Schneller TEXen mit Tastenkürzeln

Uwe Ziegenhagen

»Shortcut Expander«, auch »Text Expander« genannt, sind Programme, die es dem Nutzer ermöglichen, systemweit Tastenkürzel für häufig genutzte Textschnipsel zu definieren. So kann aus einem eingetippten »mfg« automatisch ein »Mit freundlichen Grüßen« oder einem »ä#« das aktuelle Datum werden. Da man als TEXie naturgemäß sehr häufig bestimmte Befehle eingeben muss – man denke allein an die unzähligen `\section{}` oder `\subsection{}` Befehle – sind diese Werkzeuge es auf jeden Fall wert, dass man sich näher mit ihnen beschäftigt.

Im folgenden Artikel werden drei Programme für die gängigsten Plattformen Windows, OS X und Linux kurz vorgestellt, bevor dann am Beispiel von Autohotkey sinnvolle TEX-bezogene Kürzel erstellt werden.

Übersicht über verschiedene Werkzeuge

Autokey (Linux)

Autokey[1] ist das Linux-Gegenstück zu PhraseExpress[2], einem kommerziellen Text Expander für Windows. Da Autokey-Versionen vor 0.90 einige Bugs aufwiesen, sollte man prüfen, ob die eingesetzte Linux-Distribution wirklich die neueste Version bereitstellt. Im Zweifelsfall empfiehlt es sich, gleich die Version von der Projekt-Webseite installieren.

Als Skript-Sprache setzt Autokey auf Python, es sollten sich daher alle Aufgaben meistern lassen, die auch von Python-Skripten ausgeführt werden können.

TextExpander (Mac OS)

TextExpander für Mac OS X, siehe Abbildung **??**, ist mit einem Preis von $ 34,95 die einzige kommerzielle Lösung dieses Artikels. Neben der Expansion von Tastenkürzeln kann TextExpander mit Datumswerten rechnen oder AppleScript/Shell-Skripte ausführen. Auf der OS X Plattform gilt TextExpander als *die* Referenz. Es gibt sogar eine Version für iOS, die aber leider auf die individuelle Unterstützung durch jede einzelne App angewiesen ist.

[1] `https://code.google.com/p/autokey/`
[2] `http://www.phraseexpress.com/`

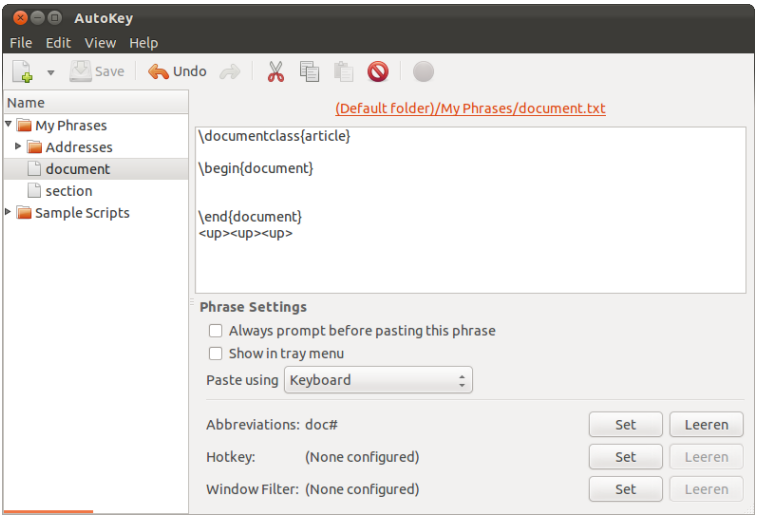

Abb. 1: Autokey Screenshot

Autohotkey (Windows)

Autohotkey gibt es seit mittlerweile zehn Jahren, es steht unter `www.autohotkey.com` zum Download bereit. Wird Autohotkey nach der Installation gestartet, so fragt es nach, ob es eine entsprechende `.ahk` Datei im Home-Verzeichnis des Nutzers erstellen soll. Alternativ kann man eine `.ahk` Datei auch an beliebiger Stelle ablegen und eine Verknüpfung im Autostart-Ordner anlegen, sodass beim Systemstart Autohotkey automatisch lädt. Beim Autor liegt diese Datei in der Dropbox, damit alle vorhandenen Windows-Rechner stets die aktuelle Version der Kürzel haben.

Die grafische Oberfläche, siehe Abbildung **??**, ist recht schlicht. Über das Kontextmenü der Applikation im System-Tray lässt sich das Skript editieren, neu laden oder pausieren. Außerdem lässt sich der »Window Spy« starten, mit dessen Hilfe man ermitteln kann, an welchen Koordinaten sich die Maus befindet oder mit welchem Namen/Typ die einzelnen offenen Fenster im Autohotkey-Code angesprochen werden können.

Ruft man den edit-Befehl auf, so startet Autohotkey Notepad.exe. Wer mehr Editor-Funktionalität benötigt, dem sei SciTE4AutoHotkey[3] empfohlen, das Funktionen wie Syntax-Highlighting und Auto-Completion bietet.

