Sveučilište u Zagrebu
PMF – Matematički odsjek

Mreže računala

Vježbe 03

Zvonimir Bujanović
Slaven Kožić
Vinko Petričević

Mrežno programiranje: SocketAPI

- Programiramo u aplikacijskom sloju, za ostale se brinu operacijski sustav i hardware.
- Aplikacijski sloj u internetskom protokolarnom stogu obuhvaća 3 sloja referentnog modela:
  - sloj sesije – uspostava konekcije između 2 računala
  - prezentacijski sloj – konverzija podataka tako da budu pogodni za prijenos preko mreže
  - aplikacijski sloj – konkretan način komunikacije između 2 računala, svojstven samo našoj aplikaciji
- **SocketAPI** – kolekcija struktura podataka i funkcija u programskom jeziku C namjenjena mrežnom programiranju.

Ponavljanje: varijable i adrese

- Svaka deklarirana varijabla zauzima određeni raspon memorijskih adresa. Veličinu zauzete memorije doznajemo operatorom **`sizeof`**.

```
int a; char b; double c;
printf( "%d %d %d", sizeof(a), sizeof(b), sizeof(c) );
// tipicno ispisuje: 4 1 8
```

- Adresu varijable doznajemo operatorom &.
- Adresu spremamo u specijalnu varijablu koju zovemo pointer ili pokazivač.

```
int a; double c;
int *pa = &a;
double *pc = &c;
pc = &a; // compile error! Pointeri nisu istog tipa.
pc = (double *) &a; // OK. Pretvorba putem cast operatora.
```

Ponavljanje: varijable i adrese

- Pomoću operatora * možemo pristupati i modificirati sadržaj varijable na koju pokazuje pointer.
- Pointer tijekom izvođenja programa može pokazivati na različite varijable.
- Adresu varijable ne možemo promijeniti.

```
int a = 5, b = 3;
int *p;
p = &a;
printf( "a=%d, *p=%d, b=%d\n", a, *p, b ); // ispis: 5 5 3
*p = 7;
printf( "a=%d, *p=%d, b=%d\n", a, *p, b ); // ispis: 7 7 3
p = &b;
printf( "a=%d, *p=%d, b=%d\n", a, *p, b ); // ispis: 7 3 3
printf( "&a=%p, p=%p &b=%p\n", &a, p, &b );
// ispis (npr) &a=7ffff428, p=7ffff42c &b=7ffff42c
```

Ponavljanje: polja i strukture

- Pointerska aritmetika: **`niz + n = &niz[0] + n = &niz[n]`**

```
int niz[5] = {1, 2, 3, 4, 5};
int *p = &niz[1];
*p = 7;
printf( "niz[1] = %d", niz[1] ); // ispis: 7
p = p+2; *p = 9; ++p; *p = 6;
printf( "niz[3]=%d niz[4]=%d", niz[3], niz[4] ); // ispis: 9 6
```

- Ako pokazivač **`p`** pokazuje na strukturu čiji je element **`e`**, onda do vrijednosti elementa **`e`** dolazimo pomoću **`p->e`**.

```
typedef struct { int i; char c; } STR;
STR s, *p;
p = &s;
s.i = 7; p->c = 'A';
printf( "s.i = %d, s.c = %c", s.i, s.c ); // ispis: 7 A
```

SocketAPI: Header datoteke

- Svaka funkcija iz SocketAPI treba neku header datoteku.
- Jednostavnosti radi, možemo uvijek uključiti sve koje bi nam mogle zatrebati:

```
#include <sys/socket.h>
#include <sys/types.h>
#include <netdb.h>
#include <netinet/in.h>
#include <unistd.h>
#include <arpa/inet.h>
```

Rad sa mrežnim adresama

- 3 načina reprezentacije adresa:
  - *host-name* – npr. www.google.com
  - IP-adresa u dekaskom zapisu – npr. 192.84.105.1
  - IP-adresa u 32-bitnom binarnom zapisu

- Interno se sav rad sa adresama mora odvijati u 32-bitnom binarnom zapisu.
- Postoje funkcije za konverziju iz jednog zapisa u drugi.

