# Mreže računala

**Rješenja zadataka** s prvog kolokvija, 24. studenoga 2023. godine

## RJEŠENJE PRVOG ZADATKA

UKUPNO BODOVA: 8

```
struct sockaddr_in klijentAdresa;
int lenAddr = sizeof( klijentAdresa );
int commSocket = accept(listenerSocket, (struct sockaddr *)&klijentAdresa,
                        &lenAddr);

if( commSocket == -1 )
    perror( "accept" );

char dekadskiIP[20];
if(recv(commSocket, dekadskiIP, sizeof(dekadskiIP), 0) <= 0){
    perror("error"); close(commSocket);
}

struct in_addr binarniIP;
if(inet_aton(dekasdkiIP, &binarniIP) == 0){
    herror("error"); close(commSocket);
}

struct hostent *hostInfo = gethostbyaddr((const char*)&binarniIP,
                           sizeof(binarniIP), AF_INET);
if(hostInfo == NULL){
    printf("error"); close(commSocket);
}

if(send(commSocket, hostInfo->h_name, sizeof(hostInfo->h_name), 0) <= 0){
    perror("error"); close(commSocket);
}

int i;
for(i = 0; hostInfo->h_aliases[i] != NULL; ++i){}

int i_n = htonl(i);
if(send(commSocket, &i_n, sizeof(i_n), 0) <= 0){
    perror("error");
}

close(commSocket);
```

## RJEŠENJE DRUGOG ZADATKA

UKUPNO BODOVA: 12

**Format poruka** 1 bod

- Svaka poruka sastoji se od dva dijela: zaglavlja i tijela poruke. Zaglavlje se sastoji od dva integera - prvi predstavlja duljinu poruke (ne uključujući zaglavlje), a drugi predstavlja kod kojim je jednoznačno određena jedna od dolje navedenih vrsta poruke.

**Jedan mogući dizajn sustava komunikacije** 5 bodova

1. *UNAJMI vrijeme datum broj_sati* - klijent želi dobiti popis modela autobusa (s ukupnom cijenom najma za svaki od njih) koji su raspoloživi za rezervaciju od danog vremena i datuma na željeni broj sati
2. *UNAJMIBROJ vrijeme datum broj_sati broj_putnika* - kao za prethodnu vrstu poruke s time da svaki od modela mora imati ukupan broj mjesta veći ili jednak *broj_putnika*
3. *UNAJMI_R popis* - server vraća niz uređenih parova oblika *(naziv_modela,cijena_najma)* odvojenih zarezom kao odgovor na poruku klijenta tipa *UNAJMI* ili *UNAJMIBROJ*
4. *REGISTRACIJE model* - klijent želi popis svih registracijskih oznaka autobusa željenog modela
5. *REGISTRACIJE_R popis* - server šalje niz registracijskih oznaka odvojenih zarezom
6. *CIJENA model* - klijent želi dobiti cijenu jednog sata najma autobusa željenog modela
7. *CIJENA_R cijena* - server šalje broj koji predstavlja odgovor na poruku tipa *CIJENA*
8. *BR_MJESTA model* - klijent želi dobiti broj sjedećih i broj stajaćih mjesta autobusa tog modela
9. *BR_MJESTA_R broj_stajaćih broj_sjedećih* - server šalje broj stajaćih i broj sjedećih mjesta
10. *DODATNO model* - klijent želi popis svih dodatnih sadržaja autobusa tog modela
11. *DODATNO_R popis* - server šalje niz dodatnih sadržaja autobusa odvojenih zarezom
12. *REZERVIRAJ vrijeme datum broj_sati registracija* - klijent želi rezervirati autobus dane registracije od danog vremena danog datuma na željeni broj sati
13. *REZERVIRAJ_R cijena poziv_na_broj kod* - ako je traženi autobus u željeno vrijeme raspoloživ za najam, server klijentu vraća ukupnu cijenu tog najma, poziv na broj te jedinstveni kod koji određuje tu rezervaciju. Inače, server šalje poruku tipa *ODGOVOR* da željeni autobus tada nije raspoloživ za najam.
14. *STANJE kod* - klijent želi saznati stanje rezervacije određene danim kodom i koliko je vremena (u minutama) ostalo za plaćanje te rezervacije
15. *STANJE_R stanje vrijeme* - server šalje klijentu stanje rezervacije (*neplaćeno*, *unajmljeno*, *isteklo*) te broj koji predstavlja koliko je minuta preostalo za plaćanje te rezervacije (0 ako stanje nije *neplaćeno*)
16. *BOK* - klijent šalje serveru znak za prekid komunikacije
17. *ODGOVOR poruka* - poruka je string koji je jednak *OK* ako je zahtjev uspješno obrađen, inače sadrži opis greške (greške su opisane u nastavku ovog rješenja!)

