# Betriebs- und Kommunikationssysteme

WS 14/15

Prof. Dr-Ing. Volker Roth

Freie Universität Berlin

# 10. Übung

| Ausgabe | Abgabe |
|---|---|
| 19.12.14 | 16.01.15 |

Bitte bei der Abgabe Name der Mitglieder einer Gruppe, Nummer der Übung/Teilaufgabe und Datum auf den Lösungsblättern nicht vergessen! Darauf achten, dass die Lösungen beim richtigen Tutor abgegeben werden. Achten Sie bei Programmieraufgaben außerdem darauf, dass diese im Linuxpool kompilierbar sind. Nutzen Sie dazu die Flags `-std=c99`, `-Wall` und `-pedantic`. Es sollten keine Warnungen auftauchen.
**Zu spät abgegebene Lösungen werden nicht berücksichtigt!**

## Aufgabe 1: Begriffe (2 Punkte)

(a) Beschreiben Sie jeden der folgenden Begriffe mit wenigen Sätzen:

- Socket
- Datagram
- Protocol stack
- Bit
- Signal
- Frequenzmodulation
- Framing
- Fehlererkennung

(b) Erläutern Sie den Unterschied zwischen Bit rate und Baud rate.

(c) Grenzen Sie den Begriff World wide web vom Internet ab.

(d) Erläutern Sie den Unterschied zwischen TCP und UDP und nennen Sie jeweils eine Anwendung.

## Aufgabe 2: Programmieren in C (3 Punkte)

In dieser Aufgabe sollen Sie einen einfachen Webserver programmieren.

Der Webserver soll auf Port 8080 hören und dort Verbindungen entgegennehmen. Der Server soll mehrere simultane Verbindungen verarbeiten können. Sie brauchen zum Lösen dieser Aufgabe nicht das HTTP-Protokoll zu verstehen! Wichtig ist nur das Sie aus der ersten Zeile der Anfrage (des Browsers) den Dateinamen parsen, den Rest können sie komplett ignorieren.

Der Server soll immer mit folgendem Header vor den eigentlichen Daten antworten:

```
HTTP/1.0 <STATUS>
Content-Type: <MIME>
Connection: close
Content-Length: <LEN>

<CONTENT>
```

Dabei ist <STATUS> entweder “200 OK” für ausgelieferte Dokumente oder 404 “Not Found“ für nicht gefundene Dokumente.

<MIME> ist der Typ der Rückgabe, also „text/html“ für Webseiten bzw. „image/jpeg“ oder „image/gif“ für Grafiken. Andere Typen braucht der Server nicht zu unterstützen.

<LEN> ist die Länge des zurückgelieferten Contents, also exklusive des Headers.

Nach dem Header senden Sie eine Leerzeile “\n“ und dann den Dateiinhalt bzw. die Errorpage im Fehlerfall. Danach schließen Sie den Socket.

Grobe Vorgehensweise (nicht zwingend):

1. Öffnen Sie einen Server-Socket.
2. Versetzen Sie den Socket in den Listen-Status und merken Sie sich die Rückgabe (den Deskriptor).
3. Machen Sie sich mit dem Konzept von `select()` vertraut.
4. Nutzen Sie `FD_ZERO` um eine leere Socket-Struktur zu erstellen.
5. Nutzen Sie `FD_SET` um ihren Server-Socket an die erste Stelle der Struktur zu schreiben.
6. Programmieren Sie eine Endlosschleife mit folgendem Inhalt:
   (a) Kopieren Sie ihre Socket-Struktur.
   (b) Rufen Sie auf der Kopie `select()` auf.
   (c) Prüfen Sie ob auf dem Server-Socket ein Verbindungswunsch aufgelaufen ist (FD_ISSET), wenn ja nehmen Sie Ihn mittels `accept()` an und merken Sie sich den Filedescriptor!
   (d) Prüfen Sie alle Ihre Filedescriptoren (von den Clients) in einer Schleife darauf, ob sie in Ihrer Kopie der Socket-Struktur enthalten sind (FD_ISSET). Wenn ja lesen Sie mittels `read()` von diesem Descriptor die Anfrage ein.
   (e) Parsen Sie den Dateinamen aus der Anfrage und geben Sie eine entsprechende Antwort (Grafik, Page oder Fehler), inkl. Header zurück.
   (f) Schliessen Sie die Verbindung!

Testen Sie Ihren Webserver mit einer einfachen HTML-Seite die mindestens 4 Bilder enthält. Sie werden sehen, dass Ihr Browser 4 Verbindungen zu ihrem Server gleichzeitig aufbaut um die Bilder zu erhalten.

Wenn Ihr Server läuft, haben Sie einen performanten, robusten (hoffe ich) TCP-Server programmiert, den Sie so auch in eigenen Projekten einsetzen können. **Herzlichen Glückwunsch!**