. - **gata de executie** (suspendat proviz. ptr. a permite executia unui alt proc.);
3. - **blocat** (asteptind un ev. extern, fara care nu poate continua).



Tranzitia 1 se petrece atunci cind un proces nu poate sa isi continue executia. La unele sisteme de operare tranzitia 1 se petrece prin lansarea unui apel sistem de tip BLOCK sau WAIT. Dupa ce a trecut in starea blocat procesul ocupa un loc intr-o coada de asteptare speciala ce cuprinde si celelalte procese care au aceeasi stare ca si el.
Tranzitiile 2 si 3 sunt provocate de SO, fara ca procesul sa fie responsabil de acest fenomen. Tranzitia 2 se petrece atunci cind procesul in executie a epuizat timpul procesor acordat de sist. de operare.
Tranzitia 3 se produce atunci cind procesul este ales pe baza unui algoritm special din coada proceselor gata de executie. El va primi dreptul de a se executa pina la epuizarea intervalului de timp acordat sau pina cind va trece in starea blocat.
Tranzitia 4 se petrece cind evenimentul exterior asteptat de un proces blocat isi face aparitia. Procesul va trece din coada proceselor blocate in coada proc. gata de executie si va astepta sa fie ales pentru executie.
Obiectivul urmarit prin multiprogramarea unitatii centrale este de a dispune in permanenta de citeva programe aflate in executie simultan. Obiectivul partajarii timpului este de a comuta unitatea centrala intre procese atit de frecvent incit utilizatorii sa poata interactiona cu programele proprii fara sa fie deranjati unul de existenta altuia.

Componenta din sistemul de operare care decide carui proces din coada proceselor gata de executie i se va acorda unitatea centrala se numeste **planificatorul de procese** (**scheduler**). El realizeaza suspendarea si relansarea proceselor precum si gestiunea intreruperilor. Conceptual se considera ca planificatorul de procese constituie un nivelul de baza deasupra caruia se afla restul componentelor sistemului de operare, structurate sub forma unor procese secventiale (figura 2.3).

Sistemul de	Proces 1	Proces 2	. . . . .	Proces n
Operare	Planificatorul de procese			

2.2.4 TABELA PROCESELOR

Planificatorul de procese realizeaza activitatea de ordonantare a proceselor pe baza informatiilor aflate intr-o tabela speciala numita tabela proceselor, **PMT**, Process Map Table.
O intrare in tabela PMT corespunde unui proces si contine in general urmatoarele informatii:

- - starea procesului;
- - contorul ordinal propriu procesului; registrii unitatii centrale (registrii acumulatori, registrii de index, pointeri de stiva); continutul acestor registri trebuie salvat ori de cite ori procesul este intrerupt, pentru ca la o reluare ulterioara el sa poata sa isi continue executia in mod corect;
- - informatii de planificare (prioritatea procesului, pointeri in diversele cozi de asteptare) ;
- - informatii privind spatiul de memorie interna acordat procesului (valori ale regi. de baza si limita ale partitiei in care este incarcat codul procesului, sau pointeri catre tabelele de paginare sau de segmentare, dupa tehnica de gestiune a mem. folosite in sistem);
- - informatii de contabilizare : timpul procesor utilizat , limitele de timp acordate procesului, s.a;
- - inform. de legate de operatiile de intrare/iesire (lista perifericelor folosite in cadrul procesului, starea lor, lista fisierele deschise si starea lor, valoarea pointerilor de pozitionare in aceste fis. deschise, etc).

Continutul unei intrari din tabela proceselor poate varia de la un sistem de operare la altul; in cadrul tabelei pot exista si informatii necesare pentru lucrul planificatorului de procese al sistemului respectiv.

