

4.2. Multitasking- sisteme real-time

4.2.1 Rolul timpului in activitatea sistemelor de operare

Timpul joaca un rol important in proiectarea si utilizarea sistemelor de operare in timp real. Trebuie clarificate insa anumite aspecte.

Timp real

Exista foarte multe definitii ale notiunii de timp real; una dintre ele este – Un sistem in timp real este capabil sa execute toate procesele sale fara a viola constringerile de timp specificate. O alta definitie este – Un sistem la care timpii de executie ai proceselor pot fi deterministic prezisi pe baza cunostintelor despre hardware-ul si software-ul sistemului. Acest lucru inseamna ca daca hardul poate indeplini sarcinile, atunci softul sistemului de operare in timp real va indeplini in timp deterministic acele sarcini. Acest determinism trebuie putin “usurat” datorita naturii stochastice inevitabile a planificarii proceselor.

Se face adesea distinctie intre “soft real time” si “hard real time”. “Soft” semnifica faptul ca nerespectarea specificatiilor constringerilor de timp nu este o catastrofa, in timp ce acest lucru este o catastrofa pentru un sistem “hard real time”. De exemplu: redarea unui fisier audio sau video este soft real time, pentru ca putine persoane vor observa cind un esantion soseste cu o fractiune de secunda mai tirziu. Controlul unei sonde spatiale, pe de alta parte, necesita hard real time, pentru ca racheta se misca cu viteze de ordinul kilometrilor pe secunda si intirzieri reduse in semnalele de ghidare adauga distante semnificative pe orbita si acestea pot cauza situatii periculoase de intrare in atmosfera. Prelucrările de precizie in coordonate numerice, radiatia de inalta precizie sau robotii chirurgicali reprezinta alte exemple in care este necesar hard real time: mutarea capului robotului cu o zecime de milimetru in plus datorita erorilor de timp poate cauza rebutarea partilor fabricate sau moartea pacientului.

Practic vorbind, distinctia intre soft si hard real time este adesea asociata scarii de timp implicate in sistem: procesele soft real time trebuie planificate cu precizie de ordinul milisecundelor, in timp ce procesele hard real-time cu precizie de ordinul microsecundelor. Totusi aceasta afirmatie are multe exceptii – de exemplu, un procesor low-cost pe 4 biti care controleaza un semafor de intersectie este mai hard real-time (in sensul caracterului determinist) decit un server e-commerce de zeci de mii de euro bazat pe mai multe procesoare Athlon/Opteron.

Latenta

Latenta unui proces este diferenta intre momentul de timp la care procesul trebuia sa inceapa(sau sa se termine) si momentul de timp la care s-a intimplat efectiv acest lucru (sau, in alt context, timpul intre generarea unui eveniment si perceperea sa). latentele apar datorita unor factori diversi:

- proprietatilor de timp ale procesorului, magistralelor, memoriei (cache intern, extern, RAM,ROM) sau dispozitivelor periferice
- proprietatile planificatorului sistemului de operare
- preemtivitatea nucleului



UNIUNEA EUROPEANĂ

MINISTERUL MUNCII, FAMILIEI ȘI
PROTECȚIEI SOCIALE
AMFOSDRUFONDUL SOCIAL EUROPEAN
POSDRU
2007-2013INSTRUMENTE STRUCTURALE
2007-2013

Structuri hardware si algoritmi specifici microsistemelor EMBEDDED

- incarcarea sistemului (de exemplu, numarul de procese concurente ce vor sa fie planificate)
- timpul de comutare a contextului

Acest ultim aspect este timpul necesar procesorului pentru a salva datele procesului curent in executie, de exemplu registri, stiva, pointerul de instructiuni, si pentru a le inlocui cu datele locale ale noului proces planificat. Citiiva din acesti factori sint constanti in timp, ceilalti aleatori, iar distributia statistica a latentelor in planificarile urmatoare se numeste jitter.

Exista numeroase activitati la nivelul sistemului de operare (de uz general) care pot introduce nondeterminism in comportamentul temporal :

Accesul la hard disk – Datorita aliniamentului sectoarelor precum si distantelor intre piste, un proces nu poate cunoaste dinainte cit ii va lua sa acceseze datele de care are nevoie. In plus, discurile sint dispozitive mecanice, timpii lor de lucru sint mult mai mari decit ai dispozitivelor electronice(de exemplu memoria RAM) iar accesul la disc se face prin buffere pentru a reduce timpul mediu de acces.

Accesul in retea. In special in cazul stivei de protocoale TCP/IP, se retransmit pachete in cazul erorilor de transmisie, astfel ca apare un nondeterminism.

