

4.3. Sisteme de gestionare a întreruperilor

4.3.1 Managementul proceselor

Managementul proceselor pentru un sistem real-time este mult mai restrictiv decât pentru un sistem de operare obisnuit pentru ca trebuie sa aloce necesarul de memorie fara intirziere, iar acea memorie trebuie sa se afle in memoria principala pentru a evita latentele datorate swapping-ului, pentru ca schimbarea priorităților de rulare a proceselor influentează comportamentul si predictibilitatea sistemului. In general procese multiple vor fi active simulta si este sarcina sistemului de operare sa imparta resursele disponibile (procesor, memorie) între procese. Alocarea procesorului este una dintre cele mai grele sarcini a unui sistem de operare pentru ca algoritmul respectiv trebuie sa fie in acelasi timp simplu si optim si exista o multitudine de criterii de optim. Algoritmii care încearca sa optimizeze alocarea tinind cont de mai multe criterii sint in general complecsi si necesita cunostinte despre un numar mare de parametri ai proceselor. Sistemele de operare real-time si embedded prefera algoritmii simpli pentru ca necesita un timp de executie scazut si deterministic si de asemenea ocupa putina memorie.

Sistemele de operare de uz general si cele in timp real difera considerabil la algoritmii de planificare. Desi utilizeaza aceleasi principii de baza, aplicarea acestora este diferita datorita criteriilor diferite de performanta. Un sistem de operare de uz general cauta sa maximizeze performanta medie, in timp ce sistemul in timp real tintește spre un comportament deterministic iar un sistem embedded are ca prima prioritate consumul redus de putere si o memorie necesara cit mai mica.

4.3.2 Deservirea intreruperilor

Un sistem de operare trebuie sa fie capabil sa gestioneze nu numai procese conform unor algoritmi de planificare deterministici, dar trebuie sa serveasca si echipamente hardware ca timere, senzori, motoare, dispozitive de comunicatie, discuri etc. Toate aceste periferice solicita atentia sistemului de operare in mod asincron, adica la momentul cind ele doresc serviciile sistemului de operare, acesta trebuie sa se asigure ca este gata sa onoreze aceste cereri. O astfel de cerere este in general semnalizata printr-o intrerupere. Exista doua tipuri de intreruperi:

- intreruperi hardware. Dispozitivul periferic poate activa o linie particulara de semnal a procesorului pe care ruleaza sistemul de operare, semnalizindu-i ca solicita servirea. Rezultatul acestei declansari este ca procesorul isi salveaza starea curenta si executa un salt la programul aflat la o adresa in memorie care a fost asociata intreruperii respective la initializarea sistemului.
- intreruperi software. Diferite procesoare dispun de instructiuni care genereaza software efectele unei intreruperi hardware. Rezultatul unei astfel de instructiuni este de asemenea declansarea executiei unei rutine aflate la o adresa predefinita

4.3.3 Comunicarea si sincronizarea interproces



UNIUNEA EUROPEANĂ

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

Structuri hardware si algoritmi specifici microsystemelor EMBEDDED

O alta responsabilitate a unui sistem de operare este comunicarea interproces (Inter-Process Communication IPC). Comunicarea interproces cuprinde o colectie larga de primitive de programare pe care sistemul de operare le pune la dispozitia proceselor care necesita schimbul de informatii cu alte procese, sau sincronizarea actiunilor lor. Trebuie subliniat din nou, un sistem de operare in timp real trebuie sa asigure o comunicare si sincronizare intr-un mod deterministic.

Pe langa comunicarea sau sincronizarea cu alte procese ce ruleaza pe acelasi calculator, anumite procese necesita de asemenea sa comunice cu alte sisteme, sau cu dispozitive periferice (de exemplu placi cu intrari/iesiri analogice). Acest fapt implica hardware de interfata, de genul linie seriala sau retea, iar acest hardware necesita un driver soft.

