Aufgabe 2: sister (12.0 Punkte)

Implementieren Sie einen einfädigen Webserver `sister` (**Si**mple **S**ingle-**T**hreaded Serv**er**), der HTTP-Anfragen auf einem wählbaren Port *p* entgegennimmt und Dateien innerhalb eines festen Verzeichnisbaums *dir* ausliefert. Das Programm wird wie folgt aufgerufen:

```
./sister -wwwpath <dir> [-port <p>]
```

Wird auf der Befehlszeile kein Port angegeben, soll der Server Verbindungen auf Port 2011 akzeptieren.
Das Programm ist modular aufgebaut und in mehrere Komponenten untergliedert, die separat zu implementieren sind. In einer späteren Aufgabe im Verlauf dieses Semesters werden Sie einzelne Module austauschen beziehungsweise um zusätzliche Funktionalität erweitern.

a) Hauptprogramm: `sister.c`

Im Hauptprogramm muss zunächst die Initialisierungs-Funktion des Verbindungs-Moduls aufgerufen werden. Anschließend wird ein TCP-Socket erzeugt, der auf dem angegebenen Port auf Verbindungsanfragen wartet (**socket(2)**, **bind(2)**, **listen(2)**). Der Socket soll sowohl IPv4- als auch IPv6-Verbindungen unterstützen.
Für jede eingehende Verbindung (**accept(2)**) wird die Funktion `handleConnection()` aufgerufen und anschließend die nächste Verbindung angenommen.

b) Verbindungs-Modul: `connection-fork.c`

Damit das Hauptprogramm sich ohne Verzögerung um die Annahme weiterer Anfragen kümmern kann, soll ein neuer Kindprozess erzeugt werden (**fork(2)**), der für das Erledigen der eigentlichen Arbeit die Funktion `handleRequest()` aufruft, die Verbindung trennt und sich anschließend beendet.
Um zu verhindern, dass beendete Kindprozesse in den Zombie-Zustand übergehen, setzen Sie die Behandlung des `SIGCHLD`-Signals auf das Standardverhalten (`SIG_DFL`) und geben dabei das Flag `SA_NOCLDWAIT` mit an (**sigaction(2)**).
Bei seiner Initialisierung soll sich das Verbindungs-Modul darum kümmern, die Initialisierungsfunktion des Anfrage-Moduls aufzurufen.

c) Anfrage-Modul: `request-http.c`

Dieses Modul kümmert sich um die Kommunikation mit dem Client. Zunächst wird die HTTP-Anfragezeile vom Socket gelesen, die wie im folgenden Beispiel aussieht:

```
GET /doc/index.html
```

Der Pfadname enthält hierbei keine Leerzeichen und verweist auf die gewünschte Datei. Weitere Wörter, die dem Pfadnamen eventuell folgen, **müssen** vom Server **ignoriert** werden. Die Anfragezeile wird entweder mit `\r\n` oder mit `\n` terminiert. Der Dateiname wird relativ zum auf der Befehlszeile angegebenen Pfad interpretiert und der Inhalt der entsprechenden Datei an den Client ausgeliefert.
Beachten Sie, dass der Client **auf gar keinen Fall** Zugriff auf Dateien erhalten darf, die sich jenseits des WWW-Verzeichnisses befinden (→ *Directory-Traversal-Attacke*)! Verwenden Sie die Funktion `checkPath()` aus dem `i4httools`-Modul, um den Pfadnamen jeder Anfrage diesbezüglich zu prüfen. Im Falle einer ungültigen Anfrage oder eines anderweitig aufgetretenen Fehlers soll der Client eine entsprechende Fehlerseite erhalten, die Sie mit Hilfe der entsprechenden Hilfsfunktionen aus dem `i4httools`-Modul erzeugen können.

Hinweise zur Aufgabe:

- Erforderliche Dateien: `sister.c, connection-fork.c, request-http.c`
- Eine ausführliche Beschreibung des HTTP-Protokolls in der Version 0.9 finden Sie unter `http://www.w3.org/Protocols/HTTP/AsImplemented.html`
- Im Verzeichnis `/proj/i4sp2/pub/aufgabe2` finden Sie Schnittstellenvorgaben für sämtliche Module – sowohl für die von Ihnen zu implementierenden als auch für die Hilfsmodule, die Sie zum Lösen der Aufgabe benutzen können. Die Schnittstellen sind verbindlich einzuhalten und die Headerdateien dürfen nicht verändert werden. Auf der Übungswebseite finden Sie außerdem eine Doxygen-Dokumentation der APIs.
- Im Vorgabeverzeichnis befinden sich ebenfalls ein Makefile sowie vorkompilierte `.o`-Dateien für die von Ihnen zu implementierenden Module, so dass Sie alle Teilaufgaben getrennt voneinander bearbeiten können.
- Mit Hilfe des Skriptes `/proj/i4sp2/bin/copy-public-files-for` können Sie sich alle nötigen Dateien in Ihr Projektverzeichnis kopieren lassen.
- Benutzen Sie die Funktionen aus dem `getargs`-Modul zum Parsen und Auswerten der Befehlszeilenargumente.
- Testen Sie Ihren Webserver z. B. mit dem WWW-Pfad `/usr/share/doc/stl-manual/html` und rufen Sie dann in einem Webbrowser (z. B. Firefox) die folgende URL auf, woraufhin eine C++-Dokumentation erscheinen sollte:
  `http://<RECHNERNAME>:<PORT>/index.html`
- Die HTTP-0.9-Antworten des Servers scheinen nur von Google Chrome richtig verarbeitet zu werden. Bei anderen Browsern kann es vorkommen, dass lediglich der HTML-Quellcode angezeigt wird und nicht die gerenderte Seite.

Hinweise zur Abgabe:

Bearbeitung: Zweiergruppen
Bearbeitungszeit: 9 Werktage
Abgabezeit: 17:30 Uhr