Hochschule RheinMain
Fachbereich DCSM - Informatik
Dr. Oliver Hahm

Verteilte Systeme
SS 2020
LV 4132

Übungsblatt 5
Praktische Übungen
Bearbeitungszeit: 2 Wochen
Abgabe: 31.05.2021, 04:00 Uhr MESZ

Aufgabe 5.1 (Projekt „Hamsterasyl mit Sun RPC“):

Das selbst gestrickte RPC-Protokoll der Open-Source Bibliothek Hamsterlib, ist zwar robust und effizient, jedoch ist die Implementierung komplex und aufwändig. Das IT-Unternehmen, welches die aktuelle Implementierung des RPC-Protokolls vertreibt, verlangt horrende Preise für die Wartung. Um auf lange Sicht Geld zu sparen, beschließt das westhessische Hamsterverwahrungsunternehmen in eine standardisierte RPC-Lösung zu investieren und somit nicht mehr von einem Unternehmen abhängig zu sein. Das Sun RPC Protokoll ist allgemein verfügbar, es ist für gute Effizienz bekannt und es erlaubt durch seine C-Schnittstelle problemlos die Weiterverwendung der Hamsterbibliothek. Deshalb entscheidet man sich für diesen Standard.

Aufgabe 5.1.1 Sun RPC

In Ihrem SVN-Repository finden Sie den neuen Ordner „5“ und darin im Verzeichnis `libsrc` die bereits bekannte Hamsterlib. Unter `src` finden Sie den C-Quellcode (`hamster_cli.c`) eines einfachen Menüprogramms, das das API der Hamsterlib verwendet. Um es zu kompilieren, rufen Sie im Verzeichnis „5“ einfach `make` auf. Das ausführbare Programm kann dann anschließend mit `./hamster_climenue` aufgerufen werden.

Ihre Aufgabe ist es nun, dieses Programm in ein verteiltes Programm, bestehend aus einem **Server** `hamster_server` und einem **Client** `hamster_cliclient`, umzuwandeln. Dabei sollen Server und Client mittels des Sun RPC Mechanismus miteinander kommunizieren.

Dazu sind folgende Schritte erforderlich:

(a) Erstellen einer RPC **Schnittstellenspezifikation** `hamster.x` im Unterverzeichnis `src`.

(b) Automatisches Generieren der **RPC stubs** `hamster_clnt.c`, `hamster_svc.c`, der Headerdatei `hamster.h`, und der **Templates** `hamster_server.c` und `hamster_client.c`. Hierzu rufen Sie das folgendes Kommando auf:

```
$ rpcgen -a hamster.x
```

(c) Manuelles Anpassen der client- und serverseitigen Templates `hamster_server.c` und `hamster_client.c` so, dass der Server die Schnittstelle der Hamsterlib aufruft und der Client eine zu dieser Schittstelle identische, auf RPC aufgesetzte Schnittstelle bietet.

(d) Kompilieren der client- und serverseitigen Programmteile `hamster_cliclient` und `hamster_server`. Dazu geben Sie im Verzeichnis `5` einfach `make all` ein.

Sun RPC Tests

Zum Testen finden Sie im Verzeichnis `scripts` zwei Dateien `refinput.txt` und `refoutput.txt`. `refinput.txt` enthält eine aufgezeichnete Sequenz von Eingaben für das Menüprogramm und `refoutput.txt` enthält die aufgrund dieser Eingaben vom Referenzprogramm erzeugten Ausgaben. Zum Testen starten Sie zunächst den Server und senden Sie die Referenz-Eingabe als Standardeingabe an den Client:

```
$ ./hamster_server &
$ cat scripts/refinput.txt | ./hamster_cliclient >myoutput.txt
```

Die von Ihrem Programm erzeugte Ausgabe `myoutput.txt` muss mit der Referenz-Ausgabe weitestgehend identisch sein. (Um sich eventuelle Unterschiede anzeigen zu lassen, können Sie eines der Tools `diff` oder `meld` verwenden).

Tipps zur Bearbeitung

Machen Sie sich zunächst anhand eines einfachen Beispiels mit der Benutzung von SunRPC vertraut. Es gibt unzählige Sun RPC Tutorials im Netz. Mit [1] sei hier nur eines davon genannt. Erproben Sie das dort gezeigte Beispiel, um die Benutzung der Tools kennenzulernen.

Hinweise:

- Es kann sein, dass Sie auf ihrem eigenen (privaten) Rechner erst noch die notwendigen Pakete installieren müssen. Für Debian-basierte Systeme sind dies:
  `rpcbind`, `libc-dev-bin`
- SunRPC benötigt einen Portmapper-Service (vgl. Vorlesung). Es kann sein, dass Sie auf Ihrem System diesen Service erst starten müssen. Je nach System heisst das Programm `portmapper`, `rpcbind`, `portmapd` o.ä. .

[1] `https://www.cs.rutgers.edu/~pxk/417/notes/rpc/index.html`