3. Identyfikacja

3.1. Określanie adresu połączonego hosta

SKŁADNIA

```
#include <sys/socket.h>
int getpeername(int socket, struct sockaddr *addr, int *addrlen);
```

OPIS

- Funkcja `getpeername` dostarcza adresu drugiej strony połączenia.
  - Parametry:
    - *`socket`* – deskryptor gniazda, przez które obsługiwane jest połączenia.
    - *`addr`* – adres, który funkcja `getpeername` wypełnia adresem odległego komputera
    - *`addrlen`* – rozmiar adresu, wypełnia funkcja `getpeername`.
  - Funkcja zwraca 0, jeśli operacja zakończy się powodzeniem, zaś gdy operacja nie powiedzie się wartość –1. Zmienna `errno` zawiera kod błędu.

```
struct sockaddr_in adres;
int dl_adres;
dl_adres=sizeof(adres);
if (getpeername(gniazdo, (struct sockaddr *) &adres, &dl_adres) != 0)
  perror("getpeername()");
else
  printf("Dane partnera: %s:%d\n",
          inet_ntoa(adres.sin_addr),ntohs(adres.sin_port));
```

3.2. Określanie adresu lokalnego hosta

SKŁADNIA

```
#include <sys/socket.h>
int getsockname(int socket, struct sockaddr *addr, int *addrlen);
```

OPIS

- Funkcja `getsockname` dostarcza adresu lokalnej strony połączenia.
  - Parametry:
    - *`socket`* – deskryptor gniazda, przez które obsługiwane jest połączenia.
    - *`addr`* – adres, który funkcja `getsockname` wypełnia adresem lokalnego komputera
    - *`addrlen`* – rozmiar adresu, wypełnia funkcja `getsockname`.
  - Funkcja zwraca 0, jeśli operacja zakończy się powodzeniem, zaś gdy operacja nie powiedzie się wartość –1. Zmienna `errno` zawiera kod błędu.

```
struct sockaddr_in adres;
int dl_adres;
dl_adres=sizeof(adres);
if (getsockname(sock, (struct sockaddr *) &adres, &dl_adres) != 0)
  perror("getsockname()");
else
  printf("Dane lokalne: %s:%d\n",
          inet_ntoa(adres.sin_addr),ntohs(adres.sin_port));
```

Uwaga: w kliencie należy wywołać funkcję po `connect()`.

3.3. Określanie nazwy lokalnego hosta

SKŁADNIA

```
#include <unistd.h>
int gethostname(char *name, size_t len);
```

OPIS

- Funkcja `gethostname` dostarcza nazwę lokalnego hosta.
- Parametry:
  - *`name`* – bufor, w którym będzie umieszczona nazwa
  - *`len`* – rozmiar bufora
  - *`addrlen`* – rozmiar adresu, wypełnia funkcja `getpeername`.
- Funkcja zwraca 0, jeśli operacja zakończy się powodzeniem, zaś gdy operacja nie zakończy się powodzeniem wartość –1. Zmienna `errno` zawiera kod błędu.

```
char nazwa[50];
if (gethostname(nazwa,sizeof(nazwa)) != 0)
  perror("gethostname()");
else
  printf("Nazwa mojego hosta: %s\n",nazwa);
```

3.4. Odwzorowanie nazwy domenowej na adres IP - `gethostbyname()`

SKŁADNIA

```
#include <netdb.h>
struct hostent *gethostbyname(const char *name);
```

OPIS

- Funkcja `gethostbyname` pobiera ciąg znaków ASCII reprezentujący nazwę domenową komputera (parametr *name*) i zwraca wskaźnik do wypełnionej struktury `hostent` zawierającej między innymi 32-bitowy adres komputera. Jeśli funkcja `gethostbyname` zostanie wywołana z adresem IP zamiast nazwy, to w strukturze IP wypełniona zostanie tylko jedna składowa – pierwsza pozycja `h_addr_list`.
- Funkcja po poprawnym wykonaniu zwraca wskaźnik do struktury `hostent`, w przypadku wystąpienia błędu zwracana jest wartość NULL i ustawiana jest zmienna globalna `h_errno`. Informację o błędzie można uzyskać za pomocą funkcji `herror()`.
- Definicja struktury `hostent` (plik `netdb.h`):

