Technische Universität Braunschweig
Institut für Betriebssysteme und Rechnerverbund
Kommunikation und Multimedia
Prof. Dr. L. Wolf

Praktikum Kommunikationssysteme im SS06

– Testaufgabe –

Betreuer: Zefir Kurtisi, NN

http://www.ibr.cs.tu-bs.de/courses/ss06/pks/

Kurzbeschreibung

Obwohl für das Praktikum „Kommunikationssysteme" explizit Programmiererfahrung in C gefordert ist, haben wir in der Vergangenheit regelmäßig erlebt, wie Studenten – bewusst oder unbewusst – diese Anforderung unterschätzen und das Praktikum, trotz intensivem Zeitaufwand, im Verlauf abbrechen.

Sicher wäre es möglich, sich Programmierkenntnisse während der Bearbeitung der Aufgaben anzueignen – wenn keine sonstigen Veranstaltungen zu besuchen wären und genug Zeit zum Programmieren bleibt. Da das eher in der vorlesungsfreien Zeit der Fall sein wird, haben wir diese Testaufgabe vorbereitet. Damit können C-Neulinge unter den Interessenten schon vor Beginn der Vorlesungen mit dem Programmieren anfangen, Erfahrene können ihren Wissensstand prüfen und den späteren Zeitaufwand für die Bearbeitung der Aufgaben abschätzen.

Grundlagen der Programmierung in C und die in der Begleitdokumentation zu diesem Praktikum enthaltene Beschreibung der Socket-Programmierung sind zum Lösen dieser Aufgabe ausreichend. Auf der Webseite zum Praktikum sind darüber hinaus Verweise auf viel umfassendere Unterlagen enthalten, darunter auch die Online-Version eines C-Handbuchs, dass die Programmierung von Grund auf beinhaltet.

Aufgabe: Yoda-Übersetzungs-Dienst

Es soll eine einfache, verbindungsorientierte Client-Server-Kommunikation über TCP implementiert werden. Der Server soll auf Nachrichten vom Client warten, diese verarbeiten und zum Client zurücksenden. Der Client liest Benutzereingaben ein, verarbeitet diese, sendet sie zum Server und gibt die erhaltene Antwort am Bildschirm aus.

Nachrichten

Die Nachrichten, die in dieser Anwendung zwischen Client und Server ausgetauscht werden, sind Sätze, die folgenden Einschränkungen genügen:

1. alle Wörter sind in Kleinbuchstaben und enthalten keine Umlaute
2. die Einzelwörter sind genau durch ein Leerzeichen voneinander getrennt
3. jeder Satz enthält genau einen Punkt, der unmittelbar dem letzten Wort folgt

Client

Der Client ist die Schnittstelle vom Benutzer zum Dienst, der vom Server bereitgestellt wird. Als Eingabe wird eine durch die Eingabetaste abgeschlossene Zeile eingelesen. Es folgt eine Prüfung und Anpassung der Nachricht, bei der

- Gross- in Kleinbuchstaben umgewandelt werden
- Bindestriche durch Leerzeichen ersetzt werden
- alle sonstigen Zeichen (außer dem Leerzeichen) gelöscht werden
- mehrfache Leerzeichen hintereinander gelöscht werden
- die Nachricht mit einem Punkt beendet wird

Diese Nachricht wird dann dem Server zur Verarbeitung zugesandt und dessen Antwort ausgegeben. Die Eingabe eines Punktes allein beendet den Client.

Insgesamt soll am Client ausgegeben werden:

1. eingegebener Satz
2. angepasste Nachricht
3. Antwort des Servers

Der Client soll mit oder ohne Parameter aufgerufen werden können. Der erste Parameter soll dabei die IP-Adresse des Servers sein, der zweite der Port. Werden diese Werte nicht angegeben, wird der lokale Rechner als Server angenommen, der auf Port 5000 lauscht.

Server

Der Server lauscht zunächst auf Clients, die sich verbinden wollen. Nach dem Verbindungsaufbau wartet er auf Nachrichten vom Client und antwortet mit einer modifizierten Nachricht. Diese Nachricht besteht aus dem eintreffenden Satz in umgekehrter Reihenfolge, abgeschlossen mit einem Punkt.

Der Server soll mit der optionalen Angabe des Ports gestartet werden können, auf dem er lauscht. Wird dieser Wert nicht angegeben, soll Port 5000 verwendet werden.

Test

Bei einer korrekten Implementierung sollte die Ausgabe beim Client beispielsweise wie folgt aussehen:

```
Eingabe des Benutzers: "Das ist super-einfach!"
Gesendete Nachricht:   "das ist super einfach."
Empfangene Nachricht:  "einfach super ist das."
```

Umfang

Der reine Programmieraufwand – also ohne die evtl. nötige Einarbeitung in C und Socket-Programmierung – für diese Testaufgabe sollte bis zu vier Stunden betragen. Liegt die tatsächlich benötigte Zeit deutlich über diesen Wert, wird empfohlen, sich die vorlesungsfreie Zeit mit ähnlichen Programmieraufgaben oder Erweiterungen zu verkürzen, denn:

Programmieren lernt man nur durch Programmieren

Probleme & Musterlösung

Treten unerwartete Probleme auf, die auch mit der verfügbaren Dokumentation eine Lösung verhindern, ist es möglich, bei den Betreuern Einsicht in die Musterlösung zu bekommen.

Empfohlene Dokumentation

Die auf der Praktikumsseite bereitgestellte Dokumentation [1] liefert einen guten Überblick über die Socket-Programmierung. Die BSD-Socket API wird mittlerweile von allen gängigen Betriebssystemen unterstützt, so dass der Quellcode meist ohne grosse Änderung auf OS/X, Linux, Windows oder Solaris übersetzbar ist. Brian Hall stellt eine umfangreiche Anleitung zur Socket-Programmierung [3] sowohl online als auch als PDF zur Verfügung. Diese englische Dokumentation enthält Beispielprogramme für verschiedene Kommunikationsszenarien.

Für das Erlernen der Programmiersprache C stehen, wenn man den Weg zur Bibliothek scheut, viele freie Online-Kurse zur Verfügung. Die auf der Webseite referenzierte HTML-Version von "C von A bis Z" [4] ist mit über 1000 Seiten sehr umfangreich. Die englische Anleitung von Brian Hall [2] ist hier mit knapp 120 Seiten deutlich kompakter und behandelt alle wesentlichen Themen.

Literatur

[1] Webseite zum praktikum kommunikationssysteme im ss06. `http://www.ibr.cs.tu-bs.de/courses/ss06/pks/`.

[2] Brian "Beej" Hall. *Beej's Guide to C Programming.* 2005. `http://beej.us/guide/bgc/`.

[3] Brian "Beej" Hall. *Beej's Guide to Network Programming.* 2005. `http://beej.us/guide/bgnet/`.

[4] Jürgen Wolf. *C von A bis Z – Das umfassende Handbuch für Linux, Unix und Windows.* Galileo Computing, 2006. online unter `http://www.galileocomputing.de/openbook/c_von_a_bis_z/`.

– Und nun: Viel Spass beim Programmieren! –