2.6 Mecanisme de control asincron sau parțial sincron

Mecanismele prezentate în secțiunile precedente au un caracter ***sincron***:

- ***procesele*** supuse controlului sunt ***active***;
- ***efectele schimbării*** contextului de către un proces ***sunt receptate*** în acelaşi timp de procesul sau procesele interesate în recepția acestei schimbări;

Recepția efectului schimbării de context ***să nu fie făcută instantaneu***.

- procesul generator al schimbării ***să anunțe*** această schimbare depunând o informație adecvată într-un anume loc: ***căsuță poştală, canal specializat etc.***
- Procesul sau procesele care trebuie să reacționeze la această schimbare au obligația ca în momentul pornirii lor, ***să consulte căsuța poştală sau canalul respectiv.***
- Dacă este prezentă informația de schimbare a contextului, atunci procesul consultant îşi ***decide evoluția ulterioară în funcție de conținutul acestei informații.***

Spunem că astfel de mecanisme au un caracter asincron.

OBS Există şi mecanisme mixte.

Mecanisme de control asincron sau parțial sincron

Schimbul de mesaje

Fluxul de octeți

Memoria partajată

Evenimente, excepții, semnale

2.6.1 Schimbul de mesaje

Mecanismele de tip ***schimb de mesaje*** au apărut cam în acelaşi timp cu ***semafoarele***.

Domenii de utilizare: - comunicarea prin *poştă electronică (e-mail)*

- comunicarea între persoane;
- comunicarea între procese:Există servere în Internet care transmit automat e-mailuri către utilizatori, după cum utilizatorii se pot adresa direct unor servere prin e-mailuri care respectă anumite structuri standard.

Sistemele de operare precum Unix, dar nu numai, îşi definesc propriile standarde de mesaje, specifice comunicării numai între procese. *Coada de mesaje* este un astfel de exemplu.

Destinație	Sursă	Tip mesaj	Conținut mesaj

Figura 2.12 **Structura unui mesaj**

Schimbul de mesaje

- ***Destinație şi Sursă*** sunt adrese reprezentate în conformitate cu convențiile de adresare ale implementării mecanismului de mesaje.
- ***Tip mesaj*** dă o caracterizare, în concordanță cu aceleaşi convenții, asupra naturii mesajului respectiv.
- Aceste trei câmpuri, eventual completate şi cu altele în funcție de specific, formează ***antetul mesajului***.
- ***Ultimul câmp*** conține ***mesajul propriu-zis***.

OBS:

- Dacă sistemul de mesaje se adresează ***factorului uman***, atunci informațiile sunt reprezentate sub formă de ***text ASCII***, iar conținutul nu trebuie să aibă o structură rigidă.

- Dacă sistemul de mesaje se adresează ***proceselor***, atunci câmpurile lui au ***structuri standardizate***, eventual chiar ***conținut binar***.

Schimbul de mesaje

Indiferent de natura corespondenților, scenariul comunicării prin mesaje este acelaşi:

• Emițătorul mesajului compune mesajul şi îl expediază spre destinatar sau destinatari, după caz. In situația în care corespondenții se află pe acelaşi sistem de calcul, expedierea înseamnă depunerea lui într-o zonă specială numită ***canal*** sau ***coadă de mesaje***.

• Sistemul care conține un destinatar recepționează mesajul. ***Recepția*** se face numai atunci când ***sistemul devine operațional***. Pe perioada dintre expediere şi recepție mesajul este ***păstrat temporar*** prin intermediul unui ***sistem de zone tampon***.

• Atunci când procesul (utilizatorul) destinatar devine activ, sistemul îl anunță de primirea mesajului respectiv. De asemenea, ***fiecare proces activ controlează din când în când căsuța poştală*** sau canalul spre a sesiza apariția de noi mesaje.

• Atunci când ***procesul consideră de cuviință***, îşi ***citeşte mesajul primit***. In funcție de conținutul lui, procesul îşi poate modifica execuția ulterioară.

Din scenariul de mai sus reiese clar caracterul asincron al acestui mecanism

Schimbul de mesaje

Caracteristici ale mecanismelor de tip schimb de mesaje

- ***Frecvența de consultare a canalului de mesaje depinde de specificul aplicației.***
- De regulă, după ce un mesaj a fost ***consultat*** de către procesul receptor el este ***eliminat*** din canalul de mesaje.
- ***Ordinea de extragere*** a mesajelor din coadă depinde de asemenea de ***specificul sistemului de mesaje***.
- Ordinea cea mai folosită este ***FIFO*** – primul venit primul servit. Ea poate fi aplicată fie:
  - ***asupra întregii cozi***;
  - ***asupra mesajelor de un anumit tip***;
  - fie asupra ***mesajelor care îndeplinesc o anumită condiție***.
- Există sisteme de mesaje în care ***procesul însuşi decide*** care mesaj îl extrage mai întâi din coadă, deci extragerea poate fi aleatorie.

