

Spring School  
›Text Processing mit TUSTEP‹  
(Tübinger System von  
Textverarbeitungs-Programmen)

Universität Trier

Skript

Matthias Schneider, M.A.  
[schneiderm@uni-trier.de](mailto:schneiderm@uni-trier.de)

Kompetenzzentrum für elektronische  
Erschließungs- und Publikationsverfahren  
in den Geisteswissenschaften  
Universität Trier  
[www.kompetenzzentrum.uni-trier.de](http://www.kompetenzzentrum.uni-trier.de)

Stand: 03.10.2018

## Gliederung

- Vorbemerkungen
- Organisatorisches
- Grundlagen: TUSTEP & Text Processing
- Erste Schritte in TUSTEP
- TUSTEP-Editor
- Import von Daten
- Einstieg und einfache Anwendungen in TUSCRIPT
- Einstieg in den hochwertigen Satz: #\*SATZ
- Dateizugriffe und Verarbeitung mit TUSCRIPT
- XML-Auszeichnungen mit TUSCRIPT
- Konvertieren bibliographischer Datenbankexporte
- Herstellen von Korrekturfassungen: Export nach TXT, RTF, HTML, EPUB, PDF

## Vorbemerkungen

☞ **Hinweis:** Dieses Skript basiert auf verschiedenen Materialien, die sukzessive für verschiedene Kurse und Workshops mit unterschiedlichen thematischen Schwerpunkten erstellt wurden. Zwar wurde versucht, Brüche in der Darstellung zu minimieren, allerdings ist davon auszugehen, dass der derzeitige Stand noch nicht zur eigenständigen Benutzung ohne Hilfestellung und weitere Erläuterungen geeignet ist. Außerdem ist mit Tippfehlern, Formulierungsschwächen, ggf. auch inhaltlichen Inkonsistenzen u.ä. zu rechnen. Anregungen, Hinweise und Fehlerkorrekturen werden gerne aufgenommen. Gerne per Mail an: [schneiderm@uni-trier.de](mailto:schneiderm@uni-trier.de).

Das vorliegende Skript entstand als Vorbereitung und Begleitmaterial für verschiedene TUSTEP-Kurse und wurde fortlaufend an die verschiedenen Kursinhalte angepasst. Gemäß der in den Veranstaltungen vorgestellten Inhalte ist die Auswahl der hier präsentierten Funktionen sehr stark selektiert.

Im vorliegenden Skript werden in technischer Hinsicht Texte als *Textdaten* (im Unterschied zu Bilddaten, Audio- und Videodaten o.ä.) verstanden, die es zu be- und verarbeiten gilt. Es werden hierbei keine philologischen und kodikologischen Unterscheidungen zwischen *Buch*, *Text*, *Textteilen* usw. vorgenommen. Vielmehr steht der technisch unterstützte Umgang mit diesen Daten im Vordergrund, der auf unterschiedlichen Ebenen erfolgen kann:

1. die Erfassung/ das Schreiben von Texten
2. die manuelle Korrektur/Bearbeitung von Texten
3. die automatische oder programmatisch unterstützte Korrektur/Bearbeitung von Texten inklusive Überführung in andere Datenformate
4. die Publikation von Texten in unterschiedlichen Formaten und Erscheinungsformen (digital/print)
5. die wissenschaftliche Analyse von Texten<sup>1</sup>

Wenn im Folgenden von »Text« gesprochen wird, sind damit immer verkürzend die *Textdaten* gemeint.

Das Skript geht davon aus, dass die Texte, die es zu be- und verarbeiten gilt, in einem XML-Format vorliegen oder in ein solches überführt werden sollen, was als eines der am weitesten verbreiteten Datenformate in den digital arbeitenden Geisteswissenschaften gelten kann. Damit ist gemeint, dass die Texte, die in TUSTEP verarbeitet werden, in unseren Beispielen mit Tagstrukturen versehen sind oder versehen werden können, die Daten selbst aber i.d.R. im TUSTEP-eigenen Datenformat gespeichert werden. Eine Ausgabe nach »XML« in UTF-8 ist leicht möglich. In den gegebenen Beispielen wird durchgehend auf die XML-konforme Deklaration von Header-Informationen sowie auf

---

<sup>1</sup> Z.B. quantitative Analysen, editionsphilologische motivierte Vergleiche u.ä.m.

ein Wurzelement verzichtet, das den gesamten Text umfasst, um die Arbeitsdaten nicht zu überfrachten. Vor einem Export der Daten nach XML können diese Informationen problemlos per Skript eingefügt werden. Ebenso wird auf die Nutzung eines spezifischen XML-Vokabulars (wie dasjenige der *Text Encoding Initiative*; = TEI) meist verzichtet und stattdessen ein Arbeits-XML genutzt, das gemäß der persönlichen Präferenzen des Anwenders ausgestaltet werden kann, um abschließend (z.B. beim Export nach HTML oder TEI-XML) in das finale Format überführt zu werden.<sup>2</sup>

Im Skript wird häufig auf das TUSTEP-Wiki verwiesen (<http://www.tustep.wikispaces.com>), das im Sommer 2018 auf eine andere Plattform umziehen muss, da der Betreiber von Wikispaces seine Dienste einstellt. Die hier angegebenen URLs haben entsprechend ein Verfallsdatum, das es zu beachten gilt. Informationen über den Umzug des Wiki werden auf der Seite der ITUG (<http://www.itug.de>) sowie über die dort verlinkte Mailing-Liste bereitgestellt. In Facebook gibt es die Gruppe »TUSTEP Community«, die ebenfalls dem Austausch unter den Nutzern dient: <https://www.facebook.com/groups/559140454216784/>.

Die Verweise auf das TUSTEP-Handbuch und die weiteren Beschreibungen von TUSTEP-Programmen erfolgen nicht seiten-, sondern kapitelweise, da sich die Seitenzählung bei jeder neuen TUSTEP-Version, für die die Beschreibungen ergänzt werden, ändern kann und eine allzu schnelle Veraltung der Angaben vermieden werden sollte. Um dem Benutzer zu ermöglichen, die Textstellen möglichst schnell aufzufinden, wird zudem der an den PDF-Lesezeichen orientierte Hierarchiebaum der Kapitel mit angegeben. Dabei werden die einzelnen Hierarchiestufen mit einem senkrechten Strich (= |) abgegrenzt, z.B.: »Editor | Tastenkombinationen für Makroaufrufe«.

---

<sup>2</sup> Dieses Vorgehen hat neben dem häufig geringeren Notationsaufwand einige Vorteile; vgl. exemplarisch die Länge der Taginhalte:

`<hi rend='italic'>` vs.

`<i>`.

---

## Organisatorisches zur Spring School 2018

---

- Raum: DM 343
- Mittwoch-Freitag, 21.–23. März 2018
- Zeitplanung: 9–12.30 Uhr und 13.30–16.30 Uhr Workshopkernzeit, anschließend ggf. Zeit für projektspezifische Aufgaben
- Notebooks: Alle Teilnehmer sollten ein eigenes Notebook mitbringen, auf dem sie auch Administratorrechte besitzen, damit TUSTEP sowie ggf. Zusatzprogramme installiert werden können.
- Download von Materialien:

<https://github.com/ms-tr/kurs18tr>

## Grundlagen: TUSTEP & Text Processing

## Grundlagen

### TUSTEP – Programmgeschichte

---

TUSTEP oder das *Tübinger System von Textverarbeitungs-Programmen* wurde maßgeblich an der Universität Tübingen vorgedacht und programmiert. Heute führen Professor Dr. Wilhelm Ott und Kuno Schälkle die Entwicklung in Kooperation mit der Uni Tübingen und der International TUSTEP User Group (ITUG) weiter. Das Programm geht auf die Anfänge der nichtnumerischen Datenverarbeitung zurück und entstand aus Vorläufern ab 1966 in der Abteilung Literarische und Dokumentarische Datenverarbeitung (LDDV) im Zentrum für Datenverarbeitung (ZDV) der Universität Tübingen. Ab dem Frühjahrskolloquium 1978 wurden die bestehenden Routinen unter der Bezeichnung »TUSTEP« zusammengeführt und weiterentwickelt.<sup>3</sup>

Die Anfänge dieser Arbeiten gehen auf das Jahr 1966 und die folgenden Jahre zurück, als nach dem Vorbild des nichtnumerischen Unterprogrammpakets des Deutschen Rechenzentrums Darmstadt ein System von Unterprogrammen zur Zeichen- und Zeichenketten-Verarbeitung in FORTRAN entworfen und auf der Tübinger Anlage implementiert wurde. Dies erleichterte die Programmierung für damals betreute Projekte wie die metrische Analyse lateinischer Hexameter, die Konkordanz zur Vulgata oder die Edition und die Register zu den Werken Heinrich Kaufringers.

Aufgrund der Erfahrung mit solchen Projekten konnte der nächste Schritt getan werden, der die Programmierung in FORTRAN oder einer anderen »höheren« Programmiersprache für diese Art von Aufgaben überflüssig machte: Für die wichtigsten Grundoperationen der Textdaten-Verarbeitung wurden Programme bereitgestellt, deren Leistung vom Benutzer über Parameter spezifiziert werden kann und die in vielfältiger Weise für die Lösung verschiedenster Aufgabenstellungen zu kombinieren sind. Den Namen TUSTEP tragen diese Programme seit 1978.<sup>4</sup>

---

<sup>3</sup> Eine Übersicht von Referenten, Themen und z.T. auch Abstracts zu den insgesamt 90 Kolloquien, die in den Jahren 1976–2005 stattgefunden haben, findet man auf der TUSTEP-Homepage:

<http://www.tustep.uni-tuebingen.de/kolloq.html>.

<sup>4</sup> <http://www.tustep.uni-tuebingen.de/tdv17.html>.

Wie der Name bereits anzeigt, handelt es sich bei TUSTEP nicht um *ein* Programm, sondern um eine Sammlung von *mehreren* Programmen, die dem Zweck der Textdatenverarbeitung dienen.

Ist im TUSTEP-Umfeld von *Textdaten* oder *Textdatenverarbeitung* die Rede, ist die wissenschaftliche, reliable, professionelle, maschinelle Be- bzw. Verarbeitung von Texten (im weiteren Sinne inklusive Programm Quellcode) gemeint. Diese Nomenklatur dient der Abgrenzung von Programmen, welche die Erfassung und ergebnisnahe Darstellung von Texten am Bildschirm ohne wissenschaftlichen Anspruch zum Gegenstand haben (z.B. What you see is what you get (WYSIWYG)-Programmen wie LibreOffice oder Microsoft Word). Bei TUSTEP stehen folglich nicht Aspekte wie *creative writing* oder die Erstellung von Vorlagen für alltägliche Office-Anforderungen im Vordergrund, sondern die Beschäftigung mit Texten als Daten, die automatisch oder teilautomatisch verarbeitet (aufbereitet, verglichen, gesetzt, codiert) werden sollen.<sup>5</sup>

Hierzu finden sich auf der TUSTEP-Homepage folgende Ausführungen:

Selbstverständlich gehören auch die für die Dokumenten-Erstellung notwendigen Funktionen wie Eingabe, Korrektur, Formatieren und Drucken von Texten (auch von fremdsprachlichen Texten in nicht-lateinischen Alphabeten) zum Leistungsangebot von TUSTEP, da diese in allen Wissenschaftsbereichen zum Zweck der Dokumentation und der Vorbereitung von Publikationen benötigt werden. TUSTEP wurde aber als Werkzeug vor allem für diejenigen Wissenschaften entwickelt, in denen Texte Objekte der Forschung sind: Philologien, Sprachwissenschaften, Literaturwissenschaften, historische Wissenschaften, Bibliothekswesen; Wissenschaften also, in denen nicht nur neue Texte als Produkt der eigenen wissenschaftlichen Arbeit erstellt und publiziert werden, sondern in denen schon existierende, überlieferte, schriftlich fixierte oder zu fixierende Texte (einschließlich literarischer Texte und historischer Quellen) durch kritische Neuedition gesichert, sprachlich und stilistisch analysiert, inhaltlich erschlossen, bibliographisch erfasst werden.<sup>6</sup>

TUSTEP entspricht einem umfangreichen Werkzeugkasten zur wissenschaftlichen Bearbeitung von Texten aller Art, egal ob es sich um die Erstellung (Schreiben und Setzen) einer Hausarbeit oder einer Dissertation, um die Anreicherung von XML-Daten, die Herstellung einer historisch-kritischen Edition<sup>7</sup> oder um die Konvertierung alter Verlags-RTF-Daten für die Herstellung einer Neuauflage in Form eines EPUB handelt. TUSTEP hält hierfür Programme vor, die der Benutzer für seinen jeweiligen Einsatz-

---

<sup>5</sup> Dabei ist nicht zu unterschätzen, dass auch komplexe und leistungsfähige Systeme wie TUSTEP, LaTeX oder ähnliche Mitbewerber durchaus gewinnbringend im akademischen Alltag eingesetzt werden können.

<sup>6</sup> <http://www.tustep.uni-tuebingen.de/tdv17.html>.

<sup>7</sup> S. beispielsweise die Editio Coloniensis des Albertus Magnus: [http://institutionen.erzbistum-koeln.de/albertus-magnus-institut/editio\\_coloniensis](http://institutionen.erzbistum-koeln.de/albertus-magnus-institut/editio_coloniensis).



zweck passend zusammenstellen und nutzen kann.<sup>8</sup> Wichtige Programme sind hierbei etwa der Editor, TUSCRIPT/Kopiere, das Satz-Programm, die Programme zur Erstellung von Registern und die Vergleichs-/Kollationierungswerkzeuge.

Entscheidend und hervorzuheben ist hierbei, dass TUSTEP keine vorgefertigten Lösungen bietet. Funktionen zum ›vollautomatischen‹ Verarbeiten sucht man vergeblich. Vielmehr werden grundlegende Funktionen bereitgestellt, die vom Benutzer parametrisiert und kombiniert angewandt werden können. Bei der Zusammenstellung kann ein sehr großes Maß an Automatisierung erreicht werden, wobei der Benutzer selbst für die Abstimmung verantwortlich ist und hierdurch die Möglichkeit erhält, seine Verarbeitung exakt auf seine Daten anzupassen. Um im Bild des Werkzeugkastens zu bleiben: Der Anwender erhält frei kombinierbare und zusammen einsetzbare Werkzeuge und kann Bohrmaschine, Hammer, Säge und Hobel passend für die ihn interessierende Aufgabe einsetzen. Wie man sich die Arbeit mit TUSTEP vorstellen kann, soll anhand der Designprinzipien verdeutlicht werden.

---

<sup>8</sup> Weitere typische Anwendungsbereiche sind: Paläographie, die Erstellung und der Satz von Wörterbüchern respektive Indizes, sowie Retrodigitalisierungsprojekte.

### Designprinzipien:<sup>9</sup>

Die Entwicklung und Benutzungsphilosophie von TUSTEP folgt einigen Prinzipien, die auf die Anfangszeit der geisteswissenschaftlichen Datenverarbeitung zurückgehen und entsprechend zum Teil mit den seither veränderten Benutzererwartungen nicht übereinstimmen. Dennoch bieten diese Grundbedingungen ein überhaus leistungsfähiges Fundament für IT-Anwendung in den Geisteswissenschaften auch nach über 40 Jahren Programmentwicklung und lassen sich auch auf individuelle Anforderungen und Wünsche einstellen.

Eigenschaft	Folge
Funktionalität	Vorrang von Leistung vor intuitiver Bedienbarkeit
Professionalität	Nutzer ist auf Einarbeitung und strikte, syntax-basierte Arbeitsweise verwiesen
Portabilität	Versions- und Plattformunabhängigkeit ist gewährleistet, fordert Kompromisse im Bereich der GUI
Modularität	Keine Gesamtlösungen, sondern einzelne, kombinationsfähige Werkzeuge

- TUSTEP kann für alle Stationen geisteswissenschaftlicher, text-basierter Arbeiten genutzt werden. Von der Datenerfassung über die Kollationierung, die Datenmodifikation bis zur Publikation in Print-, HTML-, Datenbankform oder als eBook muss die Plattform nicht verlassen werden (*Integration*).
- TUSTEP setzt auf einen *professionellen* Nutzer, der sich durch die entsprechende Konfiguration des *modularen* Programmpakets für seine Bedürfnisse eine passende Benutzeroberfläche zusammenstellen kann.

Die Benutzeroberfläche wird unter verschiedenen Windows-Versionen (u.a. XP, Vista, 7, 8, 10), unterschiedlichen Linux-Distributionen inklusive deren Terminals/Shells (optimale Ergebnisse erzielt der Terminalemulator `Konsole` aus der KDE-Desktopumge-

<sup>9</sup> Vgl. hierzu grundlegend Ott, Wilhelm, Strategies and Tools for Textual Scholarship. The Tübingen System of Text Processing Programs (TUSTEP), in: Literary and Linguistic Computing 15 (2000) H. 1, S. 92–108.

bung) und macOS betrieben, daher ist das Ausnutzen von betriebssystemseitigen Graphikmöglichkeiten nur eingeschränkt möglich. Die in der default-Einstellung nicht barrierefreie Darstellung der Benutzeroberfläche kann recht leicht angenehmer und barrierefreier gestaltet werden.<sup>10</sup>

---

<sup>10</sup> Vgl. Alberding, Stefanie / Schneider, Matthias, Barrierefreiheit in den Digital Humanities: Probleme und Lösungen am Fall TUSTEP, in: Friederike Kerkmann / Dirk Lewandowski (Hg.): Barrierefreie Informationssysteme: Zugänglichkeit für Menschen mit Behinderung in Theorie und Praxis, Berlin, Boston 2015 (Age of Access? Grundfragen der Informationsgesellschaft 6), S. 126–139 und allgemeiner [www.tustep.wikispaces.com/Erleichterungen+im+Umgang+mit+TUSTEP](http://www.tustep.wikispaces.com/Erleichterungen+im+Umgang+mit+TUSTEP).