4.3.4 Gestionarea memoriei

Cea de-a patra responsabilitate a unui sistem de operare este gestiunea memoriei: diferite procese din sistem ruleaza simultan si fiecare necesita o parte din memoria disponibila, uneori trebuind sa fie plasata la o anumita adresa hardware (pentru dispozitive I/O mapate in memorie). In acest caz sarcina sistemului de operare este multipla:

- sa acorde fiecarui proces memoria necesara – alocare de memorie
- sa mapeze memoria reala intr-o gama de adrese utilizate de diferite procese – mapare de memorie
- sa ia masurile care se impun atunci cind un proces utilizeaza memorie care nu i-a fost alocata (Cauzele cele mai obisnuite – pointeri neinitializati si indexarea tablourilor in afara limitelor). Aceasta ultima caracteristica a gestiunii memoriei este protectia memoriei. Masurile care se impun depind de aplicatie, dar in general se apeleaza la solutia cea mai simpla – oprirea procesului si notificarea utilizatorului.

4.3.5 Constrângeri in proiectarea sistemelor de operare

Proiectantii de sisteme de operare, de uz general sau de timp real, se confrunta cu o serie de constrangeri si sint obligati sa faca anumite alegeri.

Spatiu nucleu versus spatiu utilizator versus spatiu timp-real

Majoritatea procesoarelor moderne pe 16 biti si mai mult permit rulara programelor pe doua nivele de protectie hardware. In Linux acestea se numesc spatiul nucleu si spatiul utilizator. Acesta din urma are protectie mai puternica la accesele gresite la memoria fizica a dispozitivelor I/O, evident accesul la hardware fiind astfel penalizat de latente mai mari decit in cazul proceselor din spatiul nucleu ce acceseaza acelasi hard. Versiunile in timp real ale Linux adauga un al treilea nivel, spatiul timp-real; acesta nu este alceva decit o parte a spatiului nucleu, dar utilizat intr-un mod special.

Nucleu monolitic versus micronucleu

Un nucleu monolitic are serviciile sistemului de operare (drive de dispozitiv, stive protocoale de retea, sisteme de fisiere) rulind in modul privilegiat al procesorului. Micronucleul,



UNIUNEA EUROPEANĂ

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

Structuri hardware si algoritmi specifici microsistemelor EMBEDDED

pe de alta parte, utilizeaza modul privilegiat de lucru al procesorului numai pentru serviciile cu adevarat de nucleu de sistem de operare (gestiunea si planificarea proceselor, comunicarea interproces, deservirea intreruperilor, gestiunea memoriei) iar celelalte servicii ale sistemului de operare si driverele de dispozitiv ruleaza ca procese "normale" in modul neprivilegiat. Nucleul monolitic este mai usor de facut mai eficient (serviciile sistemului se executa in intregime fara a comuta din modul privilegiat in cel neprivilegiat si invers) dar un micronucleu se blocheaza mult mai rar (o eroare intr-un driver de dispozitiv care ruleaza in mod neprivilegiat are sanse mult mai scazute sa cauzeze o oprire a sistemului decit aparitia unei erori la executia in modul privilegiat). Sistemele de operare UNIX, Linux si Windows NT(2000, XP) sint nucleu monolitic; QNX, VxWorks micronucleu. Linux, la fel ca anumite distributii UNIX comerciale, permite schimbarea statica sau dinamica a anumitor servicii in nucleu: functionalitate suplimentara se poate obtine prin incarcarea unui modul, dar acesta devine apoi parte a nucleului monolitic. Un nucleu Linux minimal (ce include gestiunea memoriei, comutarea proceselor si servicii de timp) ocupa citeva sute de kiloocteti; aceasta dimesiune se apropie de cea a sistemelor embedded. Cu toate acestea, din ce in ce mai multe sisteme embedded ocupa mai mult de 1 MB memorie, pentru ca necesita si stive de protocoale de retea si diferite functiuni de comunicare.

Nucleu preemptiv sau non-preemptiv

Sistemul de operare Linux a fost initial non-preemptiv – un proces din spatiul nucleului nu poate fi oprit de un alt proces din acelasi spatiu, sau de catre procese utilizator. Nucleul este "blocat" in timpul executiei unei functii nucleu. Utilizarea mecanismului blocarilor face proiectarea nucleului mai simpla, dar introduce latente nedeterminate intolerabile intr-un sistem de timp real.

