Hochschule RheinMain
Fachbereich DCSM - Informatik
Prof. Dr. Robert Kaiser

Verteilte Systeme

SS 2022
LV 4132

Übungsblatt 1
Praktische Übungen
Bearbeitungszeit: 2 Wochen
Abgabe: 09.05.2022, 04:00 Uhr MESZ

Aufgabe 1.1 (Repository):

In diesem Praktikum werden die Abgaben in einem zentralen SVN-Repository verwaltet. Die Praktikumsleiter legen dazu für jedes Übungsblatt eine Vorlage an, die Sie am Anfang der Übung auschecken müssen. Machen Sie sich mit der Handhabung von SVN vertraut. Weiterführende Hilfe finden Sie dazu im Wiki des Studiengangs und im Internet, z.B.:

`https://doku.cs.hs-rm.de/doku.php?id=subversion_repository`

In den Vorlagen finden Sie eine einheitliche Projektstruktur vor, die Sie nicht verändern dürfen. Das Projekt beinhaltet i.d.R. Templates für alle Quelldateien und die anzulegende Dokumentation, sowie ein Makefile, um das Projekt mit den gewünschten Compiler-Settings unter Linux zu kompilieren:

`https://svn.cs.hs-rm.de/svn/vs22_mmust001/1/`	Vorlage Aufgabenblatt 1
`https://svn.cs.hs-rm.de/svn/vs22_mmust001/2/`	Vorlage Aufgabenblatt 2
`https://svn.cs.hs-rm.de/svn/vs22_mmust001/3/`	Vorlage Aufgabenblatt 3

Initialer Checkout der Aufgaben aus dem Repository:

```
$ svn checkout https://svn.cs.hs-rm.de/svn/vs22_mmust001/
```

Hinweis: *Sie müssen `mmust001` durch Ihren Username ersetzen!!*

Aktualisierung des Repositorys:

```
$ svn update
```

Anzeigen der lokalen Änderungen:

```
$ svn diff
```

Speichern der lokalen Änderungen im Repository (wichtig!):

```
$ svn commit
```

Kontrollieren Sie frühzeitig, ob das Template vollständig in Ihr Repository eingefügt wurde und Sie das Projekt mit `make` übersetzen können. Wenn es hier Probleme gibt, wenden Sie sich schnellstmöglich an Ihren Praktikumsleiter!

Bei Problemen mit dem SVN-Zugang wenden Sie sich bitte an die Laboringenieure.

Aufgabe 1.2 (Projekt „Hamsterasyl“):

Wer länger unterwegs ist, kann seine Goldhamster bei einem westhessischen Hamsterverwahrungsunternehmen abgeben. Die Verwaltung der Gasthamster soll nun durch ein neues IT-System unterstützt werden. Die Kunden können ihre Hamster abgeben und ihnen dabei optional einen Vorrat an Leckerli mitgeben. Für die Aufnahme eines Hamsters wird einmalig ein Grundbetrag von 17 € fällig. Die Hamster haben nichts besseres zu tun als ständig in ihrem Laufrad zu rennen, wodurch dieses mit einer Drehzahl von 25 Umdrehungen pro Minute rotiert. Diese Umdrehungen werden vom System erfasst und pro 1000 Umdrehungen erhöht sich der am Ende zu zahlende Betrag um 5 €. Die Kunden können anrufen und sich nach dem Wohlergehen ihrer Lieblinge erkundigen. Für jeden Anruf dieser Art wird 1 € berechnet. Ausserdem können die Kunden ihrem Hamster Leckerli geben lassen. Ist dabei der anfängliche Vorrat an Leckerli erschöpft, so stellt das Hamsterasyl natürlich gerne weitere Leckerli zur Verfügung – allerdings zum Stückpreis von 2 €.

Zufällig entdeckt die IT-Abteilung des Hamsterverwahrungsunternehmens die Open-Source-Bibliothek „Hamsterlib“, die exakt die benötigten Funktionen implementiert, und die somit verwendet werden soll. Den Quellcode zu dieser Bibliothek finden Sie in Ihrem SVN-Repository im Verzeichnis `libsrc` und die zugehörigen Header-Dateien liegen im Verzeichnis `include`. Dort finden Sie auch die Datei `hamsterlib.h`, die das API der Bibliothek beschreibt.