## Dreimal TUSTEP

TUSTEP gibt es in dreierlei Form: 1. das klassische TUSTEP [ein System von Kommandos für die Grundfunktionen der Textdatenverarbeitung], 2. TUSCRIPT, eine moderne, hochentwickelte Skriptsprache, und 3. TXSTEP, das neue XML-Frontend von TUSTEP.<sup>11</sup>

Vergleich der drei Notationen anhand eines Beispiels:

Eine Textdatei soll in eine neue Datei kopiert werden. Dabei sollen nur diejenigen Datensätze übertragen werden, in denen die Zeichenfolgen »silber« und »gold«, aber nicht »silbern« vorkommen. Die übrigen werden ignoriert.<sup>12</sup>

```

*****  ——— Dateianfang ———
1  #AN,psalm.tf
2  #DA,psalmziel.tf,SEQ-AP
3
4  #KOPIERE,QU=psalm.tf,ZI=psalmziel.tf,LO=+,PA=*
5  ZF+      |silber|gold|
6  ZF+      ||silbern||
7  *EOF
*****  ——— Dateiende ———

```

Abb.: Umsetzung mit TUSTEP-#KOPIERE (= parametergesteuert)

```

*****  ——— Dateianfang ———
1  $$ MODE TUSCRIPT, {}
2
3  SET status = OPEN ("psalm.tf", READ, -STD-)
4  SET status = CREATE ("psalmziel.tf", SEQ-E, -STD-)
5
6  ACCESS q: READ      /RECORDS "psalm.tf" sn.zn/un, txt
7  ACCESS z: WRITE/ERASE/RECORDS "psalmziel.tf" sn.zn/un, txt
8
9  LOOP/999999
10     READ/NEXT/EXIT q
11     IF (txt .CT. "|silber|gold||silbern||") THEN
12         WRITE z
13     ENDIF
14 ENDLOOP
15 ENDACCESS q
16 ENDACCESS z
*****  ——— Dateiende ———

```

Abb.: Umsetzung mit TUSCRIPT

<sup>11</sup> <http://itug.de/tustep.html>.

<sup>12</sup> Beispielskripte erstellt nach: <http://tustep.wikispaces.com/TUSTEP+-+Kopieren>, <http://tustep.wikispaces.com/TUSCRIPT+-+Kopieren>, <http://tustep.wikispaces.com/Kopieren%20omit%20TXSTEP>.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <script xmlns="http://www.xstep.org" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
3   <variables>
4     <!-- Definition einer Variablen als execution-file. Darin werden die TUSTEP-Kommandos vor der Ausführung gespeichert. -->
5     <variable type="execution-file" name="exec">file:///C:/txstep/exec.tu</variable>
6     <!-- Definition der TXSTEP-Variablen "source", die auf die Ursprungsdatei verweist. -->
7     <variable type="permanent-file" name="source">file:///C:/txstep/psalm.tf</variable>
8     <!-- Definition der TXSTEP-Variablen "dest", die auf die Zieldatei verweist (diese wird gleich angelegt bzw. zuerst gelöscht,
9       falls sie schon existiert). -->
10    <variable type="permanent-file" name="dest" option="erase">file:///C:/txstep/psalmziel.tf</variable>
11  </variables>
12  <!-- Kopieren der Daten in "source" nach "dest", mode="-" bedeutet, dass die Sätze beim Kopieren nicht neu nummeriert werden. -->
13  <transform source="$source" destination="$dest" mode="-">
14    <pass>
15      <examine-text>
16        <select>
17          <text-contains>
18            <search-table>
19              <exception-string>silbern</exception-string>
20              <search-string>silber</search-string>
21              <search-string>gold</search-string>
22            </search-table>
23          </text-contains>
24        </select>
25      </examine-text>
26    </pass>
27  </transform>
28 </script>

```

Abb.: Umsetzung mit TXSTEP

## TUSTEP: Stärken/Schwächen

Wie dargestellt gehen grundlegende Programmprinzipien von TUSTEP auf die Anfangszeit der geisteswissenschaftlichen Datenverarbeitung zurück. Hieraus folgen auch verschiedene Stärken und Schwächen des Programmpakets, die im Folgenden skizziert werden sollen.

### Stärken

#### a) Funktionen/Anwendungsmöglichkeiten

- umfassende Einsatzmöglichkeiten für Textdatenverarbeitung (keine Spezialisierung auf ein einzelnes Anwendungsfeld)
- Konvertierung und Modifikation von Textdaten (ähnlich: Python, Perl, Tcl)
- weitgehend automatisch erzeugter, typographisch hochwertiger Satz (mit teils großen Unterschieden: LaTeX, InDesign)
- Vergleich von (un-)strukturierten Textdaten (ähnlich: CollateX)
- Einbindung in andere Programme/Programmiersprachen ist möglich<sup>13</sup>

<sup>13</sup> Z.B. Java- oder Tcl/Tk-GUI zur Abfrage von Benutzereingaben, TUSTEP-Verarbeitung und Aufruf des Ergebnisses im Browser.

- Verarbeitung von Texten in *nicht-lateinischen* Alphabeten: Griechisch, Koptisch, Hebräisch, Syrisch, Arabisch, Russisch/Kyrillisch, Altkirchenslavisch, Phonetisch mittels Transliteration oder per Eingabe mithilfe einer Maske im Browser aus UTF-8-Text
- Stabilität und Ausgereiftheit aufgrund der langen Entwicklungs- und Evaluationszeit; besonders interessant für langfristige Vorhaben sowie mit Blick auf die Nachverwendbarkeit von Skripten
- Verarbeitungsgeschwindigkeit<sup>14</sup>
- Verarbeitung diverser Dateiformate (z.B. Import/Export von RTF-Daten nach MS-Word)
- Verarbeitung unstrukturierter oder nicht-wohlgeformter Textdaten<sup>15</sup>
- Verarbeitung sehr großer Daten (im Editor < 2GB, Maximalgröße für die Verarbeitung von Fremddatei nur durch die Grenzen auf Betriebssystemebene limitiert = x TB)
- kurze Wege zu den Entwicklern
- Community mit einer Mailingliste, auf der in aller Regel sehr schnell verschiedene Problemlösungsansätze vorgeschlagen werden
- open source (seit 2011: Revised BSD License)

Ein Alleinstellungsmerkmal von TUSTEP ist der Umfang an Nutzungsmöglichkeiten. Für beinahe alle textbezogenen Aufgaben hält das Programmpaket eine Lösung bzw. die Bausteine zur Konstruktion derselben bereit. Konkurrenzprodukte wie LaTeX oder CollateX hingegen haben in aller Regel ein spezifisches Anwendungsgebiet (hier nämlich den Textsatz). TUSTEP kennt grundsätzlich keine solche Fokussierung bzw. Einschränkung auf ein bestimmtes Gebiet der Textdatenverarbeitung, auch wenn sich trotz allem Schwerpunktverwendungen ausgebildet haben und das Programmpaket in einigen

---

<sup>14</sup> »All dies realisiert TUSTEP in einem atemberaubenden Tempo: So haben wir beim Satz identischer Texte mit verschiedenen Satzsystemen auf vergleichbaren Rechnern eine um den Faktor 5 bis 10 höhere Performance von TUSTEP gegenüber etwa 3B2 oder TeX gemessen.«, (pagina GmbH, <http://www.pagina-online.de/xml-hintergruende/pagina-das-kompndium/themenkomplex-iii-satz/tustep/>). Die Geschwindigkeit geht maßgeblich auf zwei Faktoren zurück; 1. den Zwang zur hochperformanten Programmierung in den Zeiten schwacher Hardware ab den 1960er Jahren und 2. die Verwendung von bis heute sehr leistungsfähigen Programmiersprachen (C und FORTRAN). Zum ersten Aspekt vgl. auch Wilhelm Ott, *Et facta test lux*. Die Anfänge des TUSTEP-Satzprogramms, in: Alexandra Geissler, Matthias Schneider (Hrsg.): *Zwischen septem artes liberales und septem artes digitales*. Festschrift gewidmet Michael Trauth zum 65. Geburtstag am 10. Mai 2015, Marburg 2016, S. 227–249.

<sup>15</sup> Diese Funktionalität ist z.B. bei der schrittweise erfolgenden XML-Auszeichnung im Rahmen von Retrodigitalisierungen oder bei Textdatenkonvertierungen sehr nützlich, wenn die Codierung insgesamt noch nicht wohlgeformt ist, aber bestimmte Tags bereits auf ihre Paarigkeit hin überprüft werden sollen.

Bereichen stärker aufgestellt ist als in anderen. Letzteres gilt für das eingeschränkte mathematische Repertoire. Derzeit sind beispielsweise keine TUSTEP-eigenen Funktionen zur Berechnung von Logarithmen oder Fakultäten implementiert. Werden solche Berechnungen benötigt, besteht die Möglichkeit, diese mithilfe einer anderen Programmiersprache (z.B. C oder Python) auszuführen und die Ergebnisse mit TUSTEP weiterzuverarbeiten.<sup>16</sup> Bedenkt man die beschriebene Funktionsfülle, relativiert sich auch die anfangs sehr hoch scheinende Einstiegshürde und der hierzu notwendige Zeitaufwand. Beherrscht man einmal die TUSTEP-Syntaxphilosophie sowie einige Grundfunktionen und bringt im besten Falle auch ein wenig Experimentierfreude mit, erschließen sich weitere Funktionen in Abhängigkeit vom wachsenden individuellen Bedarf schrittweise.

---

### Schwächen

Mit den Programmprinzipien und der Historie von TUSTEP gehen einige Besonderheiten einher, die für Neueinsteiger häufig ungewohnt oder umständlich wirken können. Hierzu zählt die sehr reduzierte Benutzeroberfläche des Editors und die syntax-basierte Arbeitsweise anstelle eines eher geläufigen ›Zusammenklickens‹ von Funktionen. Die Benutzeroberfläche kann durch eigene Makros ausgebaut und bis zu einem gewissen Grade auch für eine weitergehende Mausbenutzung optimiert werden. Darüber hinaus sollte die syntax-basierte Arbeitsweise nicht unterschätzt werden, bietet sie doch erhebliche Geschwindigkeitsvorteile, weshalb sie in Kreisen professioneller Anwender wie Systemadministratoren aufgrund von Automatisierungs- und Optimierungsmöglichkeiten weit verbreitet und geschätzt ist.<sup>17</sup>

In der Folge stellen sich für den Neueinsteiger allerdings gewisse Herausforderungen ein. So muss die TUSTEP-Syntax und -Notation (Pattern Matching, Kommandos, Anweisungen) zumindest bis zu einem gewissen Grad erlernt werden, damit das Programmpaket effizient genutzt werden kann. Dies entspricht einer anfänglich steilen Lernkurve und wird durch die nicht immer selbst erklärende Bezeichnung von Funktionalitäten verstärkt.

- teilweise steile Lernkurve
- Anfangshürde
- nicht selbsterklärend

---

<sup>16</sup> S. für eine exemplarische Implementation <http://tustep.wikispaces.com/Externe+Programme+aufrufen>.

<sup>17</sup> Vgl. bspw. <https://help.ubuntu.com/community/KeyboardShortcuts>,  
[http://linuxcommand.org/learning\\_the\\_shell.php](http://linuxcommand.org/learning_the_shell.php).




- im Vergleich zu anderen Programmiersprachen und Satzsystemen ist die Dichte an Einführungs- und Hilfsmaterialien eingeschränkt



---

## Kurzbeschreibung der einzelnen Programmteile

---


In Anlehnung an die Kurzcharakteristiken im TUSTEP-Handbuch<sup>18</sup> sollen im Folgenden knappe Erläuterungen zu den Grundoperationen der Textverarbeitung zeigen, mit welchen Werkzeugen diese in TUSTEP umgesetzt werden können. Dabei liegt der Fokus auf den  parametergesteuerten Programmen.  Standardmakros werden weitgehend ausgeblendet, die vielfältigen Verwendungsmöglichkeiten von  TUSCRIPT nur punktuell benannt.

---

## Edieren von Texten

---

Für die Eingabe, Korrektur, Aufbereitung und das Durchsuchen/Analysieren von Texten (inkl. Programmcodes) kann der TUSTEP-Editor genutzt werden.

Eine alternative Möglichkeit zur (strukturierten) Dateneingabe über frei definierbare Eingabemasken (auch über den Webbrowser und z.B. als UTF-8) ist möglich. Hierzu sind die Möglichkeiten von  TUSCRIPT (`#MAKRO`) einzusetzen.

---

## Vergleich/Kollationieren

---


Unterschiedliche Versionen eines Textes (z.B. von mittelalterlichen Handschriften, hebräischen Traktatüberlieferungen oder von Programmcodes in verschiedenen Entwicklungsstadien) können mit `#VERGLEICHE` auf Abweichungen hin überprüft und die Unterschiede zwischen den Vergleichsobjekten dokumentiert sowie ggf. mithilfe von `#VAUFBEREITE` zeilensynoptisch aufbereitet werden.

Wurden Texte (z.B. unterschiedliche Textzeugen einer Handschrift) miteinander verglichen und die Ergebnisse aufbereitet, so können mittels `#KAUSFUEHRE` Korrekturen automatisch ausgeführt bzw. die Unterschiede in einem kritischen Apparat vermerkt werden.

---

## Automatische Textverarbeitung im engeren Sinne<sup>19</sup>

---

Texte können mit `#KOPIERE` bzw. `#MAKRO/ TUSCRIPT` analysiert, angereichert, umorganisiert, aufgeteilt, zusammengeführt, ergänzt, textbezogene Maßzahlen berechnet und die Ergebnisse für die Ausgabe in verschiedene Dateiformate vorbereitet werden. Auch die Konvertierung von Texten (z.B. Umcodierung einer älteren Datenbankstruktur in das Importformat einer Nachfolgedatenbank) sind hiermit zu erledigen.

Mit dem Programm `#EINFUEGE` können nicht nur Kürzel durch umfangreiche Text-

---

<sup>18</sup> Vgl. in der TUSTEP-Beschreibung Kapitel »Grundoperationen der Textdaten-Verarbeitung mit TUSTEP«.

<sup>19</sup> Z.B. Manipulation, Anreicherung, XML-Auszeichnung/Codierung.

bestandteile aufgelöst werden, sondern z.B. auch Datenanreicherungen (Einbringen von personenbezogenen Informationen wie einer GND-Nummer<sup>20</sup> in eine XML-Datei) oder die Auflösung von Kürzeln (z.B. Entities) bewerkstelligt werden. Querverweise zu verwalten und zu aktualisieren gewährleistet #NUMMERIERE.

---

### Register/Indizes vorbereiten und aufbereiten

---

Für die Erstellung von Registern oder Indizes ist es zunächst notwendig, aus den zugrunde liegenden Texten die notwendigen Informationen zu extrahieren, sie ggf. zu ändern, zu ergänzen oder zu verkürzen und für die anschließende Sortierung Sortierfelder vorzubereiten. Hierzu dient #RVORBEREITE. Wurden Register vorbereitet und sortiert, können mit #RAUFBEREITE weiterführende Operationen realisiert werden, so etwa die Zusammenfassung von Einträgen mit gleich lautendem Stichwort, die Ergänzung oder Modifikation von Einträgen und nicht zuletzt die Berechnung von absoluter und relativer Häufigkeit von Indexeinträgen.

---

### Sortierung vorbereiten

---

Eine äquivalente Vorgehensweise wird bei #SVORBEREITE durchschritten, wenn Texte ohne vorhergehende Informationsextraktion zur Sortierung aufbereitet werden sollen. Der anwendungsrelevante Unterschied besteht darin, dass mit den Registerprogrammen aus einem Fließtext Informationen extrahiert werden (auf welcher Druckseite oder in welchem Kapitel stehen vorher gekennzeichnete Personen- oder Ortsnamen; welche Wortformen [und wie viele Vorkommen davon] sind in einem Text enthalten). Die SVORBEREITE ist dagegen für die Vorbereitung von alphanumerischen oder sonstigen Sortierungen ganzer Textabschnitte oder -Bausteine (z.B. der Einträge in einer Bibliographie) zuständig. In TUSTEP lässt sich die Sortierung von Textdaten durch den Benutzer sehr umfassend parametrisieren. Aus den Angaben zu Sortierfeldern und Sortierschlüsseln und weiteren relevanten Angaben werden Sortierschlüssel berechnet, die anschließend dem eigentlichen Sortiervorgang zugrunde gelegt werden. Sinnvoll und notwendig ist eine individuelle Festlegung der Sortierreihenfolge u.a. bei gemischt-sprachlichen Texten (Anwendungsbeispiel: sollen Namen in kyrillischer Schrift blockweise zwischen den griechischen und den in lateinischer Schrift abgefassten eingetragen oder gemäß ihrer lateinischen Umschrift einsortiert werden?).

---

<sup>20</sup> »Die Gemeinsame Normdatei (GND) ist eine Normdatei für Personen, Körperschaften, Konferenzen, Geografika, Sachschlagwörter und Werktitel, die vor allem zur Katalogisierung von Literatur in Bibliotheken dient, zunehmend aber auch von Archiven, Museen, Projekten und in Webanwendungen genutzt wird.«  
[http://www.dnb.de/DE/Standardisierung/GND/gnd\\_node.html](http://www.dnb.de/DE/Standardisierung/GND/gnd_node.html).

---

## Sortieren und Einsetzen von Sortierergebnissen

---

Die Anordnung von Datensätzen in einer ansteigenden oder abfallenden Reihenfolge aufgrund zuvor definierter Kriterien übernimmt `#SORTIERE`.

Sortierte Datensätze können mit `#MISCHE` in andere Dateien eingebracht werden.

Die korrekte Sortierung ist mittels `#SPRUEFE` zu kontrollieren.

---

## Aufbereitung für den Druck

---

Der Ausgabe von Texten ohne Interpretation von TUSTEP-Steueranweisungen dient `#DVORBEREITE`. Dieses Kommando kann z.B. zur Ausgabe von Programmcodes als PDF genutzt werden.

`#FORMATIERE` bietet darüber hinaus die Möglichkeit, einfache Formatierungsanweisungen umsetzen zu lassen.

Speziell zur Aufbereitung und Druckvorbereitung von Formularen kann `#FAUFBEREITE` Verwendung finden.

---

## PDF-Herstellung, Aufbereitung für den Druck (= Textsatz)

---

Der typographisch hochwertigen Vorbereitung und Ausgabe dient das Programm `#SATZ` inklusive den dazugehörigen Makros für die Satzumgebung. Das Zielformat Postscript (= ›Mutterformat‹ von PDF) kann zur Weitergabe problemlos in PDF überführt werden. Möglich sind u.a. die Verwendung von lebenden Kolumnentiteln, Überschriften, Grundtext, Einschaltungen, Grafiken, Fußnoten, Marginalien und bis zu neun kritischen Apparaten. Einen reduzierten Funktionsumfang mit einsteigerfreundlichen Ergänzungen bietet das Standardmakro `#*SATZ` von Friedrich Seck, das aufgrund der schlanken und niedrigschwelligen Beschreibung für Neueinsteiger besonders geeignet ist.

---

## Datenaustausch (Importieren/Exportieren von Daten)

---

TUSTEP verwendet ein eigenes Dateiformat, in dem die Daten in Datensätze (= Records) eingeteilt sind und jedem Datensatz eine Satznummer mitgegeben wird. Diese Satznummer wird für Texte in eine Seiten- und eine Zeilennummer unterteilt und ermöglicht so eine einfache Gliederung der Texte analog zu derjenigen in gedruckten Medien. Zum Einlesen und Ausgeben von Fremddaten sind die Programme bzw. Standardmakros `#UMWANDLE`, `#*IMPORT` und `#*EXPORT` einzusetzen. `#UMWANDLE` dient dem Import von Fremddaten (auch HTML, XML, JSON, TEX) nach TUSTEP sowie dem Export aus TUSTEP in diese Formate. `#*IMPORT` wird benutzt, um RTF-Dateien, die in Microsoft Word, OpenOffice o.ä. Programmen erstellt wurden, nach TUSTEP zu

importieren. Hierbei sind einige Steuermöglichkeiten vorgesehen, so können überflüssige Informationen von der Übertragung in die TUSTEP-Datei ausgeschlossen werden. `#*EXPORT` speichert schließlich TUSTEP-Daten, die geeignete Angaben zur Formatierung und Textauszeichnung enthalten, in RTF ab, wodurch Texte, die in TUSTEP geschrieben, korrigiert oder angereichert wurden, beispielsweise in Microsoft Word formatiert und an andere Nutzer weitergegeben werden können.

---

### Dateiverwaltung

---

Zum Anmelden, Abmelden (= Erlangung bzw. Freigabe von Schreib- oder Leserechten), Einrichten, Umbenennen und Löschen von Dateien sind `#DATEI`, `#ANMELDE`, `#ABMELDE`, `#AENDERE`, `#LOESCHE` und `#TITEL` zu nutzen.<sup>21</sup>

---

### Archierung/Sicherung/Versionierung von Dateien

---

TUSTEP kann Daten auf ein virtuelles Magnetband sichern,<sup>22</sup> versionieren, ältere Dateizustände wiederherstellen, Informationen über die Inhalte der Banddatei abfragen u.ä.m. mit:

`#MBEINGABE`, `#MBAUSGABE`, `#MBKOPIERE`, `#MBINFORMIERE`, `#MBLABEL`,  
`#MBTEST`, `#*MBUPDATE`, `#DATEI` (Anlegen der gesamten Banddatei),  
`#RETTE`, `#HOLE`, `#LOESCHE` (der gesamten Banddatei), `#LISTE`

---

### Ablaufsteuerung/Skripting

---

Der Steuerung von Programmabläufen u.a. im Rahmen von Skripting, der Ausführung von eigenen TUSTEP-Programmroutinen dienen `#TUE` und `#MAKRO/⌘TUSCRIPT` .

---

<sup>21</sup> Vgl. zur Notwendigkeit des Anmeldens, Abmeldens usw. das Kapitel »Befehlssyntax«.

<sup>22</sup> Vgl. als Erläuterung zu den Banddateien (= virtuellen Magnetbändern) den Eintrag ⌘Banddatei im Glossar.

## Erste Schritte in TUSTEP

## Erste Schritte

### TUSTEP installieren vs. vom USB-Stick nutzen

---

Bevor Sie mit der Benutzung von TUSTEP starten können, müssen Sie das Programmpaket installieren. Für die Betriebssysteme Linux, macOS und Windows stehen Installationsdateien bereit:

<http://www.tustep.uni-tuebingen.de/download.php>

- Zur Erläuterung der Installation und Konfiguration (insb. unter Linux und macOS) steht eine ausführliche Anleitung zur Verfügung:

<http://www.tustep.uni-tuebingen.de/inst.html>

- Mit dem Standardmakro `#*TUSTEP2STICK` kann TUSTEP von einer bereits installierten Programminstanz aus auf einem USB-Stick installiert werden.<sup>23</sup> Anschließend kann an einem anderen Rechner TUSTEP vom Stick aus benutzt werden, ohne dass eine lokale Installation notwendig wäre.
- Installieren Sie unter Windows bitte außerdem die Programme
  - Ghostscript:  
<http://www.ghostscript.com> und
  - Ghostview:
  - Diese sind notwendig, um die Ergebnisse von Satzläufen, nämlich PostScript-Dateien anzuzeigen und in PDFs zu konvertieren. Unter Linux und macOS werden die entsprechenden Tools bereits vom Betriebssystem mitgeliefert.
- Eine übersichtliche und kompakte Anleitung zur Installation und Einrichtung von TUSTEP auf allen drei Plattformen (Linux, macOS, Windows) bietet die Beschreibung »TUSTEP – Installation, Konfiguration, Aufruf«. Sie kann nach der Installation über die Kommandozeile mit folgendem Kommando aufgerufen werden:

```
#*ZEBE, CONFIG
```

Alternativ kann die Anleitung auch bereits vor der Installation (zu empfehlen insbesondere für Installationen unter Linux) von der TUSTEP-Homepage heruntergeladen werden (s.o.).

- Für erste Schritte im Programmpaket s. TUSTEP-Wiki<sup>24</sup>, die Einführungsdokumente Heidi Hein/Matthias Schneider (2014), Matthias Schneider/René Tobner (2014) sowie Tobias Ott (2017).<sup>25</sup>

---

<sup>23</sup> <http://www.tustep.wikispaces.com/Grundlagen+TUSTEP2STICK>.

<sup>24</sup> <http://tustep.wikispaces.com/TUSTEP+-+Erste+Schritte>.

<sup>25</sup> <http://tustep.wikispaces.com/Ein%C%BChrungen>.

## Einrichten eines TUSTEP-Projekts

Nach der Installation von TUSTEP können verschiedene sogenannte *Projekte* eingerichtet werden. Theoretisch können in einem TUSTEP-Projekt Dateien zu ganz unterschiedlichen Aufgaben liegen und bearbeitet werden (epigraphische Transkription, stilometrische Untersuchungen, Abfassung der Dissertation). Es hat sich allerdings bewährt, für verschiedene Arbeitsbereiche ein jeweils eigenes TUSTEP-Projekt anzulegen und die dazugehörigen Daten im jeweiligen Projektverzeichnis getrennt zu verwalten. Dies schafft Übersicht, erleichtert die Weitergabe der Daten an Kollegen, vereinfacht den Einsatz spezifischer Backup-Strategien und hält Ordnung in der Datenspeicherung. Ein ähnliches Prinzip verfolgen Programme wie Eclipse<sup>26</sup> mit unterschiedlichen Workspaces.

Grundsätzlich können Sie Ihre Projektverzeichnisse an beliebigen Orten Ihrer Festplatte abspeichern. Bewährt hat sich allerdings eine Ordnerstruktur, wie sie auf der nächsten Seite abgebildet ist.

---

<sup>26</sup> <http://www.eclipse.org/>.

```
....\proj\  
.....\diss\  
.....\kurs18tr\  
.....\p\  
.....\scr\  
.....\transkr\
```

Abb.: Schema Ordnerstruktur

Konkret könnten also die Pfade lauten:

- D:\Users\schneiderm\proj\<sup>27</sup> Verzeichnis, in dem alle TUSTEP-Projektverzeichnisse gespeichert werden
- D:\Users\schneiderm\proj\diss\ Verzeichnis mit Daten für ein Dissertationsvorhaben
- D:\Users\schneiderm\proj\kurs18tr\ Verzeichnis mit den Daten für einen TUSTEP-Kurs
- D:\Users\schneiderm\proj\p\ Verzeichnis, in dem Programme und Konfigurationsdateien hinterlegt werden, die nicht projektspezifisch, sondern von allgemeinem Interesse sind
- D:\Users\schneiderm\proj\scr\ Verzeichnis für temporäre Dateien (= Scratch-Dateien )
- D:\Users\schneiderm\proj\transkr\ Verzeichnis für ein Transkriptionsprojekt

Wenn Sie ihre Projektverzeichnisse so wie vorgeschlagen einrichten, hat dies den großen Vorteil, dass Sie sehr einfach aus den jeweiligen TUSTEP-Sitzungen auf Dateien zugreifen können, die in einem der anderen Projektverzeichnisse liegen. Hierzu ein Anwendungsbeispiel aus der Praxis:

Sie schreiben ein Programm, das mehrere RTF-Dateien nach einem für Ihre Arbeit sinnvollen Satz von Einstellungen importiert und zusammenfasst. Dieses Programm können Sie dann im Ordner \p\ ablegen und von jedem anderen Projekt aus auf diese Programmroutine zugreifen, wenn Sie RTF-Dateien importieren möchten. Sie können theoretisch auch einfach die Datei kopieren und in jedem Projekt eine Kopie ablegen, in dem das Programm benötigt wird. Dies hat allerdings einen gravierenden Nachteil,

---

<sup>27</sup> Unter Linux und macOS entsprechend; also etwa /home/schneiderm/proj/.



wenn Sie Korrekturen oder Verbesserungen an diesem wichtigen Programm vornehmen. In dem Fall müssen die Änderungen nicht an einer zentralen Stelle, sondern in diversen Projekten vorgenommen werden. Aufgrund der immanenten Vorteile der beschriebenen Struktur gehen wir im vorliegenden Skript davon aus, dass Sie ihre Projekte nach dem vorgeschlagenen Muster einrichten.

Legen Sie dazu bitte eine Ordnerstruktur wie oben beschrieben an. Hier sollte mindestens ein »p-Verzeichnis« und ein Verzeichnis für die Kursmaterialien enthalten sein; gerne auch schon ein Verzeichnis für ein bestimmtes anderes Projekt, das Sie mit TUSTEP bearbeiten möchten.

Beachten Sie bitte, dass es bei der Ordnerstruktur ein paar Regeln zu beachten gibt:

Der Ordner, in dem Sie die Daten für Ihr Projekte unmittelbar ablegen (z.B. `kurs18tr`), fungiert zugleich als »Projektname« innerhalb von TUSTEP. Im einfachsten Fall besteht der Projektname aus 1–8 Zeichen (Buchstaben, Ziffern, Minuszeichen und »\_«). Der Projektname muss mit einem Buchstaben beginnen und darf nicht mit einem Minus oder einem »\_« enden.

Der Projektname `TUSTEP` ist für TUSTEP reserviert und darf nicht vom Benutzer verwendet werden.

Der restliche Pfad zu den Projektdateien oberhalb des Projekts wird als »Träger« bezeichnet.

Dateinamen dürfen in TUSTEP 1–12 Zeichen lang sein, aus Buchstaben, Ziffern, Minuszeichen und »\_« bestehen, müssen mit einem Buchstaben beginnen und dürfen nicht auf einem »\_« oder einem Minuszeichen enden. Die Dateien können, müssen aber nicht, mit einer Dateiendung (= Extension) versehen werden. Diese darf in TUSTEP 1–4 Zeichen umfassen und aus Buchstaben, Ziffern, Minuszeichen und »\_« zusammengesetzt sein. Wie Dateinamen auch muss sie auf einen Buchstaben oder eine Ziffer enden. Es dürfen keine Dateinamen verwendet werden, die `TUSTEP.xxx` lauten, da diese für TUSTEP reserviert sind. Es hat sich bewährt, TUSTEP-Textdateien mit der Endung »tf«, Dateien mit parametergesteuerten Programmen mit der Endung »p« und Makrodateien mit »m« abzuspeichern. Diese Konventionen müssen allerdings nicht vom Benutzer eingehalten werden. Ebenso gut sind »prog«, »makr«, »text« u.ä.m. denkbar.

Wenn Sie TUSTEP zum ersten Mal aufrufen (s. hierzu oben die Hinweise in der Beschreibung zur Installation und Einrichtung), ist noch *kein* TUSTEP-Projekt eingerichtet. Sie erhalten dann beim Programmaufruf ein leeres Fenster, das ähnlich wie das folgende aussieht, allerdings noch keine Eintragungen enthält:



Abb.: Screenshot #\*DESI

Was Sie sehen, ist die Ausgabe des Standardmakros #\*DESI (= DEfiniere Sitzung), mit dem Sie Ihre TUSTEP-Projekte anlegen, ändern und löschen können. Richten Sie bitte auf Ihrem Rechner eine Sitzung für diesen Kurs ein, ähnlich dem oben gezeigten Muster. Die Variablen, denen Sie Werte zuweisen, haben folgende Bedeutung:

- TUSTEP-Sitzung    Name der Sitzung in #\*DESI
- TUSTEP\_TTL        Überschrift, die im TUSTEP-Sitzungsfenster angezeigt wird
- TUSTEP\_DSK        Pfad zum TUSTEP-Träger
- TUSTEP\_SCR        Pfad zum TUSTEP-Träger für temporäre Dateien (= Scratch-Dateien)
- TUSTEP\_PRJ        Name des TUSTEP-Projekts, das unterhalb des Trägers ggf. angelegt wird
- TUSTEP\_USR        Benutzerkürzel
- TUSTEP\_NAM        Name der TUSTEP-Sitzung; kann mit TUSTEP\_PRJ identisch sein

- 
- TUSTEP\_MEM      Sitzungsnummer; kann im Desktopbetrieb i.d.R. zufällig belegt werden, relevant für Serveranwendungen

## Grundlagen der Benutzung

Das Editorfenster umfasst folgende Teilbereiche:

- 1) Feld mit den Datensatznummern
- 2) Textfeld
- 3) Kopfzeile, in der angezeigt wird, welche Daten ediert werden bzw. in der eine vom Nutzer definierte Makroleiste angezeigt werden kann
- 4) Meldungszeile
- 5) Anweisungszeile mit Anzeige der aktuellen Satzposition und der Möglichkeit, Anweisungen einzugeben
- 6) Statuszeile mit Informationen zur Cursorposition
- 7) Einstellungsmöglichkeiten und Statusanzeigen

Wenn Sie Ihre TUSTEP-Sitzung zum ersten Mal aufrufen, stellen Sie einen grundlegenden Unterschied zu anderen aktuellen Softwareprodukten fest, der in der Hauptsache darin besteht, dass die üblicherweise vorhandenen Schaltflächen fehlen. Menüpunkte wie *Datei*, *Bearbeiten*, *Anzeige*, *Hilfe* sucht man auf der TUSTEP-Kommandoebene vergeblich. Stattdessen setzt TUSTEP konsequent auf eine reduzierte graphische Oberfläche (*graphical user interface*; = GUI), um auf allen drei Plattformen (Linux, macOS, Windows) einheitlich lauffähig zu sein.<sup>28</sup> Anstelle von graphischen Bedienelementen setzt TUSTEP auf Kommandos und Anweisungen zur Steuerung der Programmfunktionen. Statt in einem Menü »Datei« und dann »Exportieren als XML« auszuwählen, würde man in TUSTEP beispielsweise das Kommando `#UMWANDLE, datei.tf, datei.xml, CO=UTF8, LO=+` verwenden. Dies mag zunächst umständlich erscheinen, eröffnet allerdings durch verschiedene Möglichkeiten zur Parametrisierung und Anpassung an Nutzerbedürfnisse eine wesentlich höhere Mächtigkeit als bei vielen anderen Programmen.<sup>29</sup>

Zwei kurze Zeige-Anweisungen sollen nachstehend die Benutzung der Syntax auf Anweisungsebene (= im TUSTEP-Editor) illustrieren. Die Ergebnisse der Zeige-Anweisungen sehen Sie auf den Screenshots, die auf der nächsten Seite abgebildet sind.

---

<sup>28</sup> Ausnahmen sind auf Unterschiede im Betriebssystem zurückzuführen. So gestaltet sich die Interaktion zwischen der TUSTEP-Zwischenablage und der Windows-Zwischenablage einfacher als mit dem Linux-Clipboard und dem Pasteboard unter macOS. Hierzu können die Standardmakros `#*CB` bzw. `#*PB` genutzt werden. Die Spezifikation Import bzw. Export definiert, ob Inhalte in TUSTEP-Zwischenablage importiert oder in die betriebssystemseitige Zwischenablage geschrieben werden soll (s.u.).

<sup>29</sup> Vgl. für Skizzen einer Anpassung i.S. von Barrierefreiheit <http://tustep.wikispaces.com/Präsentationen%20zu%20TUSTEP%20und%20TUSTEP-Projekten>.

Exemplarische Anweisung: *Zeige **nur** diejenigen Datensätzen an, in denen die Zeichenfolge ›Frosch‹ steht.*

`ZN,,,|Frosch|`

*Zeige alle Datensätze mit der Zeichenfolge ›Frosch‹ in ihrer Umgebung an.*

`ZU,,,|Frosch|`

The screenshot shows the ZeigeNur search tool interface. The title bar indicates the file is 'KURS18TR\*FROSCH\_REG.TF'. The search results are displayed in a list on the left, with line numbers 1.1 through 1.95. The main text area shows the XML content with the search results highlighted. The search criteria are '18 Zchflg. gefunden', '18 Sätze gefunden', and '102 durchsucht'. The search criteria are '\*1.95 Gib Anweisung >'. The search criteria are '\*\*;01', 'SPLIT', 'INSERT', and 'MOUSE'.

