Kommunikationsnetze – Übung 3

Datei von einem Web-Server laden ("Browser")

Lernziele:
- HTTP Kommando `GET`
- Daten über das Netz schicken: `write()`
- Daten empfangen: `read()`
- HTTP Ergebniscodes
- Daten analysieren: `strstr()`

Wiederholung:
- Datei öffnen
- In eine Datei schreiben
- Datei schließen

HTTP = Hypertext Transfer Protocol

HTTP = Kommandos, um Dateien von einem Web-Server zu einem Browser zu senden (und umgekehrt) → RFC 2068

Beispiele: `GET ... HEAD ... POST ... PUT ...` usw.

Prinzipien:
- HTTP Befehle bestehen aus lesbarem Text (ASCII-Code)
- Befehle können aus mehreren Zeilen bestehen
- Befehle enden mit einer Leerzeile (wichtig!)

Leerzeilen in C erzeugen: Steuerzeichen `\r` und `\n`

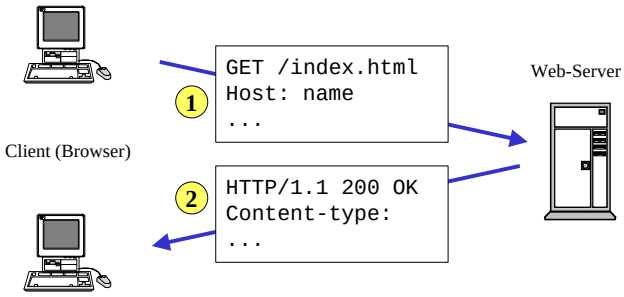
`\r` = return = CR (carriage return)

`\n` = new line = LF (line feed)

→ `\r\n` trennt die Zeilen der HTTP-Kommados

Datenfluss des HTTP-Protokolls

Prinzip: 1) Client sendet eine Anforderung, z.B. `GET dateiname ...`
2) Server antwortet mit Ergebniscode und weiteren Daten



Datenformat: Anfang der Daten (header = Kopf) → immer Text
Übertragene Datei (body = Rumpf) → evtl. binär

Eine Datei anfordern

Befehl als C-Zeichenkette:

```
"GET /hallo.txt HTTP/1.1\r\nHost: mein_PC\r\n\r\n";
```

1 2 3 4 5 6 7

1 = GET-Kommando (eine Datei anfordern)
2 = Name der gewünschten Datei
3 = Version des HTTP-Protokolls
4 = neue Zeile
5 = Parameter-Kennung für den Host-Namen (wird immer benötigt)
6 = Name des Host (frei wählbar)
7 = neue Zeile und Leerzeile (darf nicht fehlen)

Befehl abschicken

Vorraussetzung: eine mit `connect()` erfolgreich geöffnete Verbindung

1) Befehl in einer Zeichenkette ablegen:

```
char befehl[] = "GET ... ";
```

(oder mit `sprintf()` dynamisch erzeugen ...)

2) Befehle abschicken:

```
anzahl = write(socket_nummer, befehl, sizeof(befehl));
```

Parameter: - Nummer des Socket
- Zeiger auf die Befehls-Zeichenkette
- Länge der Befehlszeichenkette

Rückgabewert:
- Anzahl der gesendeten Zeichen

Antwort aus dem Socket lesen

Vorraussetzung: eine ausreichend grosse Variable zur Speicherung

1) Speicherplatz reservieren:

```
char empfangene_zeichen[65000];
```

(Problem: Grösse der Antwort ist unbekannt → dynamisch reservieren)

