Tcl-Module zu einer ausführbaren Datei verschnüren

Stern-Gucker

Starkits verpacken ganze Tcl-Anwendungen inklusive aller benötigten Dateien in einem plattformunabhängigen File. Das Ausliefern und Installieren (neudeutsch Deployment) reduziert sich so auf einfaches Kopieren. Aber auch Entwickler profitieren von der Sternenreise. Carsten Zerbst

Das vorletzte Feder-Lesen stellte schon Jean-Claude Wipplers eingebettete Datenbank Metakit vor [2]. Diese Technik nutzt der bekannte Tcl-Entwickler auch für Starkits [1], mit denen er das Ausliefern von Anwendungen vereinfacht. Tcl-Applikationen bestehen in der Regel aus einer Reihe von Skripten, Bitmaps, Konfigurationsdateien und oft auch kompilierten Erweiterungen.

Während Programmierer damit in der Regel kein Problem haben, stößt dieses Konglomerat bei Anwendern auf Unverständnis. Sie sind selten gewillt eine komplexe Dateistruktur anzulegen und Erweiterungen selbst zu kompilieren. Bei einem Starkit steckt die gesamte Anwendung inklusive aller notwendigen Erweiterungen in einer einzigen Datei. Daher auch der Name, er steht für Stand Alone Runtime Kit. Zum Starten ist eine spezielle Version des Tcl-Interpreters nötig: »tclkit«. Er steht unter [3] für die meisten Plattformen bereit; auspacken und ausführbar machen genügt:

```
wget http://www.equi4.com/pub/tk/8.4.4/➥
tclkit-linux-x86.gz
gunzip tclkit-linux-x86.gz
cp tclkit-linux-x86 tclkit
chmod 755 tclkit
```

Das Binary lässt sich wie jeder normale Interpreter verwenden. Es bringt bereits Tk, Metakit und die objektorientierte Erweiterung Incr-Tcl mit. Viele Tcl-Anwendungen gibt es als Starkit verpackt, das Starkit Distribution Archive [4] enthält zum Beispiel »tkcon«, »tkcvs« und die komplette Tcl-Entwicklungsumgebung ASED sowie Spielkram wie Keith Vetters Spirograph [5] (siehe Abbildung 1). Um eines dieser Kits zu starten, genügt »./tclkit *Programm*.kit«.

Listing 1: Hallo Welt 1

```
01 package require Tk
02
03 proc halloWelt {} {
04     puts stdout "Hallo Welt"
05 }
06 button .b -text "Hallo Welt" \
07     -command halloWelt
08 pack .b
```

Programme einpacken

Nicht nur das Starten von Starkits ist einfach, auch das Einpacken ist beinahe trivial. Als Hilfsmittel dient das SDX-Starkit »sdx.kit« [6]. Diese Kommandozeilenapplikation kennt eine Reihe von Befehlen (siehe Tabelle 1), zum Beispiel schweißt folgender Aufruf das Skript aus Listing 1 in ein Starkit ein:

```
./tclkit sdx.kit qwrap halloWelt.tcl
```

Daraus entsteht »halloWelt.kit«, der Aufruf »./tclkit halloWelt.kit« führt das neue Starkit aus. Interessanter ist der Blick in die Innereien. Mit »unwrap« ist das Paket schnell wieder ausgepackt:

```
./tclkit sdx.kit unwrap halloWelt.kit
```

Im Verzeichnis »halloWelt.vfs« entsteht dabei eine Struktur, die der in Abbildung 2 gleicht (ohne die Snack-Erweiterung): Ein Starkit ist ein virtuelles Dateisystem. Der spezielle Tcl-Interpreter »tclkit« ruft darin immer die Datei »main.tcl« auf. Dieses File wurde automatisch von »qwrap« angelegt, es besteht nur aus drei Zeilen:

```
package require starkit
starkit::startup
package require app-halloWelt
```

Die letzte Zeile startet das Originalskript »halloWelt.tcl«, da im »lib«-Verzeichnis eine Datei »pkgIndex.tcl« liegt und den Zusammenhang zwischen Paket und Skript herstellt. Die komplette Struktur hat der »qwrap«-Befehl bequemerweise angelegt. Das SDX-Kommando »./tclkit sdx.kit wrap halloWelt.kit« verwandelt sie wieder in ein Starkit.

Richtig interessant sind Starkits bei Anwendungen mit mehreren Dateien und Erweiterungen. Das Beispiel in Listing 1 erzeugt nur einen einfachen Button, der auf Klick »Hallo Welt« ausgibt. Nun soll er zusätzlich eine WAV-Datei abspielen. Es gilt, das Snack-Soundpaket und eine WAV-Datei mit in das Starkit zu packen.