Timing de joasa rezolutie. Programarea chipului timer genereaza de asemenea in anumite situatii intirzieri nepredictibile. Acestea sint de ordinul microsecundelor si pot afecta sistemele hard real-time.

Drivere de dispozitiv non real-time. Driverele de dispozitiv sint adesea ineficiente cu bugetul lor de timp – sint bazate pe asteptare ocupata sau pe estimari aproximative ale perioadelor de inactivitate in locul utilizarii intreruperilor generate de circuite timer, sau blocheaza resurse mai mult decit este strict necesar, sau se executa in spatiul utilizator cu nonpredictibilitatea corespunzatoare.

Alocarea si gestionarea memoriei. Dupa ce un proces a solicitat memorie suplimentara (de exeplu, prin apelarea functiei malloc), timpul necesar procesului de alocare a memoriei pentru a indeplini aceasta sarcina este nonpredictibil (In special in cazul in care memoria alocata a devenit puternic fragmentata si nu poate fi gasit nici un bloc de memorie contiguu suficient de mare). Mai mult chiar, un sistem de operare de uz general comuta codul si datele din memoria fizica atunci cind necesarul total de memorie al tuturor proceselor depaseste memoria fizica disponibila.

Sistemul de fisiere proc. Acest sistem de fisiere este o interfata utilizator foarte bogata la ceea ce se intimpla in interiorul nucleului Linux – toata aceasta informatie este oferita proceselor utilizator sub forma unor fisiere in acest sistem virtual de fisiere. Accesarea informatiei din aceste fisiere implica efort suplimentar semnificativ in anumite situatii, pentru ca fisierele sint virtuale – ele sint create doar atunci cind este necesar.

Magnitudinea exacta a tuturor intirzierilor mentionate anterior variaza puternic de la o platforma hardware la alta. Astfel, nu este doar sistemul de operare care face diferenta. Pentru anumite aplicatii, comutarea contextului este mai importanta – de exemplu esantionarea semnalelor audio cu 44.1 kHz – in timp ce alte aplicatii necesita putere mare de calcul la frecvente reduse de planificare – controlul miscarii robotilor la 1 kHz. Exista si procese , prelucrarea vocal de exemplu, care necesita ambele conditii.



UNIUNEA EUROPEANĂ

MINISTERUL MUNCII, FAMILIEI ȘI
PROTECȚIEI SOCIALE
AMFOSDRUFONDUL SOCIAL EUROPEAN
POSDRU
2007-2013INSTRUMENTE STRUCTURALE
2007-2013

4.2.2 Constrangerile de timp

Aplicatii diferite au constrangeri de timp diferite pe care sistemul de operare in timp real ar trebui in mod teoretic sa le satisfaca. Din pacate, nu exista un algoritm general si garantat de planificare a proceselor care sa fie capabil sa satisfaca toate clasele de constrangeri descrise in continuare:

- termenul limita. Un proces trebuie sa fie complet executat pina la un moment dat, dar cind anume va fi executat procesul intre momentul actual de timp si termenul limita nu este important pentru calitatea rezultatului final. De exemplu, procesorul trebuie sa umple memoria tampon a placii de sunet inainte ca aceasta sa se goleasca; tensiunea la un port de iesire trebuie sa atinga un anumit nivel inainte ca un alt periferic sa incerce sa citeasca aceasta valoare.
- timp de executie nul. Procesul trebuie executat complet intr-o perioada de timp care ideal ar fi zero. De exemplu, teoria controlului digital presupune ca masurarea, esantionarea, calcularea actiunii de control si trimiterea acesteia la dispozitivul periferic toate au loc instantaneu.
- calitatea serviciului (Quality of Service QoS). Procesul trebuie sa primeasca o anumita cantitate fixa de "servicii" pe unitatea de timp. Prin "serviciu" se intelege de obicei timp procesor, dar poate fi de asemenea reprezentat de notiuni ca "pagini de memorie", "latime de banda in retea" sau "latime de banda a accesului la disc". Acest aspect este important pentru aplicatii de genul multimedia (pentru a putea citi sau scrie fluxurile de date audio sau video de la dispozitivele multimedia) sau servere de retea (pentru a garanta un minim de servicii si a contracara atacuri tip "denial of service"). Calitatea serviciului se specifica printr-un numar redus de parametri: s secunde de serviciu in fiecare interval de timp de lungime t secunde. O specificatie de 5 microsecunde din 20 de microsecunde este o calitate a serviciului real-time mult mai dura decit o specificatie 5 secunde din 20 desi, in medie, ambele au ca rezultat acelasi timp alocat procesului respectiv.