```

1.1 <!-- Modifizierter Froschkönigtext, angereichertes XML, Übungsdatei -->
1.21 <head>Der <name xml:id="Prinz">Froschkönig</name> oder der eiserne <name xml:id="Diener
Heinrich">Heinrich</name>.</head>
1.31 Perlen und was es auf der Welt nur wär.</q> Wie sie so klagte, steckte ein Frosch seinen Kopf
1.33 sie, du garstiger Frosch, was kannst du mir helfen! meine goldne <sportGeraet>Kugel</sportGeraet> ist
mir in den
1.34 Brunnen gefallen.</q> Der Frosch sprach: <q>deine Perlen, deine <w attr="edel">Edel</w>steine und deine
Kleider,
1.38 <name xml:id="Prinzessin">Königstochter</name> dachte, was schwätzt der einfältige Frosch wohl, der muß
doch in seinem Wasser
1.41 seyn.</q> Der Frosch steckte seinen Kopf unter das Wasser und tauchte hinab, es dauerte auch
1.45 damit nach Haus eilte. Der Frosch rief ihr nach: <q>warte, <name
xml:id="Prinzessin">Königstochter</name>, und nimm mich mit,
1.50 war es der Frosch, an den sie nicht mehr gedacht hatte; ganz erschrocken warf sie die Thüre
1.52 und sagte: <q>warum fürchtest du dich?</q> <q>Da draußen ist ein garstiger Frosch, sagte sie, der
1.65 <p>Der <name xml:id="König">König</name> sagte: <q>was du versprochen hast, mußst du halten, geh und mach
dem Frosch die
1.66 Thüre auf.</q> Sie gehorchte und der Frosch hüpfte herein, und ihr auf dem Fuß immer nach, bis
1.69 der Frosch oben war, sprach er: <q>nun schieb dein goldenes Tellerlein näher, ich will mit dir
fürchtete sich vor dem kalten Frosch, sie getraute sich nicht ihn anzurühren und nun sollte
1.73 Sie packte den Frosch mit zwei Fingern und trug ihn hinauf in ihre Kammer, legte sich ins
1.77 mich in Ruh lassen, du garstiger Frosch!</q> </p>
1.79 <p>Aber der Frosch fiel nicht todt herunter, sondern wie er herab auf den Boden kam, da wars
1.80 <br/>als ihr eine Fretsche (Frosch) wast.</q> (wart) <w
1.95 attr="edel">Edel</w>steinmanufaktur <w attr="edel">Edel</w>stein <w
attr="edel">Edel</w>steinschleiferei

```

Abb.: Ergebnis der Suche mit ZeigeNur

The screenshot shows the ZeigeUmgebung search tool interface. The title bar indicates the file is 'KURS18TR\*FROSCH\_REG.TF'. The search results are displayed in a list on the left, with line numbers 1.6 through 1.30. The main text area shows the XML content with the search results highlighted. The search criteria are '2 Zchflg. gefunden', '2 Sätze gefunden', and '21 durchsucht'. The search criteria are '\*1.21 Gib Anweisung >'. The search criteria are '\*\*;01', 'SPLIT', 'INSERT', and 'MOUSE'.

```

1.6 <teiHeader>
1.7 <fileDesc>
1.8 <titleStmt>
1.9 <title/>
1.10 </titleStmt>
1.11 <publicationStmt>
1.12 <p>KHM, 1. Aufl. (1812/15) I: 15.</p>
1.13 </publicationStmt>
1.14 <sourceDesc>
1.15 <p/>
1.16 </sourceDesc>
1.17 </fileDesc>
1.18 </teiHeader>
1.19 <text n="1">
1.20 <body>
1.21 <head>Der <name xml:id="Prinz">Froschkönig</name> oder der eiserne <name xml:id="Diener
Heinrich">Heinrich</name>.</head>
1.22 <p>Es war einmal eine <name xml:id="Prinzessin">Königstochter</name>, die ging hinaus in den Wald und
setzte sich an einen
1.23 kühlen Brunnen. Sie hatte eine goldene <sportGeraet>Kugel</sportGeraet>, die war ihr liebstes Spielwerk,
die warf sie
1.24 in die Höhe und fing sie wieder in der Luft und hatte ihre Lust daran. Einmal war die
<sportGeraet>Kugel</sportGeraet>
1.25 gar hoch geflogen, sie hatte die Hand schon ausgestreckt und die Finger gekrümmt, um sie
wieder zu fangen, da schlug sie neben vorbei auf die Erde, rollte und rollte und geradezu in
1.26 das Wasser hinein. </p>
1.27 <p>Die <name xml:id="Prinzessin">Königstochter</name> blickte ihr erschrocken nach, der Brunnen war aber
so tief, daß kein
1.28 Grund zu sehen war. Da fing sie an jämmerlich zu weinen und zu klagen: <q>ach! Wenn ich meine
<sportGeraet>Kugel</sportGeraet> wieder hätte, da wollt ich alles darum geben, meine Kleider, meine <w
1.30 attr="edel">Edel</w>steine, meine

```

Abb.: Ergebnis der Suche mit ZeigeUmgebung

- 
- Die Vorteile einer Syntax besteht in der Präzision, Reliabilität, Geschwindigkeit (Bedienung & Programmausführung) sowie der transparenten Nachvollziehbarkeit und Dokumentierbarkeit. Die Befehle können – wenn sie in Form eines Programms niedergeschrieben werden – beliebig oft wiederholt, angepasst, neu ausgeführt und weitergegeben werden. Müsste eine gleichwertig komplexe und umfangreiche Bearbeitung hingegen über graphische Benutzermenüs ausgeführt werden, müsste bei jedem neuen Vorgang (bei neuen Daten oder wegen Modifikation der Bearbeitung) exakt die gleiche Reihenfolge der Bearbeitungsschritte eingehalten werden, manuelle Fehler vermieden und hoher Dokumentationsaufwand betrieben werden. Bei einer Syntax- und Skript-basierten Verarbeitung fällt der Aufwand entweder weg oder zumindest geringer aus.
  - Ein klarer Nachteil der kargen Oberfläche und der befehls-basierten Arbeitsweise im Unterschied zu einer umfangreichen graphischen Menüstruktur mit Benutzerführung liegt im Lernaufwand, den der Benutzer zu betreiben hat, um die für ihn relevanten Kommandos und Anweisungen zu memorieren.

## Befehlssyntax

TUSTEP arbeitet wie bereits erwähnt befehls-basiert und unterscheidet zwei Arten von Befehlen: *Kommandos* auf Kommandoebene (Gib Kommando>) und *Anweisungen* im Editor (Gib Anweisung>). Einzelne Kommandos oder ganze Kommandofolgen können in Form von Skripten im Editor in eine TUSTEP-Datei geschrieben werden, die anschließend vom Programmsystem abgearbeitet wird (Beispiele hierzu folgen). Ein solches Skript könnte folgende Aufgaben erledigen, um die stilistische Korrektur eines Aufsatzes zu unterstützen: *Einrichten einer neuen Datei mit dem Namen »register1.tf«, Anmelden der Datei »aufsatz.tf« an, Ersteller eines Wortformenregisters inklusive Angaben zur absoluten Häufigkeit der Wortformen.* Bsp.:

```
#DA,register1.tf,SEQ-AP
#AN,LE=aufsatz.tf
#= Der Text wird in einzelne Registereinträge zerlegt, es werden
#= keine Referenzen gebildet (z.B. Seite/Zeile
#= des Registereintrags; entspr. MO=-).
#= "-STD-" steht hier für eine der Standard-Dateien, die
#= TUSTEP einrichtet und die hier zur Verarbeitung genutzt
#= werden kann.
#RV,QU=aufsatz.tf,ZI=-STD-,MO=-,LO=+,PA=*
* Trennzeichen (Welche Zeichen begrenzen Registereinträge?
* Im vorliegenden Fall = Worttrennzeichen, da alle Wörter
* extrahiert werden sollen.)
TR      | |,|;|.|:|!|\?|<*>|
* Länge des Sortierschlüssels. Orientiert sich an der Länge
* des längsten zu sortierenden Registereintrags.
SSL      30
*EOF
#= Die Registereinträge werden alphabetisch sortiert.
#SO,QU=-STD-,ZI=-STD-,SO=1+30,LO=+
#= Sortierergebnis für die Ausgabe aufbereiten, zusammenfassen
#= und absolute Häufigkeit der Registereinträge ausgeben.
#RA,QU=-STD-,ZI=register1.tf,MO=-,LO=+,PA=*,PROT=-
AH      1
SSL      30
*EOF
```

Kommandos beginnen mit einem Nummerzeichen (#) und dem Kommandonamen, der



nach dem zweiten Buchstaben abgekürzt werden kann ( $\#KOPIERE = \#KOP = \#KO$ ).<sup>30</sup> Anschließend folgen die – je nach Kommando – obligatorischen oder optionalen Spezifikationen zum Kommando, die ohne Nennung des Spezifikationsnamens in einer festgelegten Reihenfolge angegeben werden können (z.B. Quell- und Zieldatei). Alternativ können diese auch in einer beliebigen Reihenfolge genannt werden, wenn der vollständige oder abgekürzte Spezifikationsname angegeben wird. Die Standardreihenfolge der Spezifikationen ergibt sich aus der Beschreibung im TUSTEP-Handbuch. Beim Kommando  $\#BEEENDE$  wird die Spezifikation  $MODUS$  als erstes, die Spezifikation  $WISCHEN$  als zweites erwartet. Gibt der Benutzer also ein:

$\#BE,MO=+,WI=-$  so wird die TUSTEP-Sitzung zwar beendet (und nicht nur unterbrochen, was mit der Eingabe der  $ESC$ -Taste auf Kommandoebene erreicht wird), die Daten in den temporären Dateien werden allerdings nicht überschrieben.

$\#BE,+, -$  hätte die gleiche Wirkung. Mit der Angabe  $+$  für die Spezifikation  $WISCHEN$  hingegen würden die Daten in den temporären Dateien (= Scratch-Dateien) aus Gründen des Datenschutzes überschrieben. Wird eine TUSTEP-Sitzung nur unterbrochen, aber nicht beendet, was der Regelfall in der täglichen Arbeit ist, so bleiben alle temporären Dateien, die Einstellungen sowie die gemerkten Anweisungen erhalten.

Einige weitere Beispiele sollen die Möglichkeiten der Formulierung illustrieren: In den folgenden gleichwertigen Beispielaufrufen wird eine Datei  $text1.tf$  nach  $text2.tf$  kopiert und der möglicherweise vorhandene Inhalt von  $text2.tf$  wird überschrieben (= der Dateiinhalt wird gelöscht):

```
#KOPIERE, QUELLE=text1.tf, ZIEL=text2.tf, MODUS=-STD-, LOESCHEN=+
#KO, text1.tf, text2.tf, , +
#KO, text1.tf, text2.tf, LO=+
```

$\#KO, quelle, ziel, +, -$  kopiert den Inhalt von Quelle nach Ziel, überschreibt das Ziel nicht und nummeriert die neu eingefügten Datensätze in »ziel« neu.

### Datensatznummerierung

In TUSTEP erhält jeder Datensatz einer Datei automatisch eine (Daten-)Satznummer. Die besteht mindestens aus einer Seitennummer und einer Zeilennummer, kann aber auch um eine Unterscheidungsnummer ergänzt werden: Seitenzahl.Zeilenzahl/Unterscheidungsnummer (z.B. 4.52 oder 19.12/3). Die Datensatznummern können so verwendet werden, um die Einteilung von Texten analog zum gedruckten Medium zu

<sup>30</sup> Das Nummernzeichen kann bei der Arbeit auf der Kommandoebene weggelassen werden, wird im Folgenden jedoch zur besseren visuellen Abgrenzung mitgeführt.

gestalten, so dass die Zeilen der S. 3 im gedruckten Buch die Datensatznummern 3.iff. erhalten (= Nummerierung im *Textmodus*).

Diese Nummerierung ist für Programm- oder Makrodateien nicht sinnvoll, daher wird im *Programmmodus* nur eine Zeilennummer sowie gegebenenfalls eine Unterscheidungsnummer genutzt: 3 oder 3/4. Die Programmmodus-Nummerierung kann auch für Texte genutzt werden, wenn kein Wert auf eine Seiten-Zeilen-Analogie gelegt wird.

Die Datensatznummerierung wird u.a. verwendet, um in den TUSTEP-Dateien zu navigieren, Such- und Austausch-Anweisungen auf bestimmte Bereiche einzugrenzen, Dateibestandteile auszuwählen, Referenzen für Register zu erstellen.<sup>31</sup>

---

## Dateiformate

---

☞ s. Skript von M. Trauth

In TUSTEP müssen Dateien vor dem Öffnen im Editor zum Lesen oder zum Schreiben (wenn die Datei auch bearbeitet werden soll) angemeldet werden. Die Berechtigung zum Lesen und Schreiben von Dateien können entweder über den Dateimanager oder über Kommandos erlangt werden:

### Dateimanager:

Lesen = Erlangen von Leserechten  
Schreiben = Erlangen von Schreibrechten

### Kommandoebene:

```
#AN,LE=datei.tf  
#AN,datei.tf (= ANMELDEN mit Leserechten)  
#AN,SCH=datei.tf  
#AN,,datei.tf (= ANMELDEN mit Schreibrechten)
```

Dieses konservative Vorgehen bei der Dateibearbeitung ist in mehrfacher Hinsicht vorteilhaft; so werden die Folgen allzu schneller Eingaben des Benutzers (z.B. Löschen oder Überschreiben) nicht unbedingt verhindert, aber zumindest gebremst, wodurch die Wahrscheinlichkeit, einen Fehler zu verhindern, größer wird. Dies kann allerdings nur dann gelingen, wenn der Benutzer sich auf die Unterscheidung von Lese- und Schreibrechten einlässt und nicht pauschal alle Dateien eines Projekts zum Schreiben anmeldet.

---

<sup>31</sup> Vgl. im TUSTEP-Handbuch vertiefend die Abschnitte »Einteilung der Daten in Sätze«, »Satznummern« sowie »Zugriff auf die Sätze«.

---

## Die wichtigsten Kommandos zur Dateiverwaltung

---

☞ s. Skript Michael Trauth

---

## Übungsaufgaben »Dateiverwaltung«

---

### Übungsdatei »bech7.tf«

1. Kopieren Sie die Übungsdatei »bech7.tf« in Ihr Projektverzeichnis und melden Sie sie auf der Kommandoebene zum Lesen an.
2. Richten Sie eine permanente Datei namens »ueb.tf« ein. Damit erhalten Sie automatisch Schreibrechte für die Datei, bis diese abgemeldet oder nur mit Lesezugriff neu angemeldet wird.
3. Kopieren Sie den Inhalt der Datei »bech7.tf« in die Datei »ueb.tf«.
4. Öffnen Sie die Datei »ueb.tf« im Editor.
5. Löschen Sie die Datei »ueb.tf« wieder.

Welche der folgenden Anweisungen führen zum mutmaßlich gewünschten Ergebnis, welche nicht?

