Kommunikationsnetze – Übung 4
Server-Socket programmieren

Lernziele:

- Programmierung eines "Server-Socket"
- Socket binden mit `bind()`
- Parameter der Funktion `bind()`
- Server aktivieren mit der Funktion `listen()`
- Parameter der Funktion `listen()`
- Verbindung annehmen mit der Funktion `accept()`
- Parameter der Funktion `accept()`
- Netzwerkverbindungen auflisten, Linux-Befehl `netstat`
- Prozesse auflisten, Linux-Befehl `ps`
- Prozesse beenden, Linux-Befehl `kill`

Prinzip

1) Socket erzeugen mit der `socket()`-Funktion

   → Adressfamilie (=Internet) und Transportprotokoll (=TCP) zuweisen und Datenpuffer anlegen

2) Socket "binden" mit der `bind()`-Funktion

   → IP-Adresse und Portnummer zuweisen

3) Socket aktivieren mit der `listen()`-Funktion

   → danach können Verbindungen aufgebaut werden

4) Verbindung annehmen mit der `accept()`-Funktion

   → die Funktion wartet auf einen Verbindungsaufbau ("blockierend")

Die Funktion `bind()`

Syntax:

```
int bind(int socketnummer, struct sockaddr *info, int laenge);
```

Übergabeparameter:

`socketnummer` = die Nummer des mit `socket()` erzeugten Sockets

`*info` = ein Zeiger auf eine Datenstruktur
die Datenstruktur enthält die IP-Adresse und die Portnummer

`laenge` = Größe der Datenstruktur in Bytes

Rückgabewert:

`-1` wenn ein Fehler auftritt

Die Funktion `listen()`

Syntax:

```
int listen(int socketnummer, int anzahl_verbindungen);
```

Übergabeparameter:

`socketnummer` = die Nummer des mit `socket()` erzeugten Sockets

`anzahl_verbindungen` = Größe der "Warteschlange"
maximale Anzahl Verbindungen, die angenommen werden
danach wird der Fehler „connection refused“ zurückgegeben

Rückgabewert:

`-1` wenn ein Fehler auftritt

Die Funktion `accept()`

Syntax:

```
int accept(int serversocket, struct sockaddr *clientinfo,
           int laenge);
```

Übergabeparameter:

`serversocket` = die Nummer des Server-Sockets

`clientinfo` = ein Zeiger auf eine Datenstruktur mit Informationen über den Client, der die Verbindung aufgebaut hat

`laenge` = Anzahl Bytes der Client-Informations-Datenstruktur

Rückgabewert:

`-1` wenn ein Fehler auftritt

sonst wird ein zusätzlicher Socket erzeugt. Der Rückgabewert ist die Nummer dieses neuen Sockets.

Die Funktion `inet_ntoa()`

Internet-Adresse vom Netzwerk-Byte-Format in ASCII wandeln

Syntax:

```
char *inet_ntoa(unsigned int IP_Adresse);
```

Übergabeparameter:

`IP_Adresse` = eine IP-Adresse im 32 Bit Binärformat

Rückgabewert:

`NULL` wenn ein Fehler auftritt

sonst: ein Zeiger vom Typ char, der auf eine IP-Adresse im Textformat zeigt.

Server starten

Problem: Server wartet nach dem Start auf Verbindungswünsche
→ keine weitere Eingabe über die Tastatur möglich

Abhilfe: Server als Hintergrundprozess starten

```
server &
```

mehrere Serverprozesse starten:

```
server & server & server &
```


Netzwerkverbindungen auflisten

Linux Befehl `netstat` listet die aktiven Netzwerkverbindungen auf

Beispiel: `netstat –l`

Eigener Server auf Port 5000 (auf PC04 in C303b)

```
pc04@pc04:~/socket > netstat -l
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 pc04.c303b:5000         *:*                     LISTEN
tcp        0      0 *:pop3                  *:*                     LISTEN
tcp        0      0 *:telnet                *:*                     LISTEN
tcp        0      0 *:ftp                   *:*                     LISTEN
tcp        0      0 *:www-http              *:*                     LISTEN
udp        0      0 pc04.c303b:netbios-dgm  *:*
udp        0      0 pc04.c303b:netbios-ns   *:*
udp        0      0 *:netbios-dgm           *:*
Active UNIX domain sockets (only servers)
Proto RefCnt Flags       Type       State         I-Node Path
unix  0      [ ACC ]     STREAM     LISTENING     84     /dev/gpmctl
unix  0      [ ACC ]     STREAM     LISTENING     124    /var/run/.nscd_socket
unix  0      [ ACC ]     STREAM     LISTENING     131    /tmp/.axnetipc
```

Prozesse auflisten und beenden

Problem: wie beendet man einen aktiven Server-Prozess ?

1) Prozessnummer ermitteln

Syntax: `ps`

```
pc04@pc04:~/socket > ps
  PID TTY          TIME CMD
  212 tty1     00:00:00 bash
  379 pts/0    00:00:00 server
  380 pts/0    00:00:00 ps
```

Eigener Server ist Prozess Nummer 379

2) Der Linux-Befehl `kill` beendet Prozesse

Syntax: `kill option prozessnummer`

Beispiel: `kill –1 379` beendet Prozess Nr. 379

Bildschirmausgabe der Musterlösung

```
pc04@pc04:~/socket > knu4a1 &
[1] 372

 Server: socket()...
 Server: bind()...
 Server; listen()...
 Server mit IP 192.168.103.24 an Port 5000 wartet...

pc04@pc04:~/socket > knu4a2

 Client: socket()...
 Server: accept()...Verbindung mit 192.168.103.24
 Server: close()...
 Serverprogramm beendet

 Client: connect()...
 Verbindung zu IP 192.168.103.24 - Port 5000
 Verbindungsaufbau erfolgreich
 Client: close()...

[1]+  Exit 26                 knu4a1
```

Start des Servers knu4a1

Start des Clients knu4a2

Server akzeptiert die Verbindung

Client meldet Verbindungsaufbau