Problema majora este ca planificatorul are nevoie de informatii complete despre cit de lung este fiecare proces si cit timp va avea nevoie pentru executie si cind va deveni gata de executie. Aceasta informatie este practic imposibil de obtinut, si, chiar daca ar fi disponibila, calculul planificarii optime este o problema de cautare de inalta complexitate, in consecinta avind cost ridicat in timp.

Procese diferite ale sistemului concureaza pentru aceleasi resurse: procesoare, retea, memorie, discuri. Mult mai atent decit in cazul unui sistem de operare de uz general, la un sistem in timp real trebuie in permanenta considerat scenariul cel mai defavorabil: daca procese diferite ar putea necesita acelasi serviciu, mai devreme sau mai tirziu ele vor avea nevoie de acel serviciu in acelasi timp.

4.2.3 Structuri de date temporale

Cea mai mica cuanta de timp utilizata in sistemele de operare de uz general depaseste 1 milisecunda. Acest ordin de marime nu apare pentru ca procesoarele nu ar fi suficient de rapide sa execute sarcini semnificative in acea cuanta, ci pentru ca masinile pe 32 de biti au numai 2 la puterea 32 cuante de timp pina la generarea depasirii in temporizatorul-numaratorului lor. La 1000 de tacte pe secunda, aceasta corespunde la mai putin de 50 de zile, cu siguranta insuficient



UNIUNEA EUROPEANĂ

MINISTERUL MUNCII, FAMILIEI ȘI
PROTECȚIEI SOCIALE
AMFOSDRUFONDUL SOCIAL EUROPEAN
POSDRU
2007-2013INSTRUMENTE STRUCTURALE
2007-2013

Structuri hardware si algoritmi specifici microsistemelor EMBEDDED

pentru servere si aplicatii embedded. Sistemul de operare Linux utilizeaza o cuanta de timp de planificare, numita jiffie, de 10 milisecunde pe majoritatea procesoarelor(1 milisecunda pe arhitectura Alpha, datorita numaratoarelor pe 64 de biti).

Constrangerile de timp ale unor procese in timp real sint adesea exprimate cu o rezolutie mult mai mare decit a planificatorului de uz general, de exemplu precizie sub microsecunda in loc de milisecunde. In acest caz, structura de date in care este memorat timpul trebuie sa fie capabila sa faca fata acestei viteze, pentru a evita depasirile. De exemplu, versiunile real-time de Linux utilizeaza o structura de date de timp de inalta rezolutie care numara timpul in nanosecunde. Un intreg pe 64 de biti ar trebui sa faca acest lucru, pentru ca 32 de biti ar fi prea periculos – un numarator pe 32 de biti genereaza depasire la fiecare 4 secunde la o perioada de numarare de 1 nanosecunda. Nu trebuie neglijat faptul ca nu toate compilatoarele pot lucra nativ cu intregi pe 64 de biti, astfel ca anumite tehnici de codare in limbaj de asamblare pot fi necesare in anumite situatii.

Specificatiile POSIX dispun de ceasuri si numaratoare standard. Timespec este o structura de date ce memoreaza timpul in doua substructuri secunde si nanosecunde, utilizeaza reprezentarea pe 64 de biti, dar separarea intre secunde si nanosecunde este un mod ineficient de reprezentare a timpului – exista numai 10^9 nanosecunde intr-o secunda, deci mai mult decit 2 biti din cimpul nanosecundelor ramine neutilizat. Aceasta inseamna ca la fiecare aditie a cuantei de timp softul trebuie sa verifice daca s-a atins limita de o secunda pentru a modifica si cimpul secundelor. Acest mod de lucru este mai complicat decit a avea un numarator pe 64 de biti care poate numara in continuu fara a verifica nimic.

4.2.4 Rolul intreruperilor in sisteme real-time si EMBEDDED

Intreruperile sint indispensabile in majoritatea sistemelor de calcul cu pretentii real time. Intreruperile sint procesate de asa numitele rutine de deservire ISR (Interrupt Service Routine). Cu cit rutina ISR isi poate face treaba, cu atit mai mare este performanta real-time a sistemului, pentru ca celelalte sarcini sint intirziate mai putin. Timer-ele sint un exemplu tipic de dispozitive periferice care genereaza intreruperi; alte asemenea dispozitive sint tastatura, placile de achizitie, placile video, porturile paralele si seriale, etc. Chiar si procesorul poate genera intreruperi, de exemplu la comutarea modului protejat, la executia unor operatii ilegale, ca parte a sesiunii de depanare sau atunci cind o exceptie este ridicata de un program aplicatie.