Binär-Erweiterungen

Erweiterungen gehören in das »lib«-Verzeichnis, am besten so, dass sie auf mehreren Plattformen funktionieren. Statt selbst Snack in diese Form zu bringen, bietet es sich an, ein fertiges Starkit

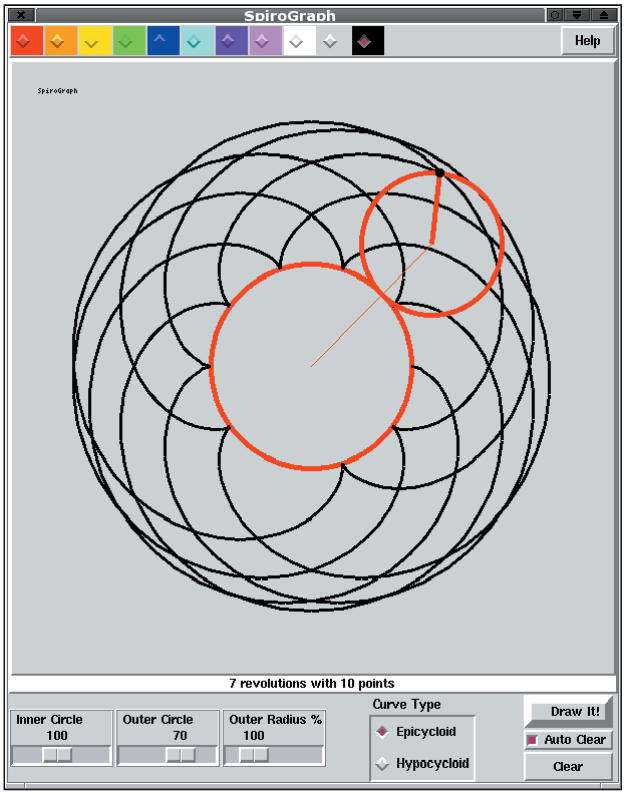

Abbildung 1: Diese nette Tcl-Spielerei zeichnet schöne Grafiken. Alle Bestandteile der Anwendung sind in einem Starkit zusammengefasst.

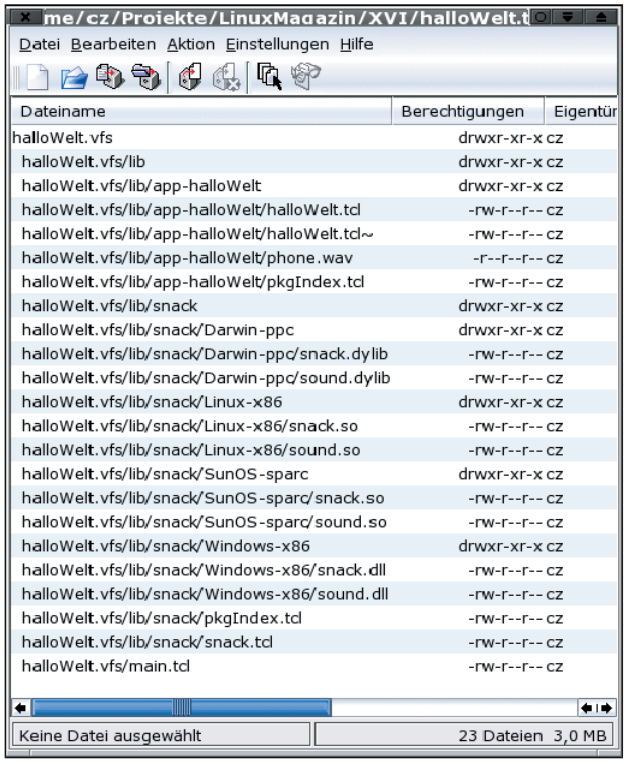

Abbildung 2: Starkits verpacken alle Bestandteile einer Anwendung. Neben den Tcl-Skripten sind hier eine WAV-Datei und die binäre Snack-Erweiterung für mehrere Zielplattformen enthalten.

mit Snack zu holen, auszupacken und in das »lib«-Verzeichnis der Anwendung zu kopieren:

```
./tclkit sdx.kit fetch ➥
http://mini.net/sdarchive/snack.kit
./tclkit sdx.kit unwrap snack.kit
cp -r snack.vfs/lib/snack ➥
halloWelt.vfs/lib/
```

Tabelle 1: SDX-Kommandos

Kommando	Bedeutung
eval *Skript*	Führt Tcl-Skript aus
fetch *URL*	Lädt Datei von einem Server
ftpd	Einfacher FTP-Server
httpd	Einfacher HTTP-Server
lsk *Starkit*	Zeigt die Struktur des Starkits
qwrap *Datei*	Erzeugt ein Starkit aus einem Skript
startsync	Startet den Starsync-Server
unwrap *Starkit*	Entpackt das Starkit
update -from *URL*	Aktualisiert das Starkit vom Starsync-Server
wrap (-writeable, -runtime *Tclkit*)	Erzeugt ein Starkit aus der *Name*-Dateistruktur im Verzeichnis »*Name*.vfs«. Mit der Option »-writeable« wird das Starkit beschreibbar, »-runtime« erzeugt ein Starpack mit dem gegebenen Tclkit als Interpreter.