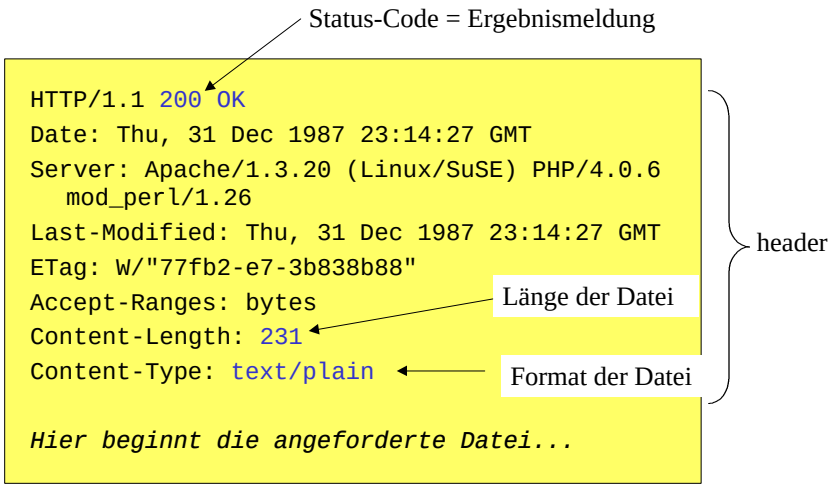
2) Daten einlesen:

```
anzahl = read(socket_nummer, empfangene_zeichen,
              sizeof(empfangene_zeichen));
```

Parameter: - Nummer des Socket
- Zeiger auf den Speicherblock
- Grösse des Speicherblocks (begrenzt den Lesebefehl)

Rückgabewert:
- Anzahl der gelesenen Zeichen

Beispiel für eine Server-Antwort



HTTP Ergebnismeldungen

Der Web-Server sendet einen Statuscode (Ergebnismeldung) zurück

Beispiel:
- `200 OK` = kein Fehler aufgetreten
- `401 Unauthorized` = kein Recht, die Datei zu lesen
- `404 Not Found` = Datei wurde nicht gefunden

allgemein:
- `1xx` = Information wird gesendet
- `2xx` = Befehl erfolgreich bearbeitet
- `3xx` = die gewünschte Datei hat eine andere Adresse
  die neue Adresse wird zurückgemeldet (redirection)
- `4xx` = Fehler (vom Client verursacht)
  z.B.: falscher Dateiname, keine Berechtigung etc.
- `5xx` = Fehler (vom Server verursacht)

Aufgabe 2

Prüfen Sie, ob der Server die gewünschte Datei gesendet hat. Dazu müssen Sie feststellen, ob in der Antwort des Servers die Zeichenkette `200 OK` vorkommt.

Hinweise:

- nach Text in Text suchen: `strstr()`
- deklariert in: `string.h`

- Beispiel: `strstr(durchsuchter_text, gesuchter_text);`

- Ergebnis: `NULL`, wenn der Text nicht gefunden wurde
  oder: ein Zeiger auf die Position des gefundenen Textes

Musterlösung Aufgabe 2

```
#include <string.h>
...
 else
   {
    printf("\n Verbindung erfolgt, sende HTTP-Befehl:\n\n%s",befehl);
    anzahl = write(socket_nummer, befehl, sizeof(befehl));
    printf(" es wurden %d Zeichen gesendet",anzahl);
    anzahl = read(socket_nummer, empfangene_zeichen, sizeof...
    empfangene_zeichen[anzahl]= '\0';
    printf("\n es wurden %d Zeichen empfangen: \n\n%s",anzahl,...

    if (strstr(empfangene_zeichen,"200 OK") == NULL)
      {
       printf("\n Der Server hat die Datei nicht gesendet");
      }
    else
      {
       printf("\n Der Server hat die Datei gesendet");
      }
   }
```

Musterlösung Aufgabe 3

```
FILE *datei;
...
else
 {
  printf("\n Der Server hat die Datei gesendet");

  datei = fopen("knu3a3.txt","wt");
  if (datei == NULL)
    {
     printf("\n Die Datei knu3a3.txt konnte nicht geöffnet werden");
    }
  else
    {
     printf("\n Empfangene Daten werden in knu3a3.txt gespeichert");
     fprintf(datei,"%s",empfangene_zeichen);
     fclose(datei);
    }
 }
```


Literaturhinweis

Gourley, David, Totty, Brian
"HTTP The Definitive Guide"
O'Reilly, 2002

FH-Bibliothek: 01 TWP 864