Suportul hardware pentru intreruperi

Un sistem tipic bazat pe intreruperi, cum sint majoritatea sistemelor in timp real si embedded, dispune de una sau mai multe din urmatoarele componente:

Vectorul de intreruperi. Majoritatea sistemelor dispun de mai mult de o linie de intrerupere (IRQ – Interrupt Request) iar producatorul procesorului asambleaza toate aceste linii de intrerupere intr-un vector de intrerupere. Pentru procesoarele INTEL x86 acest vector contine 256 de intrari si este numit tabel de descriere a intreruperilor IDT Interrupt Description Table. Vectorul de intrerupere este o arie de pointeri la rutine de deservire a intreruperilor ce vor fi declansate atunci cind intreruperea corespunzatoare apare. Vectorul contine de asemenea pentru fiecare linie de intrerupere un bit ce semnalizeaza daca este o intrerupere in asteptare pe linia respectiva, de exemplu un dispozitiv periferic a ridicat intreruperea si acum asteapta sa fie deservit.

Anumite procesoare utilizeaza procesarea nonvectorizata a intreruperilor: la aparitia unei intreruperi se transfera controlul unei rutine unice, iar aceasta va decide ce sa faca cu



Structuri hardware si algoritmi specifici microsistemelor EMBEDDED

întreruperea. Aceeași strategie este utilizată și în software de către majoritatea sistemelor de operare pentru a permite mai multor dispozitive să folosească aceeași linie de întrerupere.

Intrerupere software sau sincrona. O întrerupere sincronă, numită și întrerupere software sau capcana (trap) este o întrerupere cauzată de o instrucțiune limbaj mașină specifică, deci sincron cu execuția programului. Exemple de astfel de întreruperi sunt trap la familia de procesoare Motorola 68000, swi la ARM, int la INTEL x86, la diviziune cu zero, eroare de segment de memorie etc. Întrucât această facilitate este suportată în hardware, este de așteptat un număr mare de întreruperi software cu nume și funcții diferite, nonstandard.

Diferențele semnificative între întreruperile hardware asincrone și cele software sincrone sunt:

- eventualele următoare întreruperi sunt dezactivate imediat ce întreruperea hardware apare, dar nu sunt dezactivate în cazul întreruperilor soft
- rutina de deservire a întreruperii soft rulează în contextul procesului care a generat-o; rutinele de deservire a întreruperilor hardware nu au nici un proces de care să fie legate.

Întreruperile hardware și software partajează același vector de întrerupere, dar acest vector furnizează domenii separate pentru întreruperile hard și pentru cele soft.

Intreruperi declanșate pe nivel sau pe front. Din punct de vedere hardware, dispozitivele periferice pot transmite semnalele de întrerupere în două moduri:

- declanșare pe front. O întrerupere este trimisă atunci când linia asociată își schimbă starea de la zero la unu logic, sau invers. Acesta este un eveniment "zero time" care crește șansele pierderii hardware a întreruperii de către controllerul de întreruperi. Mai mult chiar, dacă mai multe dispozitive sunt conectate la aceeași linie de întrerupere, sistemul de operare trebuie să cheme toate rutinele de deservire a întreruperilor pentru ca altfel poate cauza o pierdere software a întreruperii: chiar dacă este detectată o singură tranziție de nivel, și prima sa rutină de deservire confirmă recepționarea acestei întreruperi, se poate întâmpla să se rateze o altă tranziție, astfel ca se poate asigura certitudinea numai după ce se da șansa tuturor rutinelor ISR să ruleze; din păcate această situație nu este eficientă.
- declanșare pe nivel. O întrerupere este semnalizată prin nivelul liniei hardware asociate. Acest fapt nu numai că reduce riscul pierderii unei tranziții, dar permite de asemenea o deservire mai eficientă a întreruperilor: fiecare ISR care deja a deservit întreruperea va confirma dispozitivului periferic, care va readuce în starea implicită linia de întrerupere. Astfel, nivelul semnalului pe linia de întrerupere se va schimba din nou după servirea întreruperii iar sistemul de operare nu trebuie să încerce toate rutinele de deservire conectate la aceeași linie de întrerupere.

