V. Sistemul de operare

Rolul sistemului de operare (1)

- program cu rol de gestiune a sistemului de calcul
- face legătura între hardware şi aplicaţii
- pune la dispoziţia aplicaţiilor diferite servicii
- supraveghează buna funcţionare a aplicaţiilor
  - poate interveni în cazurile de eroare

Rolul sistemului de operare (2)

- pentru a-şi îndeplini sarcinile, are nevoie de suport hardware
  - cel mai important - sistemul de întreruperi
- componente principale
  - nucleu (*kernel*)
  - drivere

V.1. Nucleul sistemului de operare

Nucleul sistemului de operare

- în mare măsură independent de structura hardware pe care rulează
- "creierul" sistemului de operare
- gestiunea resurselor calculatorului - hardware şi software
  - funcţionare corectă
  - alocare echitabilă între aplicaţii

Moduri de lucru ale procesorului

- utilizator (*user mode*)
  - restricţionat
  - accesul la memorie - numai anumite zone
  - accesul la periferice - interzis
- nucleu (*kernel mode*)
  - fără restricţii

Modul de rulare al programelor

- sistemul de operare - în mod nucleu
  - poate efectua orice operaţie
- aplicaţiile - în mod utilizator
  - nu pot realiza anumite acţiuni
  - apelează la nucleu

Trecerea între cele două moduri

- prin sistemul de întreruperi
- utilizator → nucleu
  - apel întrerupere software
  - generare excepţie (eroare)
- nucleu → utilizator
  - revenire din rutina de tratare a întreruperii

Consecinţe

- codul unei aplicaţii nu poate rula în modul nucleu
- avantaj: erorile unei aplicaţii nu afectează alte programe
  - aplicaţii
  - sistemul de operare
- dezavantaj: pierdere de performanţă

Structura nucleului

- nu este un program unitar
- colecţie de rutine care cooperează
- funcţii principale
  - gestiunea proceselor
  - gestiunea memoriei
  - gestiunea sistemelor de fişiere
  - comunicarea între procese

V.2. Apeluri sistem

Apeluri sistem (*system calls*)

- cereri adresate nucleului de către aplicaţii
- acţiuni pe care aplicaţiile nu le pot executa singure
  - numai în modul nucleu al procesorului
  - motiv - siguranţa sistemului
- realizate prin întreruperi software
- similar apelurilor de funcţii

Etapele unui apel sistem (1)

1. programul depune parametrii apelului sistem într-o anumită zonă de memorie
2. se generează o întrerupere software
   - procesorul trece în modul nucleu
3. se identifică serviciul cerut şi se apelează rutina corespunzătoare

Etapele unui apel sistem (2)

4. rutina preia parametrii apelului şi îi verifică
   – dacă sunt erori - apelul eşuează
5. dacă nu sunt erori - realizează acţiunea cerută
6. terminarea rutinei - rezultatele obţinute sunt depuse într-o zonă de memorie accesibilă aplicaţiei care a făcut apelul

Etapele unui apel sistem (3)

7. procesorul revine în mod utilizator
8. se reia execuţia programului din punctul în care a fost întrerupt
   - se utilizează informaţiile memorate în acest scop la apariţia întreruperii
9. programul poate prelua rezultatele apelului din zona în care au fost depuse

Apeluri sistem - concluzii

- comunicare între aplicaţie şi nucleu
  - depunere parametri
  - preluare rezultate
- acţiunile critice sunt realizate într-un mod sigur
- mari consumatoare de timp
- apeluri cât mai rare - lucru cu buffere

Cum folosim bufferele

Exemplu - funcţia *printf*

- formatează textul, apoi îl trimite spre ecran
- nu are acces direct la hardware
  - se foloseşte de un apel sistem
  - *write* (în Linux)
- de fapt, *printf* depune textul formatat într-o zonă proprie de memorie (buffer)
  - doar când bufferul e plin se face un apel sistem

V.3. Drivere

Ce sunt driverele?

