Curs 1 Introducere

- Tehnici de comunicații
- Sisteme de comunicații
  - Abordări numerice: servicii integrate la standarde de calitate superioare
    - Echipamente numerice inteligente: procesoare dedicate, procesoare de semnal, microcontrolere, micro şi minicalculatoare
    - Componenta software
    - Dezoltarea foarte rapidă a industriei microelectronicii

Curs 1 Introducere

- Limite de performanță
  - Echipamente fizice utilizate
  - Unelete software
- Sistemele de comunicații actuale:
  - Sisteme complexe
    - Gestionarea:
      - Unor serii largi de echipamente
      - Frecvent simultane
      - Reacții previzibile şi completate în timp real

Curs 1 Introducere

- Sisteme în timp real
- Sisteme de operare în timp real
- Aplicații, utilitare în timp real
- Structuri eficiente de comunicații între acestea - sincronizare

Curs 1 Introducere

Timp real. Sensuri şi limitări

- Cauzele întârzierilor
  - Datele de intrare - disponibile pentru prelucrare - cu întârziere
  - Calculator: mod de organizare a funcționării şi exploatării
  - Rezultatele prelucrărilor
- Definirea sistemului de calcul în timp real: TIMP DE RĂSPUNS

Curs 1 Introducere

Probleme specifice sistemelor de calcul în timp real

- Probleme legate de hardware
  - Sisteme de intrare/ieşire
  - Terminale specializate
  - Sisteme de întreruperi
  - Probleme specifice legate de componenta software
    - **Sisteme de operare în timp real**
    - Limbaje de programare a aplicațiilor în timp real
    - Metode şi tehnici pentru programarea(ingineria programării) aplicațiilor în timp real
  - Probleme specifice legate de proiectarea şi implementarea aplicațiilor în timp real

Curs 1. Introducere

Scopul cursului – Bazele teoretice şi unele aspecte practice ale programării concurente

Premizele apariției programării concurente

- **organizarea şi exploatarea într-un mod mai eficient a hardware-ului calculatorului;**
- **rezolvarea unor probleme foarte dificile:**
  - rezolvarea sistemelor de ecuații diferențiale;
  - calculul fluidelor de ardere în motoare termice;
  - simularea globală a fenomenelor meteo;
  - sistemele de rezervare a biletelor etc.

Soluția: calculatoare cu mai multe procesoare (***sisteme multiprocesor***), sau punerea pe treabă a mai multor calculatoare în mod simultan (prin ***programare distribuită***).

- Calculator = unitate centrală + memorie
  - evoluția tehnologica a acestora
  - orice octet lucrează numai atunci când este solicitat de la centru

***Per global, doar o mică parte din hardware este utilizată în fiecare moment* !**

Cum am putea organiza şi exploata într-un mod mai eficient acest hardware?

Unii cercetători sunt de părere că numai neştiința noastră în a ***organiza calcule paralele*** cu zeci (sute) de procese ne împiedică să realizăm computere mult mai eficiente

Multiprogramare, multitasking, programare în timp real, calcul paralel, sisteme distribuite ?

Cum utilizăm eficient calculatorul pe care îl avem?

- **multitasking** (***să lucrezi într-o aplicație în timp ce altele aflate în aşteptare continuă în paralel)*** → posibil prin: **programarea concurentă**

- *De ce se face atâta caz astăzi de **programarea concurentă**? De ce se studiază acest domeniu?*
- *Practic ce rezultate se urmăresc a fi obținute?*
- *Câteva răspunsuri:*
  - *Preocupările în ceea ce priveşte programarea concurentă nu sînt în nici un caz noi;*
  - *Nou este faptul că în prezent acestea încep **să pătrundă în sfera de preocupare a unui cerc larg de programatori şi utilizatori** şi nu mai este "privilegiul" unor "inițiați".*

Multiprogramare, multitasking, programare în timp real, calcul paralel, sisteme distribuite ?

