

DHBW Ravensburg

Praktikum Verteilte Systeme

Kurs TIT11, 6. Semester

Programmentwurf Verteilte Systeme

Aufgabenstellung

Implementieren Sie einen rudimentären HTTP-Server `tinyweb`, vom dem ein Webbrowser, z.B. Firefox, Dateien über das HTTP 1.1 Protokoll laden kann.

Berücksichtigen Sie bei der Implementierung des HTTP-Servers die folgenden funktionalen Anforderungen:

1. Der Server stellt auf einem konfigurierbaren Port einen verbindungsorientierten Dienst unter Verwendung des HTTP Protokolls¹ zur Verfügung. Der Server-Port muss als Programmparameter angegeben werden (siehe Tabelle 1).
2. Der Server implementiert ein mehrstufiges Konzept zur Behandlung von Fehlern:
 - Fehler innerhalb des Masterprozesses, die zum Beispiel bei der Allokierung von Ressourcen oder aufgrund von ungültigen Werten von Programmparametern beim Start des Servers auftreten, werden mit einer entsprechenden Fehlermeldung auf den Ausgabekanal `stderr` ausgegeben und der Server mit einem Fehlerstatus (Rückgabewert von `exit()` ungleich 0) beendet.
 - Fehler während der Abarbeitung einer Anfrage, die also innerhalb eines Kindprozesses (zum Beispiel beim Aufruf von Bibliotheksfunktionen) auftreten, werden als Status 500 in der Serverantwort an den Client zurückgemeldet. Die Verbindung wird danach beendet und ein Fehlerstatus an den Vaterprozess übergeben, wenn der Kindprozess sich beendet hat. Der Vaterprozess nimmt den Fehlerstatus des Kindprozesses zwar entgegen, im Rahmen des Programmentwurfs ist es aber nicht erforderlich, dass der Vaterprozess darauf reagiert.
 - Allgemeine Fehler in der Struktur oder dem Inhalt einer Anfragenachricht des Clients werden durch den Server mit Status 400 beantwortet.
 - Spezifische Fehlerbedingungen, die die verwendete HTTP Methode oder die angeforderte Ressource betreffen) werden vom Server mit einem entsprechend Status (siehe Tabelle 3) beantwortet und in den folgenden Anforderungen näher beschrieben.
3. Der Server muss mindestens die HTTP Methoden `GET` und `HEAD` unterstützen. Wird von einem Client in einer Anfrage eine andere HTTP Methode verwendet, so antwortet der Server mit Status 501.

¹ siehe RFC 2616 als Beschreibung des HTTP 1.1 Protokolls.

4. Der Server protokolliert sämtliche HTTP Verbindungen in eine Datei. Anzugeben sind folgende Informationen:

- IP-Adresse und Port des Clients (siehe `accept(2)`),
- die HTTP Methode,
- die Zeit der Anfrage,
- der Name der übertragenen Ressource,
- der Status (z.B. 200), sowie
- die Anzahl der übertragenen Bytes.

Der Name der Protokolldatei kann als Programmparameter angegeben werden (siehe Tabelle 1). Das Ausgabeformat soll folgender Formatierung entsprechen, wobei eine Zeile eine Verbindung repräsentiert:

```
192.168.0.99 - - [13/Jun/2014:07:24:30 +0200] "GET /index.html HTTP/1.1" 206 1004
192.168.0.99 - - [13/Jun/2014:07:25:25 +0200] "GET /index.html HTTP/1.1" 200 7768
192.168.0.99 - - [13/Jun/2014:07:31:58 +0200] "GET /source HTTP/1.1" 301 215
192.168.0.99 - - [13/Jun/2014:07:31:58 +0200] "GET /source/ HTTP/1.1" 200 7768
```

5. Der Server sendet in der HTTP-Antwortnachricht die Parameter entsprechend Tabelle 2 sowie den entsprechenden Status der Anfrage (z.B. 200, 404, usw.), siehe Tabelle 3, an den Client zurück. Bei der Methode `GET` sendet der Server zusätzlich die angeforderte Ressource an den Client.
6. Der Server bestimmt anhand der Dateiendung den MIME-Typ, z.B. HTML, GIF, JPEG, der als **Content-Type** an den Client zurück gesendet wird. Ein Programmbeispiel für die Bestimmung des MIME-Typs und der entsprechenden Datenstruktur wird Ihnen auf der Kurshomepage zur Verfügung gestellt.
7. Der Server muss mehrere HTTP Transaktionen nebenläufig bearbeiten können. Verwenden Sie für die Implementierung dieser Funktionalität Unix Prozesse (siehe `fork(2)`). Der Masterprozess nimmt Verbindungen von Clients entgegen, die dann jeweils durch einen eigenen Kindprozess bearbeitet werden.
8. Der Server überprüft mit `stat(2)`, ob die angeforderte Ressource existiert und lesbar ist.² Im Fehlerfall gibt der Server einen entsprechenden Status an den Client zurück.
9. Der Server überprüft, ob die vom Client angeforderte URI mit der Zeichenkette `'/cgi-bin'` beginnt. Falls ja, wird nicht der Inhalt der Ressource an den Client übertragen, sondern die Ressource in einem Enkelkindprozess ausgeführt und dessen Standardausgabe an den Client weitergeleitet. Der Server überprüft mit `stat(2)`, ob die angeforderte Ressource existiert und ausführbar ist.³ Im Fehlerfall gibt der Server einen entsprechenden Status an den Client zurück. Hinweis: Verwenden Sie `execle(3)`, um die Ressource auszuführen.