Neben der Erweiterung muss noch eine WAV-Datei (hier »phone.wav«) mit ins Starkit, genauer in das Verzeichnis »halloWelt.vfs/lib/app-halloWelt«. An dieser Stelle liegt auch noch das alte »halloWelt.tcl« aus Listing 1 – an seine Stelle gehört jetzt Listing 2. Diese Version erzeugt in Zeile 10 ein Snack-Sound-Objekt und füllt es mit der WAV-Datei aus demselben Verzeichnis.

Um auf Files im Starkit zuzugreifen, muss das Skript nur den Pfad richtig wählen. Die Basisadresse steht in der Variablen »starkit::topdir«.

```
./tclkit sdx.kit wrap halloWelt.kit
```

Dies Kommando verpackt die ganze Dateistruktur unter dem Verzeichnis »halloWelt.vfs« in ein neues Starkit, das sich mit »./tclkit halloWelt.kit« starten lässt – und nach jedem Knopfdruck die WAV-Datei abspielt.

Entwicklungshilfe

Die fertigen Pakete des Starkit Distribution Archive [4] sparen viel Zeit beim Entwickeln eigener Programme. Kompilierte Erweiterungen enthalten die Shared Libraries für Linux x86, Sun Sparc und Windows, damit sind die Anwendungen sofort auf drei wesentlichen Plattformen lauffähig.

Einen besonderen Blick ist Kitten [7] wert. Die Sammlung von Tcl-Erweiterungen begann einmal mit zehn Paketen, inzwischen stellt sie über 100 in der bequemen Starkit-Form zur Verfügung. Apropos Zahlen: Platzprobleme haben moderne Rechner zwar kaum noch, aber beim Download gibt es nach wie vor ei-

Listing 2: Hallo Welt 2

```
01 package provide app-halloWelt 2.0
02 package require Tk
03 package require snack
04
05 proc halloWelt {} {
06     puts stdout "Hallo Welt"
07     snd1 play
08 }
09
10 snack::sound snd1
11 snd1 read [file join $starkit::topdir \
12     lib app-halloWelt phone.wav]
13
14 button .b -text "Hallo Welt" \
15     -command halloWelt
16 pack .b
```

nen Engpass. Starkits komprimieren ihren Inhalt automatisch, er schrumpft auf etwa die Hälfte.

Einschreiben

Virtuelle Dateisysteme wie Starkits sind keine neue Erfindung, bekannt sind zum Beispiel die Open-Office-Dokumente und die Java-Archive (Jar). Während diese auf dem Archiv-Klassiker Zip beruhen, entstehen die Starkits auf Basis der Metakit-Datenbank [2]. Damit kann das Programm einzelne Dateien im Starkit zur Laufzeit ändern, etwa um seine eigene Konfiguration direkt im Starkit zu speichern. Beim Wrappen beschreibbarer Starkits ist die Option »-writeable« nach dem Kit-Namen anzugeben, andernfalls weigert sich Tclkit und gibt eine Fehlermeldung aus.

Ein einfaches Beispiel für Schreiboperationen im Starkit zeigt Listing 3. Ab Zeile 12 erzeugt das Programm für sechs Farben je einen Knopf, der die Variable »farbe« setzt und die Prozedur »speichern« aufruft. Wann immer der User eine neue Farbe wählt (siehe Abbildung 3), speichert das Skript den Wert in der Datei »farbe.tcl«. Diese Datei enthält lediglich eine Tcl-Anweisung, die einen der Farbknöpfe aktiviert:

```
.yellow invoke
```

Listing 3: In das Starkit schreiben

```
01 package provide app-listing3 1.0
02 package require Tk
03
04 proc speichern {} {
05     puts $::farbe
06     .farbe configure -background $::farbe
07     set fd [open [file join $starkit::topdir lib app-
   listing3 farbe.tcl] w]
08     puts $fd ".$::farbe invoke"
09     close $fd
10 }
11
12 foreach f {red green blue yellow magenta cyan} {
13     radiobutton .$f -value $f -variable farbe \
14         -command speichern -text $f -anchor w
15     grid .$f -sticky ew
16 }
17
18 frame .farbe -width 50
19 grid .farbe -row 0 -column 1 -rowspan 6 -sticky ns
20
21 source [file join $starkit::topdir lib app-listing3
   farbe.tcl]
```

Abbildung 3: Das Starkit merkt sich die Farbkonfiguration.