La sist. de operare UNIX informatiile tipice dintr-o intrare din tabela proceselor sunt :

Gestiunea proceselor	Gestiunea memoriei	Gestiunea fisierelor
Registre, contor ordinal	Pointer la segmentul de cod	Masca UMASK
Cuvint de stare program	Pointer la segmentul de date	Director radacina
Pointer de stiva	Pointer catre segmentul bss	Director de lucru
Starea procesului	Statutul la sfirsit de executie	Descriptori de fisiere
Data lansarii procesului	Identificator de proces	uid efectiv
Timp UC utilizat	Proces parinte	gid efectiv
Timp UC utilizat de procesele fii	Grupul procesului	Parametri apelurilor sistem
Data urmatoarei alarme	uid real, uid efectiv	Diversi indicatori
Pointer catre cozile de mesaje	gid real, gid efectiv	
Biti de semnale in asteptare	Tabela cu biti de semnal	
Identificatorul procesului (PID)	Diversi indicatori	
Diversi indicatori		

Activitatea de salvare sau incarcare a starii procesului poarta numele de comutare de context.
Durata comutarii de context tebuie sa fie cit mai mica, deoarece ea reprezinta o activitate auxiliara, dar care se repeta foarte des.
Comutarea de context consta de fapt in salvarea sau restaurarea unor registri ai unitatii centrale, si viteza ei este dependenta de hardware-ul echipamentului de calcul. In general cu cit sistemul de operare este mai complex cu atit comutarea de context este o operatie mai ampla.

2.2.5 CREAREA SI TERMINAREA PROCESELOR

Un proces poate sa creeze unul sau mai multe alte procese prin intermediul unui apel sistem special. Procesul care a efectuat crearea se numeste **proces tata**, iar noile procese se numesc **procese fii**. Fiecare proces fiu poate si el sa creeze alte procese fii, constituindu-se astfel un adevarat arbore de procese.

Un proces are nevoie de o serie de resurse pentru a efectua prelucrarile sale : unit. centrala, mem., fisiere, periferice, etc. Cind un proces creeaza un fiu, acesta din urma poate sa obtina resursele de care are nevoie fie direct de la sistemul de operare, fie un subansamblu de resurse de la procesul tata. Astfel procesul tata repartizeaza o parte din resursele sale proceselor fii. Acest partaj de resurse intre tata si fii sai are avantajul ca reduce incarcarea generala a sistemului, iar in plus procesul tata face schimb de informatii cu procesele fii.
In orice sistem de operare procesul tata trebuie sa cunoasca identitatea proceselor fii si sa aiba posibilitatea sa examineze starea acestora. Cind un proces creeaza un altul, exista doua posibilitati din punctul de vedere al executiei procesului tata:
- procesul tata continua in paralel cu procesul fiu;
- procesul tata asteapta pina cind unii dintre fii, sau chiar toti, se incheie.

Privind spatiul de adrese al procesului fiu, exista si aici doua posibilitati :
- procesul fiu este un duplicat al procesului tata;
- procesul fiu incarca un alt program si il lanseaza in executie.

Exemplu : in cazul sistemului de operare UNIX crearea unui nou proces se face cu ajutorul apelului sistem FORK. Procesul fiu rezultat este copie a procesului tata, dar are alt identificator (PID). Cele doua procese continua sa se execute de la instructiunea de dupa apelul FORK, numai ca exista deosebirea urmatoare : codul de retur intors de FORK este 0 in procesul fiu, si identificatorul fiului (diferit de 0) in procesul tata. De obicei procesul fiu executa apoi un apel sistem EXEC care inlocuieste propriul spatiu de adrese cu codul unui program executabil care este lansat in executie. In acest caz procesele fiu si cel tata sunt diferite si pot continua sa se execute in paralel.

SO VMS permite crearea de procese fii atit prin duplicare cit si prin inacrcarea unui alt program care se lanseaza in executie. Windows NT suporta ambele modele : se poate face duplicarea tatalui sau procesul tata poate sa specifice numele unui program executabil care va fi incarcat si lansat ca fiu al sau.

