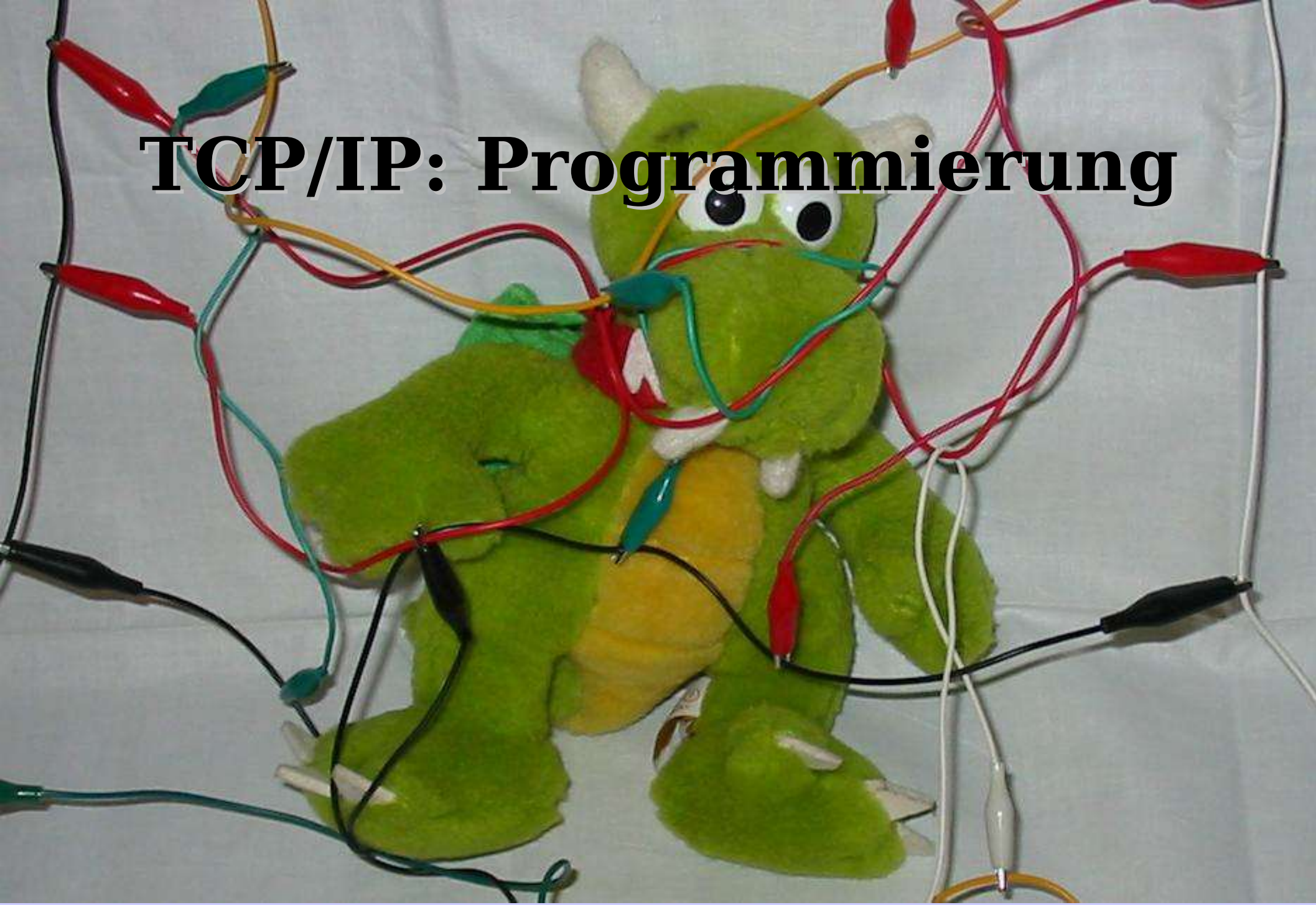

TCP/IP: Programmierung

Sockets

- verbreitetste Netzwerk-API, andere:
  - TLI (AT&T), XLI (X/Open)
- entstand in Berkeley Unix
- benutzt Datei-Handle ähnliche Funktionen

Socket erzeugen

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


int hdl;
hdl=socket(PF_INET,SOCK_STREAM,0);
```

- socket(2) erzeugt einen Handle, der weiterbenutzt werden kann
- PF_INET: TCP/IP verwenden
- SOCK_STREAM: Subprotokoll für Datenströme (TCP)

Hostname auflösen

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


int hdl;
struct hostent*hent;
hdl=socket(PF_INET,SOCK_STREAM,0);
hent=gethostbyname2(“localhost”,
    AF_INET);
if(hent==0)
    exit(1);
if(hent->h_addrtype!=AF_INET)
    exit(1);
/* hent->h_addr_list ...*/
```

- gethostbyname(3) und gethostbyname2(3) rufen transparent DNS auf und liefern Ergebnis zurück
- errno enthält Fehler
- h_addr_list ist NULL-terminierte Liste von Pointern

Client Socket verbinden

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


int hdl;
struct hostent*hent;
struct sockaddr_in sa;
hdl=socket(PF_INET,SOCK_STREAM,0);
hent=gethostbyname2(“localhost”,
    AF_INET);
/*Fehlerbehandlung*/
if(hent->h_addr_list[0]==0)
    exit(0);
sa.sin_family=AF_INET;
memcpy(sa.sin_addr,
    hent->h_addr_list[0],4);
sa.sin_port=htons(5555);
if(connect(hdl,&sa,sizeof(sa))!=0)
    exit(1);
```

- sockaddr_in enthält notwendige Daten für Verbindungsziel
- connect(2) versucht Verbindung, liefert 0 bei Erfolg
- Fehler in errno

Server Socket

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


int hdl;
int hdl2;
struct hostent*hent;
struct sockaddr_in sa;
hdl=socket(PF_INET,SOCK_STREAM,0);
sa.sin_family=AF_INET;
sa.sin_addr.s_addr=
    htonl(INADDR_ANY);
sa.sin_port=htons(5555);
if(bind(hdl,&sa,sizeof(sa))!=0)
    exit(1);
if(listen(hdl,5)!=0)
    exit(1);

/*Comm-Socket:*/
hdl2=accept(hdl);
```

- bind(2) bindet Socket an einen lokalen Port
- listen(2) versetzt Socket in Server-Mode
- accept(2) wartet auf Client und gibt Kommunikations-Socket zurück

Daten senden/empfangen

```
/*...includes*/
#include <unistd.h>


int hdl;
int ret;
char buf[1024];
/*socket, gethostbyname2,
 connect*/

/*lesen:*/
ret=read(hdl,buf,sizeof(buf));

/*schreiben:*/
ret=write(hdl,buf,DATALENGTH);
```

- read/write sind Unix-Standard-Funktionen
- Rückgabe: Länge der gelesenen/ geschriebenen Daten; 0=Ende; <0=Fehler
- Fehler in errno

Verbindung schließen

```
/*...includes*/
#include <unistd.h>
#include <sys/socket.h>


int hdl;
/*socket, gethostbyname2,
 connect, read/write*/

/*optional:*/
shutdown(hdl,SHUT_RDWR);

/*Socket löschen:*/
close(hdl);
```

- shutdown(2) beendet Kommunikation
  - SHUT_RD - kein Lesen mehr
  - SHUT_WR - kein Schreiben mehr
  - SHUT_RDWR - nix mehr
- close(2) löscht Socket und macht imliziten shutdown

Asynchrone Sockets

- fcntl(2) (GET_FL/SET_FL) wird benutzt, um das O_NONBLOCK Flag zu setzen
- select(2)/poll(2) kann benutzt werden, um auf Events zu warten
- connect(2) und accept(2) senden das “READ” Event

Winsock2

- Winsock 1.1 war 1:1 Kopie von BSD Sockets
- Winsock 2.0 hat teilweise geänderte API
- wichtige Unterschiede zu Unix-Sockets:
  - WSAGetLastError statt errno
  - WSA_* Fehlercodes
  - closesocket statt close
  - Windows-Events statt select

Fragen?

?