Die Aufgaben W7 bis W9 sind Wahlaufgaben. Entscheiden Sie sich für **eine** dieser Aufgaben. Beschreiben Sie in der Log-Datei (oder einer zusätzlichen Textdatei) zunächst Ihren Lösungsansatz und die Wahl neuer Typen und Datenstrukturen.

Achten Sie (unabhängig von der Wahl der Aufgabe) auch auf Synchronisierung: Überlegen Sie sich jeweils, ob durch parallel laufende Threads auf gemeinsam genutzte Datenstrukturen zugegriffen wird und ob diese durch einen Mutex geschützt werden müssen.

W7. Offline-Proxy

Erweitern Sie den Proxy-Server um die Möglichkeit, offline Webinhalte zu betrachten – das kann nur für URLs funktionieren, die vorab in den RAM-Cache eingelesen wurden.

- Wenn der Proxy auf dem Status-Port (8081) die Anfrage `/online` bzw. `/offline` erhält, schaltet er in den entsprechenden internen Status.
- Im Zustand „online“ arbeitet der Proxy wie gewohnt.
- Im Zustand „offline“ passiert bei einem Request (über Port 8080) folgendes:
  - Falls die Seite im RAM-Cache vorliegt, wird sie an den Client weiter geleitet.
  - Falls die Seite nicht im RAM-Cache vorliegt, wird eine Response generiert, die den Benutzer darüber informiert, dass der Proxy im Offline-Modus ist, die Seite noch nicht hat, aber bei der nächsten Gelegenheit für den Benutzer laden wird. Außerdem wird sie in eine Warteschlange eingetragen.
  - Die Warteschlange fürs Herunterladen von Seiten soll Platz für so viele URLs bieten, wie es Einträge im RAM-Cache gibt. Ist die Warteschlange voll, ersetzt ein neuer Eintrag den ältesten vorhandenen.
  - Beim Wechsel in den Zustand „Online“ arbeitet der Proxy als erstes die Warteschlange ab. Er lädt dazu die in der Warteschlange stehenden Seiten in den RAM-Cache. Verwenden Sie dafür denselben Mechanismus, der auch bei der regulären Nutzung Inhalte in den RAM-Cache schreibt.

Testen Sie die neue Offline-Funktion durch eine geeignete Sequenz von Zugriffen auf Webseiten im Offline- und Online-Modus.

Speichern Sie Ihr Programm als `projekt07.c` und dokumentieren Sie das Laufverhalten in einer Protokolldatei `projekt07.log`.

W8. Ausführliche Statusinformationen und Exit

Über den Aufruf der URL `/status` (am Status-Port 8081) soll der Proxy ausführliche Informationen über den Programmstatus ausgeben. Dazu gehören:

- Speicherplatzverbrauch (summierte Größe aller mit `malloc()` angeforderten Speicherbereiche)
- Liste der letzten 20 Anfragen mit Timestamps (wann die Anfrage kam)
- Liste aller im RAM-Cache vorgehaltenen Einträge (mit URL und Speicherbelegung)

Für das Logging von URLs und Timestamps müssen Sie u. a. `handle_connection()` anpassen, so dass es diese Zusatzinformationen speichert.

Bei der Ausgabe von Timestamps soll ein lesbares Format (Datum, Uhrzeit) und nicht die UNIX-Zeit (Sekunden seit 01.01.1970) verwendet werden. Die URLs sollen anklickbar sein, im HTML-Dokument muss also Code der Form

```
<ul>
<li>28.08.2015 14:15:55: <a href="http://th-nuernberg.de/">http://th-nuernberg.de/</a>
...
</ul>
```

stehen.

Außerdem soll das Programm auch eine Liste aller jemals (auch in früheren Läufen des Programms) erfolgten Requests speichern. Das erfordert das Ablegen der Liste in einer Datei, damit die Informationen auch nach Beenden des Programms verfügbar bleiben. Über den Aufruf der URL `/allstats` (am Status-Port 8081) erhalten Sie diese Liste (jeweils in der Form *Timestamp: URL*). Der Proxy-Server soll auch ein Löschen dieser Liste ermöglichen, wenn die URL `/deletelog` (am Status-Port) aufgerufen wird.

Zusätzlich soll die URL `/exit` erkannt werden; dann beendet sich der Proxy-Server.

Speichern Sie Ihr Programm als `projekt08.c` und dokumentieren Sie das Laufverhalten in einer Protokolldatei `projekt08.log`.

W9. Modifizierender Proxy

Die in dieser Aufgabe zu implementierende Funktion schafft die Grundlagen für Proxys, die z. B. Werbung oder andere unerwünschte Elemente aus Webseiten entfernen oder ersetzen.

Die Modifikationen sollen durch eine Konfigurationsdatei beschrieben werden, deren Pfad (z. B. `/home/fom/proxy.conf`) Sie im Programm fest verdrahten, sie wird beim Programmstart eingelesen und soll aus mehreren mehrzeiligen Einträge einer der folgenden drei Formen bestehen:

```
html              html                  jpeg
remove            replace               convert
<b>               <script>              -resize "50%"
                  <disabled>
```

Nach jedem Eintrag (auch nach dem letzten) folgt eine Leerzeile, an der Ihr Programm das Ende eines Eintrags erkennen kann.

- In der ersten Zeile steht, auf welche Dateitypen die Regel angewendet wird. Werten Sie in der HTTP-Response vom Webserver das Feld `Content-Type:` aus, darin steht z. B. `text/html; charset=...` oder `image/jpeg`.
- Die nächste Zeile gibt mit `remove` an, dass ein Muster aus der Datei entfernt werden soll (nur sinnvoll für HTML). `replace` gibt an, dass ein Muster ersetzt werden soll. `convert` gibt an, dass das externe Programm convert aufgerufen werden soll
- In der dritten und ggf. vierten Zeile folgen Optionen für die Verarbeitung. Bei `remove`- und `replace`-Regeln folgt hier der Suchbegriff. Bei `convert` folgen Optionen, die an das Programm `convert` weiter gegeben werden sollen (z. B. erzeugt `convert eingabe.jpg -resize "50%" ausgabe.jpg` eine neue Version des Bilds, die in Breite und Länge um 50% verkleinert ist).
- In der vierten Zeile steht bei `replace`-Regeln der Ersatztext.

Für die Verarbeitung der `convert`-Regel (und – wenn Sie wollen – auch der anderen beiden Regeln) reichen Sie die Inhalte über zwei Pipes an einen neuen Prozess weiter, in dem Sie `convert` oder andere Tools laufen lassen; von dort geht es mit der zweiten Pipe zurück in den Proxy-Server. Wenn Ihnen das Handling mit Pipes zu komplex ist, können Sie auch mit temporären Dateien arbeiten.

Der Proxy soll der Reihe nach alle Regeln abarbeiten, die in der Konfigurationsdatei gefunden wurden.

Speichern Sie Ihr Programm als `projekt09.c` und dokumentieren Sie das Laufverhalten in einer Protokolldatei `projekt09.log`.