- **Primele sisteme concurente** – calculatoare **cu un singur procesor central**
  - Din punct de vedere fizic o singură activitate se putea desfăşura la un moment dat
  - *procesor specializat pt. operatii I/O*

- ***Paralelism?:***

  apărea doar la nivel logic, prin faptul că se executau cu schimbul secvențe de instrucțiuni aparținând diferitelor activități

Programarea concurentă pe sisteme monoprocesor

Avantaje:

1. utilizarea eficienta a procesorului central;

- în timp ce o activitate aşteaptă un anumit eveniment (de exemplu încheierea unei operații de I/O sau a unui acces la distanță) procesorul se poate dedica altei activități

2. deservirea în paralel a mai multor utilizatori (sisteme multiuser);

- mai mulți utilizatori pot fi legați prin câte un terminal la acelaşi unic calculator. Procesorul deserveşte pentru o anumită cuantă de timp un utilizator, după care trece la următorul. În mod subiectiv, fiecare utilizator crede că are acces exclusiv la calculator deşi în realitate activitățile corespunzătoare utilizatorilor se deservesc succesiv.

Ambele se referă la creşterea eficienței în utilizarea sistemului sau la îmbunătățirea politicii de deservire a utilizatorului

Programarea concurentă pe sisteme monoprocesor

3. Aplicații care prin natura lor impun o implementare sub formă de activități paralele chiar dacă aceste activități sînt executate, pe baza unei anumite politici, de către un unic procesor

- **programe în timp real** (embedded systems)
  - sisteme dedicate controlului într-o anumită aplicație (procese industriale, automobile, aviație, ***telecomunicații***, echipament casnic, etc.);
  - aplicații care presupun ***furnizarea unui răspuns***, într-un anumit interval de timp, la un impuls exterior;
  - impulsurile exterioare provoacă lansarea unor activități care, cel puțin conceptual, se desfăşoară în paralele cu activități lansate ca răspuns la alte impulsuri.

Maniera în care procesorul se dedică diferitelor activități este rezultatul unei planificări, uneori foarte sofisticate, care trebuie să garanteze furnizarea unui răspuns în timp util.

Sisteme concurente – cu un singur procesor central

- utilizarea eficientă a unei stații de lucru

- implementarea unui sistem de control folosind un PC

- **implementarea unui sistem dedicat bazat pe un microprocesor** (embedded systems)

În sistemele bazate pe un singur procesor introducerea concurenței, deşi poate să îmbunătățească gradul de utilizare global al calculatorului, nu rezolvă accelerarea execuției unui program luat individual (față de cazul în care acest program ar fi fost executat fiindu-i dedicat în exclusivitate sistemul).

Multiprogramare multitasking, programare in timp real, calcul paralel sisteme distribuite

Limitele arhitecturilor monoprocesor tradiționale:

Prelucrarea unei cantități mari de date: calcule numerice din domeniul ştiințelor naturale, simulări, proiectare asistată de calculator, prelucrarea imaginilor, sisteme economice, etc.

Solutia:

O arhitectură nouă, bazate pe activitatea concomitentă a mai multor procesoare, care să asigure în mod efectiv desfăşurarea în paralel a prelucrării datelor.

Multiprogramare multitasking, programare in timp real, calcul paralel sisteme distribuite

Directii de dezvoltare:

- arhitecturi foarte diverse, de la sisteme cu zeci de mii de procesoare foarte simple la sisteme cu un număr redus de procesoare foarte complexe;
- procesoare foarte puternice care, deşi execută un unic flux de instrucțiuni (deci o singură activitate) pun la dispoziția programatorului instrucțiuni foarte puternice care acționează în paralel asupra unei cantități mari de date (procesoare vectoriale sau, într-un sens mai larg, calculatoare bazate pe paralelismul datelor).

Concluzie:

- **dezvoltarea arhitecturilor a luat-o înaintea dezvoltării limbajelor şi a mediilor de programare**

Programarea concurenta

Programe complexe, care să pună în valoare potențialul de performanță al sistemului de calcul şi să permită obținerea rapidă a unei soluții pentru un calcul foarte complex.

DIRECȚII

- “Limbaje paralele" - dialecte C sau Fortran
  - limbaje mai sofisticate orientate pe obiecte
  - limbaje bazate pe concepte noi mai puțin convenționale
- Algoritmi descrisi fără a ține cont de faptul că urmează să fie executați pe un sistem paralel;
  - Compilatoarele disponibile sînt cele care urmează să preia sarcina de a analiza programul şi de a-l transforma, în mod automat, astfel încât să fie executat în mod eficient, punând în valoare potențialul de paralelism al arhitecturii.

??????????

1. În viitor vom avea la dispoziție pe orice PC sisteme de operare cu facilități paralele, şi deci aplicațiile vor trebui să pună în valoare aceste facilități (în vederea exploatării eficiente a calculatorului şi a creşterii gradului de "confort psihic" al utilizatorului).

