Anexa 1 - Socketi

Contents

Socketi

Socketii sunt mijloace generalizate de comunicatie interproces deoarece: procesele comunicante pot rula pe alte masini, arhitectura masinii poate fi diferita, sistemul de operare poate fi diferit. Canalul permite comunicatia bidirectionala intre procese, dar nu se ocupa de structura datelor transmise.

In general, intr-un sistem bazat pe comunicatia cu socketi serverul urmeaza anumiti pasi pentru initierea comunicatiei:

- Creeaza un socket
- Asociaza socketului o adresa
- Asculta cererile clientilor

Când un client doreste sa initieze o comunicatie cu serverul, acesta urmeaza si el anumiti pasi:

- Creeaza un socket
- Determina locatia serverului (adresa si port)
- Incepe trimiterea si/sau primirea datelor

Tipuri de socketi Comunicatia intre procese prin socketi poate fi cu conexiune (stream) sau fara conexiune (datagram). In primul caz datele transmise sunt vazute ca secventa de bytes pe când in al doilea ca pachete (discrete) de informatie. Aceste pachete contin in general pe lânga datele utile de transmis si informatii header si trailer. Alegerea tipului de socket este importanta, deoarece fiecare are performanta si siguranta datelor diferita, iar doi socketi care comunica trebuie sa fie de acelasi tip.

- Stream. Socketii de tip stream sunt siguri, adica se garanteaza primirea datelor in ordinea in care au fost trimise. Exista un mecanism implementat care se ocupa de datele duplicate, verificarea impotriva erorilor, si controlul fluxului (cu care programatorul nu are de-a face). Când se folosesc socketi de tip stream o conexiune este stabilita. Aceasta inseamna ca doua procese trebuie sa fie in mod logic ?de acord? sa comunice inainte ca informatia sa fie transferata intre ele. Aceasta conexiune este mentinuta de ambele procese pe durata sesiunii de comunicare.
- Datagram. Socketii de tip datagrama furnizeaza o comunicatie nesigura intre cele doua procese, aceasta insemnând ca datele pot fi primite in alta ordine. Nu exista notiunea de conexiune, fiecare datagrama fiind trimisa si procesata independent. Acest lucru are multe implicatii, cea mai importanta fiind aceea ca diferite datagrame pot ajunge la aceeasi destinatie pe cai diferite. Alta implicatie de luat in seama este aceea ca nu exista un control al fluxului. Pachetele datagrama sunt in general destul de mici si de obicei de lungime fixa. Nu se realizeaza corectia erorilor, si in cazul in care este dorita neaparat, este slaba.

1 Functia WSAStartup (Windows)

Initializeaza folosirea bibliotecii WS2_32.DLL de un proces. WSAStartup trebuie sa fie prima functie apelata când se doreste lucrul cu socketi.

```
   WORD wVersionRequested,
   LPWSADATA lpWSAData
```

);

Parametri: wVersionRequested [in] Versiunea cea mai recenta de Windows Sockets care poate fi folosita. Octetul cel mai semnificativ contine partea fractionara din versiune (revision), iar cel mai putin semnificativ partea intreaga. lpWSAData [out] Pointer catre o structura de tip WSADATA pentru a primi detalii despre implementarea socketilor Windows.

Valoare returnata: Functia returneaza zero in caz de succes si un cod de eroare altfel.

Exemplu de apel: (pentru versiunea 2.0)

```
   fprintf(stderr,"WSAStartup() failed"); exit(1);
```

}

Observatie: In cazul socketilor Windows la compilare trebuie inclusa libraria wsock32. (gcc <nume.c> -lwsock32)

2 Functia socket

Creeaza un socket cu caracteristicile cerute de tipul de comunicare dorit.

2.1 Apelul Unix socket

1. include <sys/socket.h>

int socket(int family, int type, int protocol);

Parametri: family poate avea diferite valori (AF vine de la address family). Doua dintre ele ar fi:

AF_UNIX pentru protocoalele interne UNIX AF_INET pentru protocoalele TCP/IP type tipul socket-ului:

SOCK_STREAM socket orientat pe conexiune SOCK_DGRAM socket datagrama (fara conexiune) SOCK_RAW socket direct (pentru comunicarea la nivel IP) protocol are de regula valoarea 0. Exista unele aplicatii specializate in care trebuie specificata o valoare dependenta de tipul de socket si familia de protocoale folosite. Exemplu: IPPROTO_TCP (socketi orientati conexiune), IPPROTO_UDP (socketi fara conexiune)

Valoare returnata: Descriptor de socket care se utilizeaza la fel ca un descriptor de fisier sau -1 in caz de eroare.

2.2 Apelul Windows socket

SOCKET socket(

```
      int af,
      int type,
      int protocol
```

);

Parametri: af [in] Address family (vezi Unix socket). type [in] Specificarea tipului pentru noul socket: SOCK_STREAM sau SOCK_DGRAM. (vezi Unix socket) protocol [in] Protocolul ce va fi folosit cu socketul, specific familiei indicate. Exemplu: IPPROTO_TCP (socketi orientati conexiune), IPPROTO_UDP (socketi fara conexiune). Parametrul este totdeauna setat la 0 pentru IrDA.

Valoare returnata: Daca nu apar erori, functia returneaza un descriptor corespunzator noului socket. Altfel se returneaza valoarea INVALID_SOCKET si un cod de eroare specific poate fi obtinut cu ajutorul functiei WSAGetLastError.

3 Functia bind

Asociaza unui socket o adresa (realizeaza legatura intre un socket creat anterior si un punct final de comunicare). Este necesara atât in cazul comunicarii cu conexiune cât si in cazul comunicarii fara conexiune.

3.1 Apelul Unix bind

1. include <sys/socket.h>

int bind(int sockfd, struct sockaddr *myaddr, int addrlen);

Parametri: sockfd descriptor de socket obtinut printr-un apel socket anterior. myaddr pointer la o structura de adresa specifica fiecarei familii de protocoale. addrlen lungimea structurii de adresa.

Valoare returnata: Returneaza 0 in caz de succes sau -1 in caz de eroare.

Structura de adresa se gaseste in <sys/socket.h>:

```
u_short sa_family;
char sa_data[14];
```

};

Continutul celor 14 octeti de date este precizat in cazul familiei TCP/IP prin urmatoarea structura definita in <netinet/in.h>:

```
u_long s_addr; /* 32-bit netid/hostid (network byte ordered) */
```

}; struct sockaddr_in{

```
short sin_family; /* AF_INET */
u_short sin_port; /* 16-bit port number (network byte ordered) */
struct in_addr sin_addr; /* 32-bit netid/hostid (network byte  ordered)*/
char sin_zero[8]; /* unused */
```

};

Exemplu de folosire: vezi exemplul pentru Windows.

3.2 Apelul Windows bind

```
SOCKET s,
const struct sockaddr* name,
int namelen
```

);

Parametri: s [in] Descriptor ce identifica un socket creat prin apelul functiei socket. name [in] Adresa care i se va asigna socketului. namelen [in] Lungimea valorii parametrului name in bytes.

Valoare returnata: In caz de succes functia returneaza zero, altfel returneaza SOCKET_ERROR, si un cod de eroare specific poate fi vazut cu WSAGetLastError.

Structura de adresa este identica cu cea din UNIX.

```
short sin_family;
u_short sin_port;
struct in_addr sin_addr;
char sin_zero[8];
```

};

Observatii

1. In Windows Sockets 2, parametrul name nu este interpretat in mod strict ca un pointer la o structura de tip sockaddr. Se face acest cast pentru compatibilitatea cu Windows Sockets 1.1. (vezi exemplu de folosire)
2. In cazul in care nu ne intereseaza ce adresa locala va fi asignata, se va specifica valoarea INADDR_ANY pentru adresa (vezi exemplu) si va fi alocata automat o adresa disponibila.

Exemplu de folosire:

echoServAddr.sin_family = AF_INET; echoServAddr.sin_addr.s_addr = htonl(INADDR_ANY); echoServAddr.sin_port = htons(echoServPort); if(bind(sock,(struct sockaddr *) &echoServAddr, sizeof(echoServAddr))<0){

```
   fprintf(stderr,"eroare: %d\n",WSAGetLastError());
```

}

4 Functia connect

Initiaza o conexiune (pentru protocoalele cu conexiune), adica stabileste o legatura intre un proces client si server. Executia apelului implica schimbul unor mesaje intre cele doua sisteme pentru stabilirea parametrilor legaturii.

4.1 Apelul Unix connect

1. include <sys/socket.h>

int connect(int sockfd, const struct sockaddr *myaddr, int addrlen);

Parametri: asemanatori ca semnificatie cu cei de la bind, cu exceptia faptului ca myaddr trebuie sa contina adresa serverului la care clientul vrea sa se conecteze.

Valoare returnata: Revenirea din apel se face numai dupa stabilirea cu succes a legaturii (valoarea 0) sau daca se produce o eroare (valoarea -1).

4.2 Apelul Windows connect

```
SOCKET s,
const struct sockaddr* name,
int namelen
```

);

Parametri: asemanatori ca semnificatie cu cei de la bind, cu exceptia faptului ca myaddr trebuie sa contina adresa serverului la care clientul vrea sa se conecteze.

Valoare returnata: In caz de succes returneaza zero. Altfel, returneaza SOCKET_ERROR, si un cod de eroare specific poate fi vazut cu ajutorul WSAGetLastError.

5 Functia listen

Pregateste socketul pentru a primi conexiuni (numai pentru protocoalele cu conexiune).

5.1 Apelul Unix listen

Parametri: backlog Câte apeluri pot fi acceptate in timp ce serverul asteapta terminarea unui accept inceput (se consuma un anumit timp pentru ca serverul sa efectueze operatiile necesare la acceptarea unei cereri). Traditional valoarea utilizata pentru backlog este 5, maximul admis in implementarile curente.

Valoare returnata: Zero in caz de succes si -1 in caz de eroare.

5.2 Apelul Windows listen

```
SOCKET s,
int backlog
```

);

Parametri: backlog [in] Lungimea maxima a cozii de conexiuni in curs de acceptare. Daca este setat la SOMAXCONN, backlog va fi setata la valoarea maxima posibila. Nu exista o metoda standard pentru a obtine valoarea backlog efectiva.

Valoare returnata: In caz de succes functia returneaza zero. Altfel se intoarce SOCKET_ERROR. Un cod de eroare detaliat se poate obtine prin apelul functiei WSAGetLastError.

6 Functia accept

Accepta o conexiune; un canal de comunicatie este creat intre cel ce a initiat si cel ce a acceptat conexiunea (numai pentru protocoalele cu conexiune).

6.1 Apelul Unix accept

1. include <sys/socket.h>

int accept(int sockfd, struct sockaddr *addr, int *addrlen);

Parametri: sockfd Descriptor de socket pe care s-a facut listen. addr Contine dupa apel adresa procesului client care s-a conectat. addrlen [in,out] Dimensiune addr - vezi observatie importanta!

Valoare returnata: Functia returneaza fie eroare (-1), fie descriptorul de socket al unui socket nou creat, cu aceleasi proprietati ca si sockfd. Prin socketul returnat se va desfasura comunicatia intre server si clientul acceptat.

6.2 Apelul Windows accept

```
SOCKET s,
struct sockaddr* addr,
int* addrlen
```

);

Parametri: s [in] Descriptor de socket pe care s-a facut listen. addr [out] Pointer optional catre un buffer in care se pune adresa clientului care s-a conectat. addrlen [in,out] Pointer optional catre un intreg care contine lungimea valorii lui addr. (vezi observatie importanta!)

Valoare returnata: In caz de succes se returneaza o valoare de tip SOCKET care reprezinta descriptorul pentru noul socket. Prin socketul returnat se va desfasura comunicatia intre server si clientul acceptat. In caz de eroare se intoarce INVALID_SOCKET, iar un cod de eroare detaliat se poate obtine prin apelul functiei WSAGetLastError.

Observatie:

Functia accept poate bloca programul apelant pâna când o conexiune este prezenta daca nu exista cereri de conectare in coada, si socket-ul este marcat ca blocant. Daca socketul nu e blocant si nu exista conexiuni in asteptare, accept returneaza eroare.

Observatie importanta:

Intregul referit de addrlen trebuie sa contina initial dimensiunea structurii catre care puncteaza addr. La return va contine lungimea efectiva a adresei returnate.

7 Apeluri pentru transferuri cu conexiune

7.1 Unix

int recv(int sockfd, void *buff, size_t nbytes, int flags);

Parametri: sockfd Descriptor corespunzator unui socket conectat. buff [in] Buffer ce contine datele de transmis (send) / [out] Buffer pentru datele ce vor fi primite (recv). nbytes [in] Lungimea in octeti a datelor din buf (send) / [in] Lungimea in octeti a lui buf (recv). flags [in] Flag se specifica modul in care se face transmiterea/primirea datelor. Se poate seta folosind OR pe biti cu diferite valori:

Pentru send: MSG_DONTROUTE, MSG_OOB (date de tip OOB), etc.

Pentru recv: MSG_PEEK, MSG_OOB, etc.

Valoare returnata: Send / Recv returneaza numarul de caractere trimise / primite sau -1 in caz de eroare.

Observatie: Apelurile Unix read si write pot fi de asemenea folosite de protocoalele cu conexiune pentru a transmite si primi date. Se folosesc la fel ca si in cazul fisierelor.

7.2 Windows

```
   SOCKET s,
   const char* buf,
   int len,
   int flags
```

); int recv(

```
   SOCKET s,
   char* buf,
   int len,
   int flags
```

);

Parametri: s [in] Descriptor corespunzator unui socket conectat. buf [in] Buffer ce contine datele de transmis (send) / [out] Buffer pentru datele ce vor fi primite (recv). len [in] Lungimea in octeti a datelor din buf (send) / [in] Lungimea in octeti a lui buf (recv). flags [in] Flag se specifica modul in care se face transmiterea/primirea datelor. Se poate seta folosind OR pe biti cu urmatoarele valori:

Pentru send: MSG_DONTROUTE, MSG_OOB (date de tip OOB).

Pentru recv: MSG_PEEK, MSG_OOB.

Valoare returnata: In caz de succes, send returneaza numarul total de octeti trimisi, care poate fi mai mic decât len, altfel SOCKET_ERROR.

In caz de succes recv returneaza numarul de octeti primiti. Daca conexiunea a fost inchisa, se retuneaza zero. La eroare, se returneaza SOCKET_ERROR si informatii detaliate se obtin cu ajutorul WSAGetLastError.

8 Apeluri pentru transferuri fara conexiune

8.1 Unix

sockaddr *to, int tolen); int recvfrom(int sd, void *buf, size_t len, int flags, struct sockaddr *from, int fromlen);

8.2 Windows

```
SOCKET sd,
const char* buf,
int len,
int flags,
const struct sockaddr* to,
int tolen
```

); int recvfrom(

```
SOCKET sd,
char* buf,
int len,
int flags,
struct sockaddr* from,
int* fromlen
```

);

Parametri: sd, buf, len, flags - idem send / recv. to [in] Pointer catre o structura de tip sockaddr care contine adresa socketului destinatie. tolen [in] Dimensiunea adresei din to. from [out] Pointer catre un buffer dintr-o structura de tip sockaddr care va contine adresa sursei. fromlen [in, out] Pointer catre dimensiunea bufferului from.

Valoare retunata: idem send / recv.

9 Functia de inchidere socket

Inchide canalul de comunicatie si distruge socketul.

9.1 Apelul Unix close

In caz de succces se returneaza zero, altfel -1.

9.2 Apelul Windows closesocket

```
SOCKET s
```

);

In caz de succces se retuneaza zero, altfel SOCKET_ERROR si WSAGetLastError pentru cod de eroare specific.

10 Functia WSACleanup (Windows)

Functia incheie utilizarea bibliotecii WS2_32.DLL.

Returneaza zero in caz de succes si SOCKET_ERROR altfel (WSAGetLastError pentru erori specifice).

11 Functii pentru ordinea octetilor

Sunt identice atât in Windows cât si in Linux. Sunt necesare datorita diferentelor arhitecturale intre calculatoare si conventiei de transmitere a informatiilor multioctet in retea.

u_short htons(u_short hostshort); u_long ntohl(u_long netlong); u_short ntohs(u_short netshort);

Primele doua functii convertesc valori lungi, respectiv scurte, din reprezentarea interna (in calculator) in reprezentarea din retea iar celelalte doua functii fac conversia in sens invers. Valorile convertite se considera intregi, cu conventia ca un intreg scurt se reprezinta pe 16 biti iar unul lung pe 32 biti.

12 Observatii asupra succesiunii functiilor pentru server / client

Server pentru un protocol cu conexiune:

Client pentru un protocol cu conexiune:

Server pentru un protocol fara conexiune:

Client pentru un protocol fara conexiune:

13 Tratarea erorilor

13.1 Tratarea erorilor in Unix

Este realizata cu ajutorul variabilei errno care este setata de apelurile de sistem (si de unele functii de biblioteca) pentru a indica ce eroare a aparut.

extern int errno;

Mai exista de asemenea si functia perror care afiseaza la stderr intâi sirul s si apoi un mesaj cu ultima eroare produsa.

13.2 Tratarea erorilor in Windows

Este realizata cu ajutorul functiei GetLastError, care returneaza ultima eroare aparuta.

Pentru cazul functiilor care lucreaza cu socketi se foloseste functia WSAGetLastError, care returneaza eroarea aparuta la ultima operatie cu socketi.