- module de program care gestionează comunicarea cu perifericele
  - specializate - câte un driver pentru fiecare periferic
- parte a sistemului de operare
  - acces direct la hardware
  - se execută în modul nucleu al procesorului

Utilizare

- nu fac parte din nucleu
  - dar se află sub comanda nucleului
- folosite de rutinele de tratare ale întreruperilor hardware
- înlocuire periferic → înlocuire driver
  - organizare modulară
  - nu trebuie reinstalat tot sistemul de operare

V.4. Gestiunea proceselor

Procese (1)

- se pot lansa în execuţie mai multe programe în acelaşi timp (*multitasking*)
- paralelismul nu este real
  - doar dacă sistemul are mai multe procesoare
  - altfel - concurenţă
- un program se poate împărţi în mai multe secvenţe de instrucţiuni - *procese*
  - se pot executa paralel sau concurent

Procese (2)

- sistemul de operare lucrează cu procese
  - nu cu programe
- la lansare, un program constă dintr-un singur proces
  - poate crea alte procese
  - care pot crea alte procese ş.a.m.d.
- un procesor poate executa la un moment dat instrucţiunile unui singur proces

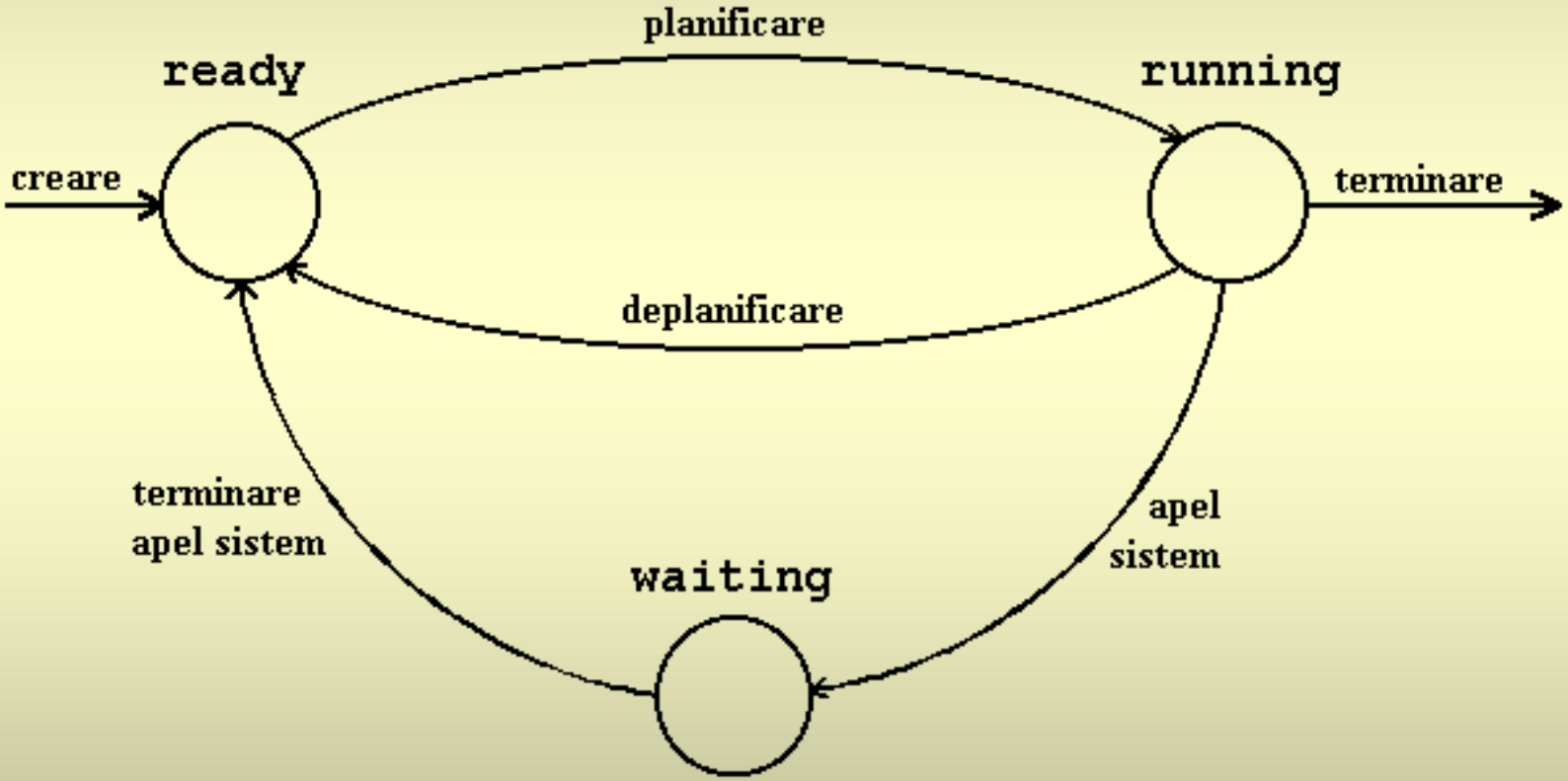
Procese (3)

