VJEŽBA 3: Soketi

Soketi (engl. *sockets*) predstavljaju krajnje točke u međuprocesnoj komunikaciji (engl. *inter-process communication, IPC*).

Sokete možete zamisliti kao telefone – svaki proces mora imati svoj telefon da bi mogao razgovarati sa nekim drugim procesom. Doslovan prijevod engleske riječi "socket" bio bi "utičnica", ali riječ "soket" se češće koristi u Računarstvu pa ćemo je i mi koristiti na laboratorijskim vježbama.

Postoje dvije vrste soketa:

- oni koji se koriste između procesa koji se izvršavaju na istom računalu (tzv. **IPC soketi** ili **soketi Unix domene**),
- i oni koji se koriste između procesa koji se ne izvršavaju na istom računalu (tzv. **mrežni soketi**). Budući da se danas većina mrežne komunikacije između računala odvija preko Internet protokola, za većinu se mrežnih soketa može reći da su **Internet soketi**. Internet soketi se najčešće koriste kod klijent-server arhitekture.

Unutar ove laboratorijske vježbe mi ćemo se baviti Internet soketima.

Zadatak. Na e-Learning portal su postavljeni .c dokumenti u kojima se nalazi programski kod za servera i klijenta. Isti je kod dan u nastavku teksta, i zatim detaljno objašnjen. Vaš zadatak je modificirati dani kod na slijedeći način:

1. klijent serveru šalje neki broj (tipa float);
2. server kvadrira primljeni broj;
3. server kvadrirani broj šalje klijentu;
4. klijent ispisuje primljeni rezultat.

Dodatne funkcije koje će vam trebati prilikom rješavanja zadatka dane su na kraju ovog dokumenta.

Programski kod: klijent.c

U ovome su poglavlju objašnjene najvažnije linije koda koji se nalazi u dokumentu klijent.c koji je dostupan preko e-Learning portala.

```
#include <stdio.h>            // potrebno za C
#include <string.h>           // potrebno za strlen()
#include <stdlib.h>           // potrebno za exit()
#include <unistd.h>           // potrebno za close()
#include <sys/socket.h>       // potrebno za rad sa soketima
#include <arpa/inet.h>        // potrebno za inet_aton()
```

```
#include <sys/types.h>        // definira sistemske tipove podataka (npr. u_char)
```

Ovu biblioteku nije potrebno uključivati na Linux-u. Međutim, budući da ne znate gdje se sve vaš program može izvoditi, ova se biblioteka ipak uključuje zbog portabilnosti koda.

```
#include <netinet/in.h>
```

U ovoj su biblioteci definirane strukture podataka (npr. sockaddr_in) te skupovi (tzv. obitelji) adresa (npr. AF_INET) i protokola (npr. PF_INET) koji se koriste prilikom rada sa soketima.

```
int main(int argc, char *argv[])
```

Funkcija main() može primiti više argumenata (iako u našem slučaju prima samo jedan, tj. prima ime programa):

- Prvi argument se naziva **argc** (engl. ***Arg****ument* ***C****ount*), i označava broj argumenata s kojima se poziva izvršna datoteka. Na primjer, ako se neki program pozove sa "./ime_programa", argc je jednak 1. Ako se program pak pozove sa "./ime_programa 161.53.166.5 5 89", argc je onda jednak 4.
- Drugi argument se naziva **argv** (engl. ***Arg****ument* ***V****ector*), i u njemu se nalaze svi argumenti s kojima se poziva program. Da program pozovemo sa "./ime_programa 161.53.166.5 50000 12", onda bi argv argument sadržavao slijedeće:
  - argv[0] = ./ime_programa
  - argv[1] = 161.53.166.5
  - argv[2] = 50000
  - argv[3] = 12

```
int sk;     // deskriptor soketa kojega ćemo kreirati
```

```
struct sockaddr_in skaddr;
```

Struktura **sockaddr_in** (engl. *Socket Address Internet*) služi za pohranu informacija o soketu. Za istu se svrhu može koristiti i struktura sockaddr, ali za Internet sokete se ta struktura ne preporuča zbog kompliciranijeg sučelja.