Un proces se incheie cind a executat ultima sa instructiune. El cere sistemului de operare sa-l suprime prin intermediul unui apel sistem de tip EXIT. In acest moment toate resursele detinute de proces sunt eliberate de sistemul de operare. Procesul tata are posibilitatea de a termina procesele fii cind:
- procesul fiu a depasit indicele de utilizare al resurselor atribuite;
- prelucrarile efectuate de procesul fiu nu mai sunt necesare;
- procesul tata este suprimat si SO nu permite ca un fiu sa se execute atunci cind procesul tata s-a incheiat.

Marea majoritate a sist. de operare nu permit existenta unui proces fiu atunci cind tatal s-a terminat. In aceste sist. incheierea normala sau anormala a unui proces duce la incheierea tuturor fiilor sai; fenomenul se numeste **terminare in cascada**.

Exemplu : sub UNIX un fiu se termina prin apelul EXIT; procesul tata poate astepta incheierea unuia dintre fii sai utilizind apelul sistem WAIT. Apelul sistem WAIT intoarce tatalui identificatorul proc.fiu care s-a incheiat.

2.3 APEL SISTEM

Apelul sistem este un mecanism prin care utilizatorul comunica cu sistemul de operare si **solicita diverse servicii acestuia.**

Apelul sistem este similar cu apelul de procedura din cadrul unui program si are ca scop:
- lansarea unei operatii de intrare/iesire;
- lansarea unei operatii de comunicare cu un alt proces.

Fiecarui apel sistem ii corespunde o **procedura de biblioteca** care :
- **executa trecerea** de la procesul utilizator la serviciul solicitat din cadrul SO;
- **mascheaza pregatirile necesare activarii SO,** astfel incit apelul sistem sa apara ca un apel obisnuit de procedura.

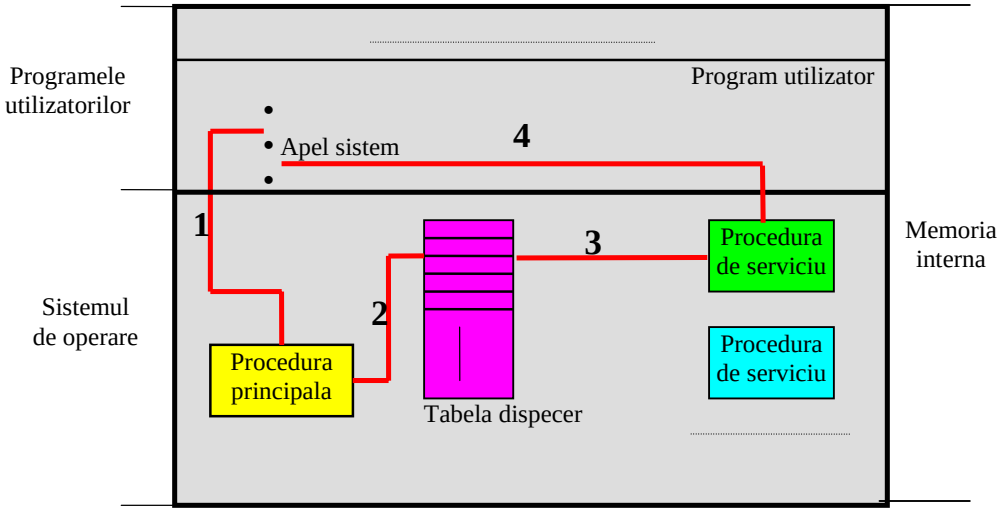
Procedura de biblioteca:
1. initiaza apelul sistem:
- plaseaza parametrii apelului sistem intr-un loc bine stabilit, cum ar fi:
  - registrii procesorului,
  - o structura de date speciala pentru transfer parametri.
- executa o instructiune speciala (TRAP sau EMT -Emulator Trap-) care invoca sistemul de operare.