- fiecare proces are propriile zone de memorie (cod, date, stivă, ...)
  - separate de ale celorlalte procese
- la crearea unui nou proces, i se alocă spațiu de memorie
- la terminarea unui proces, memoria ocupată este eliberată
  - chiar dacă programul în ansamblul său continuă

Stările unui proces (1)

- în execuţie (*running*)
  - instrucţiunile sale sunt executate de procesor
- gata de execuţie (*ready*)
  - aşteaptă să fie executat de procesor
- în aşteptare (*waiting*)
  - aşteaptă terminarea unui apel sistem
  - nu concurează momentan pentru planificarea la procesor

Stările unui proces (2)



Stările unui proces (3)

- procesul aflat în execuţie părăseşte această stare
  - la terminarea sa
    - normală sau în urma unei erori
  - la efectuarea unui apel sistem (→ *waiting*)
  - când instrucţiunile sale au fost executate un timp suficient de lung şi este rândul altui proces să fie executat (deplanificare)

Forme de multitasking

- non-preemptiv
  - nu permite deplanificarea unui proces
  - un proces poate fi scos din execuţie doar în celelalte situaţii
  - dezavantaj - erorile de programare pot bloca procesele (ex. buclă infinită)
- preemptiv

Deplanificarea

- cum ştie sistemul de operare cât timp s-a executat un proces?
- este necesară o formă de măsurare a timpului
- ceasul de timp real
  - dispozitiv periferic
  - generează cereri de întrerupere la intervale regulate de timp

Fire de execuție (1)

- un proces se poate împărți la rândul său în mai multe fire de execuție (*threads*)
  - uzual, un fir constă în execuția unei funcții din codul procesului
- firele de execuție partajează resursele procesului (memorie, fișiere deschise etc.)
  - comunicare mai simplă - prin variabile globale
  - risc mai mare de interferențe nedorite

Fire de execuție (2)

- când un proces este planificat la procesor, se va executa unul dintre firele sale
  - deci este necesară și aici o formă de planificare
- cine realizează planificarea?
- variante
  - sistemul de operare (mai rar)
  - aplicația - prin funcții de bibliotecă specializate

V.5. Gestiunea memoriei

Gestiunea memoriei

Funcţii

- alocarea zonelor de memorie către aplicaţii
- prevenirea interferenţelor între aplicaţii
- detectarea şi oprirea acceselor incorecte