Struktura sockaddr_in ima slijedeći oblik:

```
struct sockaddr_in
{
        short sin_family;                   // short = short int
        u_short sin_port;                   // u_short = unsigned short int
```

```
    struct in_addr sin_addr;
    char sin_zero[8];
};
```

Element **sin_family** (engl. *Socket Internet Family*) označava format adrese soketa, tj. kaže kojemu skupu (ili obitelji) adresa ona pripada. Mi ćemo za ovaj element koristiti vrijednost **AF_INET** (engl. *Address Family Internet*), jer ćemo raditi sa Internet soketima. Neke od vrijednosti koje element sin_family može poprimiti su prikazane u Tablici 1.

Tablica 1. Primjeri vrijednosti koje element sin_family može poprimiti

Naziv	**Za što se koristi?**
AF_INET	IP (engl. *Internet Protocol*) adrese
AF_IRDA	IRDA (engl. *Infrared Data Association*) adrese
AF_BLUETOOTH	Bluetooth adrese
...	...

Element **sin_port** (engl. *Socket Internet Port*) označava broj port-a na kojemu server čeka nove zahtjeve za komunikacijom, tj. preko kojega klijent može "razgovarati" sa serverom.

Važno je naglasiti da vi u svojim programima ne možete koristiti bilo koji broj port-a na kojemu će vaš server čekati na zahtjeve. Možete koristiti jedino neki od slobodnih i neregistriranih port-ova (pogledati Tablicu 2 za više informacija).

Tablica 2. Rezervirani, registrirani i slobodni port-ovi

Broj port-a	**Objašnjenje**
0 - 1023	Rezervirani portovi koje ne možete koristiti u programima koje sami napišete jer ih već koriste neke druge aplikacije (npr. HTTP protokol koristi port broj 80)
1024 - 49151	Portovi koje je rezervirala organizacija IANA (engl. *Internet Assigned Numbers Authority*) i koje uglavnom ne možete koristiti u programima koje sami napišete
49152 - 65535	Slobodni port-ovi koje možete koristiti u programima koje sami napišete

Element **sin_addr** (engl. *Socket Internet Address*) sadrži IP adresu na kojoj se nalazi server. Element sin_addr je tipa struct in_addr. Struktura in_addr može biti različito implementirana na različitim operacijskim sustavima, pa se može dogoditi da u nekim implementacijama ima manji broj elemenata nego u drugim. Jedan od elemenata te strukture koji je uvijek implementiran naziva se **s_addr** (engl. *Socket Address*), i jedino ćemo njega koristiti. Ovaj element je broj koji se sastoji od 4 bajta (engl. *byte*). U svaki se bajt sprema jedan od 4 dijela IP adrese.

Element **sin_zero[8]** (engl. *Socket Internet Zero*) se koristi da bi se veličina strukture sockaddr_in nadopunila nulama sve do one veličine koju ima struktura sockaddr. Ovo se radi zbog toga da se kasnije pokazivač na strukturu sockaddr_in može pretvoriti (cast operacijom) u pokazivač na strukturu sockaddr (i obrnuto).

```
if (argc != 4)
{
      printf("Pokretanje: ./klijentvj <IP servera> <port> <broj>\n");
      exit(-1);
}
```

Provjeravamo da li je izvršna datoteka klijent.exe pozvana na ispravan način. Ta se aplikacija ispravno poziva ovako: "./klijent 161.53.166.5 50000 12", gdje je:

- ./klijent - ime programa;
- 161.53.166.5 - IP adresa ADRIA servera (inače IP adresu servera može dobiti pomoću naredbe ping);
- 50000 - broj port-a na kojemu naš program server.exe (koji se nalazi na serveru ADRIA) sluša zahtjeve;
- 12 - broj koji smo odlučili poslati serveru da ga kvadrira.

```
sk = socket(PF_INET, SOCK_STREAM, 0);
if (sk < 0)
{
      printf("Problem sa kreiranjem soketa!\n");
      exit(-1);
}
```

Funkcijom **socket()** stvara se soket. Funkcija prima tri argumenta:

- **PF_INET** (engl. *Protocol Family Internet*) označava skup (ili obitelj) protokola koji se koriste kod IP adresa. (Umjesto PF_INET se teoretski može koristiti AF_INET, ali u praksi se to ipak ne radi. Razlika između AF_INET i PF_INET u literaturi nije dobro dokumentirana, jer su te vrijednosti stvorene davno i onda su trebale imati drugačija značenja od onih koje imaju danas.);
- **SOCK_STREAM** - koristimo spojno-orijentirane sokete (engl. c*onnection oriented sockets, stream sockets*). Karakteristike spojno-orijentiranog prijenosa podataka su pouzdanost (svi paketi podataka stižu na odredište osim u slučaju prekida Internet veze ili greške na serveru), te očuvanost redoslijeda isporučenih paketa (paketi stižu na odredište redoslijedom kojim su poslani). Primjer spojno-orijentiranog protokola bio bi TCP (engl. *Transmission Control Protocol*). Besposjno-orijentirani (engl. *connectionless*) prijenos podataka nije pouzdan i ne čuva redoslijed poslanih paketa. Primjer bespojno-orijentiranog protokola bio bi UDP (engl. *User Datagram Protocol*);
- **0** - koristimo samo jednu vrstu protokola za kreiranje soketa, pa ovu vrijednost postavljamo na nulu (inače bi trebali specificirati broj protokola kojeg koristimo).

Funkcija socket() vraća -1 ako je prilikom stvaranja soketa došlo do greške.

```
skaddr.sin_family = AF_INET;  // koristimo IP adrese

if (inet_aton(argv[1], &skaddr.sin_addr) == 0)
{
      printf("Kriva IP adesa: %s\n",argv[1]);
      exit(-1);
```

```
}
```

Funkcija **inet_aton()** (engl. *Internet ASCII to Network*) primljenu IP adresu (koja je zapisana u argv[1]) pretvori iz tzv. "host byte order" formata (koji može biti "little-endian" ili "big-endian" format) u tzv. "network byte order" ("big-endian") format, te je zatim zapiše u polje sin_addr koje se nalazi unutar strukture skaddr. Ako je IP adresa već primljena u "big-endian" formatu, inet_aton() će je samo zapisati u sin_addr.

Podsjetnik: u "big-endian" formatu najznačajniji bit se zapisuje prvi, a u "little-endian" formatu najznačajniji bit se zapisuje posljednji.

Funkcija inet_aton() vraća nulu u slučaju da nije uspjela prepoznati format primljene IP adrese.

Ponekad se umjesto inet_aton() funkcije koristi inet_addr() funkcija. Ona se pak ne preporuča zbog toga što broadcast adrese (255.255.255.255) smatra nevaljanima i vraća grešku u slučaju njihovog korištenja.

```
skaddr.sin_port = htons(atoi(argv[2]));
```

Funkcija **htons()** (engl. *Host to Network Short*) pretvara pozitivni cijeli broj (zapisan u 16 bitova, tj. u 2 bajta) iz "host byte order" formata u "network byte order" format. Funkcija **atoi()** (engl. *ASCII to int*) pretvara ASCII znak u cijeli broj.

```
if (connect(sk, (struct sockaddr *) &skaddr, sizeof(skaddr)) < 0)
{
      printf("Nije spojen!\n");
      exit(-1);
}
```

Funkcijom **connect()** se naš soket (označen sa svojim deskriptorom sk) spaja na soket servera (kojega smo definirali sa IP adresom i brojem port-a). Funkcija prima tri argumenta:

- sk - deskriptor soketa koji se spaja na server;
- (struct sockaddr *) &skaddr - pokazivač na skaddr strukturu (pokazivač ne pokazuje na sockaddr_in strukturu, već na sockaddr strukturu);
- sizeof(skaddr) - veličina (u bajtovima) strukture skaddr.

```
close(sk);
```

Funkcijom close() se zatvara soket.

Programski kod: server.c

U ovome su poglavlju objašnjene najvažnije linije koda **(a koje nisu objašnjene u prethodnom poglavlju)** koji se nalazi u dokumentu server.c koji je dostupan preko e-Learning portala.

```
socklen_t addrlen, length;
```

socklen_t je tip podataka koji se koristi za varijable koje sadrže informacije o duljinama IP adresa. Teoretski bi se u ovakvim slučajevima mogao koristiti i obični int, ali onda bi vam kompajler generirao upozorenja (zato što je socklen_t preporučeni tip podataka u ovome slučaju).

```
skaddr.sin_addr.s_addr = htonl(INADDR_ANY);
```

Opcijom **INADDR_ANY** kažemo programu server.exe da za svoju IP adresu koristi IP adresu računala na kojemu se trenutno nalazi (jer ne znamo gdje se sve može izvršavati naš program). Funkcija **htonl()** (engl. *Host to Network Long*) je slična funkciji htons(), samo se radi sa cijelim brojevima duljine 32 (ili 64, ovisi o operacijskom sustavu) bita, a ne sa cijelim brojevima duljine 16 bita.

```
skaddr.sin_port = htons(50000);
```

Port na kojemu server sluša zahtjeve postavljamo na 50000 (to je slobodni port). Da smo umjesto tog broja napisali nulu, program bi automatski pronašao slobodni port i onda slušao na njemu.

```
int result = 0;
result = bind(ld, (struct sockaddr *) &skaddr, sizeof(skaddr));
if (result < 0)
{
      printf("Problem sa bindingom!\n");
      printf("Port mozda nije dostupan!\n");
      exit(-1);
}
```

Funkcijom **bind()** se povezuje deskriptor soketa (u našem slučaju ld) sa IP adresom soketa. Funkcija vraća -1 ako je došlo do greške.

```
result = 0;
result = getsockname(ld, (struct sockaddr *) &skaddr, &length);
if (result < 0)
{
      printf("Doslo je do greske prilikom dohvacanja imena soketa!\n");
      exit(-1);
}
```

Funkcijom **getsockname()** (engl. *Get Socket Name*) se dohvaća ime (tj. adresa) soketa čiji je deskriptor specificiran sa ld, te se njegova adresa sprema u skaddr, a dužina adrese u length. Ova funkcija vraća -1 u slučaju greške.

```
printf("Server je na portu %d\n", ntohs(skaddr.sin_port));
```

Funkcijom **ntohs()** (engl. *Network to Host Short*) se format "big-endian" konvertira u "little-endian" (ako se "little-endian" koristi na poslužiteljskom računalu).

```
if (listen(ld,5) < 0)
{
      printf("Greska prilikom osluskivanja konekcije!\n");
      exit(-1);
```

```
}
```

Funkcijom **listen()** soket se postavlja u pasivno stanje u kojemu čeka na nove zahtjeve na prethodno specificiranom portu. Prvi argument koji funkcija prima je deskriptor soketa, a drugi je maksimalni broj zahtjeva koje klijenti upute serveru, a koje se mogu nalaziti u redu čekanja servera. Funkcija vraća -1 u slučaju greške.

```
sd = accept(ld, (struct sockaddr*) &from, &addrlen);
```

Funkcijom **accept()** se prihvaća zahtjev za konekcijom na soket čiji je deskriptor ld (to je soket na serveru). Deskriptor novog soketa se nazove sd (to je soket na klijentu). Svojstva tog soketa se postave na ista svojstva koja ima i soked ld. U varijablu from se upisuje adresa soketa na klijentu, a u addrlen se upisuje veličina (u bajtovima) te adrese.

Funkcije koje će vam biti potrebne prilikom rješavanja zadatka

```
n = read(deskriptor_soketa, buffer, 20);
```

Funkcijom **read()** se čitaju podaci koje je jedan proces (npr. klijent) poslao drugome (npr. serveru). Funkcija prima tri argumenta:

- deskriptor soketa koji šalje podatke;
- međuspremnik u koji se ti podaci zapisuju (npr. definiran kao char buffer[20]);
- broj bajtova koje je potrebno pročitati iz međuspremnika. Ako se žele pročitati svi bajtovi koji su primljeni, onda se umjesto broja (u ovom slučaju 20) može upisati strlen(buffer).

Funkcija vraća broj bajtova koji su pročitani (1 bajt po pročitanom znaku).

```
write(deskriptor_soketa, rezultat, strlen(rezultat));
```

- deskriptor soketa kojemu se šalju pdoaci;
- međuspremnik u kojemu su ti podaci zapisani (npr. definiran kao char rezultat[50]);
- broj bajtova koje je potrebno pročitati iz međuspremnika i poslati soketu određenom sa deskriptor_soketa.

```
atof(ascii_znak)
```

Funkcijom **atof()** (engl. *ASCII to Float*) se ASCII znak pretvara u decimalni broj (tipa float).

Napomena: kada budete iz argv[3] kopirali broj koji je potrebno kvadrirati (koji će biti tipa char, a ne int!), nemojte zaboraviti na kraj staviti nul-string ('\0')!