**Universitatea Tehnică a Moldovei**

**Lucrare de laborator 4**

**LA DISCIPLINA:** PROGRAMAREA IN RETEA

**CU TEMA:** Aplicație Client-Server TCP

**Student:**

Lupei Nicolae FI-171

**Profesor:**

Boldumac Oleg

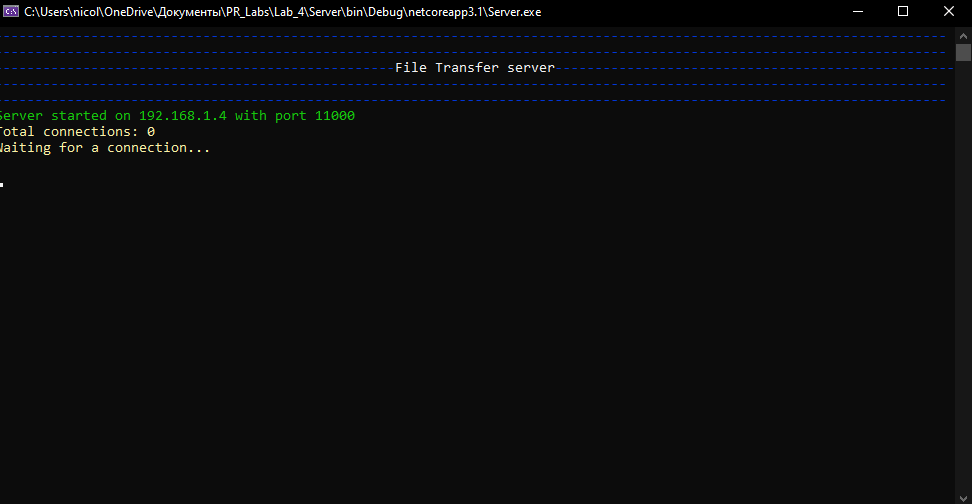
Chișinău – 2020

**Mersul lucrarii:**

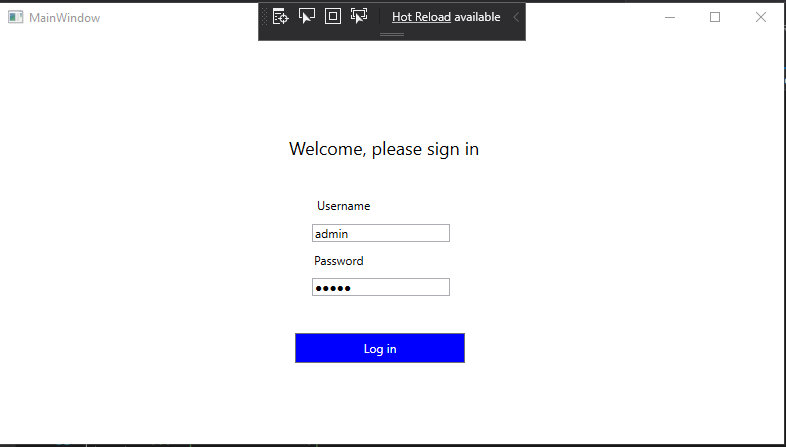
Reguli:

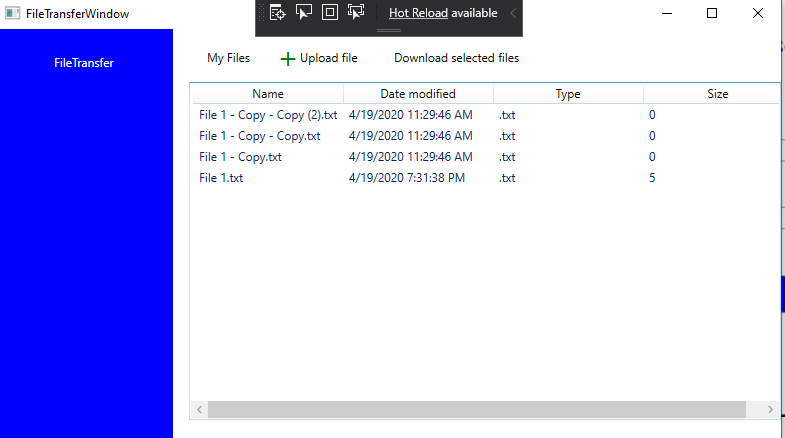
* Client: Connect
* Client: Create auth packet
* Client: Encrypt packet
* Client: Send auth packet
* Server: Receive auth packet
* Server: Decrypt packet
* Server: Authorize
* Server: Create connection
* Server: Create token
* Server: Send auth response with token
* Client: Receive auth packet
* Client: Save token
* Client: Create data packet with token
* Client: Encrypt packet
* Client: Send to Server
* Server: Receive packet
* Server: Decrypt packet
* Server: Check token
* Server: Dispach and process packet by packet type
* Server: Create response packet
* Server: Encrypt packet
* Server: Send packet to client
* Client: Receive packet
* Client: Decrypt packet
* Client: Process received data

