Softwaretechnik
2002
Prof. Dr. H. Klaeren und M E Leypold

Eberhard Karls
Universität
Tübingen

Übungsblatt 4

Theatersitze, Make

Ausgabe: 28.5.2002
Abgabe: 6.6.2002, 12:00

Inhaltsverzeichnis

Punkteverteilung

Aufgabe 2 (Makefile) wird wahrscheinlich mit einem Viertel bis einem Drittel der Gesamtpunktzahl gewertet werden. Diese Aussage ist *unverbindlich*. Wir behalten uns hier vor, den Bewertungsschlüssel den Gegebenheiten anzupassen.

1 Modellierung einer Saalbestuhlung

Die Geschichte: Azachan ist ein kleines Land in den Bergen, das bis vor kurzem einer Militärjunta regiert wurde. In einem Putsch des Kulturausschusses wurde die Junta gestürzt – und, weil Generäle nur schlechte Clowns abgeben – in Verbannung geschickt. Der Verteidigungshaushalt wurde praktisch auf Null gekürzt, wodurch plötzlich viel Geld für Wohlfahrt und Kulturförderung zur Verfügung stand.

Der Kulturausschuss beschloss, einen Teil dieses Geldes zur Verbesserung des Kartenverkaufs für Theater, Kino und Konzerte auszugeben. Und zwar sollen im ganzen Land an Bahnhöfen, Marktplätzen usw. Automaten aufgestellt werden, mittels deren Tickets zu kulturellen Veranstaltungen verkauft werden sollen.

Im Prinzip sollen diese Automaten wie ein Fahrkartenautomat funktionieren. Die Hardware wurde bereits während der Militärregierung aus russischer Produktion erworben – die einzelnen Automaten sehen in etwa so aus wie ein Geldautomat älteren Baujahres: 80*25-Zeichen Display, Funktionstasten zu beiden Seiten des Display, Geldkartenleser, Drucker und zusätzlich eine Einheit für die Münzeinwurf. Ursprünglich war beabsichtigt, sie für den Straßenverkauf von Passierscheinen und Reisegenehmigungen einzusetzen (der Junta waren die innere Sicherheit und der „Kampf gegen die Saboteure“ ein wichtiges Anliegen). Diesem Plan wurde jedoch durch den Umsturz ein jähes Ende bereitet. Im Rahmen des 5-Jahresplans „Schwerter zu Pflugscharen“ soll nun eine neue Software für den Automaten entwickelt werden, um, wie erläutert, damit nun Theaterkarten und Ähnliches verkaufen zu können.

Die neue Regierung von Azachan hat sich an *ThinkCrime Inc.* gewandt, die diesen Auftrag gern übernommen haben. So kommt es, dass eines schönen morgens Dein Telefon klingelt – Dein Chef *Bertold Brecheisen* ist am Apparat und hat einen Auftrag für Dich: Und zwar sei eine besonders wichtige Funktion der Automaten die Möglichkeit, Karten für mehrere nebeneinanderliegende Plätze reservieren und kaufen zu können, wenn beispielsweise jemand mit FreundInnen oder Familie eine Veranstaltung besuchen möchte, für die Platzkarten ausgegeben werden.

Eigentlich, so sagt Bertold Brecheisen, sei man mit der Implementation dieser Funktion schon ziemlich weit gewesen. Man habe angenommen, die Stühle seien im Rechteck, so-und-so-viele Reihen zu so-und-so-vielen Stühlen aufgestellt und auf eine bestimmte Art durchnummeriert gewesen. Leider sei der Kunde dann „zickig“ geworden: Es gäbe auch andere Aufstellungen, in den kleineren Experimentiertheatern sei es sogar so, dass die Stühle in mehreren konzentrischen Kreisen stünden, und in den Kindervorstellungen seien die Stühle nicht nummeriert, sondern mit kleinen Tiersymbolen in verschiedenen Farben versehen.

Mit einem Wort – der Ansatz sei nicht allgemein genug gewesen, und bevor man nun nochmal von vorne begänne, würde man lieber alles gleich richtig machen wollen, und die ganze Sache zuerst etwas abstrakter beschreiben. Man hätte gehört, Du hättest in Tübingen (damals) „Softwaretechnik“ gehört und stündest deshalb mit Spezifikationen auf vertrautem, um nicht zu sagen, innigem, Fuß. Ob Du nicht eine Theaterbestuhlung allgemein beschreiben könntest: Was es bedeutet, dass eine Reihe von Sitzen „zusammenhängt“, und

schließlich, ob Du nicht die Operation spezifizieren könntest, welche eine Reihe von *n* zusammenhängenden freien Sitzen sucht und belegt.

Da Du noch nicht ganz bei Bewusstsein bist, zudem tief erschüttert von der plötzlichen Erwähnung finsterer Abschnitte Deiner Vergangenheit (Softwaretechnik in Tübingen ...), sagst Du zu. Als Du später im Büro versuchst, Dich davon zu distanzieren, knallt Bertold Brecheisen nur einen Tonbandmitschnitt des morgendlichen Gesprächs auf den Tisch und macht Dir klar, dass Du keine andere Möglichkeit hast als mitzuarbeiten. Pech gehabt!

