**Introducere**

Fiecare unitate periferica are urmatoarele parti componente:

* Circuitele electronice de control al unitatii(controller-dispozitiv hard care realizeaza interfata calculator-periferic,guvernat de un program special,numit **driver**);
* Unitatea periferica propriu-zisa(ex:unitati de disc,tastatura,mouse etc.)

In ceea ce priveste conectarea placilor de extensie-periferice intr-unul dintre sloturile libere de pe placa de baza,simpla cuplare fizica a acestora e suficienta din punct de vedere hardware, daca dispozitivele sunt de tipul plug’n’play.

Exista si o integrare de tip software,pe baza driver-elor care insotesc,de obicei placile de extensie si,de asemenea,este posibila partial configurarea pe baza setarilor din CMOS, solutii care insa nu rezolva integral problema interconectarii.

Pentru a integra complet un dispozitiv periferic intr-un sistem de calcul,indiferent de rolul acestuia,este nevoie de a defini și a asigura comunicatia generala ,prin care se va semaforiza și controla,la comanda microprocesorului,activitatea totala a dispozitivului în cauza,respective,de a-l cupla in sistemul general al transferurilor de date.

In continuare voi vorbi despre implicarea sistemului de operare(acțiuni software) in ceea ce privește gestiunea de intrari și iesiri:

**Concepte.Probleme**

Software-ul dispozitivelor de I/O are o structura ierarhizata pe niveluri in care cele inferioare le ascund detaliile celor superioare nivelurile inferioare, carora le executa cererile de servicii furnizandu-le rezultate.

 Un concept important in design SO este cunoscut sub numele de “independenta dispozitivelor”. Ceea ce inseamna ca este este posibil sa se scrie un program care sa poata accesa orice dispozitiv I/O fara a fi necesara specificarea dispozitivul in prealabil.

Un astfel de exemplu este cel al programului de citire al unui fisier ca intrare indiferent daca se gaseste pe floppy disk,CD,hard disk fara a fi necesara modificarea programului pentru fiecare dispozitiv in parte.

De asemenea este posibil sa scriu urmatoarea comanda:

sort < input > output

si aceasta sa functioneze indiferent daca la intrare este vorba de floppy disk, IDD,SCSI,disk iar la iesire avem orice mediu de stocare sau monitor.Aceasta problema intra in responsabilitatea sistemului de operare,care trebuie sa rezolve problema cauzata de faptul ca aceste dispozitive sunt diferite si necesita comenzi diferite de scriere si citire.

(Comanda “sort” este o comanda in UNIX ce are rolul de a sorta datele in cod ASCII.

 Folosind caracterul “<” se poate redirectiona input-ul dintr-un fisier si nu numai de la tastatura.

De exemplu daca avem o lista “fructe”:

portocala

mar

ananas

lamaie ;

in urma comenzii “ **sort<fructe**” ni se va afisa pe ecran :

ananas

lamiae

mar

portocala .

Daca insa ne dorim ca lista sortata sa fie redirectionata catre un alt fisier, “cumparaturi”, atunci vom executa “**sort<fructe>cumparaturi**” iar apoi pentru a citi continutul fisierului “cumparaturi” vom folosi **cat**.

In esenta,in urma comenzii,se sorteaza input-ul si il redirectioneaza in output.

Aceasta comanda se scrie in terminal.

Daca in terminal, totusi, nu este recunoscuta comanda de sort exista posibilitatea sa nu avem o anumita librarile ( respectiv nu avem PATH`ul setat catre acea librarie si prin urmare terminalul nu identifica comanda ). Este la fel ca si cand as vrea din terminal sa execut o comanda java de compilare a unei clase iar terminalul imi spune ca nu recunoaste comanda java - trebuie setat pathul catre JAVA.

)

Foarte apropiat de acest concept este si scopul uniformizarii numelor.Numele unui fisier trebuie sa fie intreg si sa nu depinda in nici un fel de dispozitiv.In UNIX toate discurile pot fi incluse intr-un sistem de ierarhizare in moduri arbitrare astfel incat utilizatorul sa nu aiba nici o grija referitoare la corespondenta unui anumit dispozitiv.

De exemplu floppy disk poate fi fixat in fata directoarelor /usr/ast/backup astfel daca copiez un fisier la /usr/ast/backup/Monday sa copiez fisierul pe floppy disk.In felul acesta toate fisierele sunt indexate in acelasi fel: dupa o cale precis definita.

