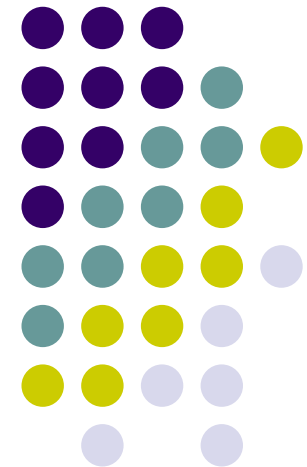


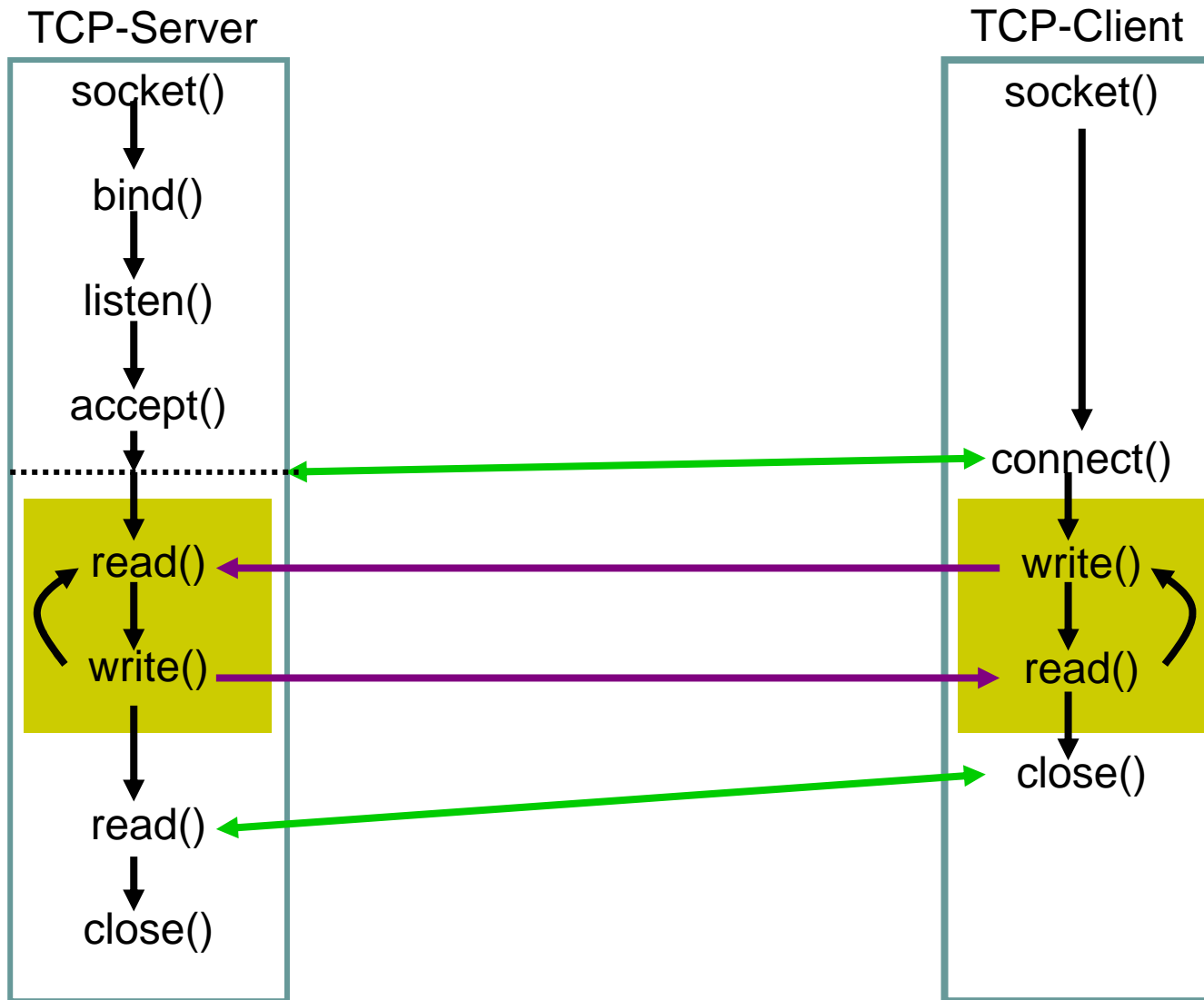
Einführung in die Systemprogrammierung

6

Netzwerkprogrammierung
Sockets



Elementare TCP-Socket Funktionen



socket()

```
#include <sys/types.h>  
#include <sys/socket.h>
```

```
int socket (int domain, int typ, int protokoll)
```

Rückgabewert: Socketdeskriptor (bei Erfolg), -1 (bei Fehler)

- domain: legt zu benutzende Protokoll- bzw. Adressfamilie fest
 - Konstante PF_ (Protokollfamilie) oder AF_ (Adressfamilie)
 - Ursprünglich sollte eine Protokollfamilie mehrere Adressfamilien unterstützen
 - PF_-Konstante sollte beim Anlegen des Sockets verwendet werden
 - AF_-Konstante sollte in den Socket-Adressstrukturen verwendet werden
 - Beispiel: PF_INET (AF_INET) für IPv4-Protokoll
 - Siehe Datei /usr/include/bits/socket.h

socket()

```
#include <sys/types.h>  
#include <sys/socket.h>
```

```
int socket (int domain, int typ, int protokoll)
```

