P5. Server beenden, Statistik

a) Ergänzen Sie in die Anfragenbearbeitung eine Abfrage, ob die angeforderte URL gleich `/EXIT` ist. Wenn ja, soll das Programm beendet werden – allerdings erst nachdem alle schon laufenden Anfragen abgearbeitet wurden. Sie verweigern also im Hauptprogramm die Annahme neuer Anfragen (indem Sie nicht mehr `accept()` aufrufen). Der Einsammel-Thread muss darüber informiert werden, dass der Server jetzt im Shutdown-Modus ist. Er wartet dann, bis es keine Threads mehr gibt, und beendet dann mit `exit()` den ganzen Prozess. Erst wenn es keine (noch beschäftigten) Server-Threads mehr gibt, darf das Programm beendet werden.

b) Bei Anforderung der URL `/STATUS` soll eine Informationsseite ausgegeben werden, die eine Liste der bestehenden Verbindungen enthält. Zu jeder Verbindung soll der Zeitpunkt des Verbindungsaufbaus, die IP-Adresse des Clients sowie die angeforderte URL angezeigt werden.

Speichern Sie Ihre Lösung in `projekt05.c` und dokumentieren Sie das Laufverhalten in einer Protokolldatei `projekt05.log`.

P6. `gzip`-Kompression

Erzeugen Sie einen neuen Prozess, in dem `gzip -c` ausgeführt wird. Die Standardeingabe und Standardausgabe dieses Prozesses müssen Sie vor dem `exec()`-Aufruf so umbiegen, dass der aktuelle Thread Daten an den `gzip`-Prozess schickt **und** der gzip-Prozess seine Ausgaben an den Socket (der aktuellen Verbindung) sendet. Dadurch kann der Thread nach dem HTTP-Header die eigentlichen Nutzdaten in komprimierter Form an den Browser schicken.

a) Erweitern Sie die Auswertung des Requests – bisher haben Sie nur die erste Zeile (`GET URL HTTP/1.1`) aus dem Request ausgelesen; jetzt suchen Sie zusätzlich nach einer Zeile, die mit `Accept-Encoding:` beginnt. Dahinter können (bis zum nächsten Zeilenumbruch) mehrere Kodierungen folgen, die durch Komma+Leerzeichen voneinander getrennt sind, z. B.

```
Accept-Encoding: gzip, deflate
```

Wenn sich in der Liste `gzip` befindet, merken Sie sich, dass für diese Anfrage eine `gzip`-Kompression gewünscht ist. Die Alternative ist, dass der `gzip`-Eintrag fehlt oder es gar keine `Accept-Encoding`-Zeile gibt.

b) Beim Transfer der Datei müssen Sie an geeigneter Stelle mit Pipes arbeiten. Sie können wie bisher den Dateiinhalt über eine Schleife in einen Puffer einlesen, schreiben ihn dann aber nicht direkt in den Socket, sondern verwenden eine Pipe, um die Daten an einen Kindprozess weiterzuleiten, der den Befehl `gzip -c` ausführt. Dieser Kindprozess liest von der Standardeingabe und schreibt die komprimierten Daten auf die Standardausgabe. Sie müssen also für den Kindprozess die Standardeingabe und die Standardausgabe so umleiten, dass er im Ergebnis aus der Pipe liest und in den Socket (zum Webbrowser) schreibt. Zur Erinnerung: Diese Umleitungen sollten Sie unmittelbar vor dem `execlp()`-Aufruf einrichten. Bevor Sie loslegen, sollten Sie sich überlegen, welche Umleitungen bereits bestehen und was beim Erzeugen eines Kindprozesses passiert. Die `exec()`-Variante `execlp()` können Sie in der Form `execlp ("gzip", "gzip", "-c", (char*)0);` aufrufen. Für das Kommando brauchen Sie die Header-Datei `unistd.h`.

Ein weiterer Hinweis: Im Vaterprozess muss die Standardausgabe (also der Socket, auf den Sie stdout umgeleitet haben) geschlossen werden, ansonsten hängt der Kindprozess (gzip).

Außerdem ist in der Antwort eine neue Response-Header-Zeile nötig, an der der Webbrowser erkennt, dass die Kompression aktiviert ist: Sie fügen dazu die Zeile

```
Content-Encoding: gzip
```

hinzu. Beides (die Kompression und die zusätzliche Header-Zeile) soll Ihr Programm nur verwenden, wenn Sie in Schritt **a)** in der Anfrage die passende Aufforderung gefunden haben.

Wenn Sie `wget` zum Testen verwenden, können Sie explizit die Kompression anfordern, z. B. mit einem Kommando der folgendem Form:

```
wget -S --header="accept-encoding: gzip" http://localhost:8080/index.html
```

Die Daten landen in der Datei `index.html`, sind aber noch komprimiert, weil `wget` selbst keine gzip-Kompression unterstützt (und darum den Response-Header nicht erkennt). Die Option `-S` bringt `wget` dazu, die gelesenen Response-Header-Zeilen im Terminalfenster auszugeben. Sie können im Beispiel den korrekten Empfang mit

```
zcat index.html
```

überprüfen. Alternativ zum Einsatz von `wget` können Sie auch (wie bisher) für Tests einen normalen Browser oder das Programm `telnet` verwenden.

Speichern Sie Ihre Lösung in `projekt06.c` und dokumentieren Sie das Laufverhalten in einer Protokolldatei `projekt06.log`.