Aufgabe 8: mt-httpd (14 Punkte) Bearbeitung in Zweier-Gruppen 17.01.2011

Implementieren Sie einen mehrfädigen Webserver `mt-httpd` (Multi-Threaded HTTP Daemon) in einer Datei *mt-httpd.c*, der http-Anfragen auf einem wählbaren Port `port` entgegennimmt und Dateien innerhalb eines festen Verzeichnisbaums `www-path` ausliefert. Das Programm wird wie folgt aufgerufen:

```
mt-httpd www-path port #threads #bufslots
```

Parameter drei und vier sind in Teilaufgabe d) beschrieben. Eine Anfrage sieht wie im folgenden Beispiel aus:

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

Der Pfadname enthält hierbei keine Leerzeichen und muss immer auf eine reguläre Datei verweisen. Weitere Wörter, die dem Pfadnamen evtl. folgen, **müssen** vom Server **ignoriert** werden. Die Anfragezeile wird entweder mit "\r\n" oder mit "\n" terminiert. Der Server liefert dann die geforderte Datei an den Client aus und beendet nach Abschluss der Übertragung die Verbindung.

a) Beginnen Sie zunächst mit einem einfädigen Webserver, in dem der Hauptthread die Anfrage sofort bearbeitet, bevor eine neue Verbindung entgegengenommen wird (**accept(2)**). 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

b) Implementieren Sie zählende Semaphore zur Koordinierung von POSIX-Threads (**pthread_mutex_lock(3)**, **pthread_cond_wait(3)**) im Modul **sem** (Datei *sem.c*).

c) Implementieren Sie einen Ringpuffer im Modul **bbuffer** (Datei *bbuffer.c*), der für einen einzelnen Produzententhread und mehrere Konsumententhreads konzipiert ist. Verwenden Sie die Semaphorimplementierung aus Teilaufgabe b) zur Synchronisation der Über- und Unterlaufsituationen, so dass Produzent bzw. Konsumenten beim Einfügen in einen vollen bzw. bei der Entnahme aus einem leeren Puffer blockieren. Verwenden Sie zur Koordinierung der Konsumenten nicht-blockierende Synchronisation mit Hilfe der vorgegebenen CAS-Funktion (*cas.h*, funktioniert nur auf x86-Rechnern), so dass mehrere Konsumenten gleichzeitig den kritischen Abschnitt durchlaufen können (keine Locks!).

d) Erweitern Sie den Server nun so, dass mehrere Arbeiter-Threads die Anfragebearbeitung übernehmen und der Hauptthread nur noch für die Verbindungsannahme zuständig ist. Es werden `#threads` Arbeiter-Threads beim Programmstart 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 mit `#bufslots` Einträgen. Der Hauptthread nimmt nun nur noch jeweils eine Verbindung an und trägt den zugehörigen Socket-Dateideskriptor in den Ringpuffer ein. Die Arbeiterthreads entnehmen dann jeweils einen Dateideskriptor aus dem Ringpuffer und führen die Bearbeitung der zugehörigen Verbindung durch.

Hinweise

- 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/i4sp/pub/aufgabe8* finden Sie Schnittstellenvorgaben für die Module **sem** und **bbuffer** sowie einige nützliche Hilfsroutinen im Modul **i4httools**. Verwenden Sie insbesondere die Funktion **checkpath()**, da es sonst leicht möglich ist, über Ihren Webserver an beliebige Dateien aus z.B. Ihrem Home-Verzeichnis zu gelangen. Sie können außerdem die bereitgestellten Funktionen zum Übertragen einer Fehlerseite bei einer nicht-existierenden Dateien bzw. einer fehlerhaften Anfrage verwenden.
- Im Vorgabeverzeichnis befinden sich auch die Semaphor-, Puffer- und Servermodule. Mit den einzelnen Vorgabemodulen können Sie die Teilaufgaben b), c) und d) getrennt bearbeiten.
- Die Schnittstellenvorgaben der Module **bbuffer** und **sem** sowie der CAS-Funktion finden Sie in einer Online-API-Beschreibung auf der Übungswebseite. Die Schnittstellen sind verbindlich einzuhalten. Die Headerdateien dieser beiden Module dürfen nicht verändert werden. Die Verwendung des Moduls **i4httools** ist optional und es kann auf eigene Bedürfnisse angepasst werden.
- Der Server soll sowohl IPv4- als auch IPv6-Verbindungen bedienen.
- Insbesondere die nicht-blockierende Synchronisation erfordert eine spezielle Denkweise. Es ist ausdrücklich erwünscht, dass Sie Ihre Lösungsansätze auch vor dem Abgabetermin im Forum zur Diskussion stellen.

Abgabe: bis spätestens Freitag, 28.01.2011, 14:00 Uhr