Kommunikationsnetze - Übung 2
Socket-Programmierung (1)

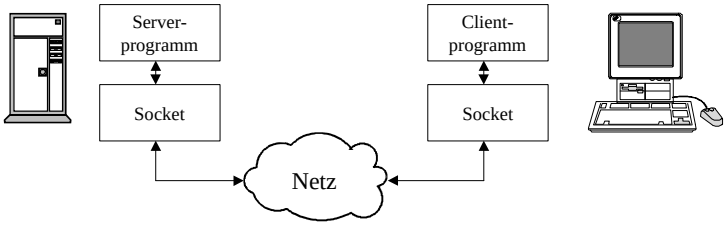
Lernziele:

- Begriffe "Client" und "Server"
- Aufgaben eines "Socket"
- Programmierung eines "Client-Socket"
- Socket öffnen mit der Funktion `socket()`
- Parameter der Funktion `socket()`
- Verbindung aufbauen mit der Funktion `connect()`
- Parameter der Funktion `connect()`
- Verbindungsaufbau prüfen
- Verbindung schließen mit der Funktion `close()`
- Ein einfacher "Port-Scanner"

Client und Server

Server stellt einen Dienst bereit

Client nutzt den Dienst



Socket = Kommunikationsendpunkt = Datenstruktur mit Pufferspeicher
- speichert ankommende und abgehende Datenbytes
- Ressource des Betriebssystem, wird einem Prozess zugeordnet

Server hat einen „Server Socket“ (listening socket) = wartet auf Anrufe
Client hat einen „Client Socket“ = kann Verbindung zum Server aufbauen

Programmierung eines Client-Socket

1.) Socket einrichten mit der Funktion `socket()`
   → Auswahl des Adressformats (z.B.: IP-Adressformat)
   → Auswahl des Transportprotokolls (TCP oder UDP)

2.) Gewünschte Ziel-Adresse festlegen = Speicherung in Datenstruktur
   → IP-Adresse des Servers
   → Portnummer des Servers-Programmes

3.) Verbindung herstellen mit der Funktion `connect()`

4.) Daten lesen und schreiben mit `read()` und `write()`

5.) Verbindung schließen mit `close()`

Beispielprogramm

Datenstrukturen: (definiert in `sys/socket.h`)

`struct sockaddr_in`	Internet Socket

Funktionen:

`socket()`	richtet einen Socket ein (`socket.h`)
`inet_addr()`	wandelt IP-Adresse von Text in 32-Bit-Binärformat (`arpa/inet.h`)
`htons()`	vertauscht die Bytes einer Integer-Zahl (htons = host to network short)
`connect()`	stellt Verbindung zum Server her (`socket.h`)
`perror()`	gibt eine Fehlermeldung aus (`stdio.h`)
`close()`	beendet die Verbindung zum Server

Die Funktion `socket()`

Deklaration in `socket.h`:

```
int socket(int domain, int type, int protocol);
```

Beispiel:

```
socket_nummer = socket(AF_INET, SOCK_STREAM, 0);
```

gibt eine Nummer (file descriptor) zur Identifikation des Socket zurück

Parameter: 1) domain (AF = Adress-Familie), z.B.:
- AF_INET = Internet-Adressen
- AF_UNIX = UNIX-Adressen für interne Kommunikation

2) type (Transport Protokoll Typ), z.B.:
- SOCK_STREAM = bei IP → TCP (verbindungsorientiert)
- SOCK_DGAM = bei IP → UDP (datagramm, verbindungslos)

3) protocol (falls weitere Protokoll-Optionen vorhanden)
- meist 0, d.h. automatische Protokollwahl

Socket-Adressformate

Wichtig: es gibt verschiedene Adressformate
Für Internet (IP): Deklaration in `netinet/in.h`:

```
struct sockaddr_in
  {
    short int           sin_family;   = Adressfamilie (16 Bit)
    unsigned short int  sin_port;     = Portnummer (16 Bit)
    struct in_addr      sin_addr;     = IP-Adresse (32 Bit)
  };
```

mit der folgenden IP-Adressstruktur:

```
struct in_addr
  {
    unsigned long int   s_addr;     = 32-Bit-Binärformat
   };
```

Die Funktion `connect()`

Deklaration in `socket.h`:

```
int connect(int sockfd, const struct sockaddr *addr,
      size_t len);
```

Beispiel:

```
ergebnis = connect(socket_nummer, &adressinfo, laenge);
```

ergibt Ergebnis = 0 falls erfolgreich, Ergebnis = -1 falls Fehler

Parameter: 1) sockfd (socket file descriptor)
= die von `socket()` vergebene Nummer
2) addr (address)
= ein Zeiger auf eine Datenstruktur mit Adressinformationen
3) len (length)
= die Länge der Adressinformation
(d.h. Anzahl Bytes der Datenstruktur)

Fehlerbehandlung

Im Fehlerfall: Funktion `connect()` schreibt Fehlercode in die externe Variable `errno` (= Error Number), eine Fehlerbeschreibung wird als Text abgespeichert.

- Fehlercodes sind in der Header-Datei `errno.h` definiert
- Funktion `strerror()` liefert einen Zeiger auf die Fehlerbeschreibung

Beispiel:

```
#include <string.h>
char *fehlertext;
int fehlernummer;
fehlertext = strerror(fehlernummer);
printf(" %s ",fehlertext);
```

- Funktion `perror()` gibt die Fehlerbeschreibung aus

Beispiel:

```
#include <stdio.h>
perror("Fehler: ");
```

Lösungshinweise Aufgabe 2

1) Portnummer als Variable definieren:

```
unsigned short int    portnummer;
```

2) Portnummer in einer `for`-Schleife erhöhen

```
for (portnummer=0; portnummer<100;portnummer++)
  {
   socket_nummer = socket(AF_INET, SOCK_STREAM, 0);
   adressinfo.sin_family = AF_INET;
   adressinfo.sin_addr.s_addr = inet_addr("192.168.103.10");
   adressinfo.sin_port = htons(portnummer);
   laenge = sizeof(adressinfo);
   ergebnis = connect(socket_nummer, &adressinfo, laenge);
   if (ergebnis == 0)
     {
      printf("\n Verbindung Port %d erfolgreich",portnummer);
     }
   close(socket_nummer);
  }
```


Literaturhinweise

Anleitung zur Socket-Programmierung:

`http://www.kryptocrew.de/archiv/linux/programming/socket-tipps.html`