Proiectarea și implementarea unui micronucleu

Alexandr VULPEA, Dumitru CIORBĂ
Universitatea Tehnică a Moldovei
alexandr.vulpea@ati.utm.md, dumitru.ciorba@ati.utm.md

Abstract – **Activitățile unui utilizator pe calculator sunt determinate în întregime de sistemul de operare. Prin urmare acesta ar trebui să fie ofere încredere și să funcționeze perfect. Cu regret, chiar și sistemele de operare moderne, cum ar fi Windows și Linux, nu reușesc să ofere nivelul superior de calitate, deoarece suferă de erori și defecte fundamentale de design. Nucleul lor monolitic are tendința de a fi supraîncărcat cu funcționalități care executabile la cel mai înalt nivel de privilegii. Astfel, un driver implementat de o persoană terță care rulează la nivelul nucleului, poate provoca un defect ce poate induce stare critică sistemului. În ceea ce urmează se va descrie o alternativă de proiectare a sistemelor de operare prin tehnica de micronucleu care elimină anumite problemele cu care se confrunta un nucleu monolitic.**

Cuvinte cheie – **micronucleu, modularitate, performanță, securitate.**

I. INTRODUCERE

Un micronucleu este un nucleu minim al unui sistem de operare. În forma sa canonică, oferă doar minimul de funcționalități pentru a implementa serviciile unui nucleu, spre deosebire de un nucleu monolitic în care toate serviciile sunt implementate în nucleu. În figura 1 este prezentată structura generală a unui micronucleu, elementele primare care fac parte din nucleu sunt: managementul memoriei, managementul procesorului și comunicarea între procese.

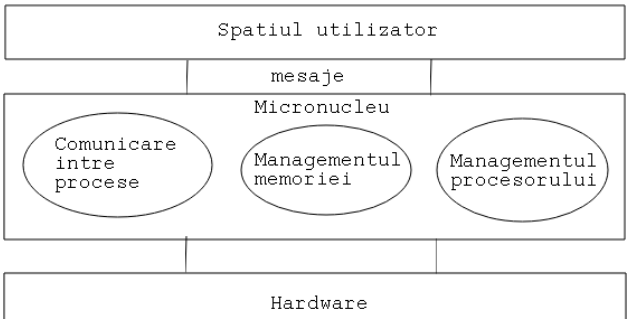

Fig. 1 - Structura unui micronucleu.

Managementul memoriei ține cont de spațiu de memorie disponibil, alocând memorie proceselor și dealocând-o acestora când nu mai au nevoie ea.

Procesele adesea comunică cu alte procese, menționându-se trei aspecte:

- Cum un proces transmite mesaje altui proces;
- Cum are loc coordonarea transmiterii reciproce de mesaje dintre două sau mai multe procese;
- Cum se *rezolvă* dependențele dintre procese, în procesul de comunicare.

Acestea și alte aspecte legate de comunicare sunt atribuite unui element dedicat al nucleului, *mecanismul de comunicare dintre procese*.

Managementul procesorului este mecanismul ce permite unui proces să folosească procesorul în timp ce executarea unui alt proces este în așteptare din cauza indisponibilității oricărei resurse (de exemplu determinate de operații de I/O), asigurând utilizarea eficientă a procesorul. Există multipli algoritmi ce pot fi folosiți la implementarea acestui mecanism, cum ar fi: primul venit – primul servit, cel mai scurt job - primul, programarea prioritară, coadă de așteptare pe mai multe niveluri etc.

Toate celelalte servicii furnizate de un nucleu monolitic (cum ar fi *managementul rețelelor*) sunt implementate în aplicații care rulează în spațiul utilizatorului denumite servere [1]. În figura 2 este prezentat o structura tipică a unui sistem de operare bazat pe micronucleu.

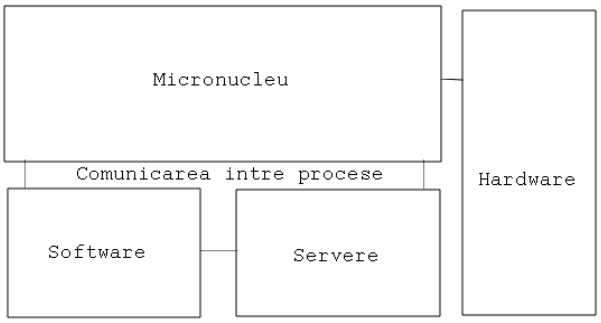

Fig. 2 – Structura unui sistem de operare bazat pe micronucleu.

Servere sunt aplicații ca oricare altele, ceea ce înseamnă că sistemul poate fi modificat prin pornirea sau oprirea acestor servere. Într-un sistem tradițional un așa mecanism se află în nucleu, astfel este necesar de recompilat un nucleu pentru a adăuga/elimina aceste servere, ceia ce depășește capacitățile unui utilizator obișnuit.

II. APELURILE DE SISTEM

Interfața dintre sistemul de operare și aplicațiile de nivelul utilizatorului este definită de un set de instrucțiuni extinse care sunt oferite de sistem. Aceste instrucțiuni sunt cunoscute ca apeluri de sistem (eng.: system calls)[2]. Caracterul apelurilor de sistem este determinat de modul în care un procesor se află la un moment dat: *nucleu* sau *utilizator*.

Cînd procesorul se află în modul nucleu, procesul executat poate accesa orice adresă din memorie și orice componentă hardware, prin urmare, modul nucleu este un mod privilegiat. Dacă un proces se blochează în modul nucleu, întregul sistem va fi oprit.

Când procesorul se află în modul utilizator procesele executate nu au acces direct către memoria și resursele hardware ale calculatorului. În cazul în care un proces care rulează în modul utilizator se blochează, sistemul va fi într-o stare sigură.

Pentru a accesa o resursă hardware, un proces ce se execută în modul utilizator trebuie să solicite nucleului să-i ofere acces spre această resursă, aceasta realizându-se prin intermediul apelurilor de sistem [3]. Odată ce un proces face un apel de sistem, procesorul trece din modul utilizator în mod nucleu. Această schimbare e foarte importantă pentru *micronucleu*: fiind o acțiune critică pentru acest tip de arhitectură, datorându-se faptului că driverele, sistemul de fișiere, managementul memoriei și alte activități ce țin de funcționarea unui sistem de operare se execută în modul *utilizator*. Astfel pentru a obține un nivel înalt de performanță într-un micronucleu, este necesar de implementat un mecanism de comunicare dintre procese cât mai eficient.

III. Comunicarea interprocese

Proiectarea unui micronucleu implică migrarea unor servicii din spațiul nucleului în spațiul utilizatorului. În rezultat, nucleul comunică mult mai des cu procesele. Astfel performanța comunicării dintre procese are un rol foarte important. Există tehnici care sporesc performanța comunicării dinte procese [4], cum ar fi:

- Transmiterea mesajelor scurte – un procent ridicat de mesaje sunt foarte scurte, prin transferarea acestor mesaje în registre se poate obține o îmbunătățire a performanței de două ori.
- Copierea mesajelor mari – prima generație de micronucleu transferă mesaje între procese printr-o dublă copie, din spațiul procesului A în spațiului nucleului în spațiul procesului B, ce duce la un impact mare asupra performanței [5, cap. 19]. O soluție ar fi „împrumutarea" temporară a regiunii țintă către expeditorul mesajului [6, cap 3.3], rezultând o îmbunătățire a performanței de două ori.

IV. Considerații arhitecturale

Șabloanele arhitecturale pot defini caracteristici de bază ale unui sistem, mentenanța acestuia, reutilizarea componentelor etc. Prin urmare este important de a alege șabloanele cât mai potrivite pentru sistemul proiectat. În cazul unui micronucleu, șablonul/șabloanele ales/alese trebuie sa răspundă la careva întrebări. Care sunt componentele care alcătuiesc sistemul? Cum interacționează aceste componente între ele? Cum aceste componente își îndeplinesc funcționalitatea și tot odată rămân decuplate una față de alta? Ies în evidență următoarele șabloane:

- Șablonul Layers – acest șablon propune un sistem împărțit în straturi aranjate vertical. Fiecare strat furnizează servicii doar statului de mai sus și utilizează servicii stratului de mai jos.
- Șablonul Indirection Layer – este un șablon care oferă un nivel între componenta care necesită un serviciu și componenta care oferă serviciul.
- Șablonul Explicit Invocation – este un șablon de tip client – server, când o componenta necesită un serviciu acesta dialoghează cererea către un mecanism a cărui rol e să retransmite cererea către o componentă care este disponibilă să execute această cerere.

Șablonul *Layers* poate fi definitoriu unui micronucleu, fiind un șablon care răspunde la întrebările enumerate mai sus și tot odată rămâne simplu de implementat, iar simplitatea este unul din punctele esențiale de atins ale unui micronucleu. Alt motiv de ce acest șablon se potrivește cel mai bine, ar fi structura sa ierarhică: o componentă poate folosi serviciile de la componenta din nivelul inferior și propune servicii componentei din nivelul superior. Șablonul *Indirection Layer* s-ar putea folosi la implementarea apelurilor de sistem, care de fapt reprezintă canalul de comunicare dintre componente nucleului.

V. Concluzii

Un micronucleu oferă doar un set minim de abstractizări care rulează la cel mai înalt nivel de privilegii. Funcționalități adiționale fiind disponibile în forma serverilor care rulează în spațiul utilizator, astfel, prin divizarea sistemului de operare în părți independente mai mici, sistemul devine mai puțin complex și mai robust, ca rezultat părțile mai mici sunt mai ușor de gestionat și ajută la izolarea defectelor. Redox [7] aduce toate inovațiile limbajului Rust în micronucleu, astfel se elimină erori care persistă în alte limbaje cum ar fi comportamentul nedefinit de la cel mai privilegiat nivel, modul nucleu.

BIBLIOGRAFIE

[1] Operating System Design/Kernel Architecture/Microkernel – 13 octombrie 2013 https://en.wikibooks.org/wiki/Operating_System_Design/Kernel_Architecture/Microkernel;

[2] **A. Tanenbaum, A. Woodhull.** *Operating Systems Design and Implementation (3rd Edition).*

[3] System Calls – citit 5 august 2017 http://www.studytonight.com/operating-system/system-calls

[4] Interprocess communication – citit 5 august 2017 http://www.cs.cornell.edu/Info/People/ulfar/ukernel/ukernel.html#current

[5] **Arpaci-Dusseau, Remzi H.; Arpaci-Dusseau, Andrea C.** Operating Systems: Three Easy Pieces.

[6] **R. Aigner.** *Communication in Microkernel-Based Operating Systems.* https://os.inf.tu-resden.de/papers_ps/aigner_phd.pdf

[7] Redox microkernel, https://github.com/AleVul/kernel