O alta problema problema importanta pt S/O este buna organizare a erorilor.In general erorile trebuie sa fie cat mai bine organizate si apropiate de hardware.Daca controlerul descopera o eroare de citire,ar trebui sa incerce sa corecteze singur eroarea daca poate.Daca nu poate, atunci driver-ul dispozitivului ar trebui sa incerce s-o repare,probabil prin incercarea de a citi din nou adresa de date.Multe erori sunt trecatoare,cum spre exemplu eroarea de citire cauzata de praf va disparea in urma repetarii de mai multe ori a operatiei de citire.Numai daca etajele inferioare nu pot sa solutioneze problema,atunci trebuie sa se apeleze la cele superioare.De cele mai multe ori,recuperarea erorilor poate fi facuta transparent la un nivel scazut fara ca etajele superioare sa stie de eroarea respectiva.

O alta problema este blocarea versus intreruperea.Majoritatea dispozitivelor I/O sunt cu intrerupere - procesorul porneste transferul si inceteaza sa faca altceva pana cand intreruperea soseste.Programele utilizatorului sunt mai usor de scris daca operatiile I/O sunt blocate - operatia de citire este suspendata pana cand datele sunt valabile in buffer.

Alta problema o reprezinta memorarea intermediara a datelor.De obicei datele care provin de la un dispozitiv nu pot fi stocate direct in locatia lor finala.De exemplu,cand se primeste un pachet de pe retea ,SO nu stie unde sa-l puna pana nu-l stocheaza undeva si il analizeaza.Memorarea intermediara a datelor implica o serie considerabila de copieri , fapt ce are un impact major asupra performantelor I/O.

 Un ultim concept prezentat este acela al dispezitivelor dedicate versus cele generalizate.Unele dispozitive ,cum ar fi discurile,pot fi utilizate de mai multi utilizatori in acelasi timp.Alte dispozitive cum ar fi discurile cu banda magnetica sunt dedicate unui singur utilizator.Introducerea dispozitivelor dedicate a adus si la aparitia unor probleme ,de genul deadlocks.Inca o data SO trebuie sa se descurce atat cu cele dedicate ,cat si cele generale fara sa apara erori.

**Cum lucreaza I/O**

Precum spuneam si in introducere, pentru a integra un dispozitiv periferic trebuie sa stabilim si sa rezervam un canal de comunicatie DMA(Direct Memory Acces),canale de intrerupere IRQ(Interrupt Request) si la final sa setam adresele de intrare-iesire (adrese I/O),la nivelul carora se efectueaza transmiterea datelor,comenzilor si parametrilor necesari proceselor executate de periferic. Astfel, exista 3 cai fundamentale (dpdv soft) diferite prin care I/O lucreaza:

1.I/O programate;

2.I/O intrerupte;

3.I/O care folosesc DMA;

**1.I/O programate**

Cea mai simpla modalitate I/O este sa aiba un processor care sa faca toata treaba pentru el;aceasta metoda se numeste I/O programate.

Cel mai simplu mod de a pune in evidentza este sa folosim un exemplu.Consideram ca utilizatorul doreste prin intermediul unui program sa printeze 8 caractere dintr-un sir “ABCDEFGH” cu ajutorul imprimantei.Prima data se pune sirul in memoria intermediara din spatial utilizatorului.Dupa aceea procesul utilizarorului face un apel catre imprimanta pt ca aceasta sa inceapa scrierea.Daca imprimanta este deja folosita de catre un alt proces,atunci aceasta incercare esueaza si se emite o eroare ori se asteapta pana cand imprimanta este disponibila(aceasta decizie depinde de sistemul de operare si de parametri procesului ).De indata ce este liber accesul la imprimanta,procesul utilizatorului face un apel catre sistemul de operare prin care ii cere sa inceapa printarea sirului de caractere cu ajutorul imprimantei.Sistemul de operare atunci, de obicei, copiaza din memoria intermediara sirul intr-o zona,sa zicem p,in nucleu principal de unde se poate accesa cel mai usor.Atunci se verifica daca imprimanta este disponibila,daca nu se asteapta pana aceasta este.Apoi,sistemul de operare copiaza primul caracter in registrul de date al imprimantei,in acest caz folosind memorie indexata I/O.Aceasta actiune activeaza imprimanta.(Se poate ca ,caracterul sa nu apara deoarece unele imprimante asteapta sa se copieze in memoria intermediara mai multe caractere chiar si o pagina pana sa inceapa printarea.)De indata ce se copiaza primul caracter la imprimanta,sistemul de operare verifica daca imprimanta este pregatita sa primeasca alt caracter.In general imprimanta are un registru in care se specifica starea ei.Scrierea de date in registru face ca starea imprimantei sa fie ocupata.Cum controlerul imprimantei a scris caracterul current,atunci starea imprimantei se schimba prin trimiterea uni bit in registrul sau prin scrierea unei valori in el.In acest punct sistemul de operare asteapta iar ca imprimanta sa fie disponibila.Acest procedeu continua pana cand se printeaza intregul sir de caractere.