Die Aufgabe: Spezifiziere nun in VDM-SL:

- *Bestuhlung*, eine Datenstruktur, die die Bestuhlung eines Theatersaals beschreibt.
- *Belegung*, eine Datenstruktur, die die Belegung (ob verkauft oder nicht) der einzelnen Plätze in einem Saal beschreibt.
- Ein boolesche Funktion *sind-nebeneinander*, die zu einer *Menge* von Plätzen (bzw. Platzbezeichnungen) angibt, ob diese Plätze in einer zusammenhängenden Reihe nebeneinanderliegen.
- Eine Operation *suche-plaetze* welche *n* nebeneinanderliegende freie Plätze in einem Theatersaal belegt (und natürlich die Identitäten der belegten Plätze auch zurückgibt, damit diese auf die Karten aufgedruckt werden können).

Im Zweifelsfall können Bestuhlung und Belegung auch in eine Datenstruktur gepackt werden.

Abzugeben ist: Eine Spezifikation in VDM-SL, die mindestens die Definitionen der Typen *Bestuhlung* und *Belegung*, des Prädikates (boolesche Funktion) *sind-nebeneinander* und der Operation *suche-plaetze*.

2 Make

In Aufgabenblatt 2 solltet Ihr ein Modul für einen *FiFo* (verkleidet als Print-Queue) implementieren.

Auf der Webseite zur Vorlesung findet Ihr unter

```
http://www-pu.informatik.uni-tuebingen.de/
              swt-2002/material/FiFo/FiFo.tgz
```

die (als tar.gz gepackten) Quelltexte einer Beipielimplementation dieser Print-Queue. Unter

```
http://www-pu.informatik.uni-tuebingen.de/
              swt-2002/material/FiFo/FiFo/
```

könnt Ihr die ausgepackten Quelltexte (6 Dateien inklusive Lizenzbestimmungen) herunterladen.

Make ist, wie in der Vorlesung besprochen, ein Tool zum inkrementellen Update von Dateien. Makefile-Regeln enthalten Informationen darüber, welche Dateien benötigt werden, um andere Dateien zu erzeugen (also direkt oder indirekt in den Erzeugungsprozess einfließen), und welche Kommandos zu diesem Zweck ausgeführt werden müssen.

Weitere Informationen über Make findet Ihr (u. a.) in den folgenden Quellen:

- *GNU Make Manual*, kann unter

  `http://www-doc.informatik.uni-tuebingen.de/software/i386_fbsd32/make-3.77/info/(make)Top`

  bzw. `http://www.gnu.org/manual/make/index.html` gelesen werden. Auf den Rechnern des WSI kann mit *info 'gnu make'* der (textorientierte) Info-Browser aufgerufen werden, der dann ebenfalls dieses Manual anzeigt.

- Stu Feldman, *Make – A Program for Maintaining Computer Programs*. Dies ist ein Essay im Band 2 des Handbuchs zum ursprünglichen Bell-Unix V7. Feldman ist der Autor von *Make*, das Essay ist (noch immer) erstaunlich lesbar und konzentriert sich auf das Wesentliche. Aber Vorsicht: Die Suffixregeln sind in modernen Make-Varianten nicht mehr aktuell und stattdessen durch *Pattern-Regeln* ersetzt worden. Das Unix-V7-Manual (mit dem erwähnten Essay) ist im PDF-Format unter `http://plan9.bell-labs.com/7thEdMan/bswv7.html` erhältlich.

- *make(1)*, ist die Kurzreferenz zum ursprünglichen Unix V7 make. Sie ist im Band 1, Abschnitt 1 des Unix-V7-Manual (siehe oben) zu finden.

- *Make: A Unix utility to create and maintain programs* unter `http://www-acs.ucsd.edu/instructional/unixprog/make.html`.

- John Locke, *Using Make to Maintain Software* erhältlich unter `http://people.cs.vt.edu/~kafura/cs2704/MakeIntro.ps`.

- Großmann, *Make It Easy: Eine kleine Einführung in Make*, kann unter `http://rcswww.urz.tu-dresden.de/~grossm/help/make.html` gelesen werden.

Schreibe nun ein (einfaches) Makefile, in dem die Regeln für die (inkrementelle) Übersetzung der Quelltexte der Print-Queue enthalten sind. *Make* hat bereits viele Regeln vorkonfiguriert, diese sollen aber in der vorliegenden Aufgabe nicht verwendet werden. Rufe *Make* deshalb immer mit *make -r* auf – der Kommandozeilenschalter '-r' deaktiviert die eingebauten Regeln.

Das Makefile soll auch die Pseudo-Ziele *all* und *clean* definieren, wobei *all* alle Programme übersetzt, *clean* dagegen alle erzeugten Dateien, bis auf die ursprünglichen Quelltexte, löscht.

Beachte, dass die Headerdateien auch in den Übersetzungsprozess einfließen (d. h. das Ergebnis – die Objektdatei) eines Compilerdurchlaufs beeinflussen. Ändern sich also Headerdateien, müssen bestimmte Objektdateien neu übersetzt werden.

Nimm an, es wäre mit Hilfe von *Make* gerade *testschedule* compiliert und gelinkt worden. Unterscheide nun die Fälle, in denen jeweils die folgenden Dateien modifiziert werden: (1) *jobs.cc*, (2) *queue.hh*, (3) *jobs.hh*. Welche Übersetzungsvorgänge laufen in diesen Fällen jeweils ab, wenn *Make* erneut aufgerufen wird?

Abzugeben sind: Ein sauberer Ausdruck des erstellten Makefiles. Die drei Antworten auf die obige Frage.

Viel Spaß!