Rad sa mrežnim adresama

- Binarni IP-zapis čuva se u specijalnoj strukturi:

```
struct in_addr
{
   unsigned long s_addr;
};
```

- host-name i dekadski IP-zapis čuvaju se kao stringovi.

dekadski IP →binarni IP

```
int inet_aton(
        const char *dekadskiIP,
        struct in_addr *binarniIP );
```

Povratna vrijednost:

0 ako dekadska adresa nije valjana, ne-nula inače.

Mnemotehnika: inet = internet, a = ASCII, n = network

Primjer:

```
char dekadskiIP[] = "161.53.8.14";
struct in_addr binarniIP;

if( inet_aton( dekadskiIP, &binarniIP ) == 0 )
   printf( "%s nije dobro zadana adresa\n", dekadskiIP );
```

binarni IP → dekadski IP

```
char *inet_ntoa( struct in_addr binarniIP );
```

Povratna vrijednost:

Pointer na memoriju unaprijed alociranu unutar funkcije. Na toj adresi je dekadska IP-adresa.

Mnemotehnika: inet = internet, a = ASCII, n = network

```
char dekadskiIP[] = "161.53.8.14";
struct in_addr binarniIP;

if( inet_aton( dekadskiIP, &binarniIP ) == 0 )
   printf( "%s nije dobro zadana adresa\n", dekadskiIP );

char natragUDekadskiIP[20];
strcpy( natragUDekadskiIP, inet_ntoa( binarniIP ) );
printf( "%s\n", natragUDekadskiIP );
```

host-name ↔ binarni IP

- koristi se specijalna struktura:

```
struct hostent {
   char *h_name;
   char **h_aliases;
   int h_addrtype;
   int h_length;
   char **h_addr_list;
};
#define h_addr h_addr_list[0]
```

- h_name – službeni host-name računala
- h_aliases – polje alternativnih host-name-ova računala (zadnji je NULL)
- h_addrtype – tip adrese, kod nas uvijek AF_INET
- h_length – duljina adrese u byteovima, kod nas uvijek 4
- h_addr_list – polje binarnih zapisa IP-adresa računala (zadnja je NULL)
- h_addr – prvi binarni zapis u gornjem polju

host-name → binarni IP

```
struct hostent *gethostbyname( const char *hostName );
```

Prima host-name kao string, vraća popunjenu hostent strukturu ili NULL ako je došlo do greške. U tom slučaju, poziv funkcije herror će na ekran ispisati poruku o greški. Struktura koja se vraća je unaprijed alocirana unutar funkcije gethostbyname!

Primjer:

```
struct hostent *hostInfo;
hostInfo = gethostbyname( "www.yahoo.com" );
if( hostInfo == NULL ) herror( "gethostbyname" );

printf( "Sluzbeni host-name: %s\n", hostInfo->h_name );
struct in_addr binarniIP=*((struct in_addr *)hostInfo->h_addr);
char *dekadskiIP = inet_ntoa( binarniIP );
printf( "Dekadska IP-adresa: %s\n", dekadskiIP );
```

binarni IP → host-name

```
struct hostent *gethostbyaddr(
   const char *binarniIP, int duljina, int tipAdrese );
```

- duljina = sizeof(binarniIP)
- tipAdrese = AF_INET (za nas, inače može biti i nešto drugo...)
- povratna vrijednost – popunjena hostent struktura

```
struct hostent *hostInfo;
struct in_addr binarniIP;
inet_aton( "161.53.8.14", &binarniIP ); // error-check...
hostInfo = gethostbyaddr(
  (const char *)&binarniIP, sizeof( binarniIP ), AF_INET );
if( hostInfo == NULL ) herror( "gethostbyaddr" );

printf( "Sluzbeni host-name: %s\n", hostInfo->h_name );
char *dekadskiIP = inet_ntoa(
   *((struct in_addr *)hostInfo->h_addr) );
printf( "Dekadska IP-adresa: %s\n", dekadskiIP );
```

Zadatak 3

- Napišite program koji se ponaša slično mrežnom alatu nslookup.
- Program sa komandne linije treba dobiti host-name nekog računala.
- Program treba ispisati sve host-name-ove i sve IP-adrese tog računala.
- Ako host-name nije bio dobar, program treba ispisati poruku o greški.