Zeile 21 in Listing 3 importiert beim Laden der Anwendung das Konfigurationsskript, sorgt also dafür, dass das Programm immer den zuletzt gespeicherten Wert darstellt. Statt die Konfiguration in einem eigenen Format zu speichern, verwendet das Beispiel direkt Tcl – und spart so eine Lese- und Parse-Funktion.

Eine interessante Entwicklung, die den Datenbankcharakter nutzt, ist Starsync [8]. Dieses Tool aktualisiert Teile eines Starkits über das Netz. Der Transaktionsmechanismus von Metakit stellt sicher, dass das Programm nur vollständig übertragene Updates verwendet.

Verschnüren

Eine Skript-Anwendung in einer Datei, die alles inklusive der (veränderbaren) Konfigurationsdatei enthält, ist nur noch durch ein komplette Anwendung in einer einzigen Datei zu übertreffen. Auch das ist möglich: Tclkit als Interpreter und Starkit als Anwendung lassen sich zu einem so genannten Starpack verbinden. Da sich die »tclkit«-Datei während des Ablaufs nicht selbst in eine weitere Datei schreiben kann, muss man sie vorher kopieren. Die Option »-runtime« verweist auf das Tclkit:

```
./tclkit sdx.kit wrap halloWelt.kit ➥
  -writable -runtime tclkit-linux-x86
```

Damit entsteht die eigenständige Anwendung »halloWelt.kit«, sie benötigt keinen weiteren Interpreter und lässt sich direkt starten. Als Nachteil ist damit allerdings die Plattformunabhängigkeit der Starkits dahin. Wer Anwendungen nur für eine Architektur ausliefern muss, spart sich mit dieser Lösung aber aufwändiges Installieren: kopieren und ausführen genügt.

Die lesenswerte Einführung von Steve Landers [9] behandelt weitere Themen wie das virtuelle Dateisystem, die eingebettete Dokumentation und das Aktualisieren. Die Kombination von einfacher Entwicklung in einer Skriptsprache, leicht verfügbaren Erweiterungen und simpler Installation ergibt eine sehr produktive Umgebung. *(fjl)* ■

Infos

[1] Starkit-Homepage: [http://www.equi4.com/starkit.html]
[2] Carsten Zerbst, „Daten im Handgepäck – Integrierte Datenbank für Tcl“: Linux-Magazin 07/03, S. 102
[3] Tclkit-Binaries: [http://www.equi4.com/pub/tk/8.4.4/]
[4] Starkit Distribution Archive: [http://mini.net/sdarchive/]
[5] Spirograph: [http://mini.net/tcl/4206]
[6] SDX, ein Packer für Starkits: [http://www.equi4.com/pub/sk/sdx.kit]
[7] Kitten: [http://mini.net/tcl/kitten]
[8] Starsync: [http://www.equi4.com/264]
[9] Beyond Tclkit: [http://www.digital-smarties.com/Tcl2002/tclkit.pdf]
[10] Activestate: [http://www.activestate.com]
[11] Sophos: [http://www.sophos.com]
[12] IpV6-Patches für Tcl: [http://www.ngn.euro6ix.org/IPv6/tcl/]
[13] Rivet: [http://tcl.apache.org/]
[14] Zinc: [http://www.openatc.org/zinc/]

Das Neueste

Viele Anwender von Skriptsprachen waren sehr überrascht, als Activestate [10] vor kurzem vom Spam- und Virenjäger Sophos [11] gekauft wurden. Activestate ist federführend bei der Tcl-Entwicklung und -Vermarktung, auch Perl- und Python-Jünger bedient die Firma. Anders als vor einigen Jahren beim Verkauf von Scriptics geschehen, wird Activestate ihr Kerngeschäft glücklicherweise wohl nicht aufgeben.

An der Tcl-Entwicklung sind viele unabhängige Programmierer beteiligt. So sorgt ein Patch [12] dafür, dass Tcl auch in IPv6-Netzwerken arbeitet. Konservativ gibt sich Rivet [13], eine Tcl-Erweiterung für Apache 1.3. David Welton hat Version 0.2.0 veröffentlicht; trotz der kleinen Versionsnummer hat Rivet ein beachtliches Alter (samt Reife) erreicht.

Dank der Starkits kann nun jeder Programmierer ohne Kompilieren interessante Erweiterungen ausprobieren, zum Beispiel das mächtige Canvas-Widget Zinc [14]. Einige Entwickler arbeiten daran, auf seiner Basis die Anzeige von SVG-Grafiken zu implementieren. Zinc bringt erste Testgrafiken ausgesprochen schnell auf den Bildschirm.