Rückgabewert: Socketdeskriptor (bei Erfolg), -1 (bei Fehler)

- typ:
 - SOCK_STREAM: Stream-Socket (z. B. TCP)
 - SOCK_DGRAM: Datagramm-Socket (z. B. UDP)
 - SOCK_RAW: Raw-Socket
 - siehe `/usr/include/bits/socket.h`

socket()

```
#include <sys/types.h>  
#include <sys/socket.h>
```

```
int socket (int domain, int typ, int protokoll)
```

Rückgabewert: Socketdeskriptor (bei Erfolg), -1 (bei Fehler)

- protokoll:
 - wählt das zu benutzende Protokoll
 - gibt man 0 an, wird das Standardprotokoll des entsprechenden Typs verwendet
 - PF_INET und SOCK_STREAM: TCP
 - PF_INET und SOCK_DGRAM: UDP
 - PF_INET und SOCK_RAW: IPv4
- Beachte: ein mit socket() kreierter Socket ist noch nicht initialisiert und noch nicht mit einer Ressource verbunden, kann somit noch nicht verwendet werden

Beispiel socket()

```
#include <sys/types.h>
#include <sys/socket.h>

int socket_nummer = -1; //Socketdiskriptor

if((socket_nummer = socket(AF_INET, SOCK_STREAM,0))<0)
{
    printf("Socket Initialisierung fehlgeschlagen\n");
    return -1;
}
```

bind()

```
#include <sys/types.h>
#include <sys/socket.h>
```

```
int bind (int sockfd, struct sockaddr *my_addr, socklen_t addrlen)
```

Rückgabewert: 0 (bei Erfolg), -1 (bei Fehler)

Zuordnen einer lokalen Protokolladresse (IP-Adresse + TCP-Port)

- sockfd:
 - Socketdeskriptor des entsprechenden Sockets
- *my_addr:
 - Adresse der entsprechenden Socket-Adressstruktur
- addrlen:
 - Länge der Socket-Adressstruktur

Die Struktur sockaddr_in

- Socket-Adressstruktur für IPv4
- Definiert in <netinet/in.h>

```
struct sockaddr_in
{
    sa_family_t sin_family;           /* AF_INET */
    in_port_t sin_port;              /* 16-Bit Port-Nummer */
    struct in_addr sin_addr;         /* 32-Bit IPv4-Adresse
    ....                             /* weitere Komponenten */
}
```

- Port in Network-Byte-Order (big-endian)
- Konvertierung von Host-Byte-Order in Network-Byte-Order mit Funktion
 - uint16_t htons(uint16_t hostshort);

Beispiel bind()

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h> // für IP Adresstruktur
#include <arpa/inet.h> // für Konvertierungsfunktionen

int port=1313;
struct sockaddr_in addr;
addr.sin_family = AF_INET;
addr.sin_addr.s_addr = INADDR_ANY; /*Server akzeptiert Anfragen
                                     von jeder IP-Adresse*/
addr.sin_port = htons(port); /*htons wandelt von Host-Byte-Order in
                               Network-Byte-Order um. (Mehr unter "man htons") */

if((bind(socket_nummer,&addr,sizeof(addr)))<0)
{
    printf("Bind fehlgeschlagen\n");
    return -1;
}
```

Warten auf Verbindungsanforderungen

- Funktion listen():
 - Nach Aufruf von listen ist Server bereit mit anderen Prozessen über diesen Socket Verbindungen einzugehen
 - Vor Aufbau einer wirklichen Verbindung, Aufruf der
- Funktion accept()
 - Akzeptieren eines Verbindungsversuchs von einem Client
 - Accept kann bereits vor einem Verbindungsversuch ausgeführt werden, blockiert dann so lange bis ein Verbindungswunsch auftritt

Warten auf Verbindungsanforderungen

```
#include <sys/types.h>
#include <sys/socket.h>
```

```
int listen (int sockfd, int backlog);
```

Rückgabewert: 0 (bei Erfolg), -1 (bei Fehler)

```
int accept(int sockfd, struct sockaddr *addr, socklen_t addrlen);
```

Rückgabewert: Socketdescriptor (bei Erfolg), -1 (bei Fehler)

- sockfd:
 - Socketdeskriptor des entsprechenden Sockets
- backlog:
 - Maximal erlaubte Anzahl von anstehenden Verbindungswünschen seitens des Clients

Beispiel listen() und accept()

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h> // für IP Adresstruktur
#include <arpa/inet.h> // für Konvertierungsfunktionen

int client; // socket-descriptor (dient als file-descriptor für write-operation)
if (listen(socket_nummer, 5)<0)
{
    printf("Listen fehlgeschlagen");
    return -1;
}
else
{
    printf("Server etabliert\nSocket Port: %d\n",ntohs(addr.sin_port));
}
if((client = accept(socket_nummer, NULL, NULL)) == -1)
{
    printf("Accept fehlgeschlagen\n");
    return -1;
}
printf("Verbindung hergestellt\n");
}
```

Aufbauen einer Verbindung zum Server

```
#include <sys/types.h>  
#include <sys/socket.h>
```

```
int connect (int sockfd, const struct sockaddr *serv_addr, socklen_t addrlen);  
Rückgabewert: 0 (bei Erfolg), -1 (bei Fehler)
```