Controllerul de întreruperi. Acest modul hardware izolează sistemul de operare de aspectele electrice ale liniilor de întrerupere. Anumite controlere sunt în stare să plaseze întreruperile în cozi de așteptare, astfel ca nici una nu este pierdută (desigur, există limite hardware în privința mărимii cozii) sau să permită configurarea priorităților întreruperilor. Circuitul Intel 8259 Programmable Interrupt Controller PIC este standardul în materie utilizat în ultimii 25 de ani. Fabricanții de PC-uri utilizează de regulă două cipuri, întrucât fiecare din ele poate gestiona numai 8 linii de întrerupere. Din păcate, utilizarea mai multor controlere implică conectarea acestora în lanț, în sistem daisy chain – linia de ieșire



UNIUNEA EUROPEANĂ

MINISTERUL MUNCII, FAMILIEI ȘI
PROTECȚIEI SOCIALE
AMFOSDRUFONDUL SOCIAL EUROPEAN
POSDRU
2007-2013INSTRUMENTE STRUCTURALE
2007-2013

Structuri hardware si algoritmi specifici microsistemelor EMBEDDED

intrerupere a primului controler este conectaata la o linie de intrare a celui de-al doilea – si apar astfel intirzieri in deservirea intreruperilor. Un alt dezavantaj al cipului este ca nu a fost proiectat pentru sisteme multiprocesor. Sistemele performante utilizeaza un Advanced Programmable Interrupt Controller (APIC). Acesta nu este doar un singur cip, ci un mic sistem hardware care gestioneaza intreruperile. Fiecare procesor trebuie sa aiba un circuit APIC local care va primi intreruperi de la sistemul APIC global. Perifericele hard au conectate liniile de intrerupere la circuitul de I/O APIC iar acesta va trimite semnale circuitelor APIC locale ale procesoarelor pentru care este destinata intreruperea. Arhitectura APIC este superioara arhitecturii standard PIC datorita mai multor aspecte:

- poate avea mult mai multe linii de intrerupere, eliminind astfel necesitatea partajarii liniilor de intrerupere
- permite prioritati programabile pentru intreruperi
- este mai rapid de programat (o singura instructiune catre registrul de prioritati al circuitului APIC local, registru aflat in procesor, spre deosebire de doua instructiuni catre PIC, care in plus nu este in procesor)
- permite lucrul cu intreruperi declansate pe nivel in locul intreruperilor declansate pe front. Magistrala PCI lucreaza cu intreruperi declansate pe nivel, active pe zero logic, astfel ca poate conlucra eficient cu circuitul APIC

Platforma PowerPC dispune de un alt standard de tratare a intreruperilor hardware, si anume OpenPIC. Acesta garanteaza o calitate hardware ridicata si, in plus, poate lucra si cu arhitecturi x86.

Suportul software pentru intreruperi

Din punct de vedere software, un sistem bazat pe intreruperi trebuie sa tina cont de urmatoarele aspecte:

Subrutinele de deservire a intreruperilor (numite si handler-e de intreruperi) trebuie sa fie cit mai scurte. Aceste subrutine sint apelate cind apare o intrerupere pe linia de intrerupere pentru care subrutina ISR a fost inregistrata in vectorul de intreruperi. De obicei aceasta inregistrare se face printr-un apel sistem, dar poate fi realizata si direct printr-o instructiune masina dintr-un program rulind cu nivel suficient de privilegii. Inregistrarea inseamna de fapt plasarea adresei subrutinei ce va fi apelata de intrerupere in cimpul adresa al vectorului de intreruperi la indexul corespunzator numarului intreruperii.

Sistemul de operare nu intervine(mai exact, nu poate sa intervina) in lansarea unei ISR, pentru ca totul este facut automat de catre procesor. Contextul in care ruleaza procesul curent este salvat in stiva acelui proces; adresa stivei este in unul din registrii procesorului iar acesta din urma are acces imediat si automat la stiva. Acest mod de lucru influenteaza structura software a sistemului: fiecare proces trebuie sa aiba suficient spatiu in stiva pentru a suporta efortul suplimentar(consum de memorie) impus de ISR. Astfel, in cel mai defavorabil scenariu, spatiul de memorie necesar stivei poate deveni foarte mare in bugetul memoriei sistemului, in special daca sistemul permite intreruperi pe mai multe nivele (intreruperile pot fi intrerupte de alte intreruperi – interrupt nesting). Din ce in ce mai des sistemele de operare furnizeaza contexte separate pentru deservirea intreruperilor, partajate de toate subrutinele ISR (Linux, vxWorks).

