Systemnahe Software (II) im Sommersemester 2009

Abgabetermin: 08. Juli 2009

Hinweis: Sockets

Verwenden Sie in allen folgenden Aufgaben die Socket-Funktionen für Netzwerkkommunikation.

38 HTTP-Client im Selbstbau (11 Punkte)

Schreiben Sie einen kleinen HTTP-Client `hc.c`, der einfache GET-Anfragen in HTTP/1.1 durchführen kann. Falls die Anfrage erfolgreich ist, soll auf der Standardausgabe der Body-Teil der Antwort ausgegeben werden.

Falls die Anfrage nicht erfolgreich ist, soll `hc.c` den Fehler (ggf. mit HTTP-Statuscode) ausgeben und sich mit einem passenden Exitstatus beenden.

Die gewünschte Webseite soll `hc.c` auf der Kommandozeile entgegennehmen. Besonderheiten im URL, wie z.B. Benutzernamen und Kennwörter oder die Angabe eines anderen Ports als Port 80 muss `hc.c` nicht beherrschen.

39 Erweiterung des HTTP-Clients (3 + 1 Punkte)

Erweitern Sie `hc.c` zu `hci.c` ("HC improved") und bauen Sie zwei weitergehende Dinge ein. Erstens sollen Anfragen, die einen Umleitungs-Statuscode liefern, mit der korrekten Adresse wiederholt werden – achten Sie hierbei auf falsch konfigurierte Server, die den Client möglicherweise im Kreis endlos umlenken! Zweitens soll auch die Angabe einer alternativen Portnummer im URL unterstützt werden.

40 Zeitserver mit Sockets (4 Punkte)

Schreiben Sie ein Programm `sockettime.c`, das analog zu Aufgabe 36 (und dem Skript) auf einer von Ihnen festgelegten TCP-Portnummer lauscht. Verwenden Sie diesmal allerdings Sockets. Anders als in Aufgabe 36 soll `sockettime.c` wie folgt arbeiten: Sobald sich ein Client verbindet, soll direkt die aktuelle Zeit übermittelt werden. Danach soll der Server auf den Befehl `quit` warten, bevor die Verbindung beendet wird.

41 Zeitclient (3 Punkte)

Schreiben Sie einen Zeitclient `timec.c`, der sich zu einem Zeitserver aus der vorigen Aufgabe verbindet, die empfangene Zeit ausgibt und dann nach einer zufälligen Zeitspanne zwischen 5 und 20 Sekunden den `quit`-Befehl sendet.

42 Concurrency im Zeitserver (8 Punkte)

In die vorigen Aufgabe müssen weitere Clients warten, bis der aktuell verbundene Client die Verbindung beendet bzw. `quit` eingibt. Um diese unnötige Wartezeit zu vermeiden, muss der Server umgebaut werden.

Erweitern Sie also das Programm aus der vorigen Aufgabe zu einem neuen Programm `concurrenttime.c`, welches als *concurrent server* mehrere Anfragen parallel bearbeiten kann. Beachten Sie auch hier die Richtlinien zum Umgang mit `fork()`.

Viel Erfolg!