2. Domeniul programării sistemelor dedicate pentru control, bazate pe PC sau pe sisteme şi mai simple formate din unul sau mai multe microprocesoare, se va dezvolta în mod exploziv în viitorul foarte apropiat.

3. Se constată o tendință de a renunța, pe cît posibil, la calculatoare paralele sofisticate şi de a utiliza în locul lor (acolo unde performanțele o permit) aşa numite "ferme de stații". Cu alte cuvinte, privim o rețea de stații (care trebuie să fie suficient de rapidă!) ca un calculator paralel şi dezvoltăm aplicații care se execută în paralel pe aceste stații. Sunt disponibile medii de programare care facilitează realizarea unor astfel de programe.

?????????????

4. În telecomunicații, sunt necesare echipamente inteligente, ce sunt de fapt nişte **calculatoare "speciale",** sisteme ***multiuser*** şi ***multitasking***, care deseori operează în ***timp real***, şi în care apare frecvent necesitatea ***comunicării inter-proces***, (asigurată prin ***programare concurentă***).

Program concurent: Un program care defineşte acțiuni care se vor desfăşura în mod simultan.

Program paralel: Un program concurent realizat pentru a funcționa pe un sistem cu mai multe procesoare.

Program distribuit: Un program paralel realizat pentru a fi executat într-o rețea de sisteme de calcul autonome.

Noțiuni fundamentale pentru programarea concurentă:

- secțiuni critice;
- excludere mutuală;
- deadlock;
- semafoare…

Dijkstra - Raport tehnic al Universității Tehnice din Eindhoven, Olanda,**1965**.

Aspectele referitoare la programarea concurentă *(**noțiunile de proces, sincronizare, excludere mutuală**)*:

- programatorii de sisteme de operare
- programatorii de aplicații numerice foarte mari – maşini paralele
  - aplicații scrise in Fortran 90 - compilatoare pt. paralelizarea codului

Acest tip de programare nu mai este pentru elite, în curând va deveni accesibil (obligatoriu ?) pentru programatorii obişnuiți

Calculatoare personale multiprocesor

Putere de calcul distribuită (calc. multi procesor→ calc. legate în rețea) la îndemâna tuturor

- sisteme de operare distribuite

- Domenii calde în programarea concurentă
  - Utilizarea sistemelor distribuite
  - Limbajele de programare concurentă????

Limbaje de programare concurentă

- Ada, Linda sau Orca

  - ascund aspecte ***legate de implementare*;**
  - utile când aplicația tratată se potriveşte cu modelul implementat de limbaj.

Limbajele pentru programare paralelă şi/sau distribuită sunt realizate pe scheletul unui limbaj de programare secvențial în care se introduc construcții specifice.

Aceste construcții se referă, în general, la descrierea paralelismului şi la mecanismele de comunicare şi sincronizare.

Sisteme de operare (SO) multitasking

execuția ***multitasking* = *execuția "întrețesută"*** a succesiunilor de instrucțiuni de la mai multe sarcini de calcul

ISTORIC

Sisteme batch - utilizau buffere de memorie şi spooling (citirea înainte de execuție a unui program din memorie) pentru a accelera execuția mai multor programe pe un processor. Bufferele şi spooling-ul permiteau o anumită suprapunere a operațiilor I/O cu programele.

Sisteme ce utilizează tehnici de multiprogramare - Aceste tehnici permiteau întrețeserea doar pe durata în care programul care se executa avea nevoie să aştepte o operație de I/O → ***Întrețesere rudimentară*.**

***Sisteme de operare multitasking* sau cu *time sharing* –** mai multe programe se execută în acelaşi timp, comutarea fiind atât de frecventă încât din punct de vedere al utilizatorului programele se execută simultan.

Sisteme de operare multitasking

Obs

- Utilizatorul ***poate interveni în activitatea sistemului de operare*** sau ale programelor în execuție;
- ***Execuția unui program*** va necesita un ***timp global superior timpului*** necesar unei execuții pe un sistem secvențial.
- ***Execuția mai multor programe*** în sistem multitasking va necesita ***global mai puțin timp*** datorită execuției întrețesute care reduce semnificativ durata în care procesorul e neutilizat.

Unitatea elementară de program se numeşte ***task*** (operațiune, sarcină de calcul) sau ***proces*** şi reprezintă :

- cea mai mică unitate de program ce se poate executa independent de către echipamentul de calcul.