Server app:



Client UI:





**Întrebări la apărarea laboratorului:**

1. **Ce este un protocol orientat pe conexiune ?**

În telecomunicații, orientarea către conexiune descrie un mijloc de transmitere a datelor în care dispozitivele din punctele finale utilizează un protocol preliminar pentru a stabili o conexiune end-to-end înainte de transmiterea oricărei date. Serviciul de protocol orientat către conexiune este uneori numit serviciu de rețea „de încredere”, deoarece garantează că datele vor ajunge în secvența corespunzătoare. Transmission Control Protocol ( TCP ) este un protocol orientat către conexiune.

1. **Ce tipuri de aplicații beneficiază în general de utilizarea protocolului TCP ?**

IP-ul este folosit de aproape toate aplicațiile care realizează comunicarea în rețea pe internet și rețelele Ethernet locale.

TCP over IP este utilizat în majoritatea cazurilor în care aplicațiile au nevoie de control asupra livrării datelor transmise. Pentru toate datele care își pierd valoarea în cazul în care părți din acestea sunt pierdute. Exemple: navigare web, chat text, tranzacții bancare, transfer de fișiere (ar putea folosi protocoale de specialitate ca FTP, SFTP, FTPS).

1. **Cum TCP garantează că datele vor fi transmise cu succes ?**

Când un utilizator emite o comandă care folosește un protocol de strat de aplicație TCP / IP, un lanț de evenimente este pus în mișcare. Comanda sau mesajul utilizatorului trece prin stiva de protocol TCP / IP pe mașina locală, apoi pe suportul de rețea la protocoalele de la destinatar. Protocoalele de la fiecare strat de pe gazda care trimite trimite adaugă informații la datele originale.

Pe măsură ce comanda utilizatorului își face drum prin stiva de protocoale, protocoalele de pe fiecare strat al gazdei expediate interacționează, de asemenea, cu colegii lor din gazda primitoare

1. **Diferența dintre blocking si non-blocking sockets**

Una dintre primele probleme pe care le veți întâmpina la dezvoltarea aplicațiilor Windows Sockets este diferența dintre socket-urile de blocare și cele care nu blochează. Ori de câte ori efectuați o operație pe o priză, este posibil să nu se poată completa imediat și să returneze controlul înapoi în program. De exemplu, o citire pe o priză nu se poate finaliza până când unele date nu au fost trimise de către gazda de la distanță. Dacă nu există date care așteaptă să fie citite, se poate întâmpla unul dintre cele două lucruri: funcția poate aștepta până când unele date au fost scrise pe soclu sau se poate întoarce imediat cu o eroare care indică faptul că nu există date care trebuie citite.

Primul caz se numește priză de blocare . Cu alte cuvinte, programul este „blocat” până când cererea de date a fost satisfăcută. Când sistemul de la distanță scrie unele date pe soclu, operațiunea de citire se va finaliza și execuția programului va fi reluată. Al doilea caz se numește priză care nu blochează și necesită ca aplicația să recunoască starea de eroare și să gestioneze în mod corespunzător situația. Programele care folosesc socluri care nu blochează utilizează de obicei una dintre cele două metode atunci când trimiteți și primiți date. Prima metodă, numită sondare, este atunci când programul încearcă periodic să citească sau să scrie date din priză (de obicei folosind un cronometru). A doua metodă, și preferată, este să folosiți ceea ce se numește notificare asincronă. Aceasta înseamnă că programul este notificat de fiecare dată când are loc un eveniment socket și, la rândul său, poate răspunde la acel eveniment. De exemplu, dacă programul de la distanță scrie unele date în priză, se generează un „eveniment citit”, astfel încât programul să știe că poate citi datele din priză în acel moment.

1. **Diferența dintre blocking multithreaded și non-blocking single thread socket**

Procesoarele sunt de sute de ori mai rapide decât rețeaua reală, deci există foarte puține, dacă există vreo diferență. Din punct de vedere al utilizării procesorului, prizele de blocare sunt mai bune, deoarece permit sistemului de operare să blocheze un fir, mai degrabă decât să-l urle de-a lungul folosirii a 100% din procesor pentru a continua să interzică o priză.

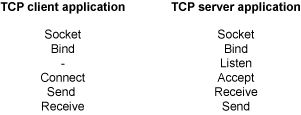
1. **Cum are loc procesul TCP Three Way Handshake ?**

Acest lucru ar putea fi văzut, de asemenea, ca o modalitate de stabilire a conexiunii TCP. Înainte de a intra în detalii, să ne uităm la câteva elemente de bază. TCP înseamnă Protocolul de control al transmisiei care indică faptul că face ceva pentru a controla transmisia datelor într-un mod fiabil.

Procesul de comunicare între dispozitivele de pe internet se întâmplă în conformitate cu modelul actual de suită TCP / IP (versiunea dezbrăcată a modelului de referință OSI). Stratul de Aplicații este un număr mare de stivă de model TCP / IP, de unde aplicația cu referire la rețea, cum ar fi browserul web, din partea clientului, stabilește conexiunea cu serverul. Din stratul de aplicații, informațiile sunt transferate în stratul de transport unde subiectul nostru apare în imagine. Cele două protocoale importante ale acestui strat sunt - TCP, UDP (User Datagram Protocol) din care TCP este predominant (deoarece oferă fiabilitate pentru conexiunea stabilită). Cu toate acestea, puteți găsi aplicația UDP în interogarea serverului DNS pentru a obține echivalentul binar al Numelui de domeniu utilizat pentru site-ul web.

1. **Numiti cele 4 apeluri de sistem necesare pentru a crea un server TCP**

Un client TCP tipic și o aplicație server emite o secvență de apeluri de sistem TCP pentru a îndeplini anumite funcții. Unele dintre aceste apeluri sistem includ socket (), bind (), listen (), accept (), send (), și receive(). Acest articol explică ce se întâmplă la nivelurile inferioare atunci când o aplicație emite apelurile sistemului TCP



1. **Care este rolul metodei bind() ?**

bind (struct proc ∗p, struct bind\_args ∗uap, int ∗retval)

struct bind\_args

{ int s;

caddr\_t name;

int namelen;

};

În bind funcția de apel sistem:

s este descriptorul soclului.

name este indicatorul către bufferul care conține adresa de transport a rețelei.

namelen este de dimensiunea tamponului.

Bind Apelul sistem asociază o adresă de rețea locală de transport cu o priză. Pentru un proces client, nu este obligatoriu emiterea unui bindapel. Nucleul are grijă să efectueze o legătură implicită atunci când procesul client emite apelul sistemului de conectare . Adesea este necesar ca un proces server să emită o cerere de legare explicită înainte de a putea accepta conexiuni sau de a începe comunicarea cu clienții.

1. **Care este rolul metodei accept() ?**

accept(struct proc ∗p, struct accept\_args ∗uap, int ∗retval);

struct accept\_args

{

int s;

caddr\_t name;

int ∗anamelen;

};

În accept apelul de sistem:

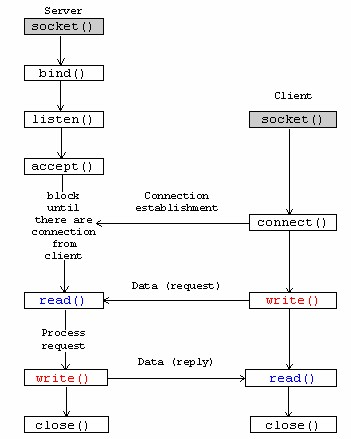
s este descriptorul soclului.

name este un tampon (un parametru OUT), care conține adresa de transport de rețea a gazdei străine.

anameleneste de dimensiunea nametamponului.

acceptApelul de sistem este un apel de blocare care așteaptă pentru conexiunile de intrare. Odată procesată o cerere de conexiune, un nou descriptor socket este returnat de accept. Această nouă priză este conectată la client, iar cealaltă priză srămâne în stare DE ASISTENȚĂ pentru a accepta conexiuni suplimentare.

1. **Ce se întîmplă cînd apelați mai întîi connect() apoi bind() ?**

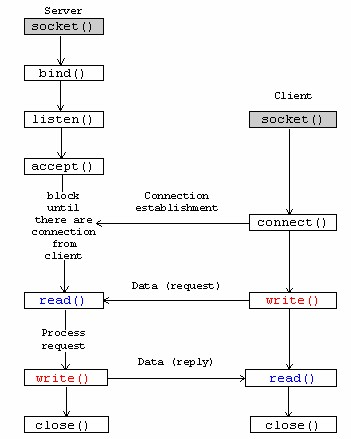


Dupa apelarea la bind, conexiunea va fi inchisa si va fi pornit procesul de accept

1. **Ați avea vreodată nevoie să implementați un timeout într-un client sau server care utilizează TCP?**

Da, fiindca pot fi probleme cu reteaua de internet si pachetul poate sa nu ajunga la destinatie iar conexiunea va fi inchisa, dar in cazul cind va fi un timeout, actiunea poate fi reluata.

1. **Într-o conexiune TCP, clientul sau serverul trimite mai întâi datele ?**



Din schema respectiva, vedem ca clientul este acela care transmite un pachet.

1. **Care este adresa de loopback IPv4 și care este rolul ei ?**

O adresă loopback este un tip de adresă IP care este utilizat pentru a testa mediul de comunicare sau de transport pe o placă de rețea locală și / sau pentru testarea aplicațiilor din rețea. Pachetele de date trimise pe o adresă de returnare a buclei sunt redirecționate înapoi către nodul orginant fără nicio modificare sau modificare

1. **De unde știe un sistem de operare ce aplicație este responsabilă pentru un pachet primit din rețea ?**

Comunicarea în rețea se bazează pe interacțiunea mai multor protocoale diferite, care implementează regulile și convențiile care reglementează modul în care dispozitivele dintr-o rețea comunică. Aceste protocoale sunt proiectate și implementate printr-o abordare stratificată, care ne permite să modificăm protocoalele individuale pentru a se adapta la schimbările tehnologice, fără a afecta funcționarea protocoalelor în straturile de mai sus și de mai jos.

Utilizarea unui model de referință de protocol de rețea stratificată face, de asemenea, mult mai ușor să înțelegeți funcția de bază a protocoalelor individuale și modul în care acestea se raportează la protocoalele din alte straturi.

Vizualizați o imagine mai mare

figura 1

Nu uitați că un model de referință este o modalitate de a înțelege interrelația dintre protocoale și nu poate susține comunicarea în rețea. Comunicarea în rețea este susținută de protocoale care respectă modelul de referință - acestea sunt denumite o suită de protocoale. Când un sistem de operare este instalat pe un dispozitiv de calcul, acesta va instala o varietate de protocoale Protocol de control al transmisiei (TCP) / Protocol de utilizator Datagram (UDP) care sunt capabile să susțină comunicațiile din rețea.

Ați văzut în lucrările anterioare că programele de pe un dispozitiv de calcul trec date la protocoalele TCP / IP atunci când datele trebuie transmise printr-o conexiune de rețea. Datele sunt acceptate de stratul de aplicație, înainte de a fi transmise straturilor de transport, Internet și acces la rețea. Fiecare strat adaugă informații suplimentare numite antet într-un proces numit încapsulare . Antetul oferă informații suplimentare care permite protocolului de încapsulare să își îndeplinească funcțiile cerute. Adăugarea informațiilor antet creează o unitate de date de protocol (PDU) și fiecare strat are un PDU identificabil:

transport (TCP): segment

transport (UDP): datagrama

Internet: pachet

acces la rețea: cadru.

Rețineți că cele două protocoale ale stratului de transport au denumiri diferite pentru PDU-urile lor, indicând ușoare diferențe de funcție între ele.

Datele primite de pe suportul de rețea sunt transmise prin intermediul straturilor TCP / IP. Fiecare strat interpretează informațiile de antet din PDU de care este interesat, înainte de a trece datele la stratul de mai sus. După ce informațiile din antet au fost interpretate, antetul este aruncat, dezvăluind PDU pentru următorul strat. Acest proces este denumit descapsulare .

2 strat de transport - TCP

TCP este un protocol de transport fiabil și realizează procese pentru a asigura livrarea fiabilă a datelor între aplicații folosind livrare recunoscută. O analogie pentru operațiune ar fi comanda online a unor articole pentru livrare la domiciliu. Veți putea urmări progresul pachetelor în timp ce acestea sunt în tranzit și vi se poate solicita să semnați pentru acestea, ceea ce oferă confirmarea livrării acestora. Veți putea, de asemenea, să contactați vânzătorul articolelor, dacă nu sunt livrate pentru a le re-livra.

Cele patru funcții de bază ale TCP sunt:

Asigurarea că segmentele de date sunt livrate în secvența corectă la protocolul de strat de aplicație corect.

Asigurarea controlului debitului , astfel încât segmentele sunt livrate cu o viteză pe care dispozitivul receptor o poate gestiona.

Multiplexarea mai multor aplicații de utilizator, permițându-le să acceseze simultan rețeaua de transmisie.

Eroare la verificarea segmentelor primite și solicitarea retransmisiei dacă segmentele au fost corupte.

TCP descompune datele primite de la stratul de aplicație în bucăți mici cunoscute sub numele de segmente. Pentru a asigura o transmisie fiabilă , segmentele sunt numerotate înainte de a fi trecute la procesul IP, care le încapsulează în pachete. TCP urmărește numărul de segmente care sunt trimise către un dispozitiv de destinație specific dintr-un protocol de strat de aplicație specific. Dacă nu primește confirmare într-o anumită perioadă de timp, TCP presupune că segmentele au fost pierdute și le vor retransmite.

TCP verifică, de asemenea, fiecare segment pentru a se asigura că conținutul nu a fost schimbat în timpul transmiterii pe suportul de rețea. Acest proces este denumit verificarea erorilor și este posibil deoarece fiecare antet de segment include o sumă de verificare, care este o semnătură matematică generată prin alimentarea datelor din segment printr-un algoritm de redundanță ciclică. Aceasta este plasată în antetul TCP de către expeditor, iar dispozitivul receptor va efectua același calcul la datele din segmentul primit. Dacă semnăturile se potrivesc, TCP va considera datele din segment ca fiind deteriorate. Dacă semnăturile nu se potrivesc, TCP va asigura retransmiterea datelor.

TCP poate servi mai multe protocoale de straturi de aplicație simultan, prelucrându-le datele în segmente și alimentându-le către stratul de internet într-un proces numit multiplexare . Pentru a permite TCP să livreze segmente primite la protocolul de strat de aplicație corect, se utilizează numere de port .

Deoarece TCP poate primi un număr semnificativ de segmente, care trebuie verificate de eroare, secvențiate și livrate la protocolul corect al stratului de aplicație, trebuie să poată controla cantitatea de segmente pe care o primește pentru a-i permite să funcționeze eficient. Un proces de primire TCP va conveni o dimensiune a ferestrei cu un expeditor TCP, care dictează cantitățile de segmente care pot fi trimise înainte de primirea unei confirmări TCP de către receptor. Receptorul TCP poate astfel controla cantitatea de segmente pe care este trimis, proces numit controlul fluxului .

1. **Datele primite prin recv() au întotdeauna aceeași dimensiune cu datele trimise cu send() ?**

Au aceeasi dimensiune, doar daca lungimea la pachet este mai mica decit marimea la buffer (marimea permisa pentru transmiterea unui pachet)

1. **Este acceptabil să închei executia programului dacă este detectată o eroare de rețea ?**

Nu, programul trebuie sa afiseze un mesaj de eroare retea si sa permita utilizatorului sa reia procesul anterior, fiindca daca utilizatorul avea de facut o tranzactie importanta, atunci el va avea de pierdut.

1. **Puteți îmbunătăți performanța aplicației prin dezactivarea algoritmului Nagle ?**

Nagle este doar un algoritm pentru partea expeditorului, astfel încât dacă puteți afecta doar receptorul, nu îl puteți dezactiva.

Chiar dacă ați putea afecta expeditorul, dezactivarea Nagle nu este foarte eficientă atunci când efectuați o comunicare direcțională. În comunicarea bidirecțională, dezactivarea Nagle poate îmbunătăți debitul, deoarece beneficiile eliminării întârzierilor se pot acumula, deoarece fiecare nod își poate trimite răspunsurile ușor mai devreme, lăsând cealaltă parte să răspundă chiar mai devreme. Cu toate acestea, în cazul unic de direcție, dezactivarea Nagle poate scădea latența cu o călătorie dus-întors, dar aceste beneficii nu se pot acumula, deoarece faptul că nu întârzieți pachetele nu încetinește generarea de pachete noi. Nu înaintați niciodată mai mult de o singură călătorie dus-întors. Pe internet, adică ~ 20-30ms. Peste o rețea LAN, de obicei ~ 1ms

Dacă sistemul dvs. este suficient de greu în timp real pentru ca o singură latență dus-întors să conteze, atunci TCP este un protocol slab și ar trebui să utilizați în schimb UDP. Nagle este un algoritm TCP, deci nu ar afecta UDP.

Doar pentru distracție: pingerea unui computer local pe LAN-ul meu este <1ms. Aceasta înseamnă că Nagle poate întârzia mesajele mele doar cu ceva sub 1ms. Cuantele pentru planificatoarele desktop pot 20-60ms [1] și chiar mai mult pentru servere, așa că m-aș aștepta ca eliminarea algoritmului nagle să nu aibă niciun efect vizibil asupra LAN-ului meu, înnegrită de efectul altor fire de pe computerul care consumă procesoarele.

1. **Ce instrumente listează socket-urile TCP deschise în sistemele de operare Windows și Linux ?**

Pentru a enumera toate porturile deschise de pe un sistem Linux , puteți utiliza comanda netstat sau utilitatea ss după cum urmează.

De asemenea, este crucial să menționăm că comanda netstat a fost depășită și în schimb comanda ss și- a luat locul în afișarea statisticilor de rețea mai detaliate.

Windows: <https://www.howtogeek.com/howto/28609/how-can-i-tell-what-is-listening-on-a-tcpip-port-in-windows/>

1. **Tehnici de sincronizare a firelor de execuții**

Sincronizarea firelor este definită ca un mecanism care asigură că două sau mai multe procese simultane sau fire nu execută simultan un anumit segment de program cunoscut sub denumirea de secțiune critică . Accesul proceselor la secțiunea critică este controlat prin utilizarea tehnicilor de sincronizare. Când un fir începe să execute secțiunea critică (segmentul serializat al programului), celălalt fir ar trebui să aștepte până când primul fir se termină. Dacă nu se aplică tehnici de sincronizare corespunzătoare [1] , poate provoca o condiție de cursă în care valorile variabilelor pot fi imprevizibile și variază în funcție de calendarul comutărilor contextuale ale proceselor sau firelor.

De exemplu, să presupunem că există trei procese, și anume 1, 2 și 3. Toate trei se execută concomitent și trebuie să partajeze o resursă comună (secțiune critică), așa cum se arată în Figura 1. Sincronizarea ar trebui să fie utilizată aici pentru evita orice conflict pentru accesarea acestei resurse partajate. Prin urmare, atunci când Procesul 1 și 2 încearcă să acceseze resursa respectivă, ar trebui să fie atribuită doar unui singur proces la un moment dat. Dacă este atribuit Procesului 1, celălalt proces (Procesul 2) trebuie să aștepte până când Procesul 1 eliberează acea resursă

O altă cerință de sincronizare care trebuie luată în considerare este ordinea în care anumite procese sau fire trebuie executate. De exemplu, nu se poate urca într-un avion înainte de a cumpăra un bilet. În mod similar, nu se pot verifica e-mailurile înainte de a valida datele de acreditare corespunzătoare (de exemplu, numele de utilizator și parola). În același mod, un bancomat nu va furniza niciun serviciu până când nu va primi un cod PIN corect.

În afară de excluderea reciprocă, sincronizarea se ocupă și de următoarele:

* Deadlock , care apare atunci când multe procese așteaptă o resursă partajată (secțiune critică) care este deținută de un alt proces. În acest caz, procesele continuă să aștepte și să nu mai execute;
* Starvation , care apare atunci când un proces așteaptă să intre în secțiunea critică, dar alte procese monopolizează secțiunea critică, iar primul proces este obligat să aștepte la nesfârșit;
* Priority inversion , care are loc atunci când un proces cu prioritate înaltă este în secțiunea critică și este întrerupt de un proces cu prioritate medie. Această încălcare a regulilor prioritare poate avea loc în anumite circumstanțe și poate duce la consecințe grave în sistemele în timp real;
* Busy waiting , care are loc atunci când un proces frecventează sondaje pentru a determina dacă are acces la o secțiune critică. Acest sondaj frecvent elimină timpul de procesare din alte procese.