Schimbul de mesaje

In funcție de numărul destinatarilor unui mesaj, se disting:

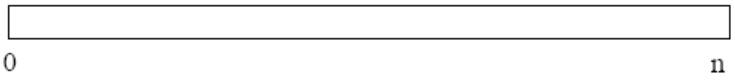
- ***Mesaje unicast****,* cu un destinatar unic: corespondență punct la punct.
- ***Mesaje multicast****,* în care există un grup de destinatari. Ei se stabilesc fie prin specificarea tuturor destinatarilor, fie există o listă predefinită de destinatari la care va fi trimis mesajul.
- ***Mesaje broadcast****,* în care mesajul este trimis automat tuturor destinatarilor care fac parte dintr-o anumită categorie:
  - utilizatorii dintr-o rețea LAN sau WAN
  - angajații unui compartiment sau companii
  - proceselor care utilizează o anumită resursă etc.

Schimbul de mesaje este folosit pe scară largă mai ales în sistemele distribuite, deci programarea distribuită operează mai des cu acest mecanism. Există totuşi şi aplicații ce se derulează concurent pe acelaşi sistem şi care comunică între ele prin mesaje specializate.

2.6.2 Fluxuri de octeți

Definiție

Un ***flux de octeți*** este un canal unidirecțional de date, de dimensiune limitată, asupra căruia acționează două categorii de procese: scriitori şi cititori. Disciplina de acces la flux este FIFO, iar octeții în care se scrie, respectiv care sunt citiți, avansează în manieră circulară în buffer.

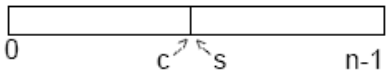


Să presupunem că avem de-a face cu un buffer de *n* octeți, numerotați de la 0 la *n*-1. ***Pentru manevrarea eficientă a fluxului, este necesară întreținerea permanentă a trei parametri***:

- ***s poziția curentă*** în care se poate scrie la următorul acces;
- ***c poziția curentă*** de unde se poate citi la următorul acces;
- ***p o variabilă booleană*** cu valoarea **TRUE** dacă poziția de **scriere** din buffer se află **înaintea** poziției de **citire**, sau valoarea **FALSE** dacă în urma unei citiri sau scrieri s-a depăşit ultimul loc din buffer şi s-a revenit la începutul său.

Diverse ipostaze ale bufferului, condițiile ce trebuie îndeplinite de cei trei parametri şi modificările acestora în urma unei operații

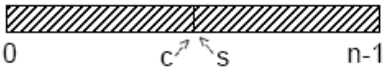
Porțiunea haşurată indică zona octeților încă disponibili în buffer.



p=true si c=s => buffer gol

Figura 2.13 **Buffer gol**

Dacă bufferul este gol – nu conține nici un octet pentru citit - atunci ***procesele cititori*** aşteaptă până când un ***proces scriitor*** depune cel puțin un octet în flux.



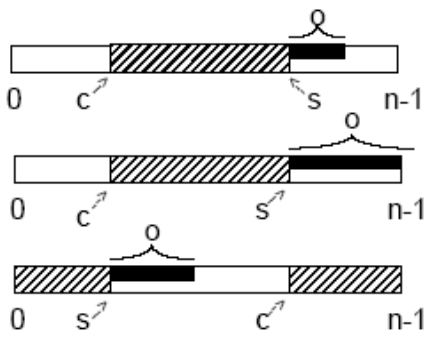
p=false si c=s => buffer plin

Figura 2.14 **Buffer plin**

Dacă bufferul este plin – nu mai este nici un loc pentru scriere – atunci procesele ***scriitori*** aşteaptă până când un ***cititor*** extrage cel puțin un octet.

Diverse ipostaze ale bufferului, condițiile ce trebuie îndeplinite de cei trei parametri şi modificările acestora în urma unei operatii

Figura 2.15 prezintă ***condițiile necesare*** pentru a se putea scrie **o** octeți în buffer. Prima şi a treia situație provoacă doar avansarea indicelui ***s*** cu ***o***, în timp ce scrierea din a doua situație îl modifică atât pe ***s*** cât şi pe ***p***.



p=true si s+o<n => s=s+o

p=true si s+o>=n si (s+o)mod n<=c => p=false s=(s+o)mod n

p=false si s+o<=c => s=s+o

Figura 2.15 **Situații de scriere în buffer**

Spre deosebire de ***scrierea unui mesaj*** în cazul ***fluxului de octeți***, regulile de citire sunt diferite şi anume

• Un octet odată scris în flux nu mai poate fi recuperat şi după el se va scrie octetul următor, în scrierea curentă sau în scrierea următoare.