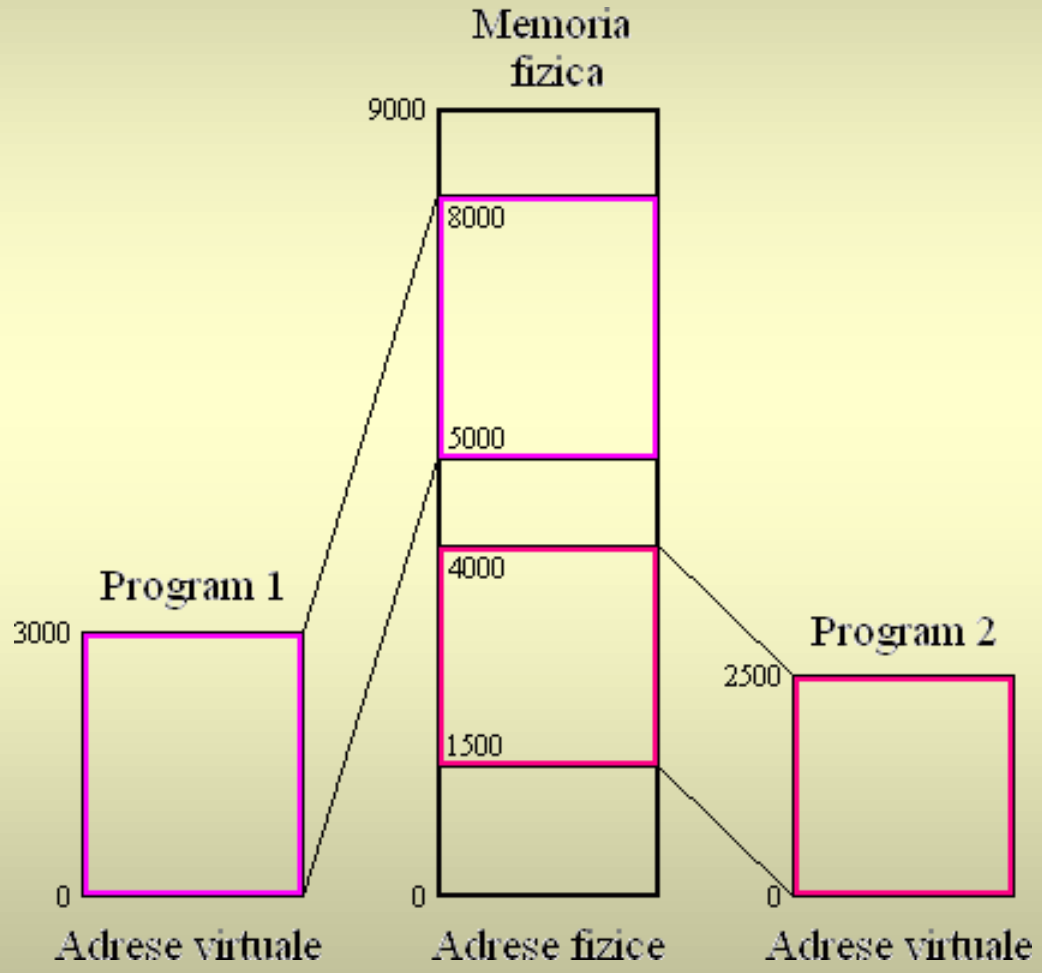
Problema fundamentală

- mai multe aplicaţii → zone de memorie disjuncte
- fiecare aplicaţie → anumite zone de memorie; care sunt aceste zone?
  - depind de ocuparea memoriei la acel moment
  - nu pot fi cunoscute la compilare

Soluţia

- două tipuri de adrese
  - virtuale - aplicaţia crede că le accesează
  - fizice - sunt accesate în realitate
- corespondenţa între adresele virtuale şi cele fizice - gestionată de sistemul de operare

Adrese virtuale şi fizice (1)



Adrese virtuale şi fizice (2)

Gestionarea adreselor fizice şi virtuale

- 2 metode diferite
  - segmentare
  - paginare
- pot fi folosite şi împreună
- componentă dedicată a procesorului - MMU (*Memory Management Unit*)

V.5.1. Segmentarea memoriei

Principiul de bază (1)

- segment - zonă continuă de memorie
- conţine informaţii de acelaşi tip (cod, date etc.)
- vizibil programatorului
- adresa unei locaţii - formată din 2 părţi
  - adresa de început a segmentului
  - deplasamentul în cadrul segmentului (*offset*)

Principiul de bază (2)

- la rulări diferite ale programului, segmentele încep la adrese diferite
- efectul asupra adreselor locaţiilor
  - adresa de început a segmentului trebuie actualizată
  - deplasamentul - nemodificat
- problema este rezolvată numai parţial
  - dorim ca adresa să nu fie modificată deloc

Descriptori (1)

- descriptor de segment - structură de date pentru gestionarea unui segment
- informaţii reţinute
  - adresa de început
  - dimensiunea
  - drepturi de acces
  - etc.

Descriptori (2)

- descriptorii - plasaţi într-un tabel
- accesul la un segment - pe baza indicelui în tabelul de descriptori (selector)
- adresa virtuală - 2 componente
  - indicele în tabelul de descriptori
  - deplasamentul în cadrul segmentului
- adresa fizică = adresa de început a segmentului + deplasamentul

Descriptori (3)

- la rulări diferite ale programului, segmentele încep la adrese diferite
- efectul asupra adreselor locaţiilor
  - nici unul
  - trebuie modificată doar adresa de început a segmentului în descriptor
  - o singură dată (la încărcarea segmentului în memorie)
  - sarcina sistemului de operare

Accesul la memorie (1)

- programul precizează adresa virtuală
- identificare descriptor segment
- verificare drepturi acces
  - drepturi insuficiente - generare excepţie
- verificare deplasament
  - dacă deplasamentul depăşeşte dimensiunea segmentului - generare excepţie

Accesul la memorie (2)

- dacă s-a produs o eroare la paşii anteriori
  - rutina de tratare a excepţiei termină programul
- dacă nu s-a produs nici o eroare
  - calcul adresă fizică (adresă început segment + deplasament)
  - acces la adresa calculată

Exemplificare (1)

Tabel descriptori (simplificat)

Indice	Adresa început	Dimensiune
0	65000	43000
1	211000	15500
2	20000	30000
3	155000	49000
4	250000	35000

Exemplificare (2)

Exemplu 1:

```
mov byte ptr ds:[eax],25
```

- ds = 3 → adresa început segment = 155000
- eax = 27348 < 49000
  - deplasament valid (nu depăşeşte dimensiunea segmentului)
- adresa fizică: 155000 + 27348 = 182348

Exemplificare (3)

Exemplu 2:

```
add dword ptr ss:[ebp],4
```

- ss = 1 → adresa început segment = 211000
- ebp = 19370 > 15500
  - deplasament invalid (depăşeşte dimensiunea segmentului)
  - eroare → generare excepţie

Cazul Intel (1)

3 tabele de descriptori

- global (GDT - *Global Descriptor Table*)
  - accesibil tuturor proceselor
- local (LDT - *Local Descriptor Table*)
  - specific fiecărui proces
- de întreruperi (IDT - *Interrupt Descriptor Table*)
  - nu este direct accesibil aplicaţiilor

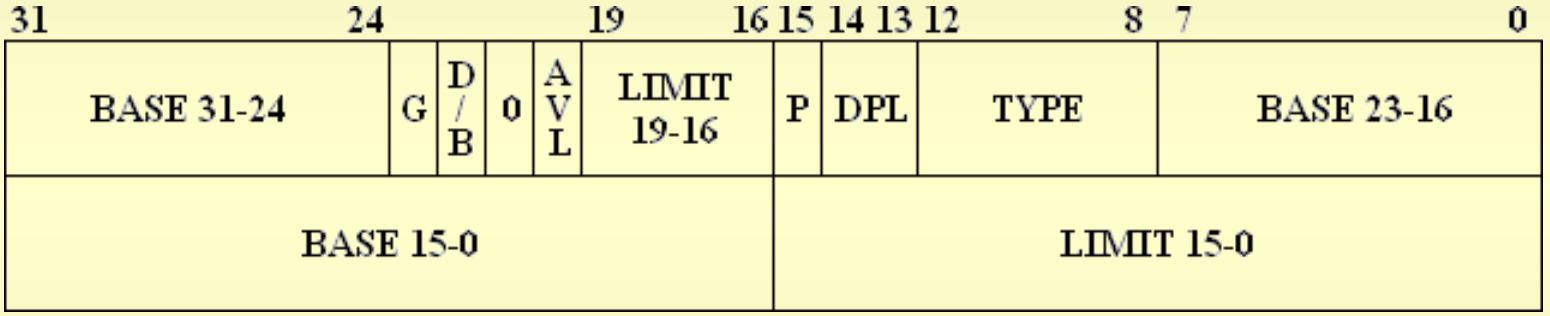
Cazul Intel (2)

Segmentele - accesate cu ajutorul selectorilor

Structura unui selector (16 biţi)

- primii 13 biţi - indicele în tabelul de descriptori
  - maximum 8192 descriptori/tabel
- 1 bit - tabelul folosit (global/local)
- ultimii 2 biţi - nivelul de privilegii
  - 0 - cel mai înalt, 3 - cel mai scăzut

Cazul Intel (3)



Cazul Intel (4)

- intervin nivelele de privilegii a 3 entităţi
  1. CPL (*Current Privilege Level*)
  – al procesului - reţinut de procesor
  2. RPL (*Requested Privilege Level*)
  – cel solicitat - preluat din selector
  3. DPL (*Descriptor Privilege Level*)
  – cel al segmentului accesat - din descriptor

Cazul Intel (5)

- relaţiile dintre aceste nivele de privilegii decid dacă se poate realiza accesul
- condiţia pentru realizarea accesului: CPL<=DPL şi RPL<=DPL (simultan)
- orice altă situaţie indică o încercare de acces la un nivel prea înalt
  - generare excepţie