```
#AN, , datei.tf  
#AN, , datei.tf, SEQ-AP  
#DA, datei.TF, SEQ-P  
#LO, datei.tf  
#LO, , datei.tf, P  
#DA, datei.tf
```

## TUSTEP-Editor

## TUSTEP-Editor

TUSTEP bringt einen eigenen Editor mit, mit dem sowohl Texte im engeren Sinne – also Aufsätze, Dissertationen, Editionen usw. – als auch Programme (Makro- oder Kommandofolgen) geschrieben und bearbeitet werden können.


**Tipp:** Es ist auch möglich, andere Editoren zu benutzen, um Texte zu schreiben und zu korrigieren, die z.B. mit TUSTEP gesetzt werden sollen. Auch TUSTEP-Programmcodes können in anderen Programmen geschrieben, anschließend konvertiert und dann von TUSTEP ausgeführt werden. Beispielsweise ist es denkbar, dass man Sublime Text, Notepad++ oder UltraEdit nutzt, um ein TUSCRIPT zu schreiben, das anschließend per `#UMWANDLE` nach TUSTEP konvertiert und dort ausgeführt wird. Dabei ist allerdings darauf zu achten, dass – zumindest bei der Erstellung parametergesteuerter Prozeduren – das für Parameter vorgesehene Format auch bezüglich der Spalteneinteilung eingehalten wird.



Abb.: Screenshot Editor mit zwei Makroleisten

## Eintragen, Korrigieren, Speichern von Daten

Der TUSTEP-Editor speichert automatisch Veränderungen am Inhalt einer geöffneten Datei u.a. bei folgenden Aktionen: Verschieben des Bildschirminhalts mit einer Anweisung (z.B. mit ZU, \* oder F5) oder dem Mauseisen/Cursor. Es gilt die Regel: Was aus dem sichtbaren Bereich heraus verschoben wurde, ist gespeichert. Eine weitere Möglichkeit ist die Tastenkombination SHIFT+RETURN. Auch bei der Eingabe oder beim Korrigieren werden gegebenenfalls Datensätze nach oben aus dem Textfeld hinaus geschoben und damit gespeichert.

TUSTEP bietet im Unterschied zu anderen IT-Programmen nicht die Möglichkeit, manuelle Veränderungen an Textinhalten in mehreren Schritten recht problemlos rückgängig zu machen. Dennoch stehen verschiedene Möglichkeiten zur Verfügung, um Änderungen zurückzunehmen:

- Funktion RESHOW, Aufruf: STRG+R: wiederholt die letzte Editor-Ausgabe und stellt die letzte Anzeige im Textfeld wieder her sowie verwirft alle Änderungen, die im

*sichtbaren* Bereich vorgenommen wurden, falls seit der letzten Editor-Ausgabe keine Änderungen in die Datei übertragen wurden. Nimmt keine Änderungen zurück, die sich mittlerweile außerhalb des sichtbaren Bereichs befinden.

- Funktion `RECALL_DEL`, Aufruf: `ALT+L`: zeigt die zuletzt im Textfeld gelöschten Datensätze an.<sup>32</sup> Diese können ausgewählt und wiederhergestellt werden. Problematisch kann diese Funktion dann sein, wenn zwischen dem Löschen und der Wiederherstellung die Datensatznummerierung geändert wurde und der gelöschte Datensatz (z.B. 1.3) mittlerweile einen anderen Inhalt hat. In diesem Fall wird der Benutzer gefragt, ob der bereits existierende Datensatz überschrieben werden soll. Will man die neuen Daten in 1.3 jedoch nicht verlieren, sollte man hier mit »Nein« antworten, die Daten von 1.3 in einen anderen Datensatz schreiben und dann erst die Wiederherstellung ausführen.
- `RECALL_MOD`, Aufruf: `ALT+K`: zeigt die Datensätze, die im Editor verändert wurden, mit dem Inhalt vor der Änderung an.<sup>33</sup> Wenn ein Satz wiederhergestellt werden soll, kann dieser mit der linken Maustaste angeklickt werden oder mit der Pfeiltaste ausgewählt und anschließend die Auswahl mit `RETURN` bestätigt werden.

### Tipps bei der Arbeit im Editor

Bei der Arbeit im TUSTEP-Editor stehen unterschiedliche Möglichkeiten zur Verfügung, um die Arbeit zu beschleunigen und insbesondere repetitive Aufgaben zu automatisieren, um neben einer Zeitersparnis auch eine geringere Fehlerquote zu erzielen. Nachstehend finden Sie eine Auflistung mit Vorschlägen zur Verwendung.

- Einfügen von Textbausteinen via Buttons<sup>34</sup>
- Einfügen von Markup via Mausaktionen (s. temporäre Makroleisten)
- Einzelzeichen verschieben `ALT+→`, `ALT+←`
- Datensätze verschieben `ALT+↑`, `ALT+↓`
- Zeile verdoppeln und einfügen `STRG+B`, `STRG+B`, Bild ↓
- Eingabe von {...} `STRG+K`+erstes Zeichen, z.B. `STRG+K+& → {&}`
- Lesezeichen/Bookmarks setzen `ALT+B`
- Lesezeichen aufrufen `ALT+I`

<sup>32</sup> Hat der Benutzer mit der Funktion `L!` Datensätze gelöscht, werden diese nicht von der Funktion `RECALL_DEL` berücksichtigt.

<sup>33</sup> Nicht angezeigt werden Datensätze, die z.B. mit einer Austausch-Anweisung verändert wurden.

<sup>34</sup> <http://tustep.wikispaces.com/Editormakros>.



- Colorierungsanweisungen (→ Syntax-Highlighting)<sup>35</sup>
- Editormakros

### Ausführen von Kommandos direkt aus dem Editor

---

Gib Anweisung > x #\*D

x → verlasse den Editor und führe das nachfolgende Kommando aus (hier Aufruf des TUSTEP-Dateimanagers)

### Einrichten des TUSTEP-Editors

Nach der Installation können Sie eine TUSTEP-Sitzung definieren, aufrufen und den Editor nach Ihren Wünschen einrichten. Sie nehmen damit die grundlegenden Einstellungen vor.

TUSTEP kann sehr umfassend an eigene Bedürfnisse und Wünsche angepasst werden. Von den Farben des Editors über die Fenstergröße, eigene ⌨Tastatur- und ⌨Mausmakros bis hin zu eigenem Syntax-Highlighting und eigenen Menüleisten in der graphischen Oberfläche kann man seine Präferenzen realisieren. Dabei kann der Benutzer für verschiedene Anwendungszwecke (z.B. jeweils für Transkriptionen, Registerbearbeitung und Satzvorbereitung) auch innerhalb eines Projektes unterschiedliche Oberflächen definieren.<sup>36</sup> Hierbei sind mehrere Ebenen der Anpassung möglich: (1) Man kann direkt im Editor innerhalb einer Sitzung temporäre Anpassungen vornehmen, (2) man kann für eine bestimmte Sitzung permanente Einstellungen in der Konfigurationsdatei `TUSTEP.INI` vornehmen oder (3) man kann sich eine `TUSTEP.INI` erstellen, die grundsätzlich in jedes neue Projekt kopiert und ggf. angepasst wird. Nutzt man Einstellungen auf diesen unterschiedlichen Ebenen, so gilt grundsätzlich, dass die konkrete Regel die abstrakte überschreibt, z.B.:

Vorschrift 1 in der allgemeinen `TUSTEP.INI`: Farbe für das ›Text Field‹: D0

Vorschrift 2 in der Editoreinstellung per `STRG+F`: Farbe für das ›Text Field‹: 07

Im Ergebnis wird Vorschrift 2 als die konkretere umgesetzt.

---

<sup>35</sup> <http://tustep.wikispaces.com/Editor+-+Colorierung>, permanente Einstellung mithilfe einer Konfigurationsdatei: <http://tustep.wikispaces.com/Editor+-+Konfiguration>.

<sup>36</sup> Betriebssystemabhängige Einstellungen wie z.B. die Anpassung des Fensters unter Windows, der KDE-Konsole (Linux) bzw. des iTerm (macOS) können in der Installationsbeschreibung nachvollzogen werden: <http://www.tustep.uni-tuebingen.de/pdf/config.pdf>. Im Folgenden beschränken wir uns weitgehend auf betriebssystemunabhängige Anpassungsmöglichkeiten.

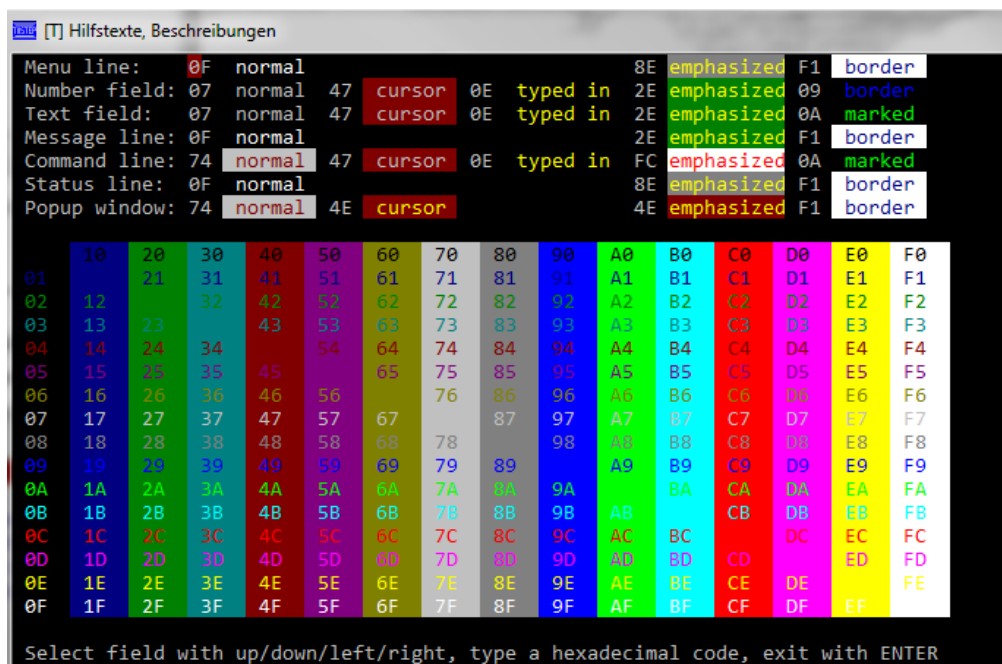


Abb.: Editorfarben

Die Bedeutung der Farbcodes erschließt sich nach Ausführung der Tastenkombination STRG+F im Editor, woraufhin man die oben abgebildete Farbpalette mit den Codes erhält (z.B. »07« für hellgraue Schrift auf schwarzem Grund oder »D0« für schwarze Schrift auf pinkfarbenem Hintergrund) inklusive der Übersicht, welche Farben gerade für welche Felder des TUSTEP-Editors eingestellt sind. In diesem Popup-Fenster können die Farben dann auch unmittelbar geändert werden, indem im oberen Drittel des Fensters für die einzelnen Positionen andere Werte eingetragen werden. Die Einstellungen können nach fertiger Eingabe mit der ENTER-Taste gespeichert werden, woraufhin sich das Fenster schließt. Das Fenster verlassen, ohne die Änderungen zu speichern, kann man stattdessen mit ESC.

Die Editor-Definitionen auf unterschiedliche Art und Weise zurückgesetzt werden:

#E,DEFINITIONEN bzw. #E,DE=+ Löscht die ggf. im Editor geänderten Definitionen und übernimmt wieder die im Segment EDIT der Datei TUSTEP.INI gespeicherten Definitionen. Gemarkte Editoranweisungen bleiben erhalten, ausschließlich im Editor definierte Zeichen- und Stringgruppen hingegen nicht (= nicht im Segment EDIT vorhanden).

#E,DE=-STD-

Funktioniert wie #E,DE=+, die gemerkten Anweisungen bleiben erhalten, allerdings werden nicht die Definitionen aus dem Segment EDIT eingestellt, sondern die von TUSTEP voreingestellten Standardwerte. Zeichen- und Stringgruppen aus dem

Segment `EDIT` sind damit ebenfalls nicht eingestellt.

`#E,DE=!`

Funktioniert wie `#E,DE=+`, allerdings werden auch die gemerkten Anweisungen gelöscht und damit der Zustand wie zu Beginn einer Sitzung wiederhergestellt (= »neu initialisieren«).

Für den Kurs wird eine `TUSTEP.INI`-Datei zur Verfügung gestellt, die über grundlegende Definitionen verfügt und nach Belieben erweitert werden kann. Hinweise hierzu finden sich im TUSTEP-Wiki.<sup>37</sup>

Die Gesamtheit der Definitionen, die temporär im Editor eingestellt werden, können zur Speicherung und Portierung in eine Datei geschrieben werden, z.B.:

`#DA,eddef.tf,seq-ap`

`#ED,MODUS=!,DEFINITIONEN=eddef.tf`

Die Datei `eddef.tf` kann dann in ein anderes Projekt kopiert werden. Für die Einstellung der enthaltenen Definitionen ist das folgende Kommando notwendig:

`#ED,DEFINITIONEN=eddef.tf`

Gegebenenfalls ist es sinnvoll, die Definitionen zuvor auf die Default-Werte zurückzusetzen (oder auch bewusst Teile der alten Definitionen beizubehalten, die nicht von den neuen Definitionen überschrieben werden):

`#ED,DEFINITIONEN=-STD-`

---

<sup>37</sup> <http://tustep.wikispaces.com/Editor%20-%20Konfiguration#Definitionen%20löschen>

## Tipps bei der Nutzung von TUSTEP

### Nützliche Tipps

---

In dieser Rubrik sollen verschiedene Hinweise gegeben werden, welche die Arbeit mit TUSTEP erleichtern und vereinfachen können und die man sonst nicht unbedingt auf Anhieb in den Beschreibungen findet.

### Fehlersignal ausschalten

---

Produziert ein Programm einen Fehler, der zum Abbruch führt, so führt dies *per default* zu einer Fehlermeldung in Verbindung mit einem akustischen Signal. Möchte man auf diesen Hinweis verzichten, reicht die Eingabe des folgenden Kommandos:

```
#DEFINIERE, SIGNAL=-
```

Hierdurch wird die akustische Warnung ausgeschaltet. Mit `SIGNAL=+` kann man sie reaktivieren.

### Information über die aktuelle Sitzung abfragen

---

Beim Start von TUSTEP wird die aktuell verwendete Version angezeigt, beispielweise: »TUSTEP Version 2015 – Open Source Software«. Nun kann es von Interesse sein, genauere Angaben über die Version zu erhalten, etwa wann diese kompiliert wurde. Diese Information lässt sich mit

```
#INFORMIERE
```

abfragen. Man erhält zusätzlich eine Information über den Startzeitpunkt der aktuellen Sitzung:

```
Start INFORMIERE   auf: ms-kenkoS860   am: 17.04.15   um: 22:03:56
**** TEST-Version Blaubeuren 2015   erstellt am: 05.03.2015   um: 11:58
**** Sitzung 1   begonnen am: 03.02.15   um: 22:31
Ende INFORMIERE   auf: ms-kenkoS860   am: 17.04.15   um: 22:03:56
```

### »O-Anweisung« nicht aktuell

---

Nach der Installation einer neuen TUSTEP-Version kann es vorkommen, dass die Fehlermeldung »O-Anweisung nicht auf aktuellem Stand« erscheint. In der sog. O-Anweisung werden verschiedene Editor-Einstellungen gespeichert, die ggf. aktualisiert werden müssen. Hierzu muss die betreffende Stelle in der Konfigurationsdatei bzw. im Segment `EDIT` der Datei `TUSTEP.INI` mit dem Editor aufgerufen und die Anweisung

```
O!, pos
```

gegeben werden. POS ist durch die aktuelle Datensatznummer zu ersetzen, in der die O-Anweisung steht, also z.B. 1.2:

```
O!, 4.11/17
```

Weitere Hinweise zur Fehlermeldung und -behebung finden sich im TUSTEP-Handbuch im Kapitel »Editor | Anpassen des Editors (Optionen) | Automatisches Einstellen der Optionen«.

#### parametrisierter Skriptaufruf mit TUE<sup>38</sup>

Mit der Parametrisierung von Programmaufrufen können regelmäßig für Standardaufgaben benutzte Skripte ohne Änderung der Programmdateien direkt ausgeführt werden. Für die Platzhalter ?1-?9 werden die beim Programmaufruf angegebenen Werte eingetragen. Die im nachfolgenden Beispiel mit »#=« eingeleiteten Zeilen dienen der Dokumentation und geben die Information zu den definierten Dateinamen im Ablaufprotokoll aus.

```
#=      ?1 = Name der Quelldatei
#=      ?2 = Name der Zieldatei
#KO, ?1, ?2, MO=-STD-, LO=+, PA=*
XX      1      |<*>| |
*EOF
```

Programmaufruf:

```
#T, skript.p, , quelle.tf'ziel.tf
```

Die Daten der Quelldatei werden in eine Zieldatei kopiert, wobei alle Tags gelöscht und die Daten der Zieldatei ggf. überschrieben werden (LO=+).

---

<sup>38</sup> TUSTEP-Handbuch, Kapitel »Kommandos | #TU: Ausführen einer Kommandofolge«.

Es können bis zu neun Parameter an beliebiger Stelle in das auszuführende Programm eingebaut werden.