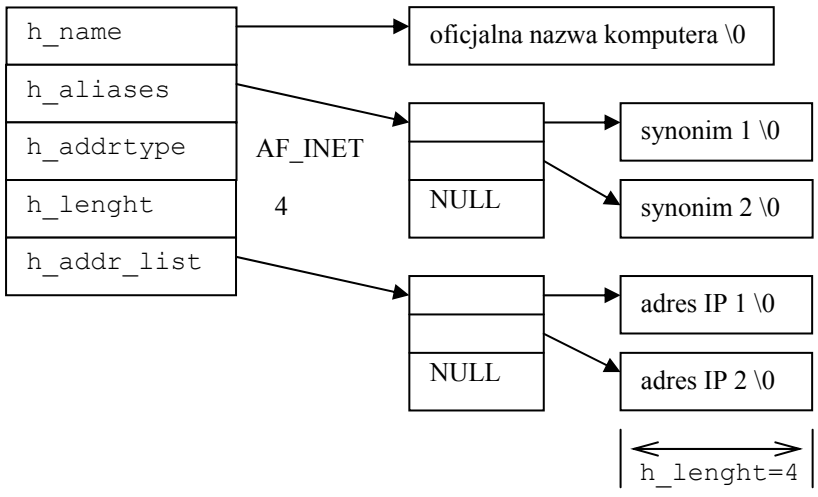
```
struct hostent {
  char *h_name;        /* oficjalna nazwa domenowa komputera */
  char **h_aliases;    /* synonimy */
  int h_addrtype;      /* typ adresu  */
  int    h_length;     /* długość adresu */
  char **h_addr_list; /* lista adresów */
};
/* pierwszy adres w tablicy adresów */
#define h_addr h_addr_list[0]
```

gdzie:

`h_name` wskazuje napis zawierający oficjalną nazwę domenową hosta
`h_aliases` jest wskaźnikiem do tablicy wskaźników zawierających inne nazwy hosta
`h_addrtype` jest określa typ adresu (stała `AF_INET` dla IPv4)
`h_length` określa długość (w bajtach) adresów w `h_addr_list` (dla IPv4 będzie zawsze równy 4)
`h_addr_list` zawiera listę adresów IP związanych z oficjalną nazwą, adresy przechowywane są w postaci sieciowej kolejności bajtów. Adresy pobierane są z pliku `/etc/hosts`, bazy NIS lub DNS – w zależności od konfiguracji.

W pliku nagłówkowym zdefiniowano również nazwę `h_addr` jako odwołanie do pierwszego elementu listy adresów IP hosta, wykorzystywane w starszym oprogramowaniu.

***Struktura* `hostent`**



- Przykład

```
struct hostent *hptr;
char *nazwa = "oceanic.wsisiz.edu.pl";
if (hptr = gethostbyname(nazwa))
{
  /* umieszczono adres IP w hptr->h_addr */
}
else
{
/* błąd w nazwie – odpowiednie działania */
}
```

- Przykład funkcji, która zwraca adres IP w postaci binarnej

```
unsigned long OdwzorujNazwe(char nazwa[])
{
  struct hostent *host; // funkcja gethostbyname zwraca
                        // wskaźnik
  if ((host = gethostbyname(nazwa)) == NULL) {
    fprintf(stderr, "gethostbyname(): nie udało się");
    exit(1);
  }
  // zwroc adres IP w postaci binarnej
   return *((unsigned long *) host->h_addr_list[0]);
}
```

