Verteilte Systeme im Sommersemester 2022

Patrick Felschen, Matr. Nr. 932056
Julian Voß, Matr. Nr. 934505

Osnabrück, 04.04.2022

Aufgabenblatt 2

Aufgabe 1 – Implementierung

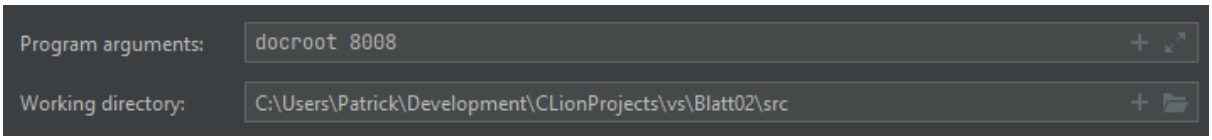
1. Mehrstufiger Server

```
// fuer jede Verbindung einen Kindprozess erzeugen
if ((pid = fork()) < 0) {
    err_abort((char *) "Fehler beim Erzeugen eines Kindprozesses!");
} else if (pid == 0) {
    close(sockfd);
    read_socket_request(newsockfd);
    _exit(0);
}
```

2. Port – Verzeichnis

```
// Verzeichnis und Port aus Übergabeparemetern auslesen
// Root Ordner
doc_root = argv[1];
// Port
char *srv_port = argv[2];
```

3. Port 8080 oder 8008



Durch zwei Program Arguments wird der http-Server gestartet. Damit das Wurzelverzeichnis „docroot“, welches alle benötigten Dateien und Verzeichnisse enthält, gefunden wird, wird zusätzlich ein Working directory festgelegt. Damit der Server vom Client aufgerufen werden kann, wird zudem der Port in diesem Fall auf 8008 festgelegt.

4. GET-Request

```c
/* Wertet ein GET-Request aus */
void get_request(int sockfd, char *uri);
```

Die *get_request* Funktion wertet eine Übergebene URI aus, indem zuerst der der Dateistatus über *lstat* abgerufen wird. Ist, die Datei verfügbar, wird der Dateityp abgefragt und ein entsprechender Header, mit Status 200 und passendem Content Type, an den Socket geschickt. Anschließend wird, falls es sich um ein Verzeichnis handelt, eine Index Seite aufgebaut. Andernfalls wird der Inhalt der Datei byteweise, in 512 Byte Blöcken, ausgelesen und an den Socket geschickt. Bei Fehlern wird im Header der Code 404 verschickt.

```c
/* Schreibt alle Header-Informationen auf den Socket */
void write_header(int sockfd, char *status_code, char *status_message, char *content_type);
```

Die *write_header* Methode erstellt anhand eines Status-Codes, einer Status-Message und einem Content-Typen einen Header, welcher zum richtigen Anzeigen des Inhalts im Browser benötigt wird. Die Informationen werden byteweise mit einer *write_bytes* Funktion auf den Socket geschrieben.

a. Dateien ausgeben (inkl. JPG)

```c
/* Schreibt den Dateiinhalt byteweise auf den Socket */
void write_file_bytewise(int sockfd, char *url)
```

Über die *read* Funktion wird der Inhalt einer Datei in 512 Byte Blöcken in einen Buffer gelesen. Dieser Buffer wird auf den Socket geschrieben. Der Vorgang wiederholt sich, solang der Rückgabewert der *read* Funktion ungleich 0 ist (0 = End of File).

b. Verzeichnis ausgeben

```c
/* Schreibt den Ordnerinhalt des root-Verzeichnisses auf den Socket */
void write_index(int sockfd, char *url) {
```

Der obere Teil der Seite bestehet aus einem Seitentitel und der Überschrift, dieser Inhalt wird als HTML-Code direkt aus den Socket geschrieben. Durch die *readdir* Funktion kann der Inhalt aus dem Übergebenen Verzeichnis ausgelesen werden. Für die Darstellung auf der Webseite wird ein <a>-Tag erstellt, welcher einen Link zu der Datei enthält.

Aufgabe 2 – Durchgeführte Tests

Zum Testen wurde durch das Beispiel Verzeichnis navigiert. Es wurde geprüft, ob alle Unterverzeichnisse erreichbar sind.

Des Weiteren wurden die einzelnen Dateiformate geöffnet und auf Richtigkeit geprüft.

Index

-
..
C_Morgenstern.jpg
der_werwolf.html
der_werwolf.txt
folder

Index

-
..
index.html
post_form.html
subfolder

Index

-
..
subsubfolder

Abschließend wurde ein GET-Request mit einer nicht vorhandenen Datei gemacht und getestet, ob die 404-ERROR Seite angezeigt wird.

Unterschiedliche GET-Requests der Browser

Internet Explorer

```
GET / HTTP/1.1
Accept: text/html, application/xhtml+xml, image/jxr, */*
Accept-Language: de-DE,de;q=0.8,en-US;q=0.5,en;q=0.2
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; rv:11.0) like Gecko
Accept-Encoding: gzip, deflate
Host: 127.0.0.1:8080
Connection: Keep-Alive
```

Google Chrome

```
GET / HTTP/1.1
Host: 127.0.0.1:8080
Connection: keep-alive
Cache-Control: max-age=0
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="99", "Google Chrome";v="99"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/99.0.4844.84 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*
;q=0.8,application/signed-exchange;v=b Xy•
```