• Un scriitor în flux precizează un număr ***o*** de octeți pe care doreşte să-i scrie. Dacă în flux mai există cel puțin ***o*** poziții libere, atunci scriitorul depune cei ***o*** octeți în flux. Dacă în flux există doar *r* octeți liberi, 0 < *r* < *o*, atunci se vor depune în flux doar **primii *r*** octeți dintre cei ***o*** şi ***scrierea se consideră terminată***. ***Rămâne la decizia procesului scriitor dacă doreşte sau nu să scrie în flux ceilalți o-r octeți printr-o scriere ulterioară.***

Condițiile necesare pentru a se putea citi următorii o octeți: primele două situații provoacă doar avansarea indicelui *c* cu *o*, în timp ce citirea în ultima situație îl modifică atât pe *c* cât şi pe *p*.

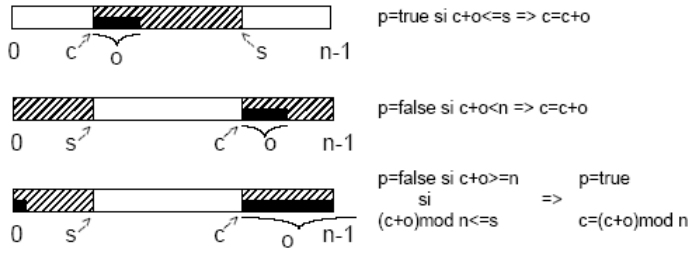


Figura 2.16 **Situații de citire din buffer**

Entitatea mesaj este ***receptată*** exact aşa cum a fost ***emisă***. În cazul ***fluxului de octeți***, ***regulile de citire*** sunt diferite şi anume:

• Un octet odată citit este eliminat din flux şi octetul următor este primul care va fi citit, în aceeaşi citire sau în citirea următoare.

• Un cititor din flux precizează un număr ***o*** de octeți pe care să-i citească. Dacă sunt disponibili cel puțin ***o*** octeți, atunci cititorul îi primeşte. Dacă în flux există doar ***r*** octeți, **0 < *r* < *o***, atunci:

- Cititorul primeşte numai cei ***r*** octeți existenți şi citirea se consideră terminată.

- Rămâne la decizia procesului cititor dacă doreşte sau nu să obțină ceilalți ***o-r*** octeți printr-o citire ulterioară.

Atât pentru citire, cât şi pentru scriere, mai trebuie precizate încă două condiții şi anume:

• Toate ***citirile şi scrierile în flux sunt operații atomice***: nici un alt proces nu poate întrerupe operația până când fluxul o declară terminată.

• ***Intre orice două operații consecutive lansate de un proces este posibil ca un alt proces să efectueze o operație asupra fluxului***.

Pentru fluxul de octeți se cunosc şi alte denumiri, ca de exemplu *coadă cu lungime limitată* sau *pipe*.

Procesul cititor trebuie să conştientizeze faptul că în condițiile în care la flux sunt ataşați mai mulți cititori şi mai mulți scriitori,

- nu poate decide de la ce procese scriitori provin octeții citiți,
- nu poate decide dacă toți octeții citiți la un moment dat provin de la un proces scriitor sau de la mai mulți.

Să presupunem că ***există 3 procese scriitor*** A, B, C, care doresc să scrie următorii 9, 5 respectiv 6 octeți:

$a_1a_2a_3a_4a_5a_6a_7a_8a_9$
$b_1b_2b_3b_4b_5$
$c_1c_2c_3c_4c_5c_6$

Presupunem că:

- procesele scriitori îşi țin evidența octeților scrişi efectiv în flux;
- la o a doua scriere vor depune în flux începând cu ultimul octet nescris;
- ***procesul cititor D*** doreşte să citească ***4 octeți***;
- ***procesul cititor E*** doreşte să citească ***13 octeți*** din flux;

Presupunem că fluxul are capacitatea de 7 octeți.