2. incheie apelul sistem prin predarea controlului de la SO la procesul apelant;
3. transmite codul de retur (care indica daca apelul sistem solicitat s-a desfasurat in mod corect sau nu) procesului apelant.

Sistemul de operare :
1. **preia controlul de la un proces** prin intermediul procedurii de biblioteca;
2. **verifica validitatea** parametrilor transmisi;
3. daca acestia sunt corecti, **executa serviciul solicitat** in modul supervizor;
4. la incheierea prelucrarii, **plaseza intr-un registru un cod de retur;**
5. **executa o instructiune** de tip **RETURN FROM TRAP** care preda controlul procedurii de biblioteca.

Numarul si forma apelurilor sistem variaza de la un sistem de operare la altul.
Apelurile sistem sunt mecanismele esentiale prin care sistemul de operare furnizeaza diverse servicii utilizatorului.

Modul de executie al unui apel sistem :
- fiecarui apel sistem ii corespunde o singura proc. de serviciu;
- un apel sistem se efectueaza din mai multi pasi;



Pasul 1 :
- controlul este transferat de la programul de aplicatie catre SO;
- executia apelului sistem se va face **in mod supervizor;**

Pasul 2 :
- SO examineaza parametrii apelului;
- SO determina ce serviciu specific a solicitat programul utilizator:
- pe baza parametrilor este gasita o intrare *n* dintr-o tabela indexata; intrarea gasita contine adresa procedurii de serviciu necesare;

Pasul 3 :
- controlul este predat proc. ce efectueaza prelucrarea solicitata;

Pasul 4 :
- dupa incheierea procedurii de serviciu, controlul este redat programului ce a lansat apelul sistem;
- se face comutarea din modul de lucru supervizor in modul utilizator.

2.4 SISTEMUL DE FISIERE

La calculatoarele din generatia a II-a SO include un **sistem de fisiere on-line** pentru stocarea si regasirea informatiilor.

Un **fisier** este o **colectie de informatii reunite sub un nume comun;** fisierele sunt o organizare specifica suporturilor magnetice.

Caracteristici:
- fisierele sunt grupate din punct de vedere logic in **directoare;**
- directoarele alcatuiesc o **structura arborescenta;**
- exista un **director radacina** de la care pornesc toate directoare.

Observatie: Fiecare proces poseda un **director implicit de lucru,** care este identic cu cel al utilizatorului proprietar al procesului.
Fiecare fisier are un nume (identificator) unic = **specificatorul fisierului:**
- **numele fisierului;**
- **pozitia sa** in cadrul arborelui de directoare (**traseul** ce trebuie parcurs in arbore pina la fisierul respectiv).

Pentru a scrie sau citi intr-un fisier acesta trebuie deschis. Deschiderea fisierului:
- se face cu un apel sistem special;
- daca drepturile de acces ale utilizatorului sunt corespunzatoare, SO reintoarce procesului apelant un **descriptor de fisier (file descriptor = file handle).**

Descriptorul de fisier:
- este folosit la orice operatie cu fisierul respectiv;
- este o valoare numerica, sau un sir de caractere, sau adresa unei tabele.

La unele SO (UNIX si MS-DOS) a fost introdusa **abstractizarea in lucrul cu anumite dispozitive periferice,** prin extinderea notiunii de fisier pentru accesul la aceste periferice.

Perifericele care nu au atasata o structura de fisiere, dar sunt accesate ca niste fisiere: **imprimanta, ecranul terminalului, tastatura terminalului, interfetele de retea**
Acestea sunt denumite **fisiere speciale de tip caracter** si suporta operatii de scriere/ citire, ca si un fisier obisnuit.
Fisierele speciale de tip caracter au un descriptor implicit si sunt alocate automat utilizatorului la atasarea la sistem.

Exemplu: la UNIX:

- **descriptorul 0** = intrarea standard (tastatura terminalului utilizat.);
- **descriptorul 1** = iesirea standard (ecranul terminalului utilizat.);
- **descriptorul 2** = iesire standard de eroare (ecranul teminalului).

