Socket-Programmierung I

Übungen zu Netzwerktechnologie
Bachelor-Studiengang Software Engineering
Vollzeitform
im Wintersemester 2019

Gerhard Jahn Stephan Leitner

25. November 2019

1 Lehrziele

Studierende sollen durch diese Übung

- praktische Erfahrung im Bereich der Socketprogrammierung sammeln,
- ihre C-Kenntnisse auffrischen bzw. verbessern.

2 Warm Up

Aller Anfang ist leicht: Spielen Sie ein wenig Hangman! Verbinden Sie sich von einem System Ihrer Wahl – jedoch innerhalb unseres Netzes – mittels telnet zu einem der beiden Server mit den Ports 4202 oder 4203 auf 10.20.30.20[1]. In Windows können Sie dazu Putty mit dem „Connection Type“ *raw* verwenden. Stellen Sie bei Putty auch die Option *Close window on exit* auf *Only on clean exit*. Überlegen Sie auch, wie das Spiel aufgebaut sein könnte.

3 Aufgaben

Es ist für eine vorhandene Programmlogik ein verbindungsorientierter Socket-Server in C zu implementieren. Laden Sie als Ausgangsbasis Datei `WordCheck.c` herunter, sie ist unter dieser Angabe platziert. Aber Achtung: Darin findet sich die

[1] z. B. `telnet 10.20.30.20 4202`

Wortliste, Sie sollten den Inhalt dieser Datei erst einsehen, nachdem Sie zuvor hinreichend lang Hangman gespielt haben.

Aufgabe *1 – Erstellung des Hangmanspiels für das lokale Terminal (1 Punkt)*

Aufwandsabschätzung (als Richtwert für Sie): geringer Aufwand
Nun sind Sie gefragt! Erstellen Sie mit Hilfe von `WordCheck.c` ein Hangman-Spiel für das lokale Terminal. Rufen Sie die in `WordCheck.c` bereitgestellte Funktion `server_process` in einem eigenen, neuen C-Modul aus einer `main`-Funktion auf. Als Parameter der Funktion `server_process` übergeben Sie: 0 (entspricht `stdin`) oder 1 (entspricht `stdout`)[2].

Aufgabe *2 – Erstellung des Socket Server (6 Punkte)*

Erstellen Sie jetzt einen Socket Server für das Hangman-Spiel. Benutzen Sie dazu wieder das Modul `WordCheck.c` und erstellen Sie ein eigenes, unabhängiges Hauptmodul, welches die gesamte Netzwerklogik übernimmt und an geeigneter Stelle wieder `server_process` aufruft, nur diesmal mit dem Netzwerk-Stream zum Client. Sie können Ihren Server wieder mit Telnet oder Putty als Client testen.

Tipps:

- In der Datei `WordCheck.c` ist im Laufe der gesamten Übung NICHTS zu verändern. Die ganze Programmlogik – also die Realierung des Applikationsprotokolls – ist darin enthalten. Die Netzwerklogik soll in ein eigenes Modul verpackt werden.
- Nehmen Sie sich die **Folien aus der Vorlesung sowie relevante man pages** zu Hilfe. Dort sind alle notwendigen Schritte zum Erstellen eines Socket-Servers beschrieben.
- Erleichtern Sie sich die Fehlersuche durch Auswertung der Fehlercodes der einzelnen Prozeduraufrufe.
- Betrachten Sie die man page von `getaddrinfo` bevor Sie eine einzige Zeile Code schreiben.

Aufgabe *3 – Kleine Erweiterung zum Socket-Server (2 Punkte)*

Erweitern Sie Ihren Socket Server aus der vorhergehenden Aufgabe: Wenn Ihr Server kurz nach dem Beenden nochmals gestartet wird, so schlägt `bind()` fehl, da der Port noch nicht freigegeben ist.

Beheben Sie diesen Schönheitsfehler durch Verwendung der Option `SO_REUSEADDR` aus der Funktion `setsockopt()`. Informationen zum Umgang mit `setsockopt` liefert seine man page.

[2] Eigentlich repräsentieren beide IDs den gleichen, bidirektionalen Stream, es ist also egal welche dieser IDs Sie nehmen – man kann auch auf stdin schreiben und von stdout lesen.

Aufgabe *4 – Erweiterung des Hangman-Servers II (3 Punkte)*

Erweitern Sie ihren Hangman Server um die Ausgabe der Adressdaten des Client (IP-Adresse und Port). Hinweise zur Lösung finden Sie auf Folie 27 in den VL-Unterlagen zur Socket-Programmierung.