O subrutina ISR trebuie sa fie cit mai scurta posibil, pentru ca de obicei ruleaza cu intreruperile dezactivate ceea ce previne deservirea altor intreruperi si, deci, rularea altor



UNIUNEA EUROPEANĂ

MINISTERUL MUNCII, FAMILIEI ȘI
PROTECȚIEI SOCIALE
AMFOSDRUFONDUL SOCIAL EUROPEAN
POSDRU
2007-2013INSTRUMENTE STRUCTURALE
2007-2013

Structuri hardware si algoritmi specifici microsistemelor EMBEDDED

proces. Subrutina ISR trebuie sa deserveasca dispozitivul periferic care a declansat-o si apoi sa-si inceteze imediat activitatea. Deservirea consta in general in citirea sau scrierea unor registre ale dispozitivului si plasarea lor in zone tampon de memorie de unde vor fi prelucrate mai departe de alte procese din afara ISR si cu intreruperile activate. Aceasta procesare ulterioara este sarcina rutinei DSR – Deferred Service Routine. Extragerea datelor din intreruperea ISR in rutina DSR trebuie sa se faca intr-un mod non-blocant; in acest scop se folosesc in general structuri de date de genul listelor FIFO sau buffere circulare.

Gestionarea intreruperilor software. O intrerupere sincrona este numita uneori trap (capcana) sau intrerupere software. Aceasta este apelata chiar de procesor ca in cazul depasirii unor registri, erori de adresare in pagini de memorie etc. si functioneaza ca o intrerupere hardware (salvarea starii, comutare in mod privilegiat, salt la subrutina de deservire) dar ruleaza cu intreruperile activate.

Intreruperile software sint foarte importante pentru ca asigura singura posibilitate pentru programele din spatiul utilizator de a executa operatii protejate sau instructiuni privilegiate. Acestea pot fi executate numai cind procesorul se afla in modul protejat (numit si mod privilegiat). Instructiunile privilegiate vizeaza operatii ca adresarea directa a spatiului fizic I/O, lucrul cu infrastructura de gestionare a memoriei cum ar fi tabelul de cautare al paginilor, activarea si dezactivarea intreruperilor sau chiar oprirea masinii. Instructiunile privilegiate sint oferite proceselor utilizator prin apeluri sistem, in fapt rutine de deservire a intreruperilor software: un apel de sistem plaseaza date in registri sau pe stiva si apoi executa o intrerupere software care comuta procesorul in mod privilegiat si executa deservirea intreruperii. Rutina de deservire poate utiliza datele din registri sau de pe stiva pentru actiuni specifice procesului. Intrucit aceasta rutina ruleaza in contextul procesului care a efectuat apelul sistem, ea are acces la toate datele procesului plasate in stiva.

Un apel sistem este numai un exemplu de intrerupere soft. Un alt nume pentru intreruperile soft este acela de cereri de servicii (SRQ – service request). Fiecare tip de procesor are o parte a vectorului de intreruperi rezervata pentru aceste rutine de deservire a intreruperilor soft (numita si trap handler). Sistemele de operare au uzual un trap handler implicit la care se ataseaza toate intreruperile software posibile din sistem. Sistemul de operare rezeva un numar de capcane ca si semnale sistem, dar ramine si pentru utilizator capcane configurabile.

Intreruperile soft sint extrem de utile in depanarea programelor: compilarea programului cu optiunea de depanare activata are ca efect, printre altele, in introducerea in versiunea compilata a codului a instructiunilor masina de generare a unei capcane dupa fiecare linie din codul sursa. Subrutina de deservire a capcanei poate informa programul de depanare la ce punct de intrerupere se afla executia , gratie informatiei registrilor pe care capcana le-a completat.

Latenta intreruperilor. Notiunea se refera la timpul dintre sosirea unei intreruperi hardware si inceperea executiei subrutinei corespunzatoare de deservire. Aceasta marime are caracter statistic pentru ca este influentata de un mare numar de efecte nedeterminate si este tot mai importanta in procesoarele moderne cu multiple nivele de cache si conduite de instructiuni, structuri care trebuie toate resetate inaintea inceperii ISR. Acest mecanism sta la baza unor fenomene oarecum ciudate la procesoarele actuale cu frecvente de ordinul GHz care au latente ale intreruperilor mai mari decit alte procesoarele mai vechi.

Activarea si dezactivarea intreruperilor. Fiecare procesor dispune de operatii atomice pentru activarea sau dezactivarea (mascarea) intreruperilor. Nume obisnuite pentru aceste



UNIUNEA EUROPEANĂ

MINISTERUL MUNCII, FAMILIEI ȘI
PROTECȚIEI SOCIALE
AMFOSDRUFONDUL SOCIAL EUROPEAN
POSDRU
2007-2013INSTRUMENTE STRUCTURALE
2007-2013

Structuri hardware si algoritmi specifici microsistemelor EMBEDDED

functii sint sti() – set interrupt enable flag si cli() – clear interrupt enable flag. In acelasi context pot fi intilnite functii ca save_flags() si restore_flags(). Acestea sint versiuni mai fine in sensul ca lucreaza la nivel de bit, deci asupra fiecarei intreruperi individual. Functia cli() dezactiveaza toate intreruperile pe toate procesoarele intr-o masina multiprocesor, abordare ineficienta intr-un sistem in timp real.