```
#KO, ?1, ?2, LO=?3, MO=?4, PA=*
XX      1      | aa | bb |
*EOF
#UM, ?2, ?5, CO=?6, LO=?7
```

exemplarischer Programmaufruf: #T, s1, , qu'zi'+'-STD-'zil.txt'ansi'+

Die Daten der Quelldatei (qu) werden in eine Zieldatei (zi) kopiert, wobei alle Zeichenfolgen »aa« in »bb« ausgetauscht werden. Außerdem wird die Zieldatei ggf. überschrieben (+), die Datensatznummerierung wird möglichst nicht geändert (-STD-) und als Zieldatei für den Export als Textdatei wird im Zeichenformat ANSI die Datei zil.txt angegeben, deren Inhalt ggf. zu überschreiben ist (+).

automatisches Backup (z.B. beim Aufruf von Skripten oder in der TUSTEP.INI)

```
#- Alle XML-/TXT- und TUSTEP-Dateien (hier xx.tf), die einen
#- TUSTEP-konformen Dateinamen haben, werden
#- vor Ausführung eines Skriptes auf eine Banddatei
#- gesichert
#- Banddatei zur Sicherung anlegen
#DA, backup.tape, TAPE-AP
#*MBUPDATE, BA=backup.tape, MO=RE, POS= | .xml | .txt | .tf |
#- exemplarische Verarbeitung:
#UM, QU=text1.tf, ZI=text1.xml, LO=+, CO=UTF8
```

Für Informationen zum Standardmakro #\*MBUPDATE kann auf Kommandoebene #INF, MBUPDATE genutzt werden. Die angegebenen Dateien werden in die Banddatei gesichert, sofern sie noch nicht oder nur in einer älteren Version in der Banddatei vorliegen.

---

## Kopieren und Einfügen (Windows)

---

Unter Windows können Inhalte ohne Umwege zwischen TUSTEP und der Windows-Zwischenablage ausgetauscht werden.

Für alle Betriebssysteme gilt, dass mittels einfachem Copy & Paste per `STRG+C` und `STRG+V` ausschließlich Inhalte ausgetauscht werden sollten, die über wenige Zeilen reichen. Zum Einfügen/Austauschen von umfangreichen Textportionen von einer TUSTEP-Datei in eine andere bzw. zwischen TUSTEP und der Zwischenablage des Betriebssystems ist die Editor-Anweisung `EZ` sehr nützlich, da hier bei der Übertragung die Datensatzstruktur beibehalten bleibt, auch wenn an der Stelle, an der sie eingefügt werden sollen, nicht genügend Datensatznummern frei sind. `EZ` funktioniert zudem auf allen Betriebssystemen:

`EZ, (+1, -1)`

→kopiert den gesamten Inhalt der im Editor geöffneten Datei in die Zwischenablage

`EZ, (54, 60)`

→kopiert den Inhalt der Datensätze mit den Zeilennummern 54–60 in die Zwischenablage

`EZ, (54/2, 60)`

→kopiert den Inhalt der Datensätze mit den Zeilennummern 54/2–60 in die Zwischenablage

Nach dem Übertragen in die Zwischenablage kann der Inhalt mittels der folgenden Anweisung an eine gewünschte Stelle (Datensatznummer) in der TUSTEP-Datei eingefügt werden:

`EZ, 5`

---

## Kopieren und Einfügen (Linux)

---

Unter Linux fungieren die Zwischenablage (= das Clipboard) des Betriebssystems und die Zwischenablage der TUSTEP-Konsole als getrennte Systeme. Um dennoch Inhalte zwischen beiden Zwischenablagen austauschen zu können, kann das Standardmakro `#*CB, i` bzw. `#*CB, e` (= Clipboard, import bzw. Clipboard, export) genutzt werden. Vorgehen beim Einfügen aus einem externen Programm in den TUSTEP-Editor:

- Text markieren, `STRG+C`
- auf der TUSTEP-Kommandoebene: `#*CB, i` (= importieren)
- Wechsel in den TUSTEP-Editor, `STRG+V` oder `EZ, datensatznummer`

Vorgehen beim Einfügen aus dem TUSTEP-Editor in ein externes Programm:

- Text markieren, STRG+C oder EZ (pos1, pos2)
- auf der TUSTEP-Kommandoebene: #\*CB, e (= exportieren)
- Wechsel in das externe Programm, STRG+V

---

### Kopieren und Einfügen (macOS)

---

Unter macOS ist die Vorgehensweise äquivalent zu derjenigen unter Linux, nur dass das benötigte Standardmakro hier #\*PB (= Pastboard) und nicht #\*CB heißt.

---

### Syntax-Highlighting<sup>39</sup>

---

Colorierungen dienen der visuellen Hervorhebung bestimmter Merkmale im TUSTEP-Editor, z.B. XML-Tags, inhaltlicher Schlagwörter, Personen u.v.m. Sie können vom Benutzer in 9 verschiedenen Farbgruppen zu je 9 Farbdefinitionen konfiguriert werden. Neben einfachen Stringfolgen ist die Benutzung von Pattern Matching zur Auswahl der hervorzuhebenden Zeichenfolgen möglich.<sup>40</sup>

- $Cg_n, f_n = xx:Definition$
- Farbauswahl anzeigen: STRG+F
- $C1, l=84: |<h1>*</h1>|<i>*</i>|$
- $Cg_n, f_n =$  Farbdefinition  $f_n$  der Farbgruppe  $g_n$  löschen
- $Cg_n =$  Alle Farbdefinitionen der Farbgruppe  $g_n$  löschen

---

### Editormakros und Funktionstasten

---

Im Folgenden werden einige Möglichkeiten vorgestellt, die Arbeit mit dem Editor nach den eigenen Bedürfnissen und Projektanforderungen zu konfigurieren. Generell gilt, dass die funktionalen Möglichkeiten bei Punktmakros, Tastaturmakros, Mausmakros, Makroleisten und Funktionstasten im Grunde identisch sind. Die Editor-Steuerbefehle, die jeweils eingesetzt werden, können ebenso gut über Tastaturmakros wie über ein Button in einer Makroleiste aufgerufen werden. Je nachdem, welches funktionale Ziel man verfolgt und wie die individuellen Benutzungspräferenzen aussehen, können die Aufrufmöglichkeiten ausgewählt werden. Nachfolgend sollen ein paar Beispiele illustrieren, welche Aktionen mit Editormakros automatisiert und vereinfacht werden können:

---

<sup>39</sup> Vgl. im TUSTEP-Handbuch Kapitel »Editor | Organisatorische Anweisungen | Colorierung definieren/wechseln/löschen/abfragen«.

<sup>40</sup> Vgl. ausführlicher <http://tustep.wikispaces.com/Editor%20-%20Colorierung>.



- Einfügen von Textbausteinen (z.B. TEI-Tag-Struktur)
- Auslesen einer markierten Bibelstelle und Erstellen eines Registereintrags dazu
- Aufruf einer Satzroutine, die aus der aktuell im Editor geöffneten Datei den Brief mit der aktuell ausgewählten Briefnummer setzt
- Export der aktuell im Editor geöffneten Datei nach RTF oder ins Textformat, anschließend öffnen der Zieldatei mit dem dazu auf dem Computer dafür vorgesehenen Programm
- Anzeige der TUSTEP-Beschreibungen
- Definieren und Aufrufen von Lesezeichen
- Tagliste erstellen
- Paarigkeit der XML-Tags überprüfen
- Syntaxhighlighting einstellen/ändern
- Makroleiste umschalten

### Punktmakros<sup>41</sup>

☞ Punktmakros sind eine spezielle Art der ☞ Editormakros, die z.B. genutzt werden können, um Abkürzungen aufzulösen, Textbausteine oder sonstige Textteile (z.B. Codesnippets, Kommandofolgen usw.) an der Cursor-Position einzufügen. Zur Vermeidung von manuellen Tippfehlern, beschleunigten Eingaben oder auch dem regelmäßigen Aufbau komplexerer Kommandofolgen beim Programmieren ist diese Möglichkeit sehr nützlich:

- Definiert werden die Punktmakros mit einer Anweisung nach dem Muster `Y,abbr.=INS:|<abbreviation>|` → Das Punktmakro »abbr« fügt <abbreviation> an der beim Aufruf ausgewählten Stelle ein (INSERT\_MODE).
- Zum Aufruf des Punktmakros wird an der passenden Stelle im Textfeld die Zeichenfolge »abbr« eingegeben oder aufgesucht, der Cursor im oder nach dem Wort platziert und die Tastenkombination → ALT+P gedrückt. Daraufhin wird »abbr« zu <abbreviation> aufgelöst.

#### – Bsp. 2:

```
y,kompze.=INS:"Kompetenzzentrum für elektronische Erschließungs-
und Publikationsverfahren in den Geisteswissenschaften"
```

#### – Bsp. 3:

```
Y,ko.=INS:"<!-- -->",3*CUR_LE
```

Dieses Punktmakro definiert ein Punktmakro »ko.«, das den Rahmen für einen XML-

<sup>41</sup> Vgl. im TUSTEP-Handbuch, Kapitel »Editor | Organisatorische Anweisungen | Makros aufrufen/definieren/löschen/abfragen«, Makro mit Abkürzungspunkt.

/HTML-Kommentar in den Text einfügen und den Cursor nach dem Einfügen um drei Stellen nach links soll, so dass der Benutzer unmittelbar mit dem Verfassen des Kommentars starten kann.

- Mit der Tastenkombination `ALT+Y` kann man sich alle definierten Punktmakros anzeigen lassen. Wählt man durch Anklicken mit der linken Maustaste oder mit den Pfeiltasten und `RETURN` das beispielsweise definierte Makro »ko.« aus, so wird das oben geschilderte Makro ausgeführt.
- Die Steuerbefehle, die benutzt zur Definition von Editormakros und deren Verwaltung werden können, findet man im Handbuch in den Kapiteln »Editor | Organisatorische Anweisungen | Makros aufrufen/definieren/löschen/abfragen«, »Editor | Steuerbefehle im Editor«.

### Allgemeine Editormakros<sup>42</sup>

Editormakros können genutzt werden, um beispielsweise Textbausteine zu schreiben, eine Folge von TUSTEP-Anweisungen oder -Kommandos auszuführen, Editoreinstellungen zu verändern oder Nicht-TUSTEP-Programme aufzurufen. Sie können mit einer Funktionstaste, einer Tastenkombination oder mit der Maus aufgerufen werden.

- Anzeigen einer Liste mit allen Makros mit der Anweisung `Y`
- einige Tastaturmakros sind standardmäßig vorbelegt
- Definition eines Makros `Y, name=makroanweisungen`
- alle definierten Makros löschen und Standardeinstellung wieder einstellen `Y, -STD-`
- Bsp.: `Y, SC_N=INDENT → SHIFT+STRG+N` rückt den aktuellen Datensatz genauso weit vom linken Rand wie den vorangegangenen Datensatz ein
- Bsp.:

`Y, CA_B=BEG_REC, INS:"<bi>", SPLIT, INS:"</bi>", CUR_UP, END_REC → STRG+ALT+B` fügt am Beginn des Datensatzes die Codierung für einen bibliographischen Eintrag ein (`<bi>`), teilt die aktuelle Zeile, fügt in der neuen Zeile die entsprechende Endkennung (`</bi>`) ein und positioniert den Cursor am Ende der ersten Zeile, damit hier gleich weitergeschrieben werden kann

- Bsp.:

`Y, CA_U=CLR_CMD_LINE, "ZU, , %, | |", CUR_LE → STRG+ALT+U` löscht den Inhalt der Anweisungszeile, fügt die Zeichenfolge `ZU, , %, | |` ein und positioniert den Cursor um eine Position nach links (`CUR_LE`).

<sup>42</sup> Vgl. TUSTEP-Handbuch die Kapitel »Editor | Organisatorische Anweisungen | Makros aufrufen/definieren/löschen/abfragen«, »Editor | Tastenkombinationen für Makroaufrufe«, »Editor | Tastenkombinationen für Steuerbefehle«, »Editor | Steuerbefehle im Editor«.

---

Menüleisten (= Makroleisten)<sup>43</sup>


---

Auch wenn in der Standardeinstellung keine Menüleisten im TUSTEP-Editor angezeigt werden, kann der Benutzer zwei Menüleisten nach eigenen Bedürfnissen konfigurieren und mithilfe einer Konfigurationsdatei oder der Datei `TUSTEP.INI` in den Editor integrieren. Diese können für diverse Aufgaben genutzt werden, z.B. zum Einfügen von Textbausteinen oder Code Snippets, als subsidiäre Buttons für die Eingabe komplexer Anweisungsfolgen, für die Änderung von Einstellungen für unterschiedliche Aufgaben (Colorierung, Funktionstasten) oder für den Aufruf externer Dokumente (TUSTEP-Beschreibungen) oder Webseiten.

---

einfache temporäre Makroleiste

---

- Aufgabe: Anlegen einer permanenten, sequenzielle Übungsdatei `ml_def1.tf`; schrittweises Vorgehen
- Konfigurierung einer temporären Makroleiste mit zwei Feldern: `HILFEN` und `DMA-NAGER`.
- Über die Felder sollen das TUSTEP-Handbuch sowie die Satzbeschreibung resp. der TUSTEP-Dateimanager angesteuert werden.
- Datei anlegen  
`#DA,ml_def1.tf,SEQ-AP`
- Datei öffnen  
`#E,ml_def1.tf`
- Folgende Zeilen sind in die Definitionsdatei linksbündig einzutragen. Fortsetzungszeilen werden durch ein Leerzeichen gekennzeichnet.  
`= Definition der Felder HBUCH, SBUCH`  
`= und DMANAGER, die gem. Syntax einen obligatorischen`  
`= Hilfstext enthalten müssen`  
`Y,*=@HBUCH:"Hilfstext",@SBUCH:"Hilfstext",`  
`@DMANAGER:"Hilfstext"`  
`= innerhalb von Definitionsdateien sind Kommentarzeilen`  
`= wie in dieser Zeile per "=" zu kennzeichnen`  
`= Beschriftung der Makrofelder:`  
`Y,@HBUCH=hbuch_auf:"Handbuch aufrufen"`  
`Y,@SBUCH=sbuch_auf:"Satzbeschreibung aufrufen"`  
`Y,@DMANAGER=dmanager_auf:"Dateimanager aufrufen"`

---

<sup>43</sup> TUSTEP-Handbuch, Kapitel »Editor | Makroleisten definieren/löschen/abfragen«.

= Festlegung der Makrofunktionen, die ausgeführt werden,

= sobald eines der Felder angeklickt wird

Y,hbuch\_auf=CLR\_CMD\_LINE, "x #\*ZEBE",ENTER

Y,sbuch\_auf=CLR\_CMD\_LINE, "x #\*ZEBE,SATZMAKRO",ENTER

Y,dmanager\_auf=CLR\_CMD\_LINE, "x #\*D",ENTER

- die Makroleiste kann mit folgendem Kommando als Editordefinition eingestellt werden:

#E,DE=ml\_def1.tf

---

einfache temporäre Makroleiste für Mauseaktionen<sup>44</sup>


---

- Aufgabe: Anlegen einer permanenten, sequenziellen Übungsdatei `ml_def2.tf`
- Konfigurierung einer temporären Makroleiste, die bei der Markierung von Text mit der rechten (!) Maustaste aufgerufen wird.
- Die Makroleiste soll drei Aktionen zur Auswahl anbieten: Text kursivieren, Text auskommentieren, Auswahl ignorieren.

- Datei anlegen

```
#DA,ml_def2.tf,SEQ-AP
```

- Datei öffnen

```
#E,ml_def2.tf
```

- Folgende Zeilen sind in die Definitionsdatei linksbündig einzutragen. Fortsetzungszeilen werden durch ein Leerzeichen gekennzeichnet.

```
= rechte Maustaste gedrückt halten, führt zu einer
```

```
= farblich hervorgehobenen Markierung
```

```
= Makro für die rechte Maustaste
```

```
Y,M_RP=SET_INS,SAVE_CUR,MRK_INI,MRK_CHG:0C
```

```
= Loslassen der rechten Maustaste →
```

```
= markierten Bereich in den Zwischenspeicher und
```

```
= Zwischenspeicher ggf. zuvor löschen; Aufruf der
```

```
= temporären Makroleiste TAGS
```

```
Y,M_RR=MRK_REP,SWITCH:C_?TAGS
```

```
Y,?TAGS="Bitte Aktion auswählen",
```

```
    KURS:"Textstelle kursivieren",
```

```
    AKOMM:"Textstelle auskommentieren",
```

```
    IGN:"Markierung aufheben"
```

```
= Festlegen der Makrofunktionen
```

```
Y,C_KURS="</i>",< i>",< i>"
```

```
Y,C_AKOMM="--->",< !---"
```

```
Y,C_IGN=MRK_IGN
```

- die temporäre Makroleiste kann mit folgendem Kommando als Editordefinition eingestellt werden:

```
#E,DE=ml_def2.tf
```

---

<sup>44</sup> TUSTEP-Handbuch, Kapitel »Editor | Mauseaktionen für Makroaufrufe«.

## Funktionstasten<sup>45</sup>

Im TUSTEP-Editor können 60 Funktionen definiert werden, von denen die ersten 52 über Funktionstasten aufgerufen werden können: Die Funktionstasten F1-F12 sind von TUSTEP vorbelegt, können jedoch beliebig anders belegt werden. Sie können über die Funktionstasten auf der Tastatur, F13-F52 über die kombinierte Verwendung von Shift, STRG, Plus aus dem Numblock bzw. STRG+B und der jeweiligen Funktionstaste aufgerufen werden. Die Funktionstaste F52 kann beispielsweise mit folgender Tastenkombination aufgerufen werden: STRG+B, STRG+B, STRG+F12. Sämtliche Funktionsaufrufe können zusätzlich als Anweisung eingegeben und die Eingabe mit RETURN bestätigt werden. Die Eingabe von F32 in der Anweisungszeile ist somit gleichbedeutend mit der Tastenkombination STRG+B, Shift+F12, die Funktionstaste F27 kann mit STRG+F7 aufgerufen werden. Die auf den Funktionen 53–62 können durch die Eingabe in der Anweisungszeile aufgerufen werden.

- Anzeige aller Funktionstastenbelegungen mit der Anweisung F
- Definition vornehmen: F<sub>n</sub>=definition
- Bsp.: Anzeige des TUSTEP-Handbuchs mit der Funktionstaste F50:  
F50=x #\*ZEBE
- Bsp.: Unterbrechen der Arbeit im Editor und Ausführung der aktuell im Editor geöffneten Datei als Programm mit TUE: F16 = x #T,<EDITOR>
- Bsp.: Aufruf eines parametrisierten Satzlaufes inkl. Konvertierung der Postscript-Datei in ein PDF:

```
F57=x #*SATZ,konfig_wo.tf,MO=satz,PAR=a4ms,AUSF=s'd,
      SCH=palatino,OPT=noghost #*ps2pdf,konfig_wo.ps,konfig_wo.pdf
```

- Löschen einer bestimmten Definition: F50=
- Löschen aller Funktionstastendefinitionen: F!
- Definitionen auf Standardeinstellung zurücksetzen: F=-STD-

<sup>45</sup> Vgl. im TUSTEP-Handbuch die Kapitel »Tastenkombinationen für Funktionsaufrufe«, »Steuerbefehle im Editor«.

## Verarbeitung sprachlicher Muster

## Zeige-Anweisungen

Zum Suchen und Finden von Zeichenfolgen (= Strings) und Mustern (= Patterns) stehen in TUSTEP verschiedene Zeige-Anweisungen zur Verfügung. Die gängigsten sind ZN für »Zeige nur« und ZU für »Zeige Umgebung«. Bei ZN werden ausschließlich die Datensätze, welche die Suchzeichenfolge enthalten, angezeigt. Sie werden dabei aus ihrem ursprünglichen Zusammenhang herausgelöst und in Listenform angezeigt. Das ist z.B. sehr hilfreich, um Uneinheitlichkeiten schnell sichtbar zu machen. Bei der Suche mit ZU werden die Treffer von Beginn der Datei an in Richtung Dateieinde abgearbeitet und die Treffer in ihrem jeweiligen Kontext angezeigt.