²Verwenden Sie hierfür `S_ISREG` und `S_IROTH`.

³Verwenden Sie hierfür `S_ISREG` und `S_IXOTH`.

10. Der Server überprüft, ob die vom Client angeforderte URI auf ein Verzeichnis verweist.⁴ Falls ja, wird die Anfrage mit Status 301 beantwortet und in der Antwort auf eine neue URI verwiesen (**Location**), die aus der ursprünglichen URI mit angehängtem '/' besteht.
11. Enthält die Anfrage den Parameter **Range**, so sendet der Server nur den angeforderten Bereich einer Ressource zurück. Bei der Auswertung des Parameters können Sie sich auf die folgende Beispiele beschränken:

```
REQUEST HEADER:
GET /index.html HTTP/1.1
Range: bytes=6764-
User-Agent: Wget/1.13.4 (linux-gnu)
Accept: */*
Host: localhost:8080
Connection: Keep-Alive

RESPONSE HEADER:
HTTP/1.1 206 Partial Content
Date: Fri, 13 Jun 2014 05:24:30 GMT
Server: TinyWeb (Build Jun 12 2014)
Accept-Ranges: bytes
Last-Modified: Thu, 12 Jun 2014 16:49:48 GMT
Content-Type: text/html
Content-Length: 1004
Content-Range: bytes 6764-7767/7768
Connection: Close
```

```
REQUEST HEADER:
GET /index.html HTTP/1.1
Range: bytes=7768-
User-Agent: Wget/1.13.4 (linux-gnu)
Accept: */*
Host: localhost:8080
Connection: Keep-Alive

RESPONSE HEADER:
HTTP/1.1 416 Requested Range Not Satisfiable
Date: Fri, 13 Jun 2014 05:29:46 GMT
Server: TinyWeb (Build Jun 12 2014)
```

12. Die Programmparameter, die von Ihrem Server zu verwenden sind, werden in Tabelle 1 beschrieben. Die Programmparameter können Sie unter Verwendung von `getopt(3)` auswerten.

Programmierungsumgebung

Verwenden Sie als Programmiersprache C und als Betriebssystem Linux (z.B. Ubuntu 14.10) auf einer x86 Architektur.⁵ Der Server darf neben der Standard C Bibliothek nur die während der Vorlesung vorgestellte Bibliothek `libsockets` einbinden. Eigene Erweiterungen dieser Bibliothek sind zulässig, aber für die Aufgabenstellung eigentlich nicht erforderlich.

⁴Verwenden Sie hierfür `S_ISREG`, `S_IROTH` und `S_IXOTH`.

⁵ISO/IEC 9899:1999 mit sog. GNU-Extensions. Die hierfür notwendige `gcc`-Option ist in dem Ihnen zur Verfügung gestellten `Makefile` bereits enthalten (`-std=gnu99`).

Parameter	Bedeutung
-d dir	Definiert die Wurzel des Teilbaums, innerhalb dessen das Programm eine Ressource entsprechend der angefragten URI lädt.
-l file	Protokolliert sämtliche Transaktionen in die Datei file . Wird als Dateiname “_” verwendet oder die Option nicht angegeben, so ist für die Ausgabe die Standardausgabe (stdout) zu verwenden.
-p port	Definiert den lokalen Port, auf dem der Server ankommende HTTP Verbindungen entgegen nimmt.
-h	Gibt eine Übersicht der gültigen Programmparameter sowie Gruppe, Namen und Kurs der Programmautoren. Diese Meldung soll zusammen mit einer Fehlermeldung auch ausgegeben werden, wenn keine oder falsche Programmparameter angegeben wurden.