Intreruperi cu prioritati. Anumite sisteme ofera ca facilitate hard, priritati statice pentru intreruperi. Aceasta inseamna ca sistemul de operare blocheaza o noua intrerupere daca o rutina de deservire a unei intreruperi cu prioritate sporita este inca in lucru. Similar, daca noua intrerupere are prioritate mai mare, va intrerupe rutina de deservire a primei intreruperi. In acest caz, aceasta rutina trebuie sa fie reentranta.

Dezactivarea bazata pe prioritati a intreruperilor. Aceasta este o facilitate soft care permite programatorului sa dezactiveze toate intreruperile cu o prioritate mai mica decit un anumit nivel. Windows NT suporta aceasta facilitate si o utilizeaza extensiv.

Imbricarea intreruperilor – Interrupt Nesting. Daca procesorul si sistemul de operare permit imbricarea intreruperilor, atunci o subrutina de deservire a unei intreruperi poate fi intrerupta la rindul ei de catre o alta intrerupere. Aceasta facilitate creste complexitatea codului, pentru ca subrutinele de deservire trebuie sa fie reentrante, adica sa fie capabile sa fie intrerupte in orice moment.

Partajarea intreruperilor. Anumite sisteme permit diferitelor dispozitive periferice sa fie legate la aceeasi linie de intrerupere. Subrutina de deservire a acesteia trebuie sa decida care a fost dispozitivul declansator si sa actioneze in consecinta. Acest lucru se face sau verificind registrul de stare al fiecarui dispozitiv sau apelind succesiv toate subrutinele asociate intreruperii respective. Partajarea intreruperilor este implementata in majoritatea sistemelor de operare de uz general. Linux accepta handleri multiple de intreruperi pe aceeasi linie de intrerupere. Nucleul leaga propria rutina de deservire de intreruperea hardware iar aceasta rutina invoca una cite una rutinele ISR inregistrate de programele aplicatie. Aceasta presupune ca ele vor fi executate dupa ce rutina ISR hardware a fost terminata, dar inaintea oricarui alt proces si cu intreruperile activate.

Ceea ce Linux permite interzic sistemele de operare real-time – RTLinux si RTAI permit numai o singura ISR pe intrerupere hardware, pentru a obtine un comportament cit mai determinist. Subrutina ISR in timp real pe care un program utilizator a inregistrat-o este direct legata de intreruperea hardware, deci ruleaza cu intreruperile dezactivate.

ISR, DSR si ASR

O subrutina ISR trebuie sa fie cit mai scurta posibil pentru a minimiza intirzierea altor ISR sau planificarea proceselor. In sistemele de operare de uz general numai subrutinele ISR pe care sistemul de operare le-a atasat cererilor de intrerupere ruleaza cu intreruperile dezactivate, dar nu si subrutinele ISR inregistrate de utilizator. Un sistem de operare in timp real, pe de alta parte, permite numai o subrutina ISR per cerere de intrerupere, altfel neputindu-se garanta determinismul deservirii celorlalte intreruperi. Acest aspect face sarcina programatorilor real-time putin mai usoara, pentru ca pot proiecta subrutine non reentrante mai simple si mai rapide (acestea pot stoca informatia locala fara pericolul de a fi suprascrisa de alta invocare a aceleiasi subrutine) si are garantia atomicitatii (intreruperea va fi deservita fara a fi intrerupta la rindul



UNIUNEA EUROPEANĂ

MINISTERUL MUNCII, FAMILIEI ȘI
PROTECȚIEI SOCIALE
AMFOSDRUFONDUL SOCIAL EUROPEAN
POSDRU
2007-2013INSTRUMENTE STRUCTURALE
2007-2013

Structuri hardware si algoritmi specifici microsistemelor EMBEDDED

ei). Cu toate acestea, daca sistemul de operare si hardul suporta intreruperi cu prioritati, subrutina de deservire de pe un nivel IRQ poate fi intrerupta de o intrerupere cu prioritate mai mare.