- Przykład funkcji, która wstawia adres IP do składowej struktury adresowej sockaddr_in

```
int resolve_name( struct sockaddr_in *addr, char *hostname )
{
  addr->sin_family = AF_INET;
  addr->sin_addr.s_addr = inet_addr(hostname);
  if ( addr->sin_addr.s_addr = 0xffffffff ) {
    struct hostent *hp;
    hp = (struct hostent *)gethostbyname( hostname );
    if (hp == NULL) return -1;
    else {
      memcpy( (void *)&addr->sin_addr,
              (void *)hp->h_addr_list[0],
              sizeof(addr->sin_addr) );
    }
  }
  return 0;
}
```

3.5. Odwzorowanie adresu IP na nazwę hosta

SKŁADNIA

```
#include <netdb.h>
#include <sys/socket.h>        /* for AF_INET */
struct hostent *gethostbyaddr(const char *addr, int len, int type);
```

OPIS

- Funkcja `gethostbyaddr` pobiera adres IP (parametr *`addr`*) i zwraca wskaźnik do wypełnionej struktury `hostent` zawierającej między innymi nazwę domenową hosta.
- Argumenty:
  - `addr` - wskaźnik do tablicy zawierającej adres IP (uwaga: adres jest liczbą binarną)
  - `len` - rozmiar adresu (4 dla IPv4)
  - `type` - typ adresu (AF_INET dla adresu IPv4)

- Przykład:

```
#include <netdb.h>

struct hostent *host;
struct in_addr in;
inet_aton("213.135.44.33", &in);
if ( (host=gethostbyaddr((char *)&in.s_addr,
                         sizeof(in.s_addr),AF_INET))) {
  printf("Host name is %s\n",host->h_name);
```

3.6. Nowe funkcje odwzorowujące

- Standard Posix wprowadza nowe funkcje odwzorowujące:

  - `getaddrinfo()` - dostarcza struktury adresowej właściwej dla protokołu (nazwa na adres IP)
  - `getnameinfo()` - łączy funkcjonalność `gethostbyaddr()` i `gethostbyport()` (adres IP na nazwę)

- Przykład 1 - Informacje uzyskiwane z `gethostbyname`

```
#include <stdio.h>          //printf() perror()
#include <string.h>         //strcmp()
#include <unistd.h>         //gethostname()
#include <netdb.h>          //gethostbyname() gethostbyaddr()
#include <sys/socket.h>     //AF_INET inet_aton() inet_ntoa()
#include <netinet/in.h>
#include <arpa/inet.h>

int main()
{
  struct hostent *host;
  struct in_addr in;
  int i;

  host=gethostbyname("www.microsoft.com");
  if (host == NULL) return 1;
  else {
    printf("h_name is %s\n", host->h_name);
    printf("h_addrtype is %d\n", host->h_addrtype);

    i=0;
    printf("Aliases:\n");
    while (1) {
      if (host->h_aliases[i]) {
        printf("h_aliases[%d] = %s\n",i,host->h_aliases[i]);
        i++;
      } else break;
    }

    i=0;
    printf("Addresses:\n");
    while (1) {
      if (host->h_addr_list[i]) {
        memcpy(&in.s_addr,host->h_addr_list[i],sizeof(in.s_addr));
        printf("h_addr_list[%d] = %s\n",i,inet_ntoa(in));
        i++;
      } else break;
    }
  }
  return 0;
}
```

- Przykład 2: Zamiana nazwy domenowej na adres IP

```
#include <netdb.h>
#include <stdio.h>          //printf() perror()
#include <string.h>         //strcmp()
#include <unistd.h>         //gethostname()
#include <netdb.h>          //gethostbyname() gethostbyaddr()
#include <sys/socket.h>           //AF_INET inet_aton() inet_ntoa()
       #include <netinet/in.h>
       #include <arpa/inet.h>
int resolve_name( struct sockaddr_in *addr, char *hostname );
int main(int argc, char *argv[])
{
  struct sockaddr_in addr;
  int ret;
  ret = resolve_name(&addr, argv[1]);
  if (ret==0) {
    printf("address is %s\n", inet_ntoa(addr.sin_addr));
  } else return 1;
  return 0;
}

int resolve_name( struct sockaddr_in *addr, char *hostname )
{
  addr->sin_family = AF_INET;
  addr->sin_addr.s_addr = inet_addr( hostname );
  if ( addr->sin_addr.s_addr = 0xffffffff ) {
    struct hostent *hp;
    hp = (struct hostent *)gethostbyname( hostname );
    if (hp == NULL) return -1;
    else {
      memcpy( (void *)&addr->sin_addr,
              (void *)hp->h_addr_list[0],
              sizeof( addr->sin_addr ) );
    }
  }
  return 0;
}
```

