Sisteme de Operare 1 - Curs 10

Curs tinut in 2012-2013 *de catre* lector dr. Sanda-Maria Dragos

cap 11 carte
pp. 297 - 310

Componenta hardware a sistemelor de calcul necesita existenta unui sistem de operare care sa poata gestiona resursele hardware, concomitent cu asistarea utilizatorului pe timpul pregatirii si lansarii in executie a lucrarilor sale. Sistemul de operare contine programe care coordoneaza si controleaza resursele hardware (memoria, CPU, canalele, dispozitivele periferice) si informatiile (progame, date), asigurand astfel utilizarea eficienta a resurselor. Din acest punct de vedere, functiile unui sistem de operare au fost grupate in 4 categorii:

1. **Gestiunea informatiei**
    » Sistemul de fisiere (organizarea, regasirea, utilizarea informatiei)
    » rutine de access
    » alocare de resurse prin deschiderea fisierului(OPEN)
    » eliberarea de resurse prin inchiderea fisierului (CLOSE)
2. **Gestiunea dispozitivelor periferice**
    » operarea dispozitivelor periferice (*I/O Traffic Controller*)
    » alocarea dispozitivelor periferice intr-un mod eficient (*I/O Scheduler*)
    » alocarea dispozitivului periferic si initierea operatiei de intrare/iesire
    » eliberearea dispozitivului periferic la terminarea operatiei de intrare/iesire
3. **Gestiunea procesorului**
    » operarea procesorului (*CPU Controller*)
    » alocarea procesorului la procese (*CPU scheduler*)
4. **Gestiunea memoriei**
    » alocarea memoriei pentru programele din sistem
    » multiprogramarea
    » metode de access si protectie
    » eliberarea zonelor de memorie alocate

Gestiunea dispozitivelor periferice

Gestiunea dispozitivelor periferice este foarte complexa datorita diversitatii si multitudinii tipurilor de periferice. Exista mii de dispozitive periferice, fiecare avand interfete diferite si viteze de acces diferite. De aceea sistemele de operare dedica o mare parte din cod gestiunii acestor periferice. Unele dispozitive pot fi de tip bloc (e.g., hardisk-ul, CD-urile, DVD-urile) iar altele de tip caracter (e.g., tastatura, mouse-ul). Unele dispozitive au acces secvential (e.g., banda magnetica, tastatura) iar altele au acces aleator (e.g., hardisk-ul, CD-urile).

slide 2

slide 3

Apoi mai exista problema verificarii daca un dispozitiv periferic este disponibil. De ex. la apasarea unei taste. Sistemul poate verifica la intervale regulate de timp daca s-a apasat vreo tasta, sau apasarea unei taste poate genera un semnal. In acest caz specific, actiunile generate de tastatura sunt rare si imprevizibile deci generarea unui semnal e mai potrivita.

Un alt exemplu este primirea pachetelor pe placa de retea. Datorita faptului ca trasmiterea pachetelor in retea e "bursty" (sub forma de ingramadiri), primul pachet dintr-un "burst" genereaza o intrerupere dupa care sistemul verifica la intervale predefinite de timp (polling) daca au mai venit si alte pachete. Nu se poate ca toate pachetele sa genereze cate o intrerupere datorita faptului ca sistemul nu mai apuca sa prelucheze pachetele.

De asemenea dispozitivele pot fi foarte rapide (placa de retea) sau foarte incete (tastatura). Si cum se vede in schema, cu cat sunt mai rapide perifericele, cu atat sunt puse mai aproape de procesor. De exemplu memoria se afla pozitionata foarte aproape de procesor. Alte periferice, care sunt mai lente se afla mai departe de procesor (de ex. coltul dreapta jos slide-ul 2).

Unul dintre scopurile principale ale gestiunii dispozitivelor periferice este acela de a oferii o interfata uniforma de lucru cu toate perifericele oricat de diferite ar fi acestea. De exemplu o cheie USB daca o atasam unui calculator o putem accesa in acelasi mod ca si un alt disc (folosind un sistem de gestiune de fisiere ca de ex: total commander). Sistemele de operare din familia Unix vad toate perifericele ca si fisiere.

Driver-ele ofera interfete standard pentru toate perifericele. Astfel 3 interfete standard:

» Periferice de tip **BLOC** (e.g., unitati de disc, banda magnetica, DVD-ROM)
  » trasferul de informatie se realizeaza la nivel de bloc de date
  » comenzile folosite sunt: *open(), read(), write(), seek()*
  » pot contine date in forma bruta sau pot fi organizate sub forma de sisteme de fisire

» Periferice de tip **CARACTER** (e.g., tastatura, mouse-ul, porturile seriale si unele periferice USB)

» unitatea de trasferul de informatie este caracterul

» comenzile folosite sunt: *get(), put()*

» librarii de functii ofera posibilitatea de a lucra si cu linii de caractere

» Periferice de tip **RETEA** (e.g., Ethernet, Wireless, Bluetooth)

» Unix si Windows includ interfata *socket*prin care se realizeaza comunicarea efectiva

slide 4

Subsistemul I/O al nucleului

Tehnica zonelor tampon temporare (buffering, caching)

» la comunicarea intre periferice datele sunt stocate in memorie

» aceasta tehnica se foloseste pentru a solutiona diferentele de viteze de acces dintre diferite periferice

» sau pentru a solutiona diferentele de dimensiune a datelor transferate

slide 5

Canalul de I/O

» aceasta componenta faciliteaza comunicarea intre procesorul central (CPU) si periferice

» vom discuta modalitati de interactiune dintre CPU si canalul I/O

Elemente specifice lucrului cu discul

» hardiscul fiind o componenta mai complexa si importanta, vom analiza elemente specifice intrarile/iesirilor pentru astfel de dispozitive

Tehnica zonelor tampon temporare

Folosita in schimbul de informatii atat intre memoria interna si periferice, cat si intre diverse nivele ale memoriei (schimb intre memoria cache si memoria interna sau intre cea interna si cea secundara).

slide 6

In functie de context se mai numeste: *metoda zonelor tampon multiple, cache-ingul paginilor web, pool de pagini recent utilizate, buffering,* etc.

--- povestea cu apa ---

» Producator, consumator (depinde de context)
» Obiectul care se produce/consuma: o zona de memorie, un spatiu pe disc, etc.
» Buffer-ul: ansamblu a n zone tampon (pool de zone tampon)
» Trecerea zonelor tampon de la producator la consumator: FIFO

Aplicatie 1: Acces bufferizat la un fisier

Unitatea de schimb intre disc si memoria interna este blocul (sectorul)

slide 7

- un pool de 2 sau mai multe zone tampon in mem. interna (pentru optimizarea accesului la fisier)

Tratarea poolului se face circular.

» La citire din fisier
» Producator: perifericul extern
» Consumator: procesul care consulta bufferele
» La scriere in fisier
» Producator: procesul care creeaza informatiile
» Consumator: perifericul extern

Aplicatie 2: Intretinerea unui cache disc

Majoritatea sistemelor de operare intretin un pool cu copii ale celor mai recente sectoare de disc accesate. Acest pool se numeste *cache disc*.

In momentul in care se solicita accesul la un sector disc, se cauta mai intai o copie a lui in cache-ul disc. Daca se gaseste copia, atunci sistemul este servit cu copia si accesul se considera incheiat. Daca se doreste modificarea continutului unui sector, acesta se face tot in cache si se marcheaza sectorul ca fiind modificat.

Cache-ul are implementata o politica de evacuare din cache atunci cand nu mai este spatiu, politica similara cu cele de inlocuire a paginilor in memoria virtuala (NRU, LRU, FIFO, etc).

slide 8

Aplicatie 3: Intretinerea unui cache Web

Navigatoarele Web intretin, dupa acelasi principiu, pool-uri cu paginile web cele mai recent folosite. Mecanismul este similar cu cel al cache-urilor de discuri, numai ca aici o zona tampon este de fapt un fisier cu pagina web dapozitata in cache.

Producatorii sunt aici furnizorii de pagini web din Internet, iar consumatorul este browserul.

slide 9

Aplicatie 4: Conectarea pipe intre doua comenzi

Fiecare comanda a unui sistem de operare are un fisier standard de intrare si unul standard de iesire. Conexiunea pipe insemna conectarea unei iesiri standard catre intrarea standard a comenzii urmatoare:

```
comanda1 | comanda2
```

Prin acest gen de conexiune se creeaza, de regula, un pool de zone tampon, unde *comanda1* e producatorul iar *comanda2* e consumatorul.

slide 10

slide 11

Aplicatie 5: Memoria cache

Memoria cache fiind parte din memoria interna, se comporta ca si un pool de copii ale unor locatii din memoria interna.

In aceasta situatie atat producatorul cat si consumatorul sunt memoria interna.

Canalul de I/O

Canalul I/O reprezinta comunicarea intre procesorul central (CPU) si periferice. Exista trei moduri fundamentale de interactiune intre procesor (CPU) si periferice.

» **Asteptare reciproca** - sincronizare prin blocare
  » la cerearea unei date (e.g., apelul sistem *read()*) procesul este pus in stare de asteptare pana cand informatia este gata
  » la scrierea unei informatii (e.g., apelul sistem *write()*) procesul este pus in stare de asteptare pana cand perifericul este pregatit pentru respectiva informatie

» **Poolingul efectuat de catre procesorul central** - metoda asincrona (adica, trasferul de date se realizeaza printr-un buffer, iar notificarea se realizeaza la umplerea bufferului)
  » aceasta notificare se produce prin verificari la intervale egale de timp ale

procesorului sa determine daca datele sunt disponibile.

» *pro:* nu incarca asa de mult sistemul

» *con:* nu e avantajos pentru evenimente rare si imprevizibile pentru ca se fac prea mult verificari (pollings) fara nici un rezultat

slide 12

» **Intrerupere lansata de catre periferic** - metoda asincrona (adica, trasferul de date se realizeaza printr-un buffer, iar notificarea se realizeaza la umplerea bufferului)

» aceasta notificare se produce prin generarea unui semnal din partea perifericului care atentioneaza CPU-ul ca datele sunt disponibile

» *pro:* in cazul evenimentelor mai rare si imprevizibile

» *con:* o intrerupere incarca sistemul (trebuie salvat tot contextul, tratata intreruperea, iar apoi refacut contextul)

Canalul de intrare-iesire este un procesor specializat de operatii de I/O care poate functiona in paralel cu procesorul central.

slide 13

» Fiecare periferic contine o zona Tampon proprie, capabila sa pastreze o inregistrare (o linie de imprimanta, un sector de disc, etc).

» Procesorul, printr-o rutina de I/O, *in cazul scrierii*, pune din memorie informatii in aceasta zona tampon, iar *in cazul citirii*, preia informatii din ea si le pune in memorie.

» Dipozitivul periferic, *in cazul scrierii*, ia informatiile din zona tampon proprie si le pune pe suport, iar *in cazul citirii*, ia informatiile de pe suport si le pune in zona tampon proprie.

» Viteza cu care lucreaza procesorul este mult mai mare decat viteza dispozitivului periferic.

Elemente specifice lucrului cu discul

Planificarea accesului la discul magnetic

Elementul de adresare pe un HDD este *sectorul*. Totusi, trasferul de date la nivel de HDD se face prin *blocuri* de date. Aceste blocuri reprezinta grupari de sectoare.

Accesarea unui bloc de date se poate face indirect. Pentru a accesa un bloc de date, unitatea de disc actioneaza in 3 etape:

» Pozitionarea bratului pe cilindrul dorit
» Rotirea discului pana cand sectorul trece prin fata capului de citire/scriere
» Schimbul propriu-zis de informatie

slide 14

slide 15

Planificarea accesului la discul magnetic. Metode de optimizare.

- optimizarea in selectarea *sectoarelor*:

» *Reducerea accesului la sectoarele vecine*
» *Reducerea asteptarii rotatiei*

- optimizarea pozitionarii *bratului cu capetele de citire/scriere*:

» *Reducerea timpului de pozitionare*

Exista asadar mai multi timpi in care se face accesul la HDD, si anume:

» timpul de **cautare** a piste/cilindrului (8-12 ms)
» timpul de **rotire** pana cand capetele de citire se afla deasupra sectorului dorit (8-16 ms per rotire - in medie e nevoie de o jumatate de rotire ca sa ajung la sectorul cautat)
» timpul de **transfer** efectiv al datelor citite/scrise (2-15 MB/s)

Reducerea accesului la sectoarele vecine

In cazul in care cererile nostre se refera la sectoare vecine, datorita timpului de transfer, capul de citire trece peste inceputul sectorului vecin. De aceea, s-a propus ca sectoarele "vecine" sa se numeroteze cu intretesere. Astfel, spre deosebire de primul caz in care se pot accesa toate cele 8 sectoare doar prin o rotiri de disc, cu intretesere se citesc toate sectoarele doar din 2 parcurgeri. Pentru cazurile in care timpul de trasfer este mai mare decat timpul de trecere peste un sector, s-

a propus intreteserea cu factor 2. In cazul acesta discul se roteste de 3 ori pentru citirea tuturor sectoarelor.

slide 16

Reducerea asteptarii rotatiei

In cazul in care se cere o secventa de sectoare, aceasta secventa se va optimiza astfel incat citirea lor sa se realizeze cu cat mai putine rotiri.

functioneaza dupa regula "cel ce asteapta cel mai putin va fi servit primul".

In cazul de fata, cele tre cereri vor fi servite dintr-o singura rotire a discului.

slide 17

Reducerea timpului de pozitionare

» FCFS - desi usor de implementat, e cel mai putin optim algoritm din punct vedere al timpilor de servire

» SSTF - poate amana indefinit unele cereri

» SCAN - unele cereri pot astepta parcurgerea de 2 ori a tuturor cilindrilor; cilindrii din mijloc sunt mai bine deserviti

slide 18

» C-SCAN - e un algorim des folosit de catre sistemele de operare Unix