La UNIX fisierele speciale **pot fi redirectate** catre alte fisiere (speciale sau obisnuite).

Perifericele care o structura de fisiere atasata (discurile magnetice, floppy disc, s.a.) **pot fi accesate in mod global,** ca un singur **fisier special de tip bloc**. In acest caz **structura de fisiere nu mai este vizibila**, accesul facindu-se direct la nivel de blocuri fizice de date. Acest mod de lucru mai special este de obicei permis doar anumitor utilizatori, cum ar fi **administratorul de sistem.**

Mecanismul de conducta (pipe):

- se refera atit la procese cit si la fisiere;
- se foloseste la **comunicatia intre procese.**

O conducta este de fapt un pseudofisier:

- un proces A **poate sa transmita** informatii unui proces B, scriindu-le in conducta ca si cum le-ar fi scris intr-un fisier obisnuit;
- procesul B **poate citi informatiile din conducta** ca si cum le-ar fi citit dintr-un fisier;
- **accesul la conducta este sincronizat,** in sensul ca un proces nu poate scrise cind conducta este plina, sau nu poate citi cind conducta este goala.

2.5 RESURSE IN CADRUL SISTEMULUI DE OPERARE

La un sistem de calcul se pot deosebi **doua tipuri de resurse:**

- **fizice :**
  - **procesorul central; memoria interna; memoria externa; canalele de intrare/iesire; dispozitivele periferice;**
- **logice (abstracte) ,** ce **s**unt create de SO prin activitatea si sunt chiar componente ale SO:
  - **module executabile; biblioteci de programe; baze de date; translatoare de programe (compilat., asambloare);**
  - **programe utilitare.**

Resursele sunt **facilitati puse la dispozitie** de SO pentru efectuarea serviciilor cerute de procese sau utilizatori.
SO contine mecanisme prin care devine posibila **utilizarea in comun a resurselor** (atit fizice cit si logice):

- administarea si gestionarea resurselor;
- asigura protectia acestora;
- rezolva conflictele legate de partajarea resurselor.

Administrarea resurselor in cadrul SO se face in tinindu-se cont de **scopuri bine precizate:**

- la calculatoarele medii si mari un scop este: **utilizarea in comun resurselor** (procesorul si memoria interna);
- la PC scade importanta utilizarii in comun a resurselor de tip procesor sau memorie, dar se urmareste **gestionarea cit mai corecta a informatiei** (prin sistemul de fisiere).

Scopurile urmarite in activitatea SO pot fi urmatoarele :

- **minimizarea timpului de raspuns** la cererile utilizatorilor;
- **maximizarea factorului de utilizare** a resurselor fizice;
- **maximizarea factorului de utilizare** a resurselor fizice, cu restrictia ca **timpii de raspuns** nu trebuie sa depaseasca o limita superioara.

Cea mai importanta decizie privind activitatea SO este alegerea elemntului primordial: **timpul de raspuns** sau **factorul de utilizare al resurselor.**
Aceste doua scopuri sunt **contradictorii**, indiferent daca deciziile sunt luate prin algoritmii incorporati in cadrul SO sau sunt luate de catre operatorul calculator.

Gestionarea resurselor de catre SO se face intotdeuna dupa o anumita **strategie,** ceea ce inseamna stabilirea precisa a scopurilor urmarite in utilizarea calculatorului pe care ruleaza acel SO.
Un principiu important: **separatia intre politici si mecanisme. Mecanismele** determina modul cum se va efectua o anumita activitate. **Politicile** decid ce anume se va efectua, in functie de scopul urmarit.
Exemplu :

- intreruperile date de ceasul intern constituie **mecanismul concret** prin care UC poate fi comutata intre procese.
- la fiecare intrerupere data de ceas, **planificatorul de procese** decide carui proces ii va fi alocata UC si pe ce durata, ceea ce este o politica de lucru.