Aufgabe 5: mother (12.0 Punkte)

Implementieren Sie ausgehend von Ihrer `sister`-Implementierung einen mehrfädigen Webserver `mother` (**Mo**dular **Th**readed Serv**er**) mit erweiterter Funktionalität, der HTTP-Anfragen auf einem wählbaren Port *p* entgegennimmt und Dateien innerhalb eines festen Verzeichnisbaums *dir* ausliefert. Das Programm wird wie folgt aufgerufen:

```
./mother -wwwpath <dir> [-port <p>] [-bufsize <b>] [-threads <t>]
```

Das Programm ist modular aufgebaut und in mehrere Komponenten untergliedert, die separat zu implementieren sind. Sie können Ihr (fehlerbereinigtes) Hauptmodul `sister.c` unverändert als `mother.c` übernehmen oder alternativ das Programm gegen das `mother.o`-Modul in der zur Verfügung gestellten Bibliothek `libmother.a` binden.

a) Verbindungs-Modul: `connection-mt.c`

Ersetzen Sie das Verbindungs-Modul durch ein neues, in dem mehrere Arbeiter-Threads (siehe *palim*) die Anfragebearbeitung übernehmen und der Hauptthread nur noch für die Verbindungsannahme zuständig ist. Bei der Initialisierung des Moduls werden *t* Arbeiter-Threads (sofern nicht in der Befehlszeile angegeben: 4) erzeugt, die dann für die Laufdauer des Programms bestehen bleiben. Verwenden Sie zum Austausch gemeinsamer Daten zwischen den Arbeiter-Threads und dem Hauptthread einen Ringpuffer (siehe *jbuffer*) mit *b* Einträgen (Standard: 8). Nach Annahme einer Verbindung trägt der Hauptthread den zugehörigen Socket-Dateideskriptor in den Ringpuffer ein. Die Arbeiter-Threads entnehmen dann in einer Endlosschleife jeweils einen Dateideskriptor aus dem Ringpuffer und führen die Bearbeitung der zugehörigen Verbindung durch.
Falls eine Socket-Verbindung getrennt wird, während ein Datenaustausch im Gange ist, stellt das Betriebssystem dem sendenden Prozess ein `SIGPIPE`-Signal zu. Dies würde standardmäßig dazu führen, dass der Webserver sich unerwartet beendet. Sorgen Sie deshalb bei der Initialisierung des Moduls dafür, dass `SIGPIPE` ignoriert wird!

b) Antworten im (vereinfachten) HTTP-1.0-Format: `request-httpx.c`

Viele aktuelle Webbrowser sind mit Version 0.9 des HTTP-Protokolls nicht mehr kompatibel und erwarten, dass alle Anfragen im HTTP-1.0-Format beantwortet werden. Hierfür muss der Server vor jeder Antwortnachricht eine Kopfzeile senden, die den Bearbeitungsstatus der Anfrage beinhaltet (z. B. *200 OK* oder *404 Not Found*).
Benennen Sie Ihr altes Anfrage-Modul in `request-httpx.c` um und passen Sie es so an, dass vor dem Ausliefern der Antwortdaten eine entsprechende Kopfzeile an den Client gesendet wird. Sie können hierfür die Hilfsfunktion `printStatusLine()` aus dem `i4httools`-Modul benutzen. Alle Hilfsfunktionen zum Generieren von Fehlerseiten (z. B. `notFound()`) geben bereits von sich aus eine Statuszeile aus – hier müssen Sie nichts tun.

c) Automatische Anzeige von Verzeichnissen: `request-httpx.c`

Erweitern Sie das Anfrage-Modul so, dass nicht nur Dateien ausgeliefert, sondern auch Verzeichnisse aufgelistet werden. Ist übergebene URL ein Verzeichnis, so überprüfen Sie zunächst, ob die URL mit einem Schrägstrich (`/`) endet. Sollte dies nicht der Fall sein, würde der Webbrowser kaputte Hyperlinks anzeigen – weisen Sie den Client dann mit Hilfe einer *Moved-Permanently*-Antwort freundlich auf die korrekte URL hin und trennen Sie die Verbindung.
Falls sich in dem angeforderten Verzeichnis eine Datei namens `index.html` befindet, senden Sie deren Inhalt an den Client. Andernfalls geben Sie die Namen aller Verzeichniseinträge aus (siehe *crawl*), die nicht mit einem Punkt (`.`) beginnen. Greifen Sie zur Formatierung der Ausgabe auf die Funktionen im Modul `i4httools` zurück.
Optional können Sie sich zwei Extrapunkte erarbeiten, indem Sie die Einträge alphabetisch sortieren (siehe *wsort*).

d) Ausführen von Perl-Skripten: `request-httpx.c`

Ermöglichen Sie nun noch die Ausführung von Perl-Skripten, deren Ausgabe an den Client gesendet wird (siehe *clash* und *josh*). Aus Sicherheitsgründen sollen nur Dateien, die auf `.pl` enden und deren Eigentümer das Ausführen-Recht hat, ausgeführt werden. Jedes Perl-Skript soll in einem eigenen Prozess gestartet werden. Achten Sie darauf, die entstehenden Zombies sofort einzusammeln.

Hinweise zur Aufgabe:

- Erforderliche Dateien: `connection-mt.c, request-httpx.c`
- Im Verzeichnis `/proj/i4sp2/pub/aufgabe5` 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 auch die Semaphor- und Puffermodule. Sie können allerdings auch Ihre Lösungen der letzten Übung verwenden.
- Testen Sie Ihren Webserver z. B. mit dem WWW-Pfad `/proj/i4sp2/pub/aufgabe5/wwwdir`. In diesem Verzeichnis finden Sie einige ausführbare Perl-Skripte sowie die Schnittstellendokumentation der Module.

Hinweise zur Abgabe:

Bearbeitung: Zweiergruppen
Bearbeitungszeit: 6 Werktage
Abgabezeit: 17:30 Uhr