Din versiunea 2.5, Linux dispune de un mecanism de blocare de granularitate mai fina si a devenit intr-o masura mai mare preemptiv; exista inca un mecanism de blocare a nucleului, numit `kernel_flag` in codul sursa Linux, dar subsistemele independente (retea, operatiuni I/O cu discurile) dispun de propriile mecanisme de blocare.

Scalabilitatea

Blocarea de granularitate mai fina este utila pentru scalabilitate, dar de obicei un efort suplimentar pentru sistemele cu un singur procesor. Sistemul de operare Solaris este un exemplu de granularitate foarte fina si scalabilitate ridicata cu performante scazute pe sisteme PC "low-end" monoprocessor. Scalabilitatea este o problema de importanta scazuta pentru aplicatiile in timp real, pentru ca obiectivele sint foarte diferite: dorinta unui sistem scalabil este de a imparti o sarcina de lucru mare transparent intre mai multe procesoare disponibile, in timp ce la un sistem in timp real se urmareste controlul tuturor aspectelor intr-un mod strict determinist.

Gestiunea memoriei si memoria partajata

Memoria virtuala si alocarea/dealocarea dinamica a paginilor de memorie sint printre cele mai utilizate servicii de gestiune a memoriei a unui sistem de operare de uz general. Totusi, gestiunea memoriei implica efort suplimentar iar anumite procesoare simple nu dispun de suport hardware pentru gestiunea memoriei. Pe aceste procesoare (care stau la baza unui numar mare de sisteme embedded) toate procesele impart acelasi spatiu de memorie astfel ca dezvoltatorii de aplicatii trebuie sa aiba grija de utilizarea corecta a memoriei. De asemenea anumite sisteme in timp real (de exemplu RTLinux) au acelasi spatiu de memorie pentru toate



UNIUNEA EUROPEANĂ

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

Structuri hardware si algoritmi specifici microsistemelor EMBEDDED

procese, chiar daca procesorul dispune de suport pentru gestiunea memoriei, pentru ca acest lucru permite obtinerea de cod mai eficient.

Dedicat sau general

Pentru multe aplicatii este mai eficient sa nu se utilizeze un sistem de operare comercial sau disponibil gratuit, ci sa se scrie unul optimizat pentru o anumita sarcina. Exemple pot fi sistemele de operare pentru telefoane mobile sau asistenti digitali personali. Sistemele de operare standard ar fi mult prea mari si nu ar dispune de suportul specific pentru prelucrarea digitala de semnal (necesara recunoasterii vorbirii si scrierii) tipice pentru aceste aplicatii. Anumite aplicatii nici macar nu necesita un sistem de operare. Decizia aici se ia intre costul dezvoltarii si portabilitatea scazuta, pe de-o parte, fata de costul unor sisteme embedded mai mici si mai ieftine.

Sistem de operare sau biblioteci runtime

Programele de aplicatie pot utiliza primitive "de nivel scazut" pentru a-si construi functionalitatea. Aceasta poate fi oferita de sistemul de operare (prin apeluri sistem) sau prin limbaje de programare (prin functii primitive ale limbajului sau bibliotecii). Limbaje ca C++, Java sau Ada ofera foarte multa functionalitate in acest fel: gestiunea memoriei, suport pentru fire de executie, sincronizarea proceselor, tratarea exceptiilor etc. Aceasta functionalitate este strinsa in asa-numitul element runtime. Avantajele utilizarii unui modul runtime sint: interfata sa este portabila peste diferite sisteme de operare si ofera solutii de-a gata si/sau sigure pentru probleme comune. Dezavantajele sint ca modulul runtime este in general de dimensiuni mari, non-determinist in timp de executie si greu configurabil. Aceste dezavantaje devin foarte importante in contextul aplicatiilor in timp real si embedded.

Bibliografie:

1. Istvan Sztojanov, Sever Pașca, Elisabeta Buzoianu, Aplicații hardware și software cu microcontrolerul PIC12F675, Editura Cavallioti, ISBN 978-973-7622-54-9, Bucuresti 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
AMFOSDRUFONDUL SOCIAL EUROPEAN
POSDRU
2007-2013INSTRUMENTE STRUCTURALE
2007-2013