Aceste actiuni prezentate mai sus sunt sintetizate in urmatoarele linii de cod:

copy\_from\_user(buffer, p, count); /\* p is the kernel buffer \*/

for (i = 0; i < count; i++) { /\* loop on every character \*/

 while (\*printer\_status\_reg != READY) ; /\* loop until ready \*/

 \*printer\_data\_register = p[i]; /\* output one character \*/

}

return\_to\_user();

**Avantaje/Dezavantaje:**

I/O programate sunt simple dar au dezavantajul de a tine CPU ocupat pana cand se termina toate operatiile de I/O.Daca timpul alocat pentrut printarea unui caracter este scurt atunci timpul de asteptare este bun.

 **Concluzie:**

De obicei asteptarea este mare, de aceea este nevoie de o alta metoda I/O.

**2.I/O intrerupte**

 Acum vom analiza cazul in care imprimanta nu copiaza caracterele in nici o memorie intermediara ci le printeaza asa cum vin.Daca imprimanta poate sa printeze, sa zicem 100caractere/secunda, atunci fiecarui character ii trebuie 10ms sa fie printat.Asta inseamna ca dupa fiecare scriere a unui caracter in registrul imprimantei, SPU intra in asteptare pt 10ms asteptand urmatorul caracter sa fie scos.Acest este un timp mai mult decat suficiet ca procesorul sa castige timp si pt alte procese care altfel s-ar fi risipit.

Felul prin care reusim sa facem procesorul sa execute si alte operatii, pana imprimanta este pregatita pentru a fi folosita dinou, este acela de a utiliza intreruperile.Cand sistemul apeleaza imprimanta, sirul de caractere este deja facut,memoria intermediara este copiata in nucleul central iar primul caracter este copiat la imprimanta de indata ce este disponibila.Pana in acest moment procesorul executa alte procese.Procesul care se ocupa de printare este blocat pana cand ne printeaza intregul sir.

Acest lucru este facut de urmatoarele instructiuni:

Copy\_from\_user(buffer, p, count);

enable\_interrupts();

while(\*printer\_status\_reg != READY) ;

\*printer\_data\_register = p[0];

scheduler();

Cand imprimanta a printat un caracter si este pregatita sa-l accepte pe urmatorul este generata o intrerupere.Acesta intrerupere opreste aceast proces si ii salveaza starea.Atunci se ruleaza procedura de intrerupere la imprimanta care este prezentata intr-o varianta in liniile urmatoare :

if(count == 0) { unblock\_user();

} else {

 \*printer\_data\_register = p[i];

 count = count – 1;

 i = i + 1;

}

acknowledge\_interrupt();

return\_from\_interrupt();

**3.I/O care foloseste DMA(Direct Memory Acces)**

 Un aspect deranjant la intreruperea I/O este ca intreruperea intervine la fiecare caracter.Intreruperea ia timp ,ceea ce face ca acest lucru sa tina mai mult timp CPU ocupat.O solutionare e sa folosirea DMA.

In esenta, cand este vorba despre DMA, se face practice referire la transferurile de date (si la controlul acestora) intre memorie si porturile I/O, procesorul nemaiavand control asupra lor.

Astfel,aceasta cale consta in trimiterea de caracterele catre imprimanta fara ca procesorul sa fie deranjat/implicat.

In esentza DMA este I/O programate,cu diferenta ca toata treaba este facuta de controlerul DMA, nu de processor.

Un exemplu de cod de prinatare a unui sir folosind DMA este urmatorul:

copy\_from\_user(buffer, p, count);

set\_up\_DMA\_controller();

scheduler();

**Avantaje/Dezavantaje:**

Marele castig al folosirii DMA este acela ca se reduce numarul intreruperilor de la unul la un caracter la unul pe memoria intermediara de printare.Daca avem multe caractere si intreruperile inzecite,acest lucru este o imbunatatire mare.Pe de alta parte ,controlerul DMA este mult mai lent decat procesorul.

**Concluzie:**

Daca controlerul DMA nu este folosit la capacitatea lui maxima si processorul nu are altceva de facut in timpul intreruperii DMA,atunci I/O programate si I/O intrerupte pot fi mai bune.

Bibliografie:

1. Andrew S. Tanenbaum “Modern Operating Systems”