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

Praktikum Computernetze im WS07/08

– Aufgabenbeschreibung –

Betreuer: Zefir Kurtisi, Habib-ur-Rehman, NN

`http://www.ibr.cs.tu-bs.de/courses/ws0708/pcn/`

– Einstiegsaufgabe **NetCalc** –

Aufgabenstellung

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

Als einfaches Beispiel einer solchen Client-Server-Anwendung soll mit *NetCalc* ein Netzwerk-Taschenrechner implementiert werden, der die vier Grundrechenarten unterstützt und mit Binär-, Dezimal- und Hexadezimalzahlen umgehen kann. Der Server bearbeitet dabei nur Anfragen, die einem definierten Format entsprechen. Als Antwort erhält der Client eine Nachricht mit dem Ergebnis der Berechnung in allen unterstützten Zahlensystemen.

Server

Der Server lauscht zunächst auf Clients, die sich verbinden wollen. Nach dem Verbindungsaufbau wartet er auf Rechenaufgaben vom Client und antwortet mit dem Ergebnis als Nachricht.

Der Server soll davon ausgehen, dass nur gültige Anfragen gesendet werden, die folgenden Regeln genügen:

- gültige Zeichen sind: `'0'-'9', 'A'-'F', 'x', 'b', '+', '-', '*', '/'`
- jede Anfrage besteht aus genau zwei Operanden, die durch einen Operator getrennt sind
- Operanden sind vorzeichenlose ganze Zahlen im Bereich von 0 bis $2^{32} - 1$
- Operanden ohne Kennzeichnung sind Dezimalzahlen
- Hexadezimalzahlen werden mit einem '`0x`' angeführt, Ziffern '`A`'-'`F`' sind Großbuchstaben
- Binärzahlen bestehen nur aus Nullen und Einsen, an denen ein kleines '`b`' angehängt ist

Der Server berechnet die Aufgabe und sendet die Antwort als Klartext zurück. Diese beinhaltet das ganzzahlige Ergebnis der Anfrage in allen drei Zahlensystemen durch Leerzeichen getrennt. Überläufe und Divisionen durch Null sollen erkannt und als Fehlermeldung übermittelt werden.

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.

Client

Der Client ist die Schnittstelle vom Benutzer zum Dienst, der vom Server bereitgestellt wird. Er ist dafür zuständig, die Eingabe zu prüfen und das Ergebnis auszugeben.

Nach erfolgtem Verbindungsaufbau nimmt der Client kontinuierlich Eingaben des Benutzers entgegen. Diese werden als durch die Eingabetaste abgeschlossene Zeilen eingelesen. Die Eingabe soll dahingehend geprüft werden, ob sie eine gültige Aufgabe für den Server ist oder aus nur einem Operanden besteht. Trifft beides nicht zu, ist die Eingabe mit einer Fehlermeldung zu quittieren.

Enthält die Eingabe nur einen Operanden, dann möchte der Benutzer diesen Wert in ein anderes Zahlensystem konvertiert haben. Um eine für den Server gültige Anfrage zu erzeugen, soll der Client in diesem Fall die Eingabe um einen Operator und einen Operanden derart erweitern, dass der Ausdruck nach der Berechnung gleich dem eingegebenen Operanden ist, z.B. “`*1`”.

Diese Nachricht wird dann dem Server zur Verarbeitung zugesandt und dessen Antwort ausgegeben. Die Eingabe einer leeren Zeile beendet den Client.

Insgesamt soll am Client ausgegeben werden:

1. eingegebene Aufgabe
2. an den Servern gesandte Nachricht
3. empfangenes Ergebnis

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 (“localhost”) als Server angenommen, der auf Port 5000 lauscht.

Test

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

```
Eingabe des Benutzers: "0xA71CE-10100110011011000011b"
Gesendete Nachricht:   "0xA71CE-10100110011011000011b"
Empfangene Nachricht:  "2827 0xB0B 101100001011b"
Eingabe des Benutzers: "0xDEAD"
Gesendete Nachricht:   "0xDEAD+0"
Empfangene Nachricht:  "57005 0xDEAD 1101111010101101b"
```

Empfohlene Dokumentation

Die auf der Praktikumsseite bereitgestellte Dokumentation 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 [2] 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” [3] ist mit über 1000 Seiten sehr umfangreich. Die englische Anleitung von Brian Hall [1] ist hier mit knapp 120 Seiten deutlich kompakter und behandelt alle wesentlichen Themen.

Literatur

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

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

[3] 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/`.