-o secvență de instrucțiuni, logic independentă, dintr-un program, care poate fi executată simultan cu alte părți ale aceluiaşi program.

Sisteme de operare multitasking

Program versus proces

Noțiunea de proces - *introdusă de modelul von Neumann*:

= Un program secvențial în execuție - el este compus (din punctul de vedere al sistemului de operare) din codul programului care se execută, datele şi resursele ocupate, şi este descris de anumite informații de stare.

Diferența dintre un program şi proces:

- **Procesul** este o **entitate dinamică**, pe când **programul** este o **entitate statică** ce poate fi regăsită pe un mediu de stocare.

- Un program în execuție poate să creeze procese care vor dura un anumit interval de timp şi vor dispărea după aceea.

- Spațiul de adrese al unui proces este inviolabil, fiind inaccesibil (în condiții normale) de către un alt proces.

Sisteme de operare multitasking

M. Conway postulează (1963) un set de trei **funcții, de bază**, care permit realizarea unui sistem de operare multitasking:***FORK*** , ***JOIN, QUIT*** .

Apelul şi efectul funcției ***FORK***

Se apelează având ca **argument o etichetă** şi determină **apariția a două procese concurente.**

- ***FORK label***

  Crează un nou proces definit de procedura care se execută la momentul curent. Acest proces începe cu instrucțiunea referită de label. Procesul inițial îşi urmează cursul normal. Odată noul proces creat, cele două procese coexistă şi se execută concurent (sau paralel).

Apelul şi efectul funcției ***JOIN***

Funcția ***JOIN*** are doi parametrii de intrare : un nr. întreg şi o etichetă.

- ***JOIN nr. label***

  Pentru a avea efect această instrucțiune trebuie să fie executată de un număr ***nr*** de procese. Cît timp numărul de execuții este inferior lui ***nr***, după întâlnirea ei se va continua cu execuția instrucțiunii imediat următoare. Dacă a fost executată de ***nr*** ori atunci procesorul va executa instrucțiunea având eticheta ***label***.

Sisteme de operare multitasking

Apelul şi efectul funcției ***QUIT***

Se apelează fără parametru.

- ***QUIT***

  Este folosită de proces pentru terminare. Procesul este distrus, şi toate resursele ocupate sunt eliberate.

Alte caracteristici:

Aplicațiile sunt ***divizate în mod explicit*** (de către programator) sau ***implicit*** (de către compilator) în taskuri care se vor executa în ***mod concurent*** dar care trebuie să poată:

- ***interacționa*** **între ele:**
  - **să-şi transmită reciproc informațiile;**
  - **să-şi sincronizeze execuțiile;**

Sisteme de operare multitasking

Obs

1. Divizarea ansamblului de programe în taskuri (sau operații/procese/ sarcini elementare) simplifică proiectarea şi elaborarea lui, uşurând totodată realizarea cerințelor în ceea ce priveşte timpii de răspuns.

2. În general, ordinea strictă de execuție a taskurilor ce compun programele din aceste aplicații este puțin previzibilă, acestea fiind legate de timpul sistem (numite şi aplicații *time driven*), de producerea unor anumite evenimente (*event - driven*), sau de sosirea unor *semnale* sau *mesaje*.

Reprezentarea în memorie a unui proces

Indiferent de platforma (***sistemul de operare***) ***reprezentarea în memorie a unui proces***, implică utilizarea mai multor zone de memorie:

- Contextul procesului
- Codul programului
- Zona datelor globale
- Zona heap
- Zona stivei

Reprezentarea în memorie a unui proces

1. **Contextul procesului** - **informatii referitoare la localizarea în memoria internă a procesului şi la starea de execuţiei a acestuia**:

- ***Legături exterioare*** cu platforma (sistemul de operare): numele procesului, directorul curent în structura de directori, variabilele de mediu etc;
- ***Pointeri*** spre adresele de început ale zonelor de cod, date stivă şi heap şi, eventual, lungimile acestor zone;
- ***Starea curentă a execuţiei procesului***: contorul de program (notat PC -program counter) ce indică în zona cod următoarea instrucţiune maşină de executat, deasemenea pointerul spre vârful stivei (notat SP - stack pointer);
- ***Zone de salvare a regiştrilor generali***, de stare a sistemului de întreruperi etc.

Reprezentarea în memorie a unui proces

2. **Codul programului** - **conţine instrucţiunile maşină care dirijează funcţionarea procesului**

   De regulă, conţinutul acestei zone este stabilit încă din faza de compilare.