Wechseln Sie in das Verzeichnis `libsrc` und geben Sie `make` ein. Dabei wird die Bibliothek kompiliert und es wird mithilfe des Tools `doxygen`[1] eine HTML-Dokumentation aus dem Quellcode generiert, die Sie anschließend im Verzeichnis `doc/html` finden, und die Sie mit einem beliebigen Webbrowser lesen können[1].

Machen Sie sich anhand der Dokumentation und dem Quellcode in `libsrc` mit den API-Funktionen der Bibliothek vertraut. Insgesamt bietet die Bibliothek sieben Funktionen.

Im Verzeichnis `src` finden Sie eine Mustervorlage für ein einfaches Hauptprogramm zum Testen der Bibliotheksfunktionen. Weiter befindet sich im Hauptverzeichnis der Aufgabe ein `Makefile`. Erstellen Sie daraus ein C-Programm `hamster`, das mit einem Kommandoparameter und (je nach Kommando) evtl. weiteren Parametern aufgerufen wird, und das folgende Funktionen unterstützt:

- ./hamster -l
  Gibt eine Liste mit dem gesamten Hamsterbestand (Namen, Preise und Leckerli-Vorrat) tabellarisch aus.

- ./hamster -l Meier
  Gibt eine Liste mit den Hamstern des Besitzers „Meier“ aus.

- ./hamster -n Schmidt Pausbackenbube
  Fügt einen neuen Datensatz für den Hamster „Pausbackenbube“ des Besitzers „Schmidt“ mit dem Standard-Anfangspreis 17 € (= Vollpension zzgl. Laufradbenutzung) hinzu.

- ./hamster -n Schmidt Pausbackenbube 55
  Wie oben, jedoch erhält „Pausbackenbube“ einen Vorrat von 55 Leckerli mit auf den Weg.

- ./hamster -f Bilbo Baggins 3
  Verfüttere 3 Leckerli an den Hamster „Baggins“ des Besitzers „Bilbo“. Falls der Leckerli-Vorrat von „Baggins“ erschöpft ist, erhöht sich der Preis um 2 € je Leckerli.

[1] Wie bei Open-Source-Projekten üblich, ist die Dokumentation in Englischer Sprache verfasst.

- ./hamster -s Dirk Dickbacke
  Zustandsabfrage des Hamsters „Dickbacke“ des Besitzers „Dirk“. Geliefert wird die Anzahl der Laufradumdrehungen, die Größe des Leckerlivorrats und der aktuelle Preis, der sich durch die Abfrage um jeweils 1 € erhöht.

- ./hamster -b Bigspender
  Gibt den vom Besitzer „Bigspender“ zu zahlenden Gesamtbetrag (Summe über alle seine Hamster) aus und löscht alle Sätze von „Bigspender“.

Hinweise:

- Falsche oder fehlende Benutzereingaben müssen abgefangen und mit einer Fehlermeldung zurückgewiesen werden. Ihr Programm sollte in solchen Fällen eine kurze Bedienungsanleitung („RTFM-Text“) ausgeben.

- Falsche Parameter (z.B. zu lange Besitzer- oder Hamsternamen) werden von der Bibliothek zurückgewiesen. Ihr Programm sollte in solchen Fällen eine qualifizierte Fehlermeldung ausgeben.

- Jede Kombination aus Hamster- und Besitzernamen darf nur ein Mal vorkommen. Versuche, dieselbe Namenskombination ein weiteres Mal einzutragen werden von der Bibliothek erkannt und müssen mit einer qualifizierten Fehlermeldung zurückgewiesen werden.

- Verwenden Sie die Funktion `strtoul(3)` zum Umwandeln einer Ganzzahl aus der Kommandozeile in Stringdarstellung in einen `int`-Wert.

- Im Verzeichnis `scripts` finden Sie ein Shell-Script `testhamster.sh`, mit dem Sie Ihr Programm gründlich testen können. Es ruft Ihr Programm mit verschiedenen Optionen auf und vergleicht die erhaltene Ausgabe mit der Ausgabe der Referenz-Implementierung.

- Im Verzeichnis `testsuite` finden Sie darüber hinaus eine in Testsuite, die auf dem Open Source Test-Framework DejaGnu[2] basiert. Diese bietet in etwa die gleiche Testtiefe wie das Shell-Script, sie arbeitet jedoch nicht-interaktiv und generiert übersichlichere Testberichte. Um sie zu benutzen muss das Paket `dejagnu` installiert sein.

[1] `http://www.doxygen.org` [2] `https://www.gnu.org/software/dejagnu/`