Tabelle 1: Übersicht der Programmparameter

Parameter	Header
Date	Response
Server	Response
Last-Modified	Response
Content-Length	Response
Content-Type	Response
Connection	Response
Accept-Ranges	Response
Location	Response
Range	Request

Tabelle 2: Übersicht der zu implementierenden HTTP Parameter

Status	Bedeutung
200	OK
206	Partial Content
301	Moved Permanently
400	Bad Request
401	Forbidden
404	Not Found
406	Requested Range Not Satisfiable
500	Internal Server Error
501	Not Implemented

Tabelle 3: Übersicht der zu implementierenden HTTP Status-Codes

Bewertung des Programmentwurfs

Die Prüfungsleistung für die Vorlesung Verteilte Systeme wird durch einen Programmentwurf mit Schwerpunkt Unix Socket Programmierung erbracht.

Der Programmentwurf wird von Ihnen gruppenweise erstellt. Die Namen der einzelnen Gruppenmitglieder sind im Quelltext zu vermerken. Ausserdem ist zu vermerken, welches Gruppenmitglied welche Module schwerpunktmässig bearbeitet hat.

Der Programmentwurf wird anhand der folgenden Kriterien bewertet:

1. Funktionalität, Korrektheit, Effizienz und Robustheit (50 Punkte)
2. Modularisierung, Verständlichkeit und Kommentierung (25 Punkte)
3. Testdurchführung und -beschreibung (25 Punkte)

Die *Gesamtpunktzahl* wird anteilig reduziert, falls einzelne Anforderungen nicht implementiert werden. Sollten während des Programmlaufs ein Speicherzugriffsfehler auftreten, wird die Punktzahl für Kriterium 1 um 50% reduziert.

Wenn Sie Programmfragmente aus der Literatur, dem Internet oder von anderen Quellen verwenden, ist die Quelle als Kommentar im Quelltext kenntlich zu machen und zusätzlich in der Datei `README.txt` anzugeben.

Abgabe des Programmentwurfs

Der Programmentwurf ist als gezipptes Archiv im tar-Format (`tar.gz`) bis spätestens

Montag, 14.07.2014, 23.59 Uhr MESZ

per Email an `kontakt@ralfreutemann.name` zu schicken. Abzugeben sind:

- der Quelltext sämtlicher Module im Unterverzeichnis `src`,
- das ausführbare Programm `tinyweb`,
- eine Verzeichnisstruktur `web`, die Beispieldateien für den Server enthält (HTML-Dateien, CGI-Skripte),
- eine Datei `README.txt` mit Hinweisen zum Programm und den Autoren,
- eine Datei mit der Beschreibung und dem Protokoll (bestehend z.B. aus Request- und Response-Header für jede Anfrage) der durchgeführten Tests.

Der Name der abzugebenden Archivdatei ist `pe_tit11_gruppe_x.tar.gz`, wobei `x` der Ihrer Gruppe zugewiesenen Nummer entspricht. Die Beschreibung der Testfälle und der Testergebnisse ist ausdrücklich Bestandteil der Aufgabenstellung und wird somit bewertet.

Programmierrichtlinien

Es wird von Ihnen erwartet, dass Sie die bereits aus anderen Vorlesungen bekannten, allgemein üblichen Prinzipien der Softwareentwicklung hinsichtlich der Modularisierung und Kommentierung von Programmen beachten. Ferner sind die folgenden Programmierrichtlinien zu erfüllen:

1. Die Programmodule müssen unter der Verwendung *aller* folgenden Optionen ohne Fehlermeldungen bzw. Warnungen mit dem `gcc` C Compiler übersetzt werden können:
 - `-Wall`
 - `-Werror`
 - `-Wunreachable-code`
 - `-Wswitch-default`
 - `-std=gnu99`
 - `-pedantic`
 - `-O2`
2. Alle Fehlermeldungen Ihrer Programme sind auf die Standardfehlerausgabe (`stderr`) zu schreiben.
3. Liefert eine Funktion einen für den Programmablauf relevanten Fehlercode zurück, so ist dieser im Programm zu überprüfen und geeignet zu reagieren.
4. Die Verwendung der `goto`-Anweisung ist *nicht* erlaubt. Ausserdem sollen `break` und `continue` in Schleifenkonstrukten nur ausnahmsweise verwendet und dann aber entsprechend kommentiert werden.
5. Vermeiden Sie die Verwendung von globalen Variablen soweit wie möglich.
6. Die Anweisungsblöcke nach `if`, `else` usw. müssen stets in geschweiften Klammern stehen, auch dann, wenn die Blöcke nur eine einzelne Anweisung enthalten.
7. Am Kopf eines Moduls (`*.c`, `*.h`) sind der Modulname, die Namen der Autorinnen/Autoren, sowie der Zweck des Moduls anzuführen.
8. Vor jeder Funktion muss ein Kommentarblock stehen, der die folgenden Punkte enthält:
 - Name und Zweck der Funktion,
 - Beschreibung der Parameter (Input, Output),
 - Beschreibung der Verwendung globaler Variablen,
 - Beschreibung des Rückgabewertes.