   Programatorul descrie programul într-un limbaj de programare de nivel înalt (asamblare).

   Textul sursă al programului este supus procesului de compilare care generează o secvenţă de instrucţiuni maşină echivalentă cu descrierea din program.

   Conţinutul acestei zone este folosit de procesor pentru a-şi încărca rând pe rând instrucţiunile de executat. Registrul PC indică, în fiecare moment, locul unde a ajuns execuţia.

3. **Zona datelor globale** - **conţine constantele şi variabilele vizibile de către toate instrucţiunile programului.**

   Constantele şi o parte dintre variabile primesc valori încă din faza de compilare. Aceste valori iniţiale sunt încărcate în locaţiile de reprezentare din zona datelor globale în momentul încărcării programului în memorie.

Reprezentarea în memorie a unui proces

4. Zona heap - zona variabilelor dinamice - găzduieşte spații de memorare a unor variabile a căror durată de viață este fixată de către programator.

Crearea (operația *new*) unei astfel de variabile înseamnă rezervarea în heap a unui şir de octeți necesar reprezentării ei şi întoarcerea unui pointer / referințe spre începutul acestui şir. Prin intermediul referinței se poate utiliza în scriere şi/sau citire această variabilă până în momentul *distrugerii* ei (operație *destroy*, *dispose* etc.).

Distrugerea înseamnă eliberarea şirului de octeți rezervat la creare pentru reprezentarea variabilei. În urma distrugerii, octeții eliberați sunt plasați în lista de spații libere a zonei heap.

Reprezentarea în memorie a unui proces

5. Zona stivă

În momentul în care programul apelelează o procedură sau o funcție, se depun în vârful stivei o serie de informații:

parametrii transmişi de programul apelator către procedură sau funcție:

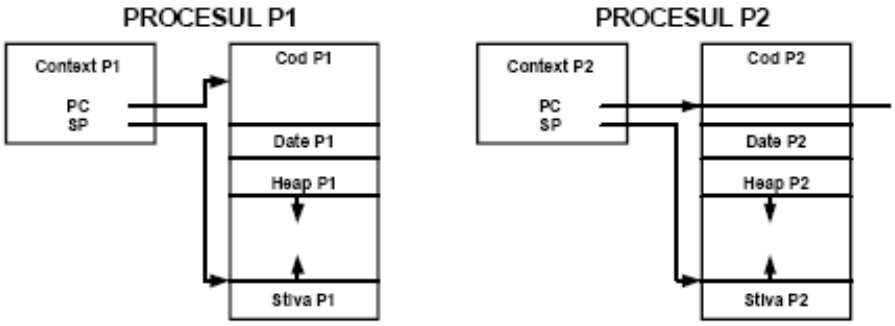
- adresa de revenire la programul apelator

- spațiile de memorie necesare reprezentării variabilelor locale declarate şi utilizate în interiorul procedurii sau funcției etc.

După ce procedura sau funcția îşi încheie activitatea, spațiul din vârful stivei ocupat la momentul apelului este eliberat.

În cele mai multe cazuri, există o stivă unică pentru fiecare proces. Există însă platforme, DOS este un exemplu, care folosesc mai multe stive simultan: una rezervată numai pentru proces, alta (altele) pentru apelurile sistem.

Reprezentarea în memorie a unui proces



- **procesele sunt aşadar entităţi de sine stătătoare**
- **faptul că în condiţii normale spaţiul de adrese al unui proces trebuie să fie inviolabil produce o serie de probleme**
- **anumite procese trebuie să comunice între ele (procese cooperante).**

Principii de proiectare care au stat la baza UNIX

Sistem de operare interactiv, cu divizarea timpului

Interfaţa cu utilizatorul cât mai simpla, uşor de extins şi înlocuit

Structura fişierului să nu mai fie impusă de sistemul de operare – fisierul-flux (şir de octeţi)

Perifericele tratate cât mai uniform posibil fiind incorporate în sistemul de fişiere general – Operatiile de I/O sunt încapsulate în driverele perifericelor care prezintă o interfaţă uniformă pentru programatori.

Sistem multi-utilizator şi multiproces

- mai multi utilizatori simultan în sistem

- mai multe programe ale utilizatorilor active

- un utilizator poate avea simultan în executie mai multe procese

Planificarea timpului procesorului între procesele active se face printr-un algoritm simplu bazat pe priorităţi.

Gestionarea memorie – memorie virtuală