```

TUSTEP KURS18TR 1 KURS18TR
KURS18TR*FROSCH REG.TF
*****
1.1      <!-- Modifizierter Froschkönigtext, angereichertes XML, Übungsdatei -->
1.21     <head>Der <name xml:id="Prinz">Froschkönig</name> oder der eiserne <name xml:id="Diener
Heinrich">Heinrich</name>.</head>
1.31     Perlen und was es auf der Welt nur wär.</q> Wie sie so klagte, steckte ein Frosch seinen Kopf
1.33     sie, du garstiger Frosch, was kannst du mir helfen! meine goldne <sportGeraet>Kugel</sportGeraet> ist
mir in den
1.34     Brunnen gefallen.</q> Der Frosch sprach: <q>deine Perlen, deine <w attr="edel">Edel</w>steine und deine
Kleider,
1.38     <name xml:id="Prinzessin">Königstochter</name> dachte, was schwätzt der einfältige Frosch wohl, der muß
doch in seinem Wasser
1.41     seyn.</q> Der Frosch steckte seinen Kopf unter das Wasser und tauchte hinab, es dauerte auch
1.45     damit nach Haus eilte. Der Frosch rief ihr nach: <q>warte, <name
xml:id="Prinzessin">Königstochter</name>, und nimm mich mit,
1.50     war es der Frosch, an den sie nicht mehr gedacht hatte; ganz erschrocken warf sie die Thüre
1.52     und sagte: <q>warum fürchtest du dich?</q> <q>Da draußen ist ein garstiger Frosch, sagte sie, der
1.65     <p>Der <name xml:id="König">König</name> sagte: <q>was du versprochen hast, mußt du halten, geh und mach
dem Frosch die
1.66     Thüre auf.</q> Sie gehorchte und der Frosch hüpfte herein, und ihr auf dem Fuß immer nach, bis
1.69     der Frosch oben war, sprach er: <q>nun schieb dein goldenes Tellerlein näher, ich will mit dir
fürchtete sich vor dem kalten Frosch, sie getraute sich nicht ihn anzurühren und nun sollte
1.73     Sie packte den Frosch mit zwei Fingern und trug ihn hinauf in ihre Kammer, legte sich ins
1.77     mich in Ruh lassen, du garstiger Frosch!</q> </p>
1.80     <p>Aber der Frosch fiel nicht todt herunter, sondern wie er herab auf den Boden kam, da wars
1.95     <br/>als ihr eine Fretsche (Frosch) wast.</q> (wart)
attr="edel">Edel</w>steinmanufaktur
attr="edel">Edel</w>steinschleiferei
*****
Dateiende
*****
18 Zchflg. gefunden      18 Sätze gefunden      102 durchsucht
*=1.95 Gib Anweisung >
**;01      SPLIT      INSERT      MOUSE

```

Abb.: Ergebnis der Suche mit ZeigeNur





`ZN,,, $Buch$`

`ZN,,, -Buch-`

`ZN,,, aBucha`

Wie geschildert wurde, können in einer Zeige-Anweisung auch mehrere Suchstrings auf einmal gesucht und bestimmte Zeichenfolgen aus der Treffermenge ausgeschlossen werden:

`ZU,,, |Buch|Tisch|`

`ZU,,, -Buch-Bücher-`

`ZN,,, |Buch|Bücher|Regal|Tische|`

`ZN,,, |Bücherregal|Bücher|Regal|`

Die letzte Zeige-Anweisung findet alle Vorkommen von »Bücher« und »Regal« inklusive der entsprechenden Bestandteile von Komposita wie »Bücherliebhaber«, »Regalbrett« usw. mit Ausnahme von »Bücherregal«.

Bsp.: Anweisung *Zeige nur diejenigen Datensätze, in denen der String ›Frosch‹ steht.*<sup>46</sup>

`ZN,,, |Frosch|`

*Zeige alle Datensätze in ihrer Umgebung an, die den String ›Frosch‹ enthalten.*

`ZU,,, |Frosch|`

*Zeige nur diejenigen Datensätze an, in denen der String ›Frosch‹ als eigenständiges Wort vorkommt.*

`ZN,,, __, |Frosch|`

Eine Zeichenfolge (sie kann auch Worttrenner enthalten) gilt dabei als ganzes Wort, wenn sie zwischen Worttrennern bzw. am Satzanfang und/oder Satzende steht. Werden zwei Unterstreichungsstriche<sup>47</sup> [zwischen dem zweiten und dritten Komma] angegeben, so gelten alle Zeichen außer Buchstaben und Ziffern als Worttrenner; wird nur einer angegeben, so gilt der Unterstreichungsstrich nicht als Worttrenner (z.B. für Variablenamen, die Unterstreichungsstriche enthalten können).<sup>48</sup>

Mit der Zeige-Anweisung `ZN,,, __, |Frosch|` wird also »Frosch« nur dann gefunden, wenn es als eigenständiges Wort vorkommt, nicht aber in Zusammensetzungen wie »Froschteich« oder »Ochsenfrosch«.

Die Suche muss nicht in der gesamten Datei durchgeführt werden, sondern kann auch auf bestimmte Datensatzbereiche beschränkt werden:

<sup>46</sup> Der nachstehende Suchstring würde auch »Froschkönig«, »Froschlauch«, »Froschteich« und ähnliche Komposita sowie flektierte Formen wie »des Froschs« als Treffer ausgeben, weil nach der Zeichenkette »Frosch« gesucht wird, die in den genannten Beispielen enthalten ist.

<sup>47</sup> Alternative Bezeichnung für das auch als Underscore oder Unterstrich bezeichnete Zeichen bzw. die entsprechende Taste.

<sup>48</sup> TUSTEP-Handbuch, Kapitel »Editor | Nur ganze Wörter suchen/austauschen«.

*Zeige alle Vorkommen von »Frosch« innerhalb der Datensätze 1–20, allerdings nicht innerhalb von Komposita.*

```
ZN, (1,20), __, |Frosch|
```

In Suchzeichenfolgen unterscheidet TUSTEP grundsätzlich nicht(!) zwischen Groß- und Kleinschreibung. Soll die Schreibweise unterschieden werden, so kann ein Backslash vor das betreffende Zeichen geschrieben werden, z.B.:

```
ZN,,, |\A|
```

Diese schreibweisenspezifische Suche gilt dann allerdings auch nur für den unmittelbar nachfolgenden Buchstaben. Möchte man die abgekürzte Aktiengesellschaften im Plural (»AGs«) finden, so liefert der folgende Suchauftrag das gewünschte Ergebnis:

```
ZN,,, |\A\Gs|
```

Möchte man darüber hinaus sichergehen, dass nicht auch die Zeichenfolge »AGS« erfasst wird, kann die Suche wie folgt verändert werden:

```
ZN,,, |\A\G\s|
```

Beachten Sie bitte, dass der vorgesetzte Backslash (»\«) nur vor Buchstaben die Wirkung hat, eine schreibweisenspezifische Suche durchzuführen. In anderen Fällen wird mit dem Backslash ein Zeichen maskiert, dass sonst als Steuerzeichen des Pattern Matchings interpretiert würde, etwa bei der Suche nach dem Backslash selbst oder nach der Zeichenfolge »[abc]«:

```
ZN,,, |\ \ |
```

```
ZN,,, |\ [abc\ ] |
```

Bei einer Ersatzzeichenfolge im Rahmen einer Austausch-Anweisung werden die Zeichen in ihrer jeweiligen Schreibweise ersetzt:

Ausgangstext:

```
Hier steht ein Text.
```

Austausch-Anweisung:<sup>49</sup>

```
A,,, |text|TEXT| (oder A,,, |text|{=0+}|)
```

Ergebnis:

```
Hier steht ein TEXT.
```

Weitere Zeige-Anweisungen sind ZA (Zeigen ab dem gefundenen Datensatz mit nachfolgendem Kontext) und ZE (Zeigen bis zu dem gefundenen Datensatz mit vorangehendem Kontext). ZF zeigt fortlaufend alle Datensätze an und hebt diejenigen Zeichen-

<sup>49</sup> Vgl. zur Funktionsweise der Ersatzzeichenfolgen auch das »Cheat Sheet« im Anhang.

folgen hervor, die den Suchbedingungen entsprechen. Die übrige Syntax inklusive der Formulierung von Separatoren und der Möglichkeit, Bereichsangaben und sonstige Suchparameter anzugeben, bleibt genauso wie bei ZN und ZU.

### Nutzung von Character Groups/String Groups in Zeige-Anweisungen

Zur Verkürzung von Zeige- und Austausch-Anweisung können zusätzlich zu den vordefinierten auch benutzerdefinierte Character Groups oder String Groups genutzt werden. Diese helfen dabei, komplexe Anweisungen zu gliedern, zu vereinfachen wiederholte Aufrufe zu verkürzen.

#### Beispiele:

Anweisung *Zeige alle Doppelvokale an.* →

Definition einer Character Group + Suchanweisung

C:vo=aeiouyääüü

ZN,,,|{C:vo}{+1=}| oder ZN,,,|aa|ee|ii|oo|uu|yy|ää|öö|üü|

*Zeige Wörter der lateinischen A-Deklination in ihrer Umgebung an.* →

Definition einer String Group + Suchanweisung

S:ad=|a|ae|am|arum|is|as|

ZU,,,|{00}{&a}{S:ad}{}}|

Die Angabe der Randbedingung {} ist notwendig, damit Wörter wie »quamque« nicht gefunden werden. Stellt man anschließend fest, dass mit dieser Zeige-Anweisung auch Wörter gefunden werden, die nicht zur A-Deklination gehören, muss die Zeige-Anweisung weiter modifiziert werden; etwa durch die zusätzliche Angabe einer Stringgruppe mit den Stämmen der A-Deklination.

- Das Pattern Matching ermöglicht die Erstellung eigener Character Groups und String Groups, um Such- bzw. Austausch-Anweisung zu steuern, z.B.:

Character Group für Vokale im Deutschen

C:vo=aeiouyääüü

Suche nach allen Wörtern, die ein Suffix enthalten:

Definition von Suffixen

S:su=|heit|heiten|keit|keiten|nis|nisse|nissen|ung|ungen|

Suche nach Wörtern (= beliebig viele Buchstaben), die ein Suffix enthalten:

ZN,,,|{00}{&a}{S:su}|

Mit dieser Anweisung würden auch Wörter gefunden, die ein Suffix enthalten, aber zu

Komposita erweitert wurden (z.B. »Erkenntnisgewinn«. Möchte man diese Wörter aus der Treffermenge ausschließen, kann folgendes zielführend sein:

Suche nach beliebig vielen Buchstaben, gefolgt von einem Suffix, wobei der Trefferstring als ›ganzes Wort‹ im Text (= von Worttrennzeichen umgeben, s.o.) auftauchen muss:

```
ZN, , __, | {00} {&a} {S:su} |
```

Sehr nützlich ist darüber hinaus die Verwendung von lokalen Character Groups (= Zeichengruppen, die nur temporär in einer Zeige- oder Austausch-Anweisung definiert werden). Diese eröffnen die Möglichkeit, an einer bestimmten Stelle einer Anweisung mehrere dort mögliche Zeichen zu definieren. So können die verschiedenen Schreibweisen des Familiennamens Meyer wie folgt gefunden werden:

```
ZN, , , | \M[ae][iy]er|
```

Gefunden wird damit: Maier, Mayer, Meier und Meyer. Ähnliches gilt für die Suche nach y/i-Varianten in frühneuhochdeutschen Texten, wenn z.B. nach allen Vorkommen von »Reych«, »Reich«, »Rych« und »Rich« gesucht werden soll:

```
ZN, , , | \R{0}e[iy]ch|
```

*Zeige die Vorkommen des Namens Johannes in seinen unterschiedlichen Schreibweisen an.*

Suchanweisung mit lokalen Zeichengruppendefinitionen

```
ZN, , , | [\I\J]{0}o{0}h{0}a{1-2}n{0}[ei]{0}s|
```

→ gefunden werden hiermit u.a.: Johannes, Johannis, Johanne, Johann, Jan, Joan, John, Ioannis, Ioannes, Iohannes, Iohannis, Ioanne, Iohanne, Ioanni, Iohanni.

Das französische Jean kann ggf. durch eine Erweiterung der Suchzeichenfolge beim ersten Vokalvorkommen ebenfalls gefunden werden.

## Übungsaufgaben Character und String Groups in Zeige-Anweisungen

Formulieren Sie korrekte und möglichst kurze Zeige-Anweisungen für folgende Aufgaben:

- In einem Text sollen alle Doppelkonsonanten gefunden werden.
- In einem Text kommt eine Person (Stefan Meier) in unterschiedlichen Schreibweisen vor. Finden Sie alle.

Übungsdatei: bech7.tf

- Wie oft wird im »Seelenlos« der »Menschenfresser« erwähnt?
- Welche Wörter im Text beginnen mit einem großgeschriebenen »S«?

- Wie oft wird ein »König« (nicht die »Königstochter«) thematisiert?
- Zählen Sie alle XML-Tags im Text. [Achtung Anzfz.!]
- Suchen Sie alle Verben, die ein Präfix enthalten.
- Wie viele wörtliche Zitate sind im Märchen enthalten?
- Wie viele Wörter enden auf ein Suffix? (dekliniert/undekliniert)?

### Ignorieren von bestimmten Zeichenfolgen bei der Suche

Bei der Suche kann es sinnvoll sein, bestimmte Codierungen zu ignorieren, um alle Vorkommen des Suchstrings zu finden. Sucht man nach »Email«, möchte man vermutlich auch »E-Mail« oder »E-<i>Mail</i>« finden, und sei es nur, um die verschiedenen Vorkommen anschließend vereinheitlichen zu können. Mit der Spezifikation %% zwischen dem zweiten und dem dritten Komma werden folgende Codierungen bei der Suche ignoriert, d.h. auch solche Suchstrings werden gefunden, die diese Codierungen enthalten:

- Akzentcodierungen (`lev%/ee`)
- Punktierungs-codes für die Transliteration des Hebräischen
- das Minuszeichen sowie runde und eckige Klammern: `( , ) , [ , ] , -`
- native Codierungen des TUSTEP-Satzprogramms für Schriftauszeichnungen nach dem Muster `#x+`
- native Codierungen des TUSTEP-Satzprogramms für Hoch- und Tiefstellungen wie `#h: , #g: #o: , #t:`
- ein Tag, dessen Name nur aus einem Buchstaben besteht (`<i>`, `<b>...`)
- native Codes des TUSTEP-Satzprogramms für An- und Abführungszeichen, Silbentrennung (`^ , \ , #. , # , , #'`)

### Übungsaufgaben Zeige-Anweisungen

Wie das obige Beispiel zeigt (s. »Beispiele für korrekte Zeige-Anweisungen mit verschiedenen Separatoren«), können auch Buchstaben (oder Zahlen) als Separator genutzt werden, solange sie nicht Teil der Suchzeichenfolge sind. Die folgenden Zeige-Anweisungen dagegen sind z.T. fehlerhaft. Welche und wieso?