[3] `http://fincs.ahk4.net/scite4ahk/`

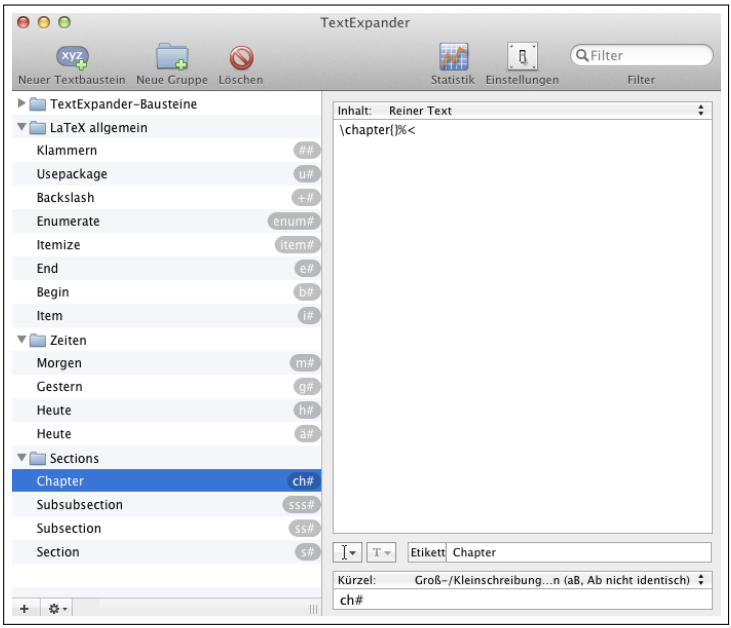

Abb. 2: Textexpander Screenshot (Mac OS X)

Der Funktionsumfang von Autohotkey ist ausgesprochen hoch, neben den in diesem Artikel besprochenen Tastenkürzeln bietet Autohotkey eine komplette Skriptsprache, mit der neben der Automatisierung vieler Windows-Funktionen sogar grafische Benutzeroberflächen gestaltet werden können.

Kürzel-Expansion und mehr mit Autohotkey

Im zweiten Teil dieses Artikels soll es um die konkrete Definition verschiedener TEXnischer Kürzel mit Autohotkey unter Windows gehen, die gezeigten Funktionen sollten sich aber größtenteils auch auf den anderen Plattformen umsetzen lassen.

Listing **??** zeigt einfache Beispiele, wie die Tastenkürzel – auch »Hotstrings« genannt – definiert werden müssen, damit aus `s#`, `ss#` und `sss#` entsprechende Gliederungsbefehle werden. Die Kürzel werden dabei so gesetzt, dass unmittelbar nach der Eingabe des Kürzels ersetzt wird.

```
:*:s#::\section{{}{}}{LEFT}
:*:ss#::\subsection{{}{}}{LEFT}
:*:sss#::\subsubsection{{}{}}{LEFT}
```

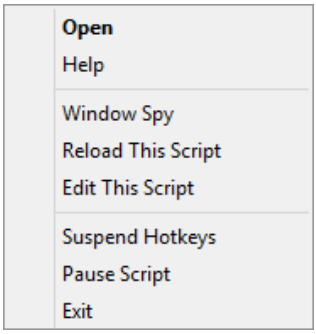

Abb. 3: Autohotkey Kontextmenü

Listing 1: \section-Expansionen

Es gibt auch die Möglichkeit, erst nach einem Auslöse-Zeichen (wie Leer- oder Satzzeichen) zu expandieren, mehr dazu im Hilfesystem.[4]

Da geschweifte Klammern für Autohotkey aktive Zeichen darstellen (die Befehle zum Bewegen des Cursors stehen beispielsweise in geschweiften Klammern) müssen geschweifte Klammern, die in unserem LaTeX-Dokument auftauchen sollen, entsprechend ausgezeichnet werden. Das `{LEFT}` am Ende schließlich weist Autohotkey an, den Cursor um ein Zeichen nach links zu bewegen. Schon dieses einfache Beispiel zeigt, mit Autohotkey lässt sich jede Menge Tipp-Arbeit sparen!

Tabelle **??** zeigt die Kürzel, die ich für mich definiert habe und als sinnvoll erachte. Das sichtbare Leerzeichen ␣ zeigt dabei, wo der Cursor nach der Expansion steht.

Modifier	Beschreibung
#	Windows
!	Alt
+	Shift
^	Control-Taste (Strg)

Tab. 1: Nützliche Kürzel für LaTeX

[4] `www.autohotkey.com/docs/Hotstrings.htm`

Tab. 2: Nützliche Kürzel für LaTeX

Kürzel	Expansion
`:*:c#::\chapter{{}{}}{LEFT}`	\chapter{␣}
`:*:s#::\section{{}{}}{LEFT}`	\section{␣}
`:*:ss#::\subsection{{}{}}{LEFT}`	\subsection{␣}
`:*:sss#::\subsubsection{{}{}}{LEFT}`	\subsubsection{␣}
`:*:b#::\begin{{}{}}{LEFT}`	\begin{␣}
`:*:e#::\begin{{}{}}{LEFT}`	\end{␣}
`:*:i#::\item{Space}`	\item ␣
`:*:l#::\label{{}{}}{LEFT}`	\label{␣}
`:*:r#::\label{{}{}}{LEFT}`	\ref{␣}
`:*:(::(){LEFT}`	(␣)
`:*:{::{}{}}{LEFT}`	{␣}
`:*:[::[]{LEFT}`	[␣]

Nicht nur »normale« Tasten lassen sich über Autohotkey ansteuern. Tabelle **??** zeigt einige wichtige »Keyboard-Modifier«, die vollständige Liste ist unter `www.autohotkey.com/docs/Hotkeys.htm#Symbols` verfügbar:

Listing **??** zeigt die praktische Anwendung. Für die F3- bzw. F4-Taste werden kurze Programme definiert, die aktuell markierten Text in die Zwischenablage kopieren, den `\emph{}` bzw. `\enquote{}` ins Dokument setzen und den Text aus der Zwischenablage an der richtigen Stelle wieder einfügen. Somit lassen sich Funktionen definieren, die in der eigentlichen Software nicht vorhanden oder nur umständlich verfügbar sind.

```
F3::
Send ^c
Send \emph{{}{}}{LEFT}
Send ^v
Return

F4::
Send ^c
Send \enquote{{}{}}{LEFT}
Send ^v
Return
```

Listing 2: Programmierung der F-Tasten mit Autohotkey

Listing **??** zeigt zwei Befehle, die von mir in TEXworks schmerzlich vermisste Funktionen nachrüsten: das Löschen einer Zeile über `Strg-k` und die Duplikation einer Zeile über `Strg-d`.

```
#IfWinActive ahk_class QWidget
{
^k::
Send {HOME}
Send {SHIFT}+{END}
Send {DEL 2}
return

^d::
Send {HOME}
Send {SHIFT}+{END}
Send ^c
ClipWait, 2
Send {END}
Send {ENTER}
Send ^v
return
}
#IfWinActive
```

Listing 3: Zeilen duplizieren und löschen

Oberflächen mit Autohotkey

Zum Abschluss des Artikels zeigt Listing **??** noch ein relativ einfaches Script für die Programmierung einer grafischen Oberfläche. Es erzeugt über die Tastenkombination `Alt+e` ein grafisches Menü, in dem der Nutzer eine Zeichenkette der Form `UmgebungZahl` eingeben kann. Diese Zeichenkette wird dann intern über einen regulären Ausdruck in »Umgebung« und »Zahl« geteilt. »Umgebung« wird dann mittels `\begin/\end` als LATEX-Umgebung ins Dokument geschrieben; es werden »Zahl« verschiedene `\item` Befehle ausgegeben und der Cursor hinter dem ersten `\item` positioniert.

Fazit

Wer einmal mit einem Text Expander gearbeitet hat, lernt sehr schnell die Annehmlichkeiten dieser Werkzeuge schätzen. Mit wenigen Schritten lassen sich Kürzel definieren, die schon nach kurzer Eingewähnung erheblich Zeit sparen können und dem Nutzer mehr Zeit geben, sich auf das Wesentliche – den Text – zu konzentrieren.

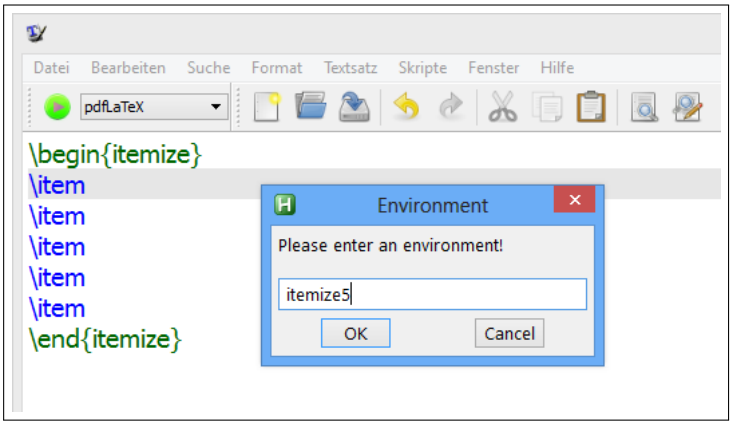

Abb. 4: Grafische Oberfläche aus Listing **??**

Die genutzte Autohotkey-Datei steht unter `www.uweziegenhagen.de` zur Verfügung, für Kommentare und Anregungen bin ich stets dankbar.

```
!e::
InputBox, UserEnv, Environment, Please specify an environment!, , 240, 120
If ErrorLevel return
Else
if( RegExMatch(UserEnv, "(.*?)(\d+)$", splitted) ) {
  Send \begin{{}%splitted1%{}}{Enter}
    Loop %splitted2% {
      Send \item {Enter}}
  Send \end{{}%splitted1%{}}{Up}
  count2 := splitted2 - 1
  Loop %count2% {
    Send {Up}
  }}
Else
  Send \begin{{}%UserEnv%{}}{Enter 2}\end{{}%UserEnv%{}}{Up}
return
```

Listing 4: Oberfläche mit Autohotkey