O subrutina de deservire a intreruperii in mod normal executa doar urmatoarele operatii:

- citește sau scrie datele implicate in comunicare cu dispozitivul periferic sau capcana ce a cauzat intreruperea
- confirma intreruperea daca dispozitivul periferic are nevoie
- eventual, trezeste alt "proces" pentru a efectua procesarile ulterioare

De exemplu, majoritatea driverelor pentru placi de retea executa doar transferul pachetelor de date de la sau la placa in ISR si delega toata interpretarea acestor date altui proces. Scheletul unei aplicatii ISR-DSR este:

```
dsr_thread()
{
while (1) {
    wait_for_signal_from_isr();
    process_data_of_ISR (); // including all blocking stuff
}
}

interrupt_handler( )
{
reset_hardware();
do_isr_stuff();
send_signal_to_wake_up_dsr();
re_enable_interrupts() // some RTOSs do this automatically
}
```

In nucleul Linux rutina DSR este denumita bottom half, in timp ce ISR este denumita top half. Conceptul este totusi depasit si inlocuit in distributiile recente de tasklets si softirqs. Motivul abandonarii conceptului de bottom half este ca Linux are o limita fizica de 32 functii bottom half. In plus, ele ruleaza cu blocarea intregului sistem, ceea ce nu este eficient in sistemele multiprocesor. Conceptul de softirq este versiunea multiprocesor a bottom half; sint limitate la 32, astfel ca programatorii nu ar trebui sa le utilizeze ci sa se bazeze pe tasklet-uri.

Un tasklet este o functie apelata de nucleu la cererea unei "continuari" din partea unei ISR. In afara lumii Linux, aceasta functie de continuare se numeste Deferred Service Routine, sau, (Windows NT) Deferred Processing Call. Numarul de tasklet-uri este nelimitat.

Un tasklet trebuie creat mai intii prin functia tasklet_schedule(&tasklet) si initializat prin apelul tasklet_init; acesta conecteaza un identificator de tasklet cu functia ce va fi executata si structura de date in care ISR va stoca informatia de procesat de catre tasklet. Tasklet-ul (sau rutina DSR) ruleaza cu intreruperile activate, dar in afara contextului oricarui proces particular, la fel ca rutina ISR care l-a solicitat.

Sistemele de operare de uz general executa subrutinele DSR in secventa dupa rutinele ISR si inaintea iesirii din nucleu pentru returnarea in spatiul utilizator.

O subrutina ISR nu are voie sa utilizeze semafoare sau orice alte apeluri sistem potential blocante: o subrutina ISR blocata in asteptarea unui alt proces cauzeaza probleme extrem de mari, pentru ca intreruperile sint dezactivate in cadrul ISR iar conditia de deblocare nu va mai apare niciodata.

Anumite sisteme de operare dispun de un nivel suplimentar de partajare a intreruperilor – pe linga functii ISR si DSR, ele ofera posibilitatea utilizarii de Asynchronous Service Routines (ASR) / Anynchronous Procedure Calls (Windows NT). ASR ruleaza dupa terminarea



UNIUNEA EUROPEANĂ

MINISTERUL MUNCII, FAMILIEI ȘI
PROTECȚIEI SOCIALE
AMFOSDRUFONDUL SOCIAL EUROPEAN
POSDRU
2007-2013INSTRUMENTE STRUCTURALE
2007-2013

Structuri hardware si algoritmi specifici microsystemelor EMBEDDED

tuturor DSR, dar înainte ca procesele normale să poată fi planificate. Scopul funcțiilor ASR este de a executa acea parte a reacției la întrerupere care necesită contextul procesului.

Bibliografie:

1. Istvan Sztojanov, Sever Pașca, Elisabeta Buzoianu, Aplicații hardware și software cu microcontrolerul PIC12F675, Editura Cavallioti, ISBN 978-973-7622-54-9, București 2008
2. Istvan Sztojanov, Alexandru Vasile, Elisabeta Buzoianu, Sever Pașca, *Programarea microcontrolerelor din familia Intel, Aplicații practice hardware cu 80C552*, Editura Man-Dely, ISBN 973-85681-5-3, București 2004.
3. <http://vega.unitbv.ro/~romanca/EmbSys/>
4. <http://facultate.regielive.ro/cursuri/electronica/>
5. www.microcip.com
6. Andrei Drumea, Teza de doctorat, UPB 2009



UNIUNEA EUROPEANĂ



MINISTERUL MUNCII, FAMILIEI ȘI
PROTECȚIEI SOCIALE
AMFOSDRU



FONDUL SOCIAL EUROPEAN
POSDRU
2007-2013



INSTRUMENTE STRUCTURALE
2007-2013