```
ZN,,,|Frosch|
ZN,__,|Froschkönig|
ZN,__,|Froschkönig
ZA,,,|Stein|
ZU,%,__,|Stein|
ZN,,, $Stein$
ZU,%, -Prinzessin$
```

## Pattern Matching

Will man Texte automatisch verarbeiten, ist es notwendig, dem Computer formale Beschreibungen zu liefern, damit dieser in die Lage versetzt wird, die Texte nach den gewünschten Vorgaben zu ändern. Im Gegensatz zu einem intuitiven Verständnis, wie es in der zwischenmenschlichen Kommunikation vorausgesetzt werden kann, arbeitet der Rechner mit starren Zuordnungen, die letztlich auf das binäre Konzept *true/not true* bzw. *0/1* zurückgeführt werden. Ein Mensch versteht in aller Regel den Arbeitsauftrag »Markiere alle Datumsangaben im Text«, auch wenn die Angaben im Text möglicherweise unvollständig oder uneinheitlich sind. Einem Programm hingegen muss zunächst mitgeteilt werden, in welchem Format Jahresangaben erwartet werden. Nachstehend finden sich einige formal unterschiedliche, aber allesamt korrekte Möglichkeiten zur Darstellung derselben Datumsinformation:

»1. August 1914, 1.08.1914, 01.08.1914, 1914-08-01, 1. Aug. 14, 1.8.14, der erste Samstag im August 1914 ...«

Was sich dem menschlichen Leser in der Regel schnell erschließt, muss dem Rechner erst in Form einer formellen Musterbeschreibung (= Pattern Matching) definiert werden. Die gesuchten Muster können entweder konkret sein (Suche nach der Zeichenfolge »1. August 1914«) oder abstrakt (Suche nach »eine ein- bis zweistellige Zahl, gefolgt von einem Punkt, danach ein Leerzeichen, ein Monatsname [der definiert werden muss], anschließend ein Leerzeichen und eine vierstellige Zahl«).

Eine einfache, aber gegebenenfalls zeitaufwendige Möglichkeit, dem Computer für die Suche Muster zu definieren, ist die iterative Methode (nachstehend einmal natürlichsprachlich und einmal im TUSTEP Pattern Matching für das Beispiel deutscher Vokale), bei der alle gesuchten Vorkommen in eine Definition geschrieben werden:

Es seien Vokale: a, e, i, o, u, ä, ö, ü, y.

C:VO=aeiouäöüy

Eine weitere Möglichkeit besteht in der Verwendung von vorbereiteten Definitionen bzw. Platzhaltern, wie sie beispielsweise in den Regular Expressions<sup>50</sup> oder auch im TUSTEP Pattern Matching verwendet werden. Die Idee ist dabei, formale Platzhalter zu benutzen, um damit ein Muster zu beschreiben, das alle Vorkommen findet, die dem Muster entsprechen.

---

<sup>50</sup> <http://programminghistorian.org/lessons/understanding-regular-expressions>,  
[http://www.themacroscopic.org/?page\\_id=521](http://www.themacroscopic.org/?page_id=521),  
<http://www.regular-expressions.info/>.

A regular expression (also called regex) is a powerful tool for finding and manipulating text. At its simplest, a regular expression is just a way of looking through texts to locate patterns. A regular expressing can help you find every line that begins with a number, or every instance of an email address, or whenever a word is used even if there are slight variations in how it's spelled. As long as you can describe the pattern you're looking for, regular expressions can help you find it. Once you've found your patterns, they can then help you manipulate your text so that it fits just what you need.<sup>51</sup>

Bsp. für eine Suche mithilfe des TUSTEP Pattern Matching:

Finde alle Jahreszahlen im ISO-8601-Format (vier Ziffern-zwei Ziffern-zwei Ziffern; z.B. 1914-08-01).<sup>52</sup>

`ZN, , , | {4}{\0}-{2}{\0}-{2}{\0} |`

Formale Beschreibungen von Sprachmustern helfen somit bei der Verarbeitung und Identifikation von Phänomenen in eigenen und fremden Texten. Sie dienen der Übersetzung natürlichsprachlicher Anweisungen und Fragen in eine dem Rechner bzw. dem verwendeten Programm verständlichen Syntax. Der Weg hierzu besteht aus drei Schritten: 1. Formulierung der Aufgabe in natürlicher Sprache, 2. Formalisierung der Anweisung, 3. Übersetzung der Formalisierung in die Programmsyntax.

Bsp.: Suche nach fehlerhaften Wortanfängen am Beginn eines Satzes.

Formale Umsetzung: Suche nach einem Satzzeichen, gefolgt von einem oder mehreren Leerzeichen, hinter dem zwei Großbuchstaben sowie anschließend beliebig viele Kleinbuchstaben stehen.

`ZN, , , | [ . ! \ ? ] {00} {2}{\A}{00}{\a} |`

### Zusammenfassung: Such- und Austausch-Anweisungen

Schritt 1: Formulierung einer natürlichsprachigen Suchanforderung (s. »Beispielhafte Suchanfragen«)

Schritt 2: Abstraktion von Einzelfällen

Schritt 3: Ggf. Formulierung einer natürlichsprachigen Verarbeitungsanforderung

Schritt 4: Formalisierung und Formulierung in einer für Software verständlichen Form

Weitere exemplarische Suchanfragen, in natürlicher Sprache formuliert:

- Finde alle Vorkommen von »Hund« oder »Katze« im Text.
- Finde alle Sätze, in denen sowohl »Hund« als auch »Katze« vorkommt.
- Finde alle Wörter der lateinischen a-Deklination.

<sup>51</sup> Graham, Shawn; Milligan, Ian; Weingart, Scott B. (2016): Exploring Big Historical Data. The Historian's Macroscopic. London, [http://www.themacroscopic.org/?page\\_id=521](http://www.themacroscopic.org/?page_id=521).

<sup>52</sup> [https://de.wikipedia.org/wiki/ISO\\_8601](https://de.wikipedia.org/wiki/ISO_8601).



- Finde alle Vorkommen des Strings »Streit«, allerdings nur solche, die als eigenes Wort vorkommen.

Bsp.: Wörterbuchencoding

- 1. »Gesucht werden alle Lemmata in einem Wörterbuchtext, die aussehen wie: **EINKUPPELMANÖVER**.«
- 2. »Lemmata sind stets als halbfette Versalien ausgeführt.«
- 3. »Alle Lemmata sollen in XML-Tags mit dem Elementnamen »lemma« eingeschlossen werden.«
- 4. Austausch-Anweisung:

A, , , -<b>{ 00 } { \A }</b>-<lemma>{=0=}</lemma>-

Details zur Codierung von Häufigkeitsangaben, Verweisen, Zahlenwerten sowie von Zeichen- und Stringgruppen in Such- und Ersatz-Zeichenfolgen können im Editor mit der Tastenkombination STRG+K+Leertaste aufgerufen werden. Die Leertaste kann dabei auch leicht verzögert betätigt werden.

vgl. zur Beschreibung im Handbuch »Parameter-Modi seit TUSTEP Version 2012«

## Weiterführende Links zu Regulären Ausdrücken

<http://programminghistorian.org/lessons/understanding-regular-expressions>

[http://www.themacroscopic.org/?page\\_id=521](http://www.themacroscopic.org/?page_id=521)

<http://www.regular-expressions.info/>

## Übungsaufgaben

Übungsdatei: `frosch_reg.tf`

- Bestimmen Sie das längste Wort im Text.
- Lesen Sie die im »Froschkönig« enthaltenen XML-Attributwerte aus.
- Wie viele Gedichte enthält der Text?
- Suchen Sie alle Zahlen mit einem beliebigen Wert.
- Suchen Sie alle vierstelligen Zahlen. Wie viele Zahlen sind im Text enthalten, die mehr oder weniger als vier Stellen haben?

## Austausch-Anweisung

Nachdem man Textstellen mithilfe von Zeige-Anweisungen finden konnte, will man diese in aller Regel auch automatisch verarbeiten. Ein sehr nützliches Werkzeug sind hierfür die Austausch-Anweisung im TUSTEP-Editor, die in ihrer grundsätzlichen Funktionweise genauso in den parametergesteuerten Programmen (insb. #KOPIERE) wie auch in TUSCRIPT angewendet werden können.

Eine Austausch-Anweisung wird durch den Anweisungsnamen (a, , , ) eingeleitet und enthält immer ein oder mehrere Zeichenfolgen-Paare, die aus einer Suchzeichenfolge und einer Ersatz-Zeichenfolge zusammengesetzt ist. Beide Zeichenfolgen werden durch ein Begrenzungszeichen voneinander getrennt und darin eingeschlossen. Eine ganz einfache Version könnte wie folgt aussehen:

```
A, , , |Frosch|Froschkönig|
```

## Übungsaufgaben

Formulieren Sie Austausch-Anweisung für folgende Aufgaben und testen Sie die Funktionalität in der Datei dhd\_wo.tf.

- Die Schriftauszeichnungen für Petit-Satz und Kapitälchen sollen gemäß der TEI-Konvention umcodiert werden:

```
<pt>...</pt>, <c>....</c>
```

soll werden:

```
<hi rend="petit">....</hi>
```

und

```
<hi rend="small-caps">...</hi>
```

- Für alle Vorkommen des Begriffes »Digitale Geisteswissenschaften« und »Digital Humanities« soll ein einheitlicher Registereintrag hinter dem betreffenden Wort in den Text eingefügt werden:

```
<reg>Digital Humanities</reg>
```

- Die Hyperlinks im Text sollen in die für HTML übliche Notation umgebaut werden:

```
<a href="Sprungziel">Text</a>
```

- Sie haben sich vorhin alle Doppelvokale anzeigen lassen. Diese sollen für eine Analyse nun folgendermaßen getaggt werden:

```
<dvok>ee</dvok>
```

- Definieren Sie eine Character Group mit Zeichen, welche den Anfang und das Ende eines Wortes kennzeichnen. Definieren Sie anschließend eine String Group mit deutschen Suffixen inklusive der flektierten Formen. Bilden Sie schließlich eine Austausch-Anweisung, mit der sie alle Wörter, die auf ein Suffix enden, taggen:

```
<suffix>Entwicklungsumgebung</suffix>
```

- Die Adressdatenbank am Ende von `dhd_wo.tf` soll nach folgendem Muster umgebaut werden, wobei die alten Codes zur Datenfeldabtrennung entfernt werden können:

```
<entry><name>....</name><surname>...</surname>  
<street>...</street><place>...</place>
```

Finden Sie jeweils den oder die (vermutlichen) Fehler in den folgenden Anweisungen:

```
A,,,|*|&#x002A;|
```



```
A,,,|<afz>{|}Cuius regio, eius religio{|}</afz>|<q>{=2=}</q>|
```

```
A,(129,98),,|Etsi omnes, ego non.:
```

## Import von Daten

## Importieren von Daten nach TUSTEP

### #\*IMPORT: Konvertierung von RTF-Dateien

TUSTEP bietet die Möglichkeit,  Fremddateien (= Nicht-TUSTEP-Dateien), die in MS-Word oder OpenOffice/LibreOffice erstellt wurden, mithilfe eines  Standardmakros (= Zusammenstellung von grundlegenden TUSTEP-Kommandos, die bereits im Installationsumfang enthalten ist) zu importieren, um sie in TUSTEP zu bearbeiten und ggf. wieder als RTF zu speichern (oder in einem anderen Format wie XML zu exportieren).<sup>53</sup> Hierzu ist es unerlässlich, die Dateien in ihrem ursprünglichen Erstellungsprogramm zu öffnen und als RTF bzw. als XML-Datei im Format »Word 2003 XML-Dokument« (und nicht(!) etwa als »Word XML-Dokument«) zu speichern.

Das Vorgehen ist dabei wie folgt:

- Ausgangsdatei als RTF abspeichern und im Projektverzeichnis ablegen
- RTF-Datei anmelden: `#AN, dateinam.rtf`
- TUSTEP-Datei einrichten, in die importiert werden soll:

`#DA, imp.tf, SEQ-AP`


- Import starten:

`#*IMPORT, QUELLE=dateinam.rtf, ZIEL=imp.tf, LOESCHEN=+, IGNORIEREN=-`

Beim Import werden neben dem Text auch die Formatierungsinformationen (Farbe, Schriftauszeichnung [fett, kursiv, unterstrichen, durchgestrichen, Hochstellung...], Schriftarten usw.) übernommen und in eine XML-Notation umgewandelt. Als Ergebnis erhält man eine TUSTEP-Datei mit wohlgeformter XML-Syntax inklusive entsprechender Metadaten. Zusammen mit dem Standardmakro `#*EXPORT` für den umgekehrten Weg (Ausgabe einer TUSTEP-Datei als RTF) ist ein Round-tripping von Word (z.B. für Manuskripterstellung) über TUSTEP (z.B. für den `#SATZ`) nach Word (z.B. für die Erstellung einer zweiten Auflage durch den Autor) möglich.<sup>54</sup>

Viele Formatierungstags aus Word sind für die Weiterverarbeitung nicht notwendig und tendenziell störend. Daher kann mit der Spezifikation `IGNORIEREN` bestimmt werden, ob alle Informationen übernommen werden sollen (`IGN=-`), ob die normalerweise überflüssigen Informationen ignoriert werden sollen (`IGN=+`), oder ob die normaler-

<sup>53</sup> Vgl. hierzu das separate »TUSTEP Import & Export«-Handbuch (`#*ZEBE, IMPORT`). Eine exemplarische RTF-Verarbeitung mit TUSTEP zeigt das Video »RTF-Processing«:

<http://tustep.wikispaces.com/Videos+zu+TUSTEP+und+TUSTEP-Projekten>. Vgl. zu den unterschiedlichen Dateiformaten im Handbuch die Beschreibungen Grundlagen → Dateien → TUSTEP-Dateien, Fremd-Dateien und Grundlagen → Dateien → Dateistruktur sowie die Ausführungen im  Glossar.

<sup>54</sup> Einschränkungen gibt es hier z.B. bei in Word eingebundenen Graphiken, die beim Import nach TUSTEP lediglich als Platzhalter übernommen werden.

weise überflüssigen Informationen bis auf die vom Benutzer definierten Ausnahmen ignoriert werden sollen (`IGN=!name`). Die Spezifikation `IGN=!color` würde somit die Informationen zu Schriftfarben importieren, die übrigen Daten allerdings ignorieren, die in der Regel nicht benötigt werden. Welche das sind, ist in der Beschreibung zum Standardmakro `#*IMPORT` dokumentiert. Beim Import wird zudem eine Liste mit den importierten Codes ausgegeben; standardmäßig ins Ablaufprotokoll auf Kommandoebene, diese Ausgabe kann aber auch in eine Datei umgeleitet werden (s. Angaben zur Spezifikation `PROTOKOLL`). Es empfiehlt sich, bei einem unbekannten Text, zunächst `IGN=-` zu nutzen, sich einen Überblick zu verschaffen und anschließend die ignorierbaren Tags und Attribute wegzulassen bzw. bestimmte Tags und Attribute von dieser Regel auszuschließen.

Die importierten Daten können dann z.B. mit `TUSCRIPT` weiterverarbeitet, aufbereitet und schließlich als TEI-XML exportiert werden.

Häufig sind nach dem Import von externen Daten Bereinigungen notwendig, die auf Erfassungsfehler, unsaubere bzw. uneinheitliche Programmbedienung oder Flüchtigkeiten zurückzuführen sind. Eine Auswahl solcher Fälle sind: Verwendung unterschiedlicher Formatvorlagen für das gleiche inhaltliche Phänomen, Vermischung verschiedener Dokument- und Formatvorlagen aus unterschiedlichen älteren Dateien, Nutzung von direkter Formatierung anstelle von Formatvorlagen.<sup>55</sup>

## Übungsaufgaben

Importieren Sie die Datei `froschl.rtf` nach `frimp1.tf`. Analysieren Sie das Importergebnis, löschen Sie den Header sowie weitere überflüssige Codierungen und codieren Sie die Auszeichnung im Editor so um, dass die Datei `frimp1.tf` als Quelldatei für `#*SATZ` genutzt werden kann.<sup>56</sup> Führen Sie anschließend einen Satzlauf mit `#*SATZ` aus.

---

<sup>55</sup> Wenn in einem Text Kapitälchen genutzt werden, um Autoren und gleichzeitig auch Orte zu kennzeichnen, und der Verfasser des Textes nicht im einen Fall die Formatvorlage »Autor« und im anderen Fall die Formatvorlage »Ort« zugewiesen hat, sondern mit der direkten Formatierung die fraglichen Stellen in Kapitälchen gesetzt hat, ist im Nachhinein sehr großer Aufwand zu treiben, um Autoren und Orte zu differenzieren und sauber in TEI auszeichnen zu können.

<sup>56</sup> Die hierfür benötigten Satztags können Sie in der Beschreibung zum Standardmakro nachschlagen: `#*ZEBE, SATZMAKRO`.

---

## #UMWANDLE: Konvertierung von XML und anderen text-basierten Dateiformaten

---

XML, TXT, HTML, MD und ähnliche text-basierte Formate – mit oder ohne bestimmten Strukturanforderungen – können mit dem Kommando #UMWANDLE nach TUSTEP konvertiert werden. Umgekehrt können TUSTEP-Dateien auch mittels #UMWANDLE als Text, XML... ausgegeben werden. TUSCRIPT kann hingegen text-basierte Fremddatenformate unmittelbar (d.h. ohne vorherigen Import) bearbeiten und auch wieder abspeichern.

Kommandosyntax (verkürzt):

```
#UMANDLE, QUELLE, ZIEL, CODE, LOESCHEN
```

Bsp.:

```
#UM, QU=frosch-1.xml, ZI=frosch-1.tf, CO=utf8, LO=+
```

Umgekehrtes Umwandle (TUSTEP → XML):

```
#UM, QU=froschl.tf, ZI=froschl.xml, CO=utf8, LO=+
```

---

## Übungsaufgaben

---

Aufgabe 1:

Importieren Sie die Datei `frosch-1.xml` in die TUSTEP-Datei `frimp2.tf`. Geben Sie dabei den korrekten Zeichensatz an, den Sie z.B. mithilfe von Notepad++, Atom oder UltraEdit herausfinden können, indem Sie die Quelldatei mit einem dieser Editoren öffnen und den angegebenen Zeichensatz in Erfahrung bringen. Öffnen Sie anschließend `frimp2.tf` im Editor: Was fällt Ihnen auf?

Überprüfen Sie testweise, was passiert, wenn Sie einen anderen Zeