Erinnerung

Die Veranstaltung findet am
17.5, 24.5 und 31.5
NICHT statt.

Netzwerk-Programmierung

Concurrent Clients

Jan Krueger
jkrueger@cebitec.uni-bielefeld.de
Alexander Sczyrba
asczyrba@cebitec.uni-bielefeld.de

Aufgabe

Schreibe mit Hilfe der `socket` API einen Client, der

- zeilenweise Eingaben an den Server schickt
- und Ausgaben vom Server zeilenweise ausgibt

Wie verhält sich Dein Client bei einem echo-Server (vgl. Material zu Programmieren mit Sockets)?
Wie verhält er sich bei einem whoisServer (z.B. 'whois.denic.de', Port 43)?

Vergleiche dazu die Ausgabe mit z.B. whois -h whois.denic.de google.com.

Wo liegt das Problem?

Problematischer Client

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, port))

byteswritten = 0

while byteswritten < len(data):
     startpos = byteswritten
     endpos = min(byteswritten + 8, len(data))
     byteswritten += s.send(data[startpos:endpos])
     sys.stdout.write("Wrote %d bytes\n" % byteswritten)
     sys.stdout.flush()
     buf = s.recv(8)
     if not len(buf):
          break

s.shutdown(1)
```

Deadlocks

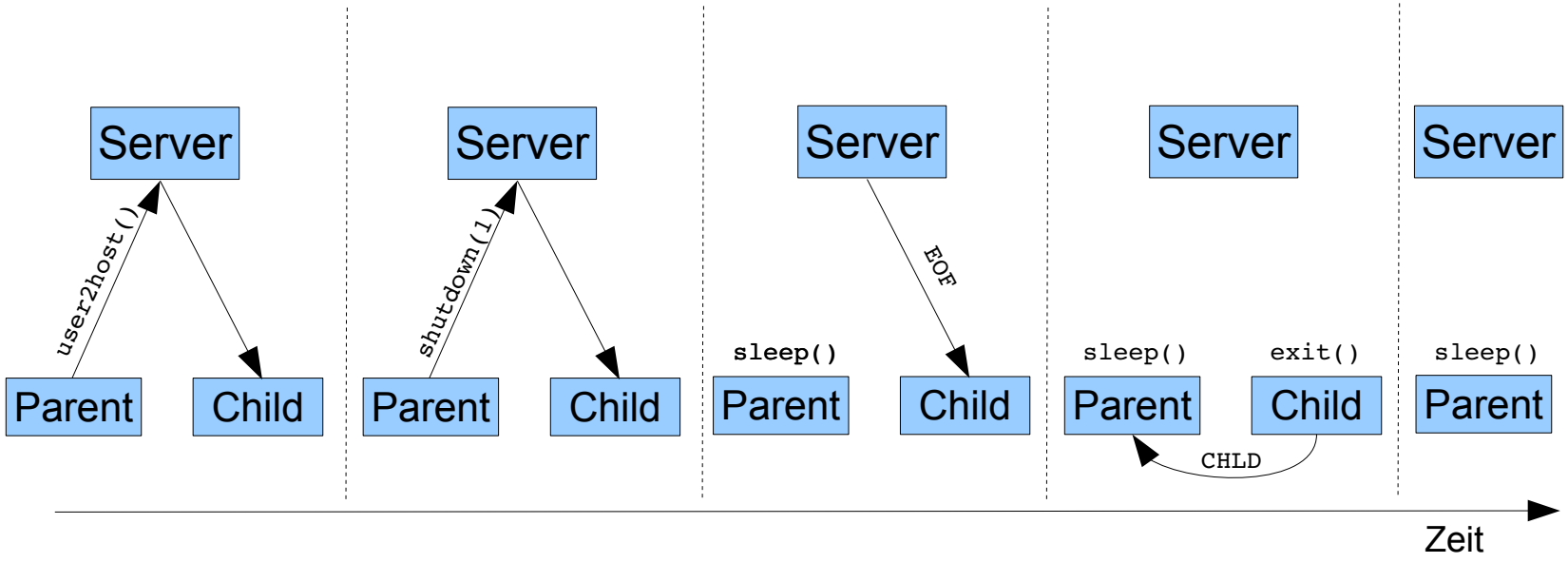
Server und Client warten gleichzeitig auf Eingaben und Erzeugen so einen Deadlock.

Concurrent Clients

Lösung : Entkoppeln der lesenden und schreibenden Prozesse durch fork

- Parent kopiert die Daten vom Client zum Server
- Child liest Daten von Server
- Problem: korrektes Beenden der Prozesse

Verbindungsabbau forked Client



Aufgabe

Schreibe einen Client, der folgendermassen aufgebaut ist:

- Socket erzeugen
- `fork()`
- Parent kopiert von STDIN zum Socket (`user2host()`)
- Child kopiert vom Socket nach STDOUT (`host2user()`)
- Parent reagiert auf Signal `CHLD` und beendet sich
- Parent initiiert `shutdown(1)`, wenn User-Eingabe beendet
- Child beendet sich bei `EOF` vom Server