3.7. Odwzorowanie nazwy usługi na numer portu - `getservbyname()`

SKŁADNIA

```
#include <netdb.h>
  struct servent *getservbyname(const char *name, const char *proto);
  struct servent *getservbyport(int port, const char *proto);
```

OPIS

- Funkcja `getservbyname` pobiera dwa napisy reprezentujące odpowiednio nazwę usługi oraz nazwę protokołu komunikacyjnego warstwy transportowej i zwraca wskaźnik do struktury `servent`. W przypadku wystąpienia błędu (brak definicji usługi w pliku `/etc/services`) zwracana jest wartość NULL.
- Funkcja `getservbyport` dokonuje odwzorowania numery portu na nazwę usługi.
- Definicja struktury `servent` (plik `netdb.h`):

```
struct servent
{
   char  *s_name;    /* oficjalna nazwa usługi */
   char **s_aliases; /* synonimy */
   int    s_port;    /* port dla tej usługi w sieciowym porządku
                        bajtów */
   char  *s_proto;   /* protokół, którego należy użyć */
};
```

- Przykład

```
struct servent *sptr;
if  (sptr = getservbyname("smtp","tcp"))
{
  /* numer portu umieszczono sptr->s_port */
}
else {
  /* błąd – odpowiednie działania */
}
```

- UWAGA: Numer portu w strukturze `servent` reprezentowany jest w sieciowym porządku bajtów. Aby poprawnie odczytać jego wartość należy dokonać konwersji do postaci obowiązującej na lokalnym komputerze (funkcja `ntohs`).
- Przykład funkcji, która zwraca numer portu w postaci binarnej

```
unsigned short OdwzorujUsluge(char usluga[],
                              char protokol[])
{
  struct servent *usl;
  unsigned short port;

/* Czy port podany jako liczba?  */
  if ((port = atoi(usluga)) == 0)  {
    if ((usl = getservbyname(usluga, protokol)) == NULL)
    {
      fprintf(stderr, "getservbyname() failed");
      exit(1);
    }
    else
      port = usl->s_port;
  }
  else
    port = htons(port);
  return port;
}
```

3.8. Odwzorowanie nazwy protokołu na jego numer - `getprotobyname()`

SKŁADNIA

```
#include <netdb.h>
  struct protoent *getprotobyname(const char *name);
  struct protoent *getprotobynumber(int proto);
```

OPIS

- Funkcja `getprotobyname` pobiera napis reprezentujący nazwę protokołu i zwraca wskaźnik do struktury `protoent` zawierającej między innymi liczbę całkowitą przypisaną temu protokołowi. W przypadku wystąpienia błędu (brak nazwy protokołu w pliku `/etc/protocols`) zwracana jest wartość NULL.
- Funkcja `getprotobynumber` dokonuje odwzorowania numeru protokołu na nazwę.
- Definicja struktury `protoent` (plik `netdb.h`):

```
struct protoent
{
   char  *p_name;     /* oficjalna nazwa protokołu */
   char **p_aliases;  /* synonimy */
   int    p_proto;    /* oficjalny numer protokołu */
};
```

- Przykład

```
struct protoent *pptr;
if  (pptr = getprotobyname("udp"))
{
  /* numer protokołu umieszczono w pptr->p_proto */
}
else
{
  /* błąd – odpowiednie działania */
}
```

Należy przeczytać:

Douglas E. Comer, David L. Stevens: *Sieci komputerowe TCP/IP, tom 3*: str. 89-96

W. Richard Stevens: *Unix, programowanie usług sieciowych, tom 1: API gniazda i XTI*: str. 277-299