Die Datei sendkeys.dll wurde von Jeffrey Richter geschrieben und im Microsoft System Journal März/April 1993 veröffentlicht (mit Sourcen).
Die Datei wurde von mir im wesentlichen unverändert übernommen (ein kleiner Patch war notwendig, da die DLL, wie sie auf der Diskette mitgeliefert wurde, nicht problemlos lief; sie hatte Probleme mit dem Wiederholungszähler).

Die Weitergabe der Sendkeys-DLL im Rahmen des shell93 Paketes geschieht mit freundlicher Genehmigung der Synergy Verlag GmbH, Redaktion Microsoft System Journal.

Der Name SendKeys ist etwas irreführend. Eigentlich werden keine Tastenanschläge versendet, sondern in einer systemweiten Art generiert. D.h. jedes Fenster kann diese Tasten empfangen, das Fenster das die Tasten tatsächlich annimmt ist immer das mit dem Eingabefokus.
Oder umgekehrt, wenn Sie die Anschläge kontolliert an ein Fenster senden wollen (was wohl die Regel ist), müssen Sie dafür sorgen, daß das Fenster den Eingabefokus hat.
Der Unterschied zwischen der Versendung von Tastaturanschlägen und ihrer Erzeugung wird dann am deutlichsten, wenn die Tastensequenz den Eingabefokus an ein anderes Fenster weitergibt.

Der Einsatz der gkbd() Funktion ist eine Möglichkeit
ungewollte Veränderungen oder gar Beschädigungen Ihres System hervorzurufen.
Setzen Sie die Funktion nur dann ein, wenn Sie genau wissen, welches Fenster den Eingabefokus hat, und wie es auf die von Ihnen erzeugten Tastenanschläge reagiert.

Aufbau des gkbd() Parameters

Grundsätzlich wird jede Taste durch einen Buchstaben realisiert.

```
gkbd("e")
gkbd("E")
```

simuliert beidemale die Taste "E".

Die Zeichen ~, +, ^ und % (Tilde, Plus, Caret und Prozent) haben spezielle Bedeutungen. Um die entsprechenden Tasten zu simulieren müssen die Zeichen in geschweifte Klammern eingeschloßen werden:

```
gkbd("{+}")
```

für die "+" Taste, etc.
Das Gleiche gilt für geschweifte Klammer { }, sowie für runde Klammern (und):

```
gkbd("{{}abcd{}}")
```

erzeugt die Tastensequenz "{abcd}".

Die Tilde ~ ist ein Synonym für die enter-Taste.

Das Plus Zeichen erzeugt die nachfolgende Taste zusammen mit der Shift-Taste, also

```
gkbd("+E")
```

für shift+"E".
Beachten Sie hierbei aber, daß das Plus Zeichen nur die nächste Taste shiften:

```
gkbd("+EX")
```

erzeugt shift+"E" und "X" (kleines x).
Wollen Sie mehrere geschiftete Tasten (mit 'heruntergehaltener' shift-Taste) absetzen, so müssen Sie die Tasten in runde Klammern einfassen:

```
gkbd("+(EX)")
```

löst shift+"EX" aus, wobei die shift-Taste zwischen beiden Buchstabentasten nicht freigegeben wird.
Im Gegensatz hierzu liefert

```
gkbd("+E+X")
```

shift+"E" und shift+"X".

Die Ctrl-Taste wird analog durch den Caret `^`, die Alt-Taste durch das Prozent `%` gesteuert.
Plus, Caret und Prozent sind kombinierbar.

```
gkbd("+^E")
```

liefert shift/ctrl+"E".

Sondertasten (Funktionstasten, Cursorsteuerung etc.) werden durch Namen realisiert. Diese Namen sind in geschweifte Klammern einzufassen:

```
gkbd("{home}")
```

'bedient' die home-Taste.

Folgende Sondertasten sind definiert:

backspace bksp bs
break
capslock
clear
delete del
down
escape esc
end
enter
F1 - F16 (16 Funktionstasten)
help
home
insert
left
numlock
pgdn

pgup
prtsc
scrollock
right
tab

Groß/Kleinschreibung ist egal,

```
gkbd("ESC"), gkbd("esc"), gkbd("escape")
```

löst immer ein die Escape-Taste aus.

Innerhalb von geschweiften Klammern kann nur eine Taste aufgeführt werden, geben Sie mehrere Zeichen an, so wird diese Sequenz automatisch als Sondertaste behandelt:

```
gkbd("{x}"), gkbd("{home}"), gkbd("{xy}")
```

Der dritte Aufruf liefert Ihnen einen Fehler zurück, da xy keine Sondertaste bezeichnet.

In geschweiften Klammern kann hinter der Taste ein Wiederholungszähler (von dem Tastenbezeichner durch genau ein Leerzeichen getrennt) folgen:

```
gkbd("{down 10}")
```

'betätigt' 10 mal die cursor-down-Taste.