Universitatea Politehnica Bucuresti
Facultatea de Electronica si Telecomunicatii

GESTIUNE DE INTRARI/IESIRI

Student : Miu Madalina Mihaela

Grupa : 431 A

Cuprins

Intrari/Iesiri

Printre principalele functii al unui sistem de operare este aceea de a controla dispozitivele de Intrare/Iesire ale calculatorului, care trebuie sa elibereze comenzi dispozitivelor, sa raspunda intreruperilor si sa se ocupe de erori. Pe langa aceste functii, sistemul de operare trebuie sa implementeze o interfata usor de folosit intre dispozitive si restul sistemului.

Sistemul de Intrari/Iesiri din punct de vedere software va fi prezentat din punct de vedere al structurarii nivelelor, fiecare nivel avand rolul specific.

1. Nivelele Software ale sistemului de Intrari/Iesiri

Sistemul de Intrari/Iesiri, din punct de vedere software, este organziat in patru nivele (Fig 1), fiecare nivel avand o functie si o interfata de comunicare cu celelalte nivele bine definite. De la sistem la sistem difera functionalitatile si interfetele. Nivelele analizate in continuare nu sunt specifice nici unei masini in mod particular.

Spatiul utilizatorului in software-ul de Intrari/Iesiri
Sistem de operare software independent de dispozitiv
Drivere de dispozitive
Stivuitori de intreruperi
Hardware

Fig. 1 - Nivelele analizate

1.1 Stivuitorii de intreruperi

Sistemele de Intrare/Iesire prezinta intreruperi, care sunt neplacute pentru majoritatea si nu pot fi evitate. Este necesar ca ele sa fie ascunse, astfel incat o mica parte a sistemului de operare sa stie de existenta lor.

Cand se activeaza o intrerupere, atunci procedura de intreupere actioneaza in sensul manipularii acesteia. Astfel, procedura poate sa deblocheze drivere-le care au activat aceasta intrerupere. In celelalte cazuri, se poate transmite un semnal pe conditie variabila intr-un

monitor sau se poate trimite un mesaj drivere-ului blocat. Efectul intreuperilor este acela de a debloca driverele anterior blocate, in toate cazurile. Daca driverele sunt structurate ca procese kernell, cu propriile stari, stive si numaratoare de program, acest model lucreaza cel mai bine.

Pe langa aceste activitati, procesarea unui intreruperi nu consta numai in depistare, schimbarea unui semafor si executarea unui instruciuni de tip IRET pentru intoarcerea la procesul anterior. In realitate, sistemul de operare are mult de lucrat.

Pasii necesari a fi efectuati din punct de vedere software, dupa o intrerupere hardware sunt urmatorii **[1]**:

1. Salveaza orice registru (incluzand Program Status Word) care nu a fost salvat anterior de intreruperea hardware;
2. Realizeaza un context pentru procedura intrerupere. Sunt implicate TLB-ul, MMU-ul (Memory Management Unit) si pagina de tabela;
3. Realizeaza o stiva pentru procedura de intreruperi;
4. Instiinteaza controller-ul de intreruperi. In cazul in care nu exista un controller de intreruperi activat, reactiveaza intreruperile;
5. Copiaza registrele de unde erau salvate in tabela de procesare;
6. Executarea procedurii de intrerupere. Ea va extrage informatii din registrele de intrerupere ale controller-ului dispozitivului;
7. Alege care proces sa fie executat in continuare. Daca intreruperea a fost la nivelul unui proces cu prioritate mare, acesta va fi in continuare executat;
8. Seteaza contextul MMU al procesului care va fi executat in continuare;
9. Incarca registrele noului porces, incluzand PSW;
10. Executa noul proces.

Astfel, o intrerupere conduce la executarea unui numar mare de instructiuni, mai ales in cazul masinilor care au memoria virtuala organizata in tabele de pagini.

[1]: Andrew S. Tanenbaum, Sisteme de operare moderne - Capitolul 5- Input/Output, Byblos, 2004;

1.2 Driverele dispozitivelor

Dispozitivele de Intrare/Iesire atasate unui calculator au nevoie de cate un cod specific pentru a fi controlat, pentru fiecare. Acest cod se numeste driver al dispozitivului. Codul este de obicei scris de fabricantul dispozitivului si trimis odata cu acesta.

Fabricantii furnizeaza drivere pentru mai multe sisteme de operare cunoscute, deoarece fiecare sistem de operare necesita propriul driver necesita propriile drivere.

Dispozitivul trebuie sa faca parte din nucleul sistemului de operare, macar cu arhitecturile curente, pentru a se putea accesa hardware-ul dispozitivului, adica registrele controller-ului.

Designerii sistemelor de operare configureaza o arhitectura care sa permita instalarea driverelor scrise de fabricanticare se vor instala in sistem. Asta inseamna un model bine definit a ceea ce face driverul si cum interactioneaza cu restul sistemului de operare. Driverele dispozitivelor sunt pozitionate, in mod normal, sub restul sistemului de operare, ca in figura urmatoare (fig. 2):

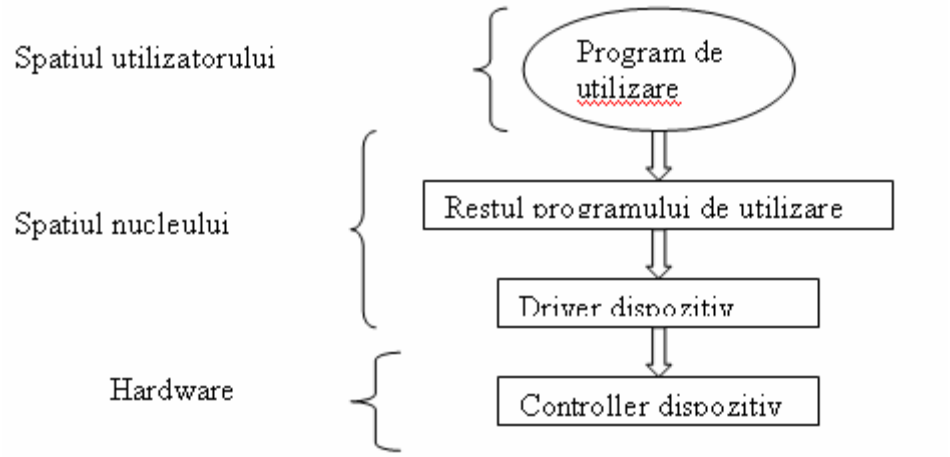

Fig. 2- Arhitectura configurata

Sistemele de operare clasifica dispozitivele intr-un numar mic de categorii. Cele mai comune sunt driverele de tip bloc, cum ar fi disk-urile, care contin date organizate in blocuri si care pot fi adresate independent, si dispozitivele de tip caracter cum ar fi tastaturile, imprimantele care accepta sau genereaza siruri de caractere. **[2]**

Cele mai multe sisteme de operare definesc o interfata standard pe care toate driverele de tip bloc trebuie sa o suporte si a doua interfata pe care toate driverele de tip caracter trebuie sa o suporte. Aceste interfete consista intr-un numar de proceduri pe care restul sistemului de operare le poate apela pentru a pune driverul in functiune. Cele mai comune proceduri sunt de citire al unui bloc sau de scriere al unui caracter.

Driverul unui dispozitiv are cateva functii. Cel mai evident este acela de accepta cereri de scriere si citire de la software-ul dispozitivului independent si de a observa daca acestea sunt efectuate. De asemenea, driverul trebuie sa initializeze dispozitivul daca este nevoise si sa gestioneze cererile de putere si evenimente. **[3]**

Multe drivere de dispozitive au , in general, aceeasi structura. Un driver incepe prin a verifica parametrii de intrare, pentru a verifica daca acestia sunt valizi; daca nu, acesta va returna o eroare. Daca acestia sunt valizi, se va face o translatare din abstract in concret a termenilor. Asta inseamna, pentru un driver de disk, ca conversia unui bloc liniar de numere intr-un head, track, sector si numere cilindrice pentru geometria disk-ului.**[4]**

In etapa urmatoare, driverul trebuie sa verifice daca dispozitivul este activ. Daca este, cererea va fi pusa intr-o coada pentru o procesare ulterioara. Daca dispozitivul este inactiv, se va examina starea hardware-ului in vederea manipularii cererii. Poate fi necesara comutarea dispozitivului in starea de pornire sau un motor de start inainte ca transferurile sa inceapa.

A controla un dispozitiv inseamna a emite o secventa de comenzi spre el. Driverul este locul in care comanda este determinata, depinzand de ce trebuie sa se faca. Dupa ce driverul stie ce comenzi urmeaza sa fie emise, incepe sa le scrie in registrele controllerului. Dupa scrierea fiecarei comenzi in controler, trebuie sa se verifice daca controlerul a acceptat comanda si este pregatit sa o accepte pe urmatoarea. Aceasta secventa continua pana cand toate comenzile au fost emise. **[5]**

[2] [3] [4][5[: Andrew S. Tanenbaum, Sisteme de operare moderne - Capitolul 5- Input/Output, Byblos, 2004;

Dupa ce comenzile au fost emise, poate aparea una dintre cele doua situatii. In multe cazuri, driverul dispozitivului trebuie sa astepte pana cand controlerul lucreaza, intrucat se autobloceahza pana cand intreruperile il vor debloca. In celelalte cazuri, operatia se termina fara intarzieri si driverul nu necesita blocare.

Acest model este o aproximare dura a realitatii. Codul este mult mai complicat. Pentru un singur lucru, un dispozitiv de Intrare/Iesire se poate completa cat timp un driver ruleaza, intrerupand driverul. Intreruperea poate cauza rularea driverului dispozitivului. De fapt, poate determina rularea unui driver al dispozitivului. De exemplu, cat timp driverul de retea proceseaza un pachet ce vine, alt pachet poate ajunge. Prin urmare driverul care este in rulare trebuie sa astepte sa fie apelat a doua ora, inainte ca prima apelare sa se fi terminat.

Intr-un sistem conectabil, dispozitivele pot fi adaugate sau indepartate cat timp calculatorul este in executie. Ca rezultat, cat timp un driver este ocupat cu cititul de la un dispozitiv, sistemul il informeaza ca utilizatorul a indepartat dispozitivul din sistem. Nu numai ca transferul de Intrare/Iesrie curent va fi anulat fara sa se deterioreze vreun nucleu al structurii de date, dar orice cerere in asteptare pentru acel dispozitiv va fi indepartata din sistem. Asadar, orice adaugare neasteptata de dispozitive noi determina nucelul sa „jongleze” cu resursele, indepartandu-le pe cele vechi din driver si adaugandu-le pe cele noi in locul lor.

Driverele nu pot apela sistemul, dar deseori au nevoie sa interactioneze cu restul nucelului. De obicei, apelurile catre un anumite proceduri ale nucleului sunt permise. De asemenea, sunt apeluri necesare pentru a administrarea MMU-ului, timere, controllerul DMA (Direct Memory Acces), controllerul de intreruperi s.a.m.d. **[6]**

[6]: . Andrew S. Tanenbaum-Operating Systems: Design&Implementation – Prentice HAll 1988

1.3 Sisteme de operare software pentru dispozitiv independente

Desi unele software-uri ale dispozitivelor de Intrare/Iesire sunt specifice, alte parti sunt independente de la dispozitiv la dispozitiv. Granita intre drivere si software-ul independent al dispozitivului este dependent de sistem, deoarece anumite functii care pot fi facute in mod independent pot fi realizate in drivere, pentru eficienta sau alte motive. In figura 3 se ilustreaza actiunile desfasurate in software-ul independent al dispozitivului:

Interfatare uniforma pentru driverele dispozitivelor
Buffering
Reportare de erori
Alocare si eliberare de dispozitive dedicate
Alocarea unei marimi de bloc independente de dispozitiv

Fig. 3 Actiunile desfasurate in software-ul independent al dispozitivului

Functia de baza al sofware-ului independent de dispozitiv este aceea de a efectua functii care sunt comune tuturor dispozitivelor si de a furniza o interfata uniforma la nivelul de utilizator al software-ului.

1.3.1 Interfatarea Uniforma pentru Driverele Dispozitivelor

O problema majora intr-un sistem de operare este cum sa se realizeze dispozitivele de Intrare/Iesire si driverele acestora sa semene mai mult sau mai putin. Daca disk-urile, imprimantele, tastaturile etc, sunt interfatate in diferite feluri, de fiecare data cand apare un nou dispozitiv, sistemul de operare trebuie modificat pentru fiecare dispozitiv.

Un aspect al acestei probleme este interfata intre driverul dispozitivului si restul sistemului de operare. In Figura 4 e ilustrata o situatie in care driverul dispozitivului are o interfata diferita, ceea ce inseamna ca functiile driverului disponibile pentru a putea fi apelate sistem difera de la driver la driver si ca functiile nucleului pe care necesare driverului difera de la unul la altul. Asadar, rezulta ca interfatarea fiecarui nou driver cere un efort de programare.

In Figura 5 observam exact opusul. Un design diferit in care toate driverele au aceeasi interfata. Este mult mai usor de adaugat un nou driver. Cei care realizeaza driverele stiu ce functii sa implementeze si ce functii ale nucleului trebuie sa apeleze. In practica, nu toate sunt absolut identice, dar de obicei exista un numar mic de tipuri de dispozitive si pana si acestea se aseamana in mare parte. De exemplu, pana si dispozitivele de bloc si caractere au multe functii in comun. **[7]**

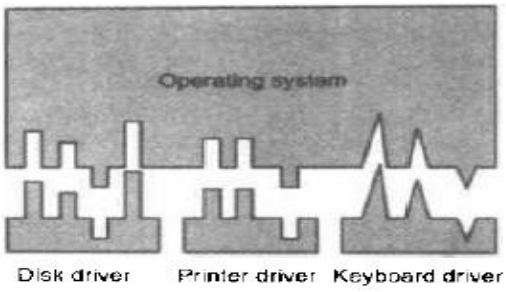

Fig. 4- Driverul dispozitivului cu interfata diferita

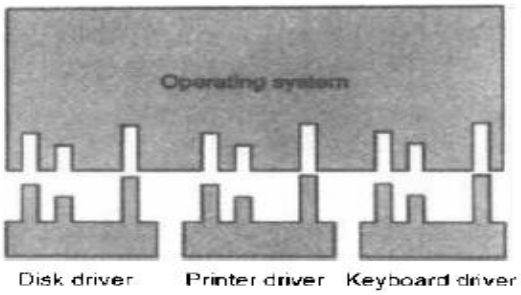

Fig. 5- Driverul dispozitivului cu aceeasi interfata

Alt aspect legat de uniformitatea interfetelor este legata de modul de denumire al dispozitivelor. Software-ul independent de dispozitiv se ocupa de alocarea pe driverele proprii a unor nume simbolice ale dispozitivelor.

Strans legat de denumire este protectia. Cum se apara sistemul de utilizatori care vor sa acceseze dispozitive la care nu au acces? Dispozitivele UNIX cat si Windows 2000 apar in sistem ca niste obiecte, ceea ce inseamna ca regulile obisnuite de protectie se aplica si dispozitivele de Intrare/Iesire. Administratorul sistemului poate seta o permisiune proprie fiecarui dispozitiv in parte.

[7] Andrew S. Tanenbaum-Operating Systems: Design&Implementation – Prentice HAll 1988

1.3.2 Buffering

Buffering-ul reprezinta de asemenea o problema atat pentru dispozitivele block, cat si pentru cele caracter. De exemplu, consideram un proces care vrea sa citeasca informatie dintr-un modem. Fiecare caracter care ajunge provoaca o intrerupere. Procedura de intrerupere duce caracterul in zona procesului de utilizator si il deblocheaza. Dupa plasarea undeva a caracterului, procesul citeste alt carcater si se blocheaza din nou. Procesul e ilustrat in Figura 5.

O metoda de imbunatatire este prezentata in Figura 5 b). Aici utilizatorul furnizeaza un buffer de n-caractere si face o citire de n-caractere. Procedura de intreruperi pune caracterele e intrare in acest buffer pana cand se umple. Apoi activeaza procesul utilizatorului. Metoda este mult mai eficienta decat cea anterioara, dar poate degrada performanta sistemului**.[8]**

O alta abordare este de a crea un buffer in interiorul nucleului in care stivuitorii de intreruperi sa pune caracterele ca in Figura 6 c). Cand bufferul este plin, pagina cu bufferul utilizatorului este adus si copiat intr-o singura operatie. In cazul in care bufferul este plin si un caracter ajunge in timp ce pagina cu bufferul utilizatorului este adusa de pe disk. Din moment ce bufferul este plin, nu exista loc pentru caracter. O metoda de a rezolva aceasta metoda este de a avea un alt buffer la nivelul nucleului. In timp ce primul buffer este copiat, al doilea este folosit in cazul unor caractere ajunse (Figura 6 d).).Aceasta metoda este cunsocuta sub numele de dublu buffering.

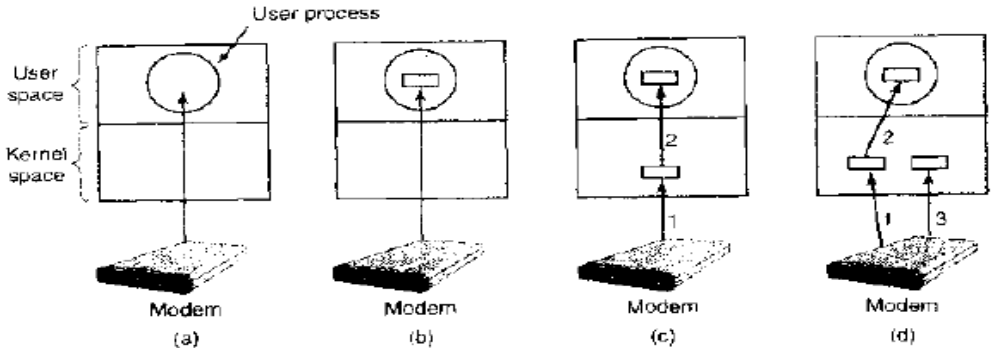

Figura 6. Metoda de dublu buffering

[8]: CS360 Lecture notes -- Introduction to System Calls (I/O System Calls) by Jim Plank

1.3.3 Reportarea erorilor

Multe erori sunt specifice dispozitivelor si de ele se ocupa driverul corespunzator, dar cadrul in care se ocupa de ele este independent de dispozitiv.

O prima clasa de erori le reprezinta erorile de programare. Acestea apar cand un proces cere ceva imposibil, cum ar fi sa se scrie pe un dispozitiv de intrare (tastatura, mouse, scanner) sau sa se citeasca de pe un dispozitiv de iesire (imprimanta).

Alte erori includ furnizarea unei adrese de buffer invalida sau un alt parametru si specifica un dispozitiv inexistent (de ex. un disk 3 atunci cand exista numai doua diskuri). In aceste cazuri se raporteaza un cod de eroare apelantului. **[9]**

Alta clasa de erori este clasa erorile de Intrare/Iesire actuala. De exemplu, incercarea de a se scrie pe un disc care a fost stricat sau sa se citeasca de pe o camera video care a fost inchisa. In aceste circumstante, driverul decide ce actiunea va lua. Atunci cand driverul nu stie ce se va intampla, paseaza problema software-ului independent de dispozitiv.[**10]**

Sofware-ul va actiona luand in calcul imprejurarile si natura erorii. Daca este cazul unei erori simple de citire si exista un utilizator disponibil, se afiseaza un dialog care intreaba utilizatorul despre actiunile ulterioare. Optiunile pot fi: reincercarea actiunii un numar de ori, ignorarea erorii sau inchiderea apelarii procesului. Daca nu exista un utilizator disponibil, probabil singura optiune reala este de a incheia apelul cu un cod de eroare.

In orice caz, cateva erori nu pot fi manipulate in felul acesta. De exemplu, in cazul unor structuri de date critice, cum ar fi directorul radacina sau lista de blocuri libera, distruse se afiseaza un cod de eroare si apelul se termina. .(*Beck L.L. – Sistem Software: An introduction to systems programming – Addison wesley Pub. Co. Inc. 1985*)

[9],[10] CS360 Lecture notes -- Introduction to System Calls (I/O System Calls)by Jim Plank

1.3.4 Alocarea si eliberarea dispozitivelor dedicate

Anumite dispozitive, cum ar CD-ROM, pot fi utilizate de un singur proces la un moment dat. Sistemul de operare se ocupa cu analizarea cererilor si acceptarea sau respingerea lor, in functie de ce e dispozitivul disponibil. O cale simpla te a manipula aceste cereri este de a efectua deschideri in fisierele speciale ale dispozitivelor. Daca dispozitivul este indisponibil, deschiderea nu reuseste. Se inchide dispozitivul dedicat, dupa care se elibereaza.

O abordare alternativa este de a avea mecanisme dedicate pentru cererile si eliberarile dispozitivelor dedicate. O incercare de a accesa un dispozitiv indisponibil, blocheaza apelarea, in locul esuarii ei. Procesele blocate sunt puse intr-o coada. Mai devreme sau mai tarziu, dispozitivul cerut devine disponibil si primului proces din coada ii este permisa continuarea executiei.**[11]**

[11] Operating Systems System Calls and I/O by Henry Newman

1.3.5 Alocarea unei marimi de bloc independente de dispozitiv

Diferite discuri pot avea marimi de sectoare diferite. Software-ului independent de dispozitiv ii revine rolul de a ascunde acest lucru si dea furniza nivelelor superioare o marime de bloc uniforma, de exemplu, sa trateze cateva sectoare ca un singur bloc logic. In acest fel, nivelele superioare lucreaza un dispozitive abstracte cu aceeasi marime a blocurilor logice, independenta de marimea sectoarelor fizice. Similar, cateva dispozitive de tip caracter trimit un byte de date pe unitatea de timp (de exemplu modemurile), in timp ce altele trimit date in unitati mai mari (interfetele de retea). Acestea diferente sunt, de asemenea, ascunse.

1.4 Spatiul utilizatorului in software-ul de Intrari/Iesiri

Desi o mare parte a software-ului de Intrari/Iesiri este in cadrul sistemului de operare, o mica parte din el consta in librarii legate cu programe ale utilizatorului. Apelurile sistemului, incluzand apelurile sistemului de Intrare/Iesire sunt realizate, in mod normal, de proceduri de biblioteca.

Atunci cand un program in C contine apelul:

Count = write(fd, buffer, nbytes), procedura de biblioteca *write* va fi legata de un program si va fi continuta in programul binar aflat in memorie la momentul rularii.**[12]**

[12] Operating Systems System Calls and I/O by Henry Newman

Alt exemplu din C este *printf* care duce un sir si, posibil, cateva variabile de intrare, construieste un sir ASCII, si dupa se efectueaza scrierea in sirul de iesire. De exemplu, instructiunea

*printf("The square of %3d is %6d\n", i, i*i");* formateaza un sir care contine: un sir de 14 caractere "The square of", urmat de valoarea i ca un format de 3 caractere si de sirul de 3 caractere " is ", iar , in final, i^2 ca un sir de 6 caractere**.[13]**

Un alt exemplu de procedura pentru intrare este *scanf* care citeste intrarea si o stocheaza descrise intr-un sir de variabile cu acelasi format ca la *printf*.

Nu toate spatiile utilizatorilor in software-ul de Intrari/Iesiri consta in proceduri de biblioteca. Alta categorie importanta este sistemul de tip **spooling** [spool - s(imultaneous) p(eripheral) o(perations) o(n) l(ine)] . Spoolingul este o metoda care se ocupa cu dispozitivele de Intrare/Iesire dedicate intr-un sistem multiprogramat. In cazul unui dispozitiv de tip spooled(de exemplu: imprimanta) este foarte usor din punct de vedere tehnic sa fie perimis oricarui proces al utilizatorului sa deschida fisierele speciale de tip caracter; procesul deshide fisierul si nu face nimic pe durata a ore intregi. Niciun alt proces nu poate printa nimic.

In schimb, ceea ce se face, este crearea unui proces special, numit **daemon**, si un director special numit **dierctor de spooling**. Pentru a printa un fisier, un proces genreaza intreg fisierul care urmeaza a fi printat si il pune in directorul de tip spooling. Protejand fisierul special impotriva folosirii directe a utilizatorilor, problema pastrariii lui deschis fara sa fie necesar a fost eliminata.[**14]**

Spoolingul nu este folosit numai de imprimante. De exemplu, transferul fisierelor intr-o retea utilizeaza o retea de tip daemon. Pentru a trimite un fisier undeva, utilizatorul il pune intr-un director de tip spooling. Mai tarziu, reteaua daemon il preia si il transmite. O intrebuintare particulara a transmisiei de a fisierelor tip spooled este sistemul USENET. Aceasta retea consta in milioane de masini din lumea intreaga care comunica prin intermediul Internetului. Pentru a posta un mesaj , utilizatorul invoca un program care accepta mesajul care urmeaza sa fie postat, apoi il depoziteaza intr-un director de spooling pentru trimiterea lui ulterioara catre alte masini. Intregul program de ruleaza in afara sistemului de operare.

[13] Operating Systems System Calls and I/O by Henry Newman

[14] Andrew S. Tanenbaum, Sisteme de operare moderne - Capitolul 5- Input/Output, Byblos, 2004;

Figura urmatoare rezuma sistemul de Intrari/Iesiri, aratand toate nivlele si functiile specifice.

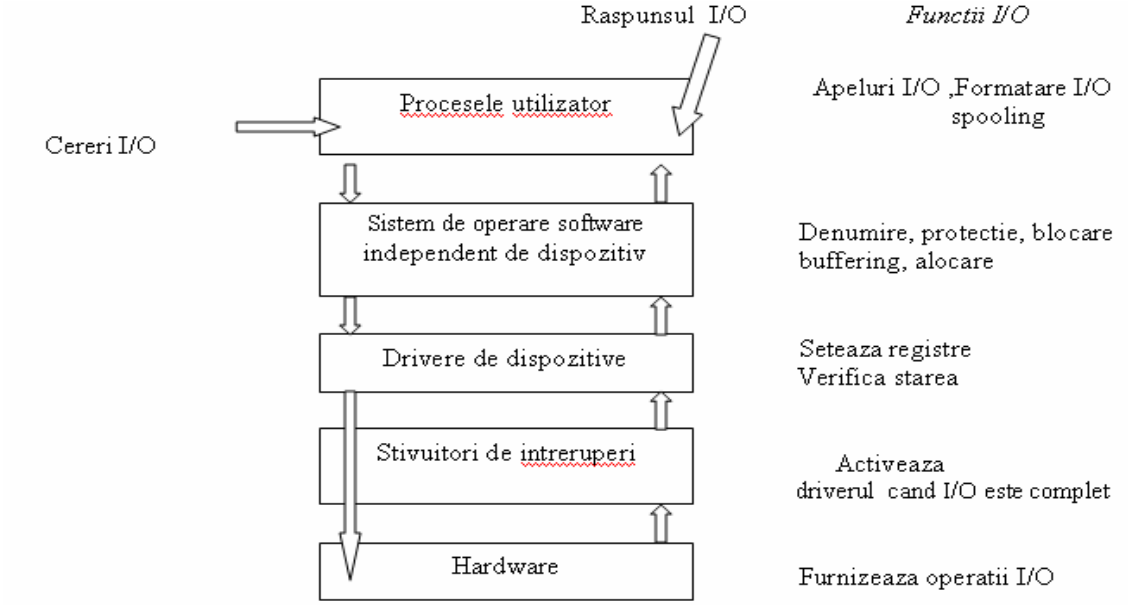

Figura 7- Schema sistemului Intrari/Iesiri

De exemplu, atunci cand un program utilizator incearca sa citeasca un bloc dintr-un fisier, sistemul de operare este invocat sa transporte apelul. Software-ul independent de dispozitiv verifica existenta blocului in bufferul cache, de exemplu. Daca blocul cerut nu este acolo, se apeleaza driverul dispozitivului in vederea emiterii cererii spre hardware, pentru a lua blocul de pe disk. Procesul este blocat pana cand operatia de pe disk este completa. Cand s-a terminat, hardware-ul genereaza o intrerupere. Stivuitorul de intreruperi are rolul de a depista ce dipozitiv are nevoie de atentie, dupa care se extrage starea dispozitvului si activeza procesul in vederea finalizarii cererii sistemuuil de Intrare/Iesire; urmatorul pas il reprezinta continuarea procesului utilizator.**[15]**

[15] Andrew S. Tanenbaum-Operating Systems: Design&Implementation – Prentice HAll 1988

BIBLIOGRAFIE

1. Andrew S. Tanenbaum, Sisteme de operare moderne - Capitolul 5- Input/Output, Byblos, 2004;

2. Andrew S. Tanenbaum-Operating Systems: Design&Implementation – Prentice HAll 1988

3. John Levine - "Linkers and Loaders" (http://www.iecc.com/linker)

4. CS360 Lecture notes -- Introduction to System Calls (I/O System Calls)
by Jim Plank

5.Operating Systems System Calls and I/O by Henry Newman

6. Beck L.L. – Sistem Software: An introduction to systems programming – Addison wesley Pub. Co. Inc. 1985