Universitatea Politehnica București

Facultatea de Electronică, Telecomunicații și Tehnologia Informației

Proiect

Sisteme de Operare Avansate

GESTIONAREA PERIFERICELOR UTILIZÂNDU-SE KERNEL-UL

Coordonator:

Conf. dr. ing. Ştefan Stăncescu

Student:

Mimi-Nicoleta Ruse

Master IISC, anul I

2016-2017

CUPRINS

1. Introducere

Unix este cel mai vechi sistem de operare multitasking ce poate fi folosit atât pentru mai multe tipuri de calculatoare, cât și pentru rețele de calculatoare. Sistemul de operare este interfața dintre utilizator și hardware-ul calculatorului și asigură dezvoltarea de aplicații, execuția programelor, accesul la dispozitivele de intrare/ieșire, accesul la fișiere și la sistem. [1]

Unix poate funcționa în același timp pe microcalculatoare și pe supercalculatoare. Acesta a stat la baza Internetului prin includerea setului de servicii pentru transferul informațiilor între calculatoare (TCP/IP). [1]

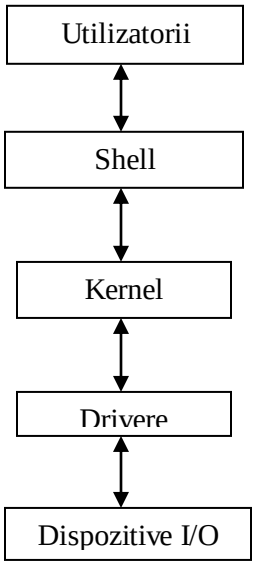
În ziua de astăzi s-au conceput mai multe variante ale acestui sistem de operare, cum ar fi: System V, Solaris, AIX, ULTRIX, OSF, HP-UX, IRIX, MaxOS X, BSD și Linux care a fost elaborat în 1992 și este free software. [1]

În cele ce urmează se vor prezenta câteva aspecte legate de arhitectura Unix-ului, istoria, structura și compilarea kernel-ului împreună cu câteva avantaje ale folosiri kernel-ului compilat și, de asemenea, modul de gestionare a dispozitivelor periferice.

2. Structura și arhitectura sistemului UNIX

Unix este format din mai multe nivele, fiecare având câte o serie de componente. Pe lângă kernel, Unix mai conține interpretorul de comenzi (shell) fiind și interfața sistemului de operare cu utilizatorul, setul de biblioteci tipice, aplicații canonice și codul sursă utilizat pentru portabilitate.

Kernel-ul este intermediarul dintre echipamentele fizice ale sistemului și interfața dată de apelurile de sistem, așa cum se poate vedea și în Figura 2.1. El asigură supervizarea și gestionarea resurselor sistemului de calcul, resurse cum ar fi memoria internă, dispozitivele de intrare/ieșire, etc. [1]



În shell sunt introduse comenzile și pornește automat în momentul în care utilizatorul se autentifică, dar distribuțiile mai noi vin cu un server grafic care controlează tastatura, mouse-ul și sistemul de ferestre.

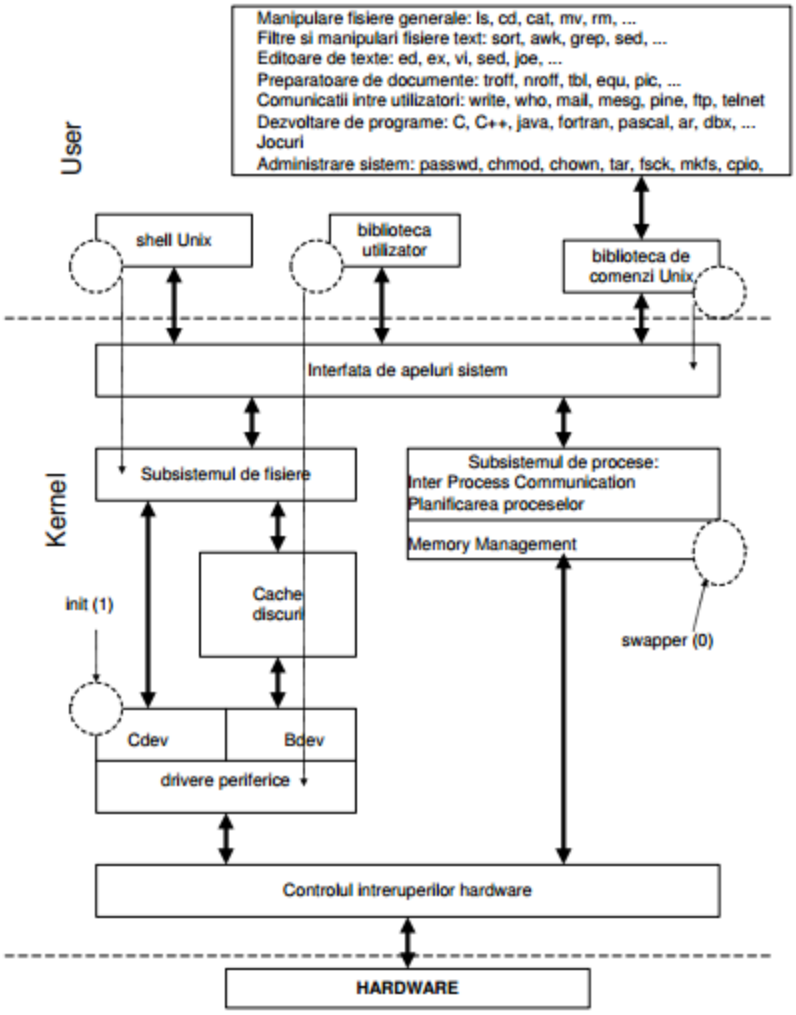


Figura 2.2 Arhitectura Unix-ului [3]

În Figura 2.2 se poate observa că Unix este format din 2 părți. User-ul poate accesa sistemul folosind shell-ul, prin comenzi standard sau cu ajutorul programelor proprii compilate și completate cu subprogramele din biblioteca Unix.

Una dintre diferențele dintr Unix și Windows este faptul că primul permite accesul simultan mai multor utilizatori la aceleași resurse, în timp ce Windows permite accesul la resurse doar unui utilizator la un moment de timp.

3. Kernel

3.1. Istoric

Primul Kernel a fost scris în 1991 de Linus Trovalds (kernel 0.02).

Kernel-ul este componenta sistemului de operare responsabilă cu gestionarea resurselor calculatorului, el făcând legătura între aplicații și hard. Kernel-ul poate fi atât monolitic, cât și microkernel. Kernel-ul monolitic folosește aceeași zonă de memorie ca și sistemul, iar în cazul microkernel-ului, acesta susține numai funcționalitățile de bază ale rulării de servere și în cadrul acestei arhitecturi se permite rularea, pe același kernel, a mai multor sisteme de operare. [4]

La ora actuală există 2 tipuri de versiuni pe care dezvoltatorii le distribuie, dar versiunea cu a doua cifră pară este cea stabilă, cealaltă având suficient de multe bug-uri.

Fiind distribuit sub GNU General Public Liceense, Kernel-ul poate fi atât distribuit, cât și modificat fără restricții, permițându-le programatorilor să îmbunătățească codul sursă, ceea ce a dus la o dezvoltare rapidă. [4]

Se recomandă să avem ultima versiune de kernel, dar de la apariția unei versiuni la alta este o perioadă de timp destul de mare, timp în care apar patch-uri care rezolvă bug-urile din versiunea curentă.

Există riscul ca un driver să fie scris doar pentru anumite versiuni de Kernel, dar pe www.kernel.org sunt ținute toate versiunile.

3.2. Structura

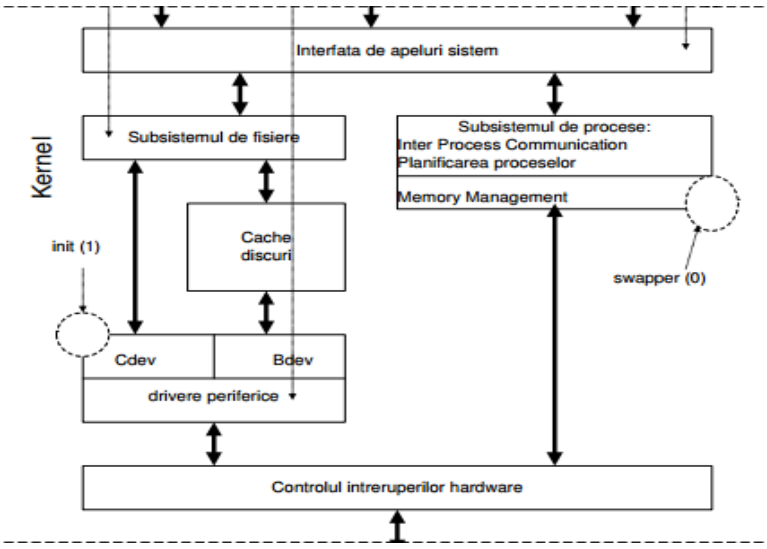


Figura 3.2.1 Structura nucleului [3]

În Figura 3.2.1 este reprezentată structura Kernel-ului.

Așa cum se poate observa și în figură, interfața cu exteriorul se realizează prin componenta numită "Interfața de apeluri sistem" care este realizată din funcții apelabile în limbajul C.

Gestionarea tuturor fișierelor se realizează prin "Subsistemul de fișiere", fiind cea mai importantă parte a Unix-ului.

Optimizarea accesului se realizează cu ajutorul blocului "Cache discuri", întrucât acesta reține porțiunile de pe disc folosite recent și solicitate mai des.

Init este un proces cu PID (Process Identification) 1 și reprezintă părintele proceselor utilizatorilor. Șirul de procese de la intrarea în sistem a utilizatorului începe de la acest proces care este activ în permanență.

Legătura dintre echipamentul hardware și Kernel se realizează indirect prin componentele Cdev și Bdev. Cdev este ansamblul de rutine prin care perifericele de tip caracter sunt legate de sistem, iar Bdev este tot un ansamblu de rutine, dar prin ele se realizează legătura dintre perifericele de tip bloc și sistem.

A doua componentă importantă, după "Subsistemul de fișiere", este "Subsistemul de procese" care se ocupă planificarea proceselor.

Swapper este un proces cu PID-ul (Process Identification) 0 și se ocupă cu extinderea virtuală a spațiului memoriei interne, păstrând pagini de memorie internă nefolosite. El poate evacua temporar alte pagini interne și poate pune în locul lor paginile solicitate.

3.3. Pregătirea pentru compilarea Kernel-ului

Pentru compilarea kernel-ului trebuiesc îndeplinite anumite cerințe și înainte de compilare trebuie verificat mereu dacă acestea sunt satisfăcute. De exemplu, cerințele hardware sunt:

- Memoria RAM minimă, pentru ultimele versiuni, trebuie să fie între 128-256 MB.
- 1 GB spațiul liber pe hard-disk întrucât spațiul ocupat pe disc poate depăși 500 MB.

Cunoașterea hardware-ului, alegerea opțiunilor de compilare și alegerea driver-elor sunt utile pentru un kernel rapid.

3.4. Compilarea Kernel-ului

Este mai utilă compilarea Kernel-ului decât folosirea unei versiuni precompilate. Unele dintre cele mai importante motive sunt:

- La folosirea versiunilor precompilate există riscul incompatibilității cu hardware-ul sau poate lipsi suportul pentru unele componente.
- Dacă suportul hardware este mai mult decât ar fi necesar, Kernelul precompilat consumă prea multă memorie, iar timpul de lansare ar fi mai lung decât timpul optim. Astfel, dacă ne compilăm noi kernel-ul, avem posibilitatea deselectării elementelor de care nu avem nevoie sau putem activa noi elemente.
- Prin compilarea Kernel-ului putem imbunătății performanțele setând modulele astfel încât să se potrivească bine cu sistemul hardware.

Pentru compilarea Kernel-ului va trebui să descărcăm și să instalăm sursele pentru el printr-o serie de comenzi: Ex. apt-get install kernel-source-2.6.8, cd /usr/src, tar –xyzf kernel-source-2.6.8.tar.bz2, cd kernel-source-2.6.8. După aceasta va trebui să creăm un fișier pentru configurare. [5]

Pentru a ne asigura că nu mai avem fișiere de configurare de la vechiul kernel, putem rula comanda make mrproper, apoi putem începe configurarea care se poate face în mai multe feluri. De exemplu, după comanda make config, utilizatorul trebuie să răspundă la un set de întrebări despre sistem, iar după make menuconfig utilizatorul poate selecta elementele pe care dorește să le activeze. Prin comenzile make xconfig sau make gconfig, cofigurarea se face sub un server, iar prin rularea comenzii make defconfig se vor folosi fișiere default. [5]

Componentele pe care dorim să le compilăm pot fi atât integrate în Kernel fiind încărcate în memorie, cât și ca modul, fiind încărcată doar când este nevoie de ea, dar sunt și componente care nu pot fi compilate decât ca integrate sau decât ca modul.

4. Gestiunea dispozitivelor periferice

Cu ajutorul dispozitivelor periferice se poate realiza comunicarea sistemului de operare cu mediul exterior, prin intermediul conectorilor din placa de bază, iar conexiunea plăcilor de bază se realizează prin sloturi sau prin magistrale (PCI, AGP, USB, etc.).

Ele interacționează cu sistemul de operare prin primirea și efectuarea comenzilor venite din partea acestuia. Dispun și de un controller, controllerul având un număr de registre folosite pentru comunicare. Sistemul de operare nu vede decât interfața către utilizator, interfață care poate fi diferită de cea a dispozitivului în sine. [9]

Dispozitivul periferic cel mai important este HDD-ul (hard disk-ul), fiind un dispozitiv care stochează permanent date, fiind considerat suportul pentru sistemele de fișiere.

4.1. Tipuri de dispozitive periferice

Dispozitivele periferice pot fi împărțite în 2 categorii, apartenența lor la una dintre ele putându-se vedea pe prima coloană a rezultatului rulării comenzii ls –l. Aceste categorii sunt:

- Dispozitive bloc fiind dispozitivele care stochează informația în blocuri cu dimensiuni fixe, fiecare având adresă proprie. Dispozitivele de stocare fac parte din această categorie (hard-disk-ul, CD-ROM-ul sau DVD-ROM-ul, dispozitivele virtuale și cele de memorie).
- Dispozitive caracter, fiind acele despozitive care acceptă și oferă fluxul de caractere fără să țină cont de structura de bloc. Ele transmit pe rând câte un caracter, fiind folosite pentru transmiterea fluxurilor de date (tastatura).

Pe lângă aceste 2 tipuri de dispozitive, mai există și speudo-dispozitive, care sunt generate de kernel și nu corespund componentelor hardware, ci sunt echivalentelor dispozitivelor virtuale, dar cu același comportament cu cele de tip caracter. [10]

4.2. Driver-ul și modulele dispozitivelor periferice

Kernel-ul ascunde dispozitivele reale de intrare/ieșire sub forma fișierelor aparente, întrucât operațiile de I/O sunt văzute de sistem ca niște operații cu fișiere. Utilizatorul sistemlui de operare poate comunica foarte bine cu orice dispozitiv de intrare/ieșire atașat folosindu-se de driver-ul acestuia, fiecare dispozitiv trebuind să aibă un astfel de program care este realizat de producătorii de hardware împreună cu programatorii de sistem. [10]

Acest driver, pentru a putea rula în kernel, trebuie să fie inclus în sistemul de operare, sistemele mai recente neacceptând rularea lor în afara kernel-ului, sau pot fi disponibile în module. [9]

Listarea modulelor se poate face prin comanda lsmod, comandă ce afișează numele modulului, spațiul pe care acesta îl ocupă în memorie și, de asemenea, atât numele, cât și numărul altor module ce depind de el. În funcție de ce dispozitive hardware sunt conectate, modulele se încarcă automat, totuși ele pot fi încărcate și manual prin comanda insmod sau modprobe, aceasta din urmă încărcând și modulele care depind de cel dorit. Pentru a putea folosi comanda insmod, este nevoie să se cunoască și calea unde se află modului, cale ce va fi folosită ca argument al comenzii, în timp ce pentru comanda modprobe este nevoie doar de numele modului. Descărcarea lui se poate face prin comanda rmmod, sau modprobe –r, comenzi urmate de numele modului care se dorește a fi descărcat. [10]

Prin utilizarea unor comenzi specifice, se pot afla atât informații generale despre un anumit dispozitiv, cât și starea în care acesta se află în momentul respectiv. De exemplu, pentru a afișarea atât a magistralelor, cât și a dispozitivelor USB prezente, se va utiliza comanda lsusb sau, dacă se dorește aflarea dispozitivelor conectate prin magistrala PCI, se va utiliza comanda lspci. [10]

În directorul /dev din Linux, există câte un fișier pentru fiecare dispozitiv, fișiere pe care le crează udev (managerul de dispozitive). Acest manager primește notificări de la kernel în momentul în care apare sau dispare un dispozitiv și, în funcție de schema care se utilizează, el poate să numească dispozitivul cu următorul nume disponibil. După ce dispozitivului i s-a dat un nume, se rulează un program care pune sistemul de fișiere de pe dispozitiv într-un subdirector. Tot cu acest manager de dispozitive, se pot modifica permisiunile pentru fișierele create. De exemplu, se configurează ca, pentru scannerele introduse în sistem, fișierul care se crează să aparțină doar acelui grup care conține numele scanner-ului respectiv, astfel, vor putea lucra cu

acel fișier doar utilizatorii care aparțin acelui grup, dar totodată, un utilizator poate să aparțină mai multor grupuri în același timp. [10]

4.3. Gestionarea

Intrarea/ieșirea se poate face în mai multe feluri. Fie prin așteptare ocupată, caz în care se emite un apel de sistem pe care kernel-ul îl traduce în apel de procedură către driver-ul corespunzător, urmând ca acesta să înceapă procesul de intrare/ieșire și să aștepte să vadă dacă se realizează sau dacă dispozitivul este ocupat, iar după finalizare, dacă este necesar, se pun datele, fie prin metoda în care driver-ul pornește dispozitivul și îi cere să dea o întrerupere atunci când termină, urmând ca driver-ul să se întoarcă. [9]

Kernel-ul este informat despre activitatea dispozitivelor periferice prin cererile de întrerupere care sunt trimise dacă apar defecte ale unui dispozitiv sau chiar la începutul și sfârșitul unui transfer de date. Pentru a se determina sursa unei astfel de cereri cel mai simplu este să se interogheze toate perifericele, dar cel mai eficient este vectorul de întreruperi deoarece fiecare element al său conține adresa rutinei de tratare a excepției. Acest vector este inițializat odată cu încărcarea sistemului, iar conținutul său poate fi modificat în timpul execuției proceselor.

Pașii pentru tratarea întreruperilor provocate de periferice sunt: inițial se determină de unde a apărut întreruperea, apoi se determină adresa rutinei care va trata această întrerupere și se salvează parametrii procesului care este deja în execuție pentru a se putea relua procesul. După salvarea parametrilor, controlul este preluat de rutină, apoi se transferă datele. După terminarea transferului, controlul este preluat de procesul care fusese întrerupt.

Fișierele de intrare/ieșire pot fi accesate pe trei nivele: pe primul nivel este accesarea prin utilitare, pe al doilea nivel este accesarea prin unele funcții de bibliotecă și pe cel de-al treilea nivel este accesarea prin apelurile de sistem, folosind comenzi de tipul: read/write, open/close, etc. Folosindu-se apelurile de sistem, procesul care rulează în acel timp este oprit pe toată durata execuției apelului, urmând a fi repornit la terminarea acestuia.

În gestionarea acestor dispozitive se folosesc tampoanele cache (buffere), zone din memoria internă, care sunt structuri de date software, organizate din două liste dublu înlănțuite. Aceste tampoane cache ajută la reducerea numărului de operații de I/O.

4.4. Buffer-ul

Buffer-ele sunt zone cu probleme atunci când vine vorba de dispozitivele periferice, din mai multe motive. De exemplu, se consideră un proces prin care se citesc datele de pe un modem. O posibilitate este ca utilizatorul să facă un apel de sistem (apel de citire) pentru un singur caracter, dar fiecare caracter produce o intrerupere. Procesul citește un alt caracter numai după ce cel precedent a fost pus undeva. Pentru ca acest lucru să fie posibil, este necesar ca procesul să fie pornit pentru fiecare caracter de intrare. [9]

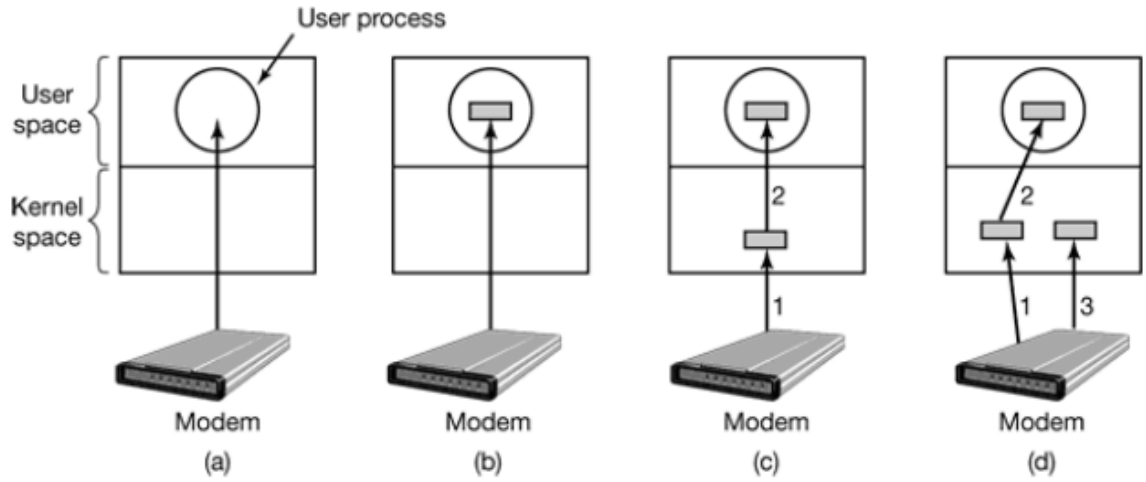


Fig. 4.1 Metode de citire a datelor de pe un modem [9]

În figura 4.1 sunt reprezentate 4 metode de citire a datelor de pe un modem. În prima imgine din partea stângă este reprezentat procesul fără folosirea buffer-ului, în imaginea a doua se introduce buffer-ul în spațiul de utilizare, în a treia imagine buffer-ul este pus în kernel și apoi caracterul este copiat în spațiul de utilizare, iar în ultima imagine este reprezentat buffer-ul dublu în kernel.

Utilizare buffer-ului este eficientă, dar are dezavantajul că, în cazul în care buffer-ul este paginat atunci când primește un caracter, buffer-ul ar putea fi blocat în memorie. În cazul în care apar mai multe procese, paginile încep să se blocheze în memorie, iar numărul de pagini rămase disponibile se micșorează, astfel scăzând performanța. Acest lucru poate apărea în cazul în care buffer-ul este în spațiul de utilizare. [9]

O abordare mai eficientă este crearea unui buffer în kernel și punerea caracterelor în el. În momentul în care buffer-ul se umple, pagina care se dorește este adusă, iar buffer-ul este copiat in spațiul de utilizare printr-o singură operație. Totuși și această metodă are dezavantajul că, atât

timp cât buffer-ul este plin, sau este adus de pe disc, nu se va putea pune un caracter nou venit. O alternativă este un al doilea buffer în kernel care se va folosi atunci când primul buffer s-a umplut și invers, primul buffer este folosit când al doilea este ocupat. [9]

5. Concluzii

- Diferența principală dintre Unix și Windows constă în faptul că Unix-ul permite accesul simultan a mai multor utilizatori la aceleași resurse, pe când Windows-ul nu permite acest lucru.
- Kernel-ul face legătura dintre aplicații și partea hard a sistemului, fiind responsabil de gestionarea resurselor calculatorului.
- Compilarea Kernelului este mai avantajoasă decât folosirea versiunii precompilate întrucât se elimină riscul incompatibilității cu partea hardware, folosirea memoriei poate fi optimizată, iar performanțele sistemului pot fi optimizate.
- Dispozitivele periferice ajută la comunicarea sistemului cu mediul exterior, ele interacționând cu acesta prin intermediul comenzilor primite și efectuate.
- Comunicare cu dispozitivele de intrare/ieșire se realizează prin intermediul driver-elor acestora, driver-e ce sunt realizare de producători. Acestea trebuie să fie incluse în sistemul de operare sau pot fi disponibile în module.
- Intrarea/ ieșirea se poate realiza atât printr-un apel de sistem, apel interpretat de kernel ca fiind un apel de procedură, cât și prin pornirea dispozitivului de către driver.
- Pentru ca gestionarea să fie eficientă a dispozitivelor periferice, este de dorit să se introducă 2 buffere în Kernel, fiecare dintre acestea fiind folosit în momentul în care celălalt este ocupa și astfel se reduce timpul de așteptare până la golirea buffer-ului care s-a umplut.

6. Bibliografie

[1] http://ebooks.unibuc.ro/informatica/Seiso/3.3.htm

[2] https://ro.wikipedia.org/wiki/UNIX

[3] http://www.cs.ubbcluj.ro/~rares/course/os/res/lectures/01UnixVersiuniStructuri.pdf

[4] http://ro.tldp.org/html/baza/kernelfaq/i1.html

[5] http://elf.cs.pub.ro/so2/wiki/laboratoare/lab_compilare

[6] http://andrei.clubcisco.ro/cursuri/1uso/lab2005/L11-K.pdf

[7] http://carment.ase.ro/so/cap2.5.UNIX-kernel.pdf

[8] http://upm.ro/intranet/ecalin/cd_educational/cd/s_o/sist_op.html

[9] MODERN OPERATING SYSTEMS SECOND EDITION, Andrew S. Tanenbaum

[10] Utilizarea sistemelor de operare, Răzvan Rughiniș, Răzvan Deaconescu, George Milescu, Mircea Bardac