Proces Doreşte să scrie *l* octeți	Octeți nescrişi	Conținut flux	Octeți Citiți	Proces doreşte să citească *l* octeți
A 9	a_8a_9	$a_1a_2a_3a_4a_5a_6a_7$		
B 5 wait	$b_1b_2b_3b_4b_5$			
C 6 wait	$c_1c_2c_3c_4c_5c_6$			
			$A_1a_2a_3a_4a_5a_6a_7$	E 13
				D 4 wait
A 2	-	a_8a_9		
B 5	-	$a_8a_9b_1b_2b_3b_4b_5$		
		$b_3b_4b_5$	$A_8a_9b_1b_2$	D 4
C 6	$c_4c_5c_6$	$b_3b_4b_5c_1c_2c_3$		
C 3 wait				
			$B_3b_4b_5c_1c_2c_3$	E 6
C 3	-	$c_4c_5c_6$		

2.6.3 Memorie partajată

Conceptul de memorie partajată reprezintă o modalitate utilă ***de a separa fluxul de control al execuției unei aplicații de comunicarea datelor dintre entitățile implicate (procese, care se execută pe acelaşi sistem (uniprocesor sau multiprocesor) sau pe sisteme diferite).***

Procesele se află pe acelaşi sistem

- segmentul de memorie partajată face parte din ***spațiul global de memorie al sistemului***.

- mai multe procese colaborează prin intermediul acestui spațiu de memorie comun, în care sunt memorate o serie de variabile accesibile tuturor proceselor implicate. Unele dintre procese scriu date în acest spațiu comun, iar altele citesc aceste date.

Unul dintre principiile programării pe orice platformă este acela că *"**fiecare proces îşi accesează în mod exclusiv propriul spațiu de memorie (internă)".***

Accesul a două procese la un segment de memorie partajată

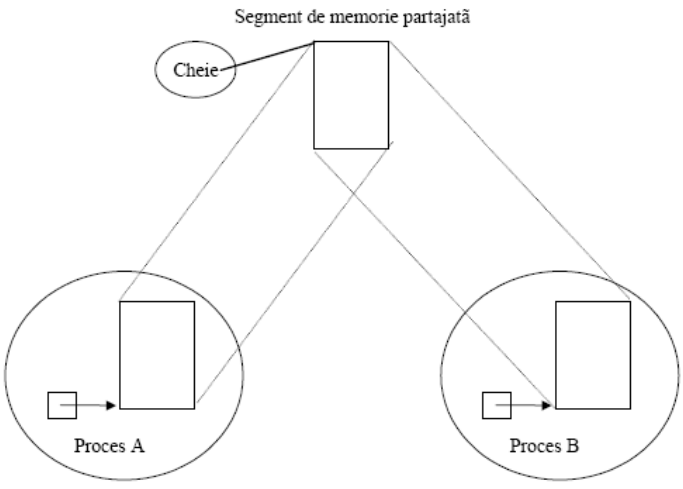


Figura 2.17 **Accesul la un segment de memorie partajată**

Accesul a două procese la un segment de memorie partajată

Principiul comunicării prin memorie partajată constă în:

1. Un ***proces crează*** un segment de ***memorie partajată***. Descriptorul zonei este trecut în structura de date ataşată zonei.

2. Procesul creator asociază segmentului de zonă partajată o ***cheie numerică*** pe care toate procesele care accesează zona trebuie să o cunoască.

3. ***Procesul creator*** *asociază zonei* ***drepturile de acces.***

4. Orice proces care doreşte să acceseze segmentul de memorie partajată, inclusiv creatorul, trebuie să *şi-o ataşeze la* ***spațiul lui virtual de adrese.*** In urma acestei ataşări obține adresa zonei de memorie partajată.

OBS:

- Astfel comunicarea între procese se realizează direct.
- Uneori este necesară protecția datelor din segment pentru a se evita rezultate inconsistente. Pentru aceasta, se pot folosi mecanismele de sincronizare cunoscute: semafoare, monitoare etc.
- Mecanismul de memorie partajată poate fi privit în acelaşi timp atât ca unul *sincron*, cât şi ca unul *asincron*, în funcție de *specificul aplicației*.

2.6.4 Evenimente, excepții, semnale

- Execuția unui proces, atât în context secvențial cât şi în context concurent, poate fi "***perturbată***" de ***intervenții externe independente de el şi la momente de timp imprevizibile***. Indiferent de natura perturbării, procesul este anunțat de aceasta folosindu-se ***sistemul de întreruperi*** existent pe platforma pe care se execută procesul.

Unele dintre aceste perturbări au ***cauze intrinseci***, dependente de funcționarea echipamentelor hard, altele apar datorită faptului ***că în interacțiunea proceselor cu mediul înconjurător au apărut situații deosebite***.

In context concurent, o parte dintre perturbările din a doua categorie sunt folosite în ***acțiuni de sincronizare şi coordonare a proceselor***.

În funcție de "sursele" acestor perturbări se poate face o clasificarea acestora

- ***eveniment*** - generat de o interfață grafică sau de un ceas.
- ***excepție*** - programul solicită resurse interne (variabile, operații) sau externe (fişiere, periferice) inexistente.
- ***eroare*** - din rațiuni hardware procesul trebuie oprit.
- ***semnal*** - procesul primeşte un anunț din partea altui proces.

Eveniment este utilizat mai ales în legătură cu funcționarea **interfețelor grafice utilizator (GUI – Graphics User Interface).** Sunt binecunoscute exempl.:

-**interfețele grafice oferite de platformele Microsoft Windows**;

- **interfețele de tip X-window operaționale în principal pe platforme din familia Unix**;
- **platformele Java oferă şi ele posibilitatea de a defini şi utiliza astfel de interfețe grafice**.

Utilizatorul unui ***GUI*** **provoacă un eveniment** atunci când ***acționează din exterior*** asupra lui prin intermediul tastaturii, a mouse-ului, track-ball-ului sau a altor periferice specifice.

Evenimente, excepții, semnale

Efecte:

- modificarea conținutului informației unuia dintre obiectele grafice de pe monitor;

- redimensionarea unei ferestre;

- închiderea unei aplicații prin distrugerea ferestrei ei principale, etc.

Aceste evenimente sunt ***prelucrate***, de către sistemul de programe aferent, într-o manieră ***asincronă***.

- Conceptul principal în jurul căruia gravitează tratarea evenimentelor este ***coada de evenimente***.
- Prelucrarea acestei cozi se face prin intermediul ***buclei de evenimente***. O astfel de buclă funcționează conform descrierii de mai jos:

```
while (TRUE) {
      Extrage următorul eveniment din coadă
            sau aşteaptă până când apare un eveniment
      switch (eveniment)
            case eveniment de tip 1:
                  tratează evenimentul de tip 1
                  - - - - - - - - - - -
            case eveniment de tip n:
                  tratează evenimentul de tip n
      }// switch
}//while
```

- Sursa evenimentului pune in coada o inregistrare specifica evenimentului respectiv.
- Tratarea evenimentului se face mai tirziu atunci cand el este scos din coada
- De regula se extrage numai un eveniment odata
- Daca evenimentul asteapta un raspuns de la sistem atunci se goleste intreaga coada.
- !!!!!! Daca in coada sunt evenimente si apare unul care asteapta raspuns de la sistem, atunci este procesata mai intai toata coada dupa care se asteapta raspunsul

Categorie aparte de evnimente: cele legate de scurgerea unui interval de timp:

- programele pot cere sistemului sa fie puse in adormire si sa fie trezite dupa un interval de timp sau la aparitia unui anume eveniment.

Prin exceptie se intelege o situatie deosebita generata de cauze interne:

- tentativa de deschidere a unui fisier care nu exista;

- tentativa de utilizare a unui indice de tablou a carui valoare este in afara limitelor fixate

- tentativa de impartire la zero.

In majoritatea situatiilor executia programului este suspendata la aparitia unei exceptii si se asteapta ca programul sau sistemul de operare sa rezolve situatia.

Daca se lasa rezolvarea pe seama sistemului, solutia acestuia este terminarea anormala a programului.

OBS: ATUNCI CIND EXCEPTIA ESTE TRATATA DE PROGRAMUL INSUSI, PROIECTANTUL TREBUIE SA PREVADA O SOLUTIE DE REZOLVARE SPECIFICA PENTRU FIECARE EXCEPTIE IN PARTE.

Eroare – se considera o situatie in care din ratiuni hardware procesul nu-si mai poate continua activitatea:

- codul unui proces este corupt si ofera spre executie instructiuni care nu exista,
- cand se solicita o adresa de memorie inexistenta

Tratarea unei erori este intotdeauna o operatie sincrona cu executia programului

Semnalul poate fi asociat cu emiterea unei intreruperi adresate unui proces.

- De regula, procesul care primeste un semnal nu stie sursa acestuia, in schimb primeste semnalul in maniera sincrona, imediat ce a fost emis.
- Procesul poate:
  - sa-si modifice functionarea imediat ce a primit semnalul,
  - sa ignore semnalul primit;
  - sa amane prelucrarea acestuia.

- Sistemele de calcul / de operare mai vechi au folosit ca suport pentru semnale chiar sistemul de intreruperi.
- Sistemul de operare UNIX ofera un mecanism simplu si elegant de utilizare a semnalelor ca instrumente pur software.