**Napomena**. Nije bilo potrebno pisati gornja objašnjenja za svaku vrstu poruke ukoliko je iz naziva dijelova tijela poruke bilo jasno što oni predstavljaju. Primjer: *broj_stajaćih*, *broj_putnika*, *model* i slično - ti nazivi dijelova tijela poruke ne zahtijeva dodatno objašnjenje, no nazivi dijelova tijela poruke poput *popis*, *poruka*, *info* i slično zahtijevaju detaljnije objašnjenje. Također, nije bilo potrebno uz svaku vrstu kao gore pisati neki broj koji ju predstavlja (npr. uz *ODGOVOR* imamo broj 17), no u tom slučaju je u donjim primjerima poruka trebalo negdje napomenuti koji broj korišten u tim primjerima predstavlja koju vrstu poruke (osim ako je umjesto 17 navedno *ODGOVOR* i sl.).

**Moguće greške** 3 boda

- neispravan format vremena, datuma, broja sati
- navedeno vrijeme i datum u prošlosti
- broj putnika negativan ili prevelik (tj. ne postoji autobus te tvrtke s dovoljnim ukupnim brojem mjesta)
- tvrtka ne posjeduje model s traženim nazivom modela
- ne postoji rezervacija za navedeni kod
- tvrtka ne posjeduje autobus s traženom registracijom ili je autobus s traženom registracijom nedostupan za najam u traženo vrijeme
- nije dostupan niti jedan model za najam u traženo vrijeme

**Primjeri klijentskih poruka i pripadni odgovori servera** 3 boda

**Napomena**: Ispod je prikazana po jedna klijentska poruka i pripadni odgovor servera za svaku točku u 3. funkcionalnosti iz teksta zadatka te za svaku od preostalih 5 funkcionalnosti (to uključuje i 6. funkcionalnost - završetak komunikacije). Izostavljene su poruke servera *[2,ODGOVOR] OK* ako je zahtjev klijenta uspješno obrađen. Naravno, u slučaju greške potrebno je prikazati odgovarajući odgovor servera (ovdje su odabrani primjeri koji ilustriraju uspješno obrađene zahtjeve klijenta).

| | |
|---|---|
| klijent: | [20,UNAJMI] 14:00h 28.12.2024 26 |
| server: | [38,UNAJMI_R] (Setra 515,367.29),(Setra 200,1546.00) |
| klijent: | [23,UNAJMIBROJ] 14:00h 28.12.2024 26 52 |
| server: | [38,UNAJMI_R] (Setra 515,367.29),(Setra 200,1546.00) |
| klijent: | [9,REGISTRACIJE] Setra 515 |
| server: | [15,REGISTRACIJE_R] VŽ314LL,ZG616FH |
| klijent | [9,CIJENA] Setra 515 |
| server: | [5,CIJENA_R] 52.73 |
| klijent: | [9,BR_MJESTA] Setra 515 |
| server: | [5,BR_MJESTA_R] 30 16 |
| klijent: | [9,DODATNO] Setra 515 |
| server: | [5,DODATNO_R] TV,WC |
| klijent: | [28,REZERVIRAJ] 14:00h 28.12.2024 26 VŽ314LL |
| server: | [36,REZERVIRAJ_R] 367.29 604060-404-VŽ314LL 1234567890 |
| klijent: | [10,STANJE] 1234567890 |
| server: | [12,STANJE_R] neplaćeno 35 |
| klijent: | [0,BOK] |
| server: | (ovdje može [2,ODGOVOR] OK) |

## RJEŠENJE TREĆEG ZADATKA

UKUPNO BODOVA: 17

1. 4 boda

```
typedef struct {
    char ime[9], pass[9];
} user;
user djelatnici[100];
int br_djelatnika;

char poslovnice[10][101];

typedef struct {
    char naziv[51], kod[11];
    int rezervirano[10], raspolozivo[10];
} artikl;

artikl* artikli;
int br_artikala;
```

**Napomene**:

- Kako ćemo u nastavku rješenja koristiti da je, primjerice, korisničko ime djelatnika string koji završava s `'\0'` umjesto `char ime[8]` u gornjem smo napisali `char ime[9]` (kako bi imali mjesta za taj dodatni znak `'\0'`).
- U prikazanom rješenju svaki artikl pamti koliko ga ima u `i`-toj poslovnici:
  - `rezervirano[i]` - koliko je tog artikla već rezervirano u `i`-toj poslovnici,
  - `raspolozivo[i]` - koliko je tog artikla raspoloživo za rezervaciju u `i`-toj poslovnici.
- Nije poznat maksimalan broj artikala pa će taj dio biti dinamički alociran na serveru.

2. Prvo jedna pomoćna funkcija koju ćemo koristiti u nastavku: 7 bodova

```
void saljiPoruku(int sock, char* user,
    char* pass, char* poruka) {
        char odgovor[1000];
        sprintf(odgovor,"4,%s,%s,%s,", user, pass, poruka);
        posalji(sock, odgovor);
}
```

**Napomene**:

- Prema zadatku možemo sami odrediti kako su odijeljene komponente tijela poruke - u ovome rješenju će između svakog dijela tijela poruke (pa i iz zadnjeg radi jednostavnosti) biti točno jedan znak zareza (`','`).
- U sljedećoj funkciji `rezerviraj` bismo mogli imati razne provjere, no napravit ćemo samo one koje se traže u zadatku.

```
void rezerviraj(int sock, char* poruka) {
    int br, vrsta, i, j;
    char kod[11], poslovnica[101], ime[9], lozinka[9];
    sscanf(poruka, "%d,%[^,],%[^,],%d,%[^,],%[^,]", &vrsta,
           ime, lozinka, &br, kod, poslovnica);
    for(i = 0; i < br_djelatnika; ++i)
        if(strcmp(djelatnici[i].ime,ime) == 0
           && strcmp(djelatnici[i].pass, lozinka) == 0)
            break;
    if(i == br_djelatnika) {
        saljiPoruku(sock,ime,lozinka,"Neispravan user/pass!");
        return;
    }
    for(i = 0; i < br_artikala; ++i)
        if(strcmp(artikli[i].kod,kod) == 0)
            break;
    if(i == br_artikala) {
        saljiPoruku(sock,ime,lozinka,"Ne postoji taj artikl!");
        return;
    }
    /* i = indeks trazenog artikla; trazimo poslovnicu (j) */
    for(j = 0; j < 10; ++j)
        if(strcmp(poslovnice[j], poslovnica) == 0) {
            if(artikli[i].raspolozivo[j] >= br) {
                artikli[i].raspolozivo[j] -= br;
                artikli[i].rezervirano[j] += br;
                saljiPoruku(sock, ime, lozinka, "OK");
                return;
            }
        }
    /* inace nema dovoljno artikla u toj poslovnici */
    saljiPoruku(sock, ime, lozinka,
        "Nedovoljno artikla u trazenoj poslovnici!");
}
```

3. 6 bodova

```
void upit(int sock) {
    char ime[9], lozinka[9], naziv[51], poruka[100];
    scanf(" %[^\n]", ime);
    scanf(" %[^\n]", lozinka);
    scanf(" %[^\n]", naziv);
    /* pitamo server za informacije o artiklu */
    sprintf(poruka, "1,%s,%s,%s,", ime, lozinka, naziv);
    posalji(sock, poruka);
    char* odgovor;
    primi(sock, &odgovor); /*primi alocira dovoljno memorije*/
    int vrsta_poruke, i;
    sscanf(odgovor,"%d", &vrsta_poruke);
```

```
    if(vrsta_poruke == 4) {
        sscanf(odgovor,"%*d,%*[^,],%*[^,],%[^,],",poruka);
        printf("Greska: %s\n", poruka);
        return;
    }
    /* izvadimo i prikazemo dobivene informacije o artiklu */
    char *popis_kodova = malloc(strlen(odgovor)),
         *popis_parova = malloc(strlen(odgovor));
    sscanf(odgovor,"%*d,%*[^,],%*[^,],%[^,],%[^,]",
           popis_kodova, popis_parova);
    printf("Kodovi:\n");
    char z;
    for(i = 0; i < strlen(popis_kodova); ++i) {
        z = popis_kodova[i];
        printf("%c", z == '.' ? '\n' : z);
    }
    printf("\nDostupnost po poslovnicama:\n");
    for(i = 0; i < strlen(popis_parova); ++i) {
        z = popis_parova[i];
        if(z == '-')
            printf(", broj komada: ");
        else if(z == '.')
            printf("\n");
        else
            printf("%c", z);
    }
    free(popis_kodova);
    free(popis_parova);
}
```

**Napomena**: Šifre su u tijelu poruke odvojene znakom `'.'` kao i parovi poslovnica i dostupnih količina pojedinog artikla u njima (pri čemu su u svakom paru naziv poslovnice i količina traženog artikla odvojeni znakom `'-'`). Uočimo da te znakove možemo koristiti za odvajanje spomenutih dijelova s obzirom da prema tekstu zadatka niti jedan od naziva u ovome zadatku ne sadrži znakove `','`, `'-'` i `'.'`.

## RJEŠENJE ČETVRTOG ZADATKA

UKUPNO BODOVA: 13

```
int sendUnion(sock, struct Set A, struct Set B){
    struct Set C;
    C.n = A.n;
    C.elements = (int*)malloc((A.n+B.n)*sizeof(int));

    for(int i = 0; i < A.n; ++i) C.elemenst[i] = A.elements[i];

    for(int i = 0; i < B.n; ++i){
        int postoji = 0;
        for(int j = 0; j < C.n; ++j)
            if(C.elements[j] == B.elements[i]) postoji = 1;
        if(!postoji) C.elements[C.n++] = B.elements[i];
    }

    int n_n = htonl(C.n);
    if(send(sock, &n_n, sizeof(n_n), 0) != sizeof(n_n)) return 1;

    for(i = 0; i < C.n; ++i){
        n_n = htonl(C.elements[i]);
        if(send(sock, &n_n, sizeof(n_n), 0) != sizeof(n_n)) return 1;
    }

    free(C.elements);
    return 0;
}
```

**Napomene**:

- Prvo je trebalo poslati broj elemenata u uniji.
- Nismo mogli pretpostaviti da su skupovi `A` i `B` disjunktni pa je trebalo provjeriti postoji li nešto u njihovom presjeku te takve elemente poslati samo jednom.
- Nije ispravno prvo poslati sve elemente iz `A` i `B` pa onda u funkciji `receiveUnion` provjeravati ima li duplića jer u zadatku piše da funkcija `sendUnion` šalje uniju.
- Nakon što se pošalje broj elemenata, treba se poslati točno toliko elemenata jer će funkcija `receiveUnion` očekivati toliko elemenata.

```
int receiveUnion(sock, struct Set *C){
    int n_n;
    if(recv(sock, &n_n, sizeof(n_n), 0) != sizeof(n_n)) return 1;
    C->n = ntohl(n_n);
    C->elements = (int*)malloc(C->n*sizeof(int));

    for(int i = 0; i < C->n; ++i){
        if(recv(sock, &n_n, sizeof(n_n), 0) != sizeof(n_n)) return 1;
        C->elements[i] = ntohl(n_n);
    }

    return 0;
}
```

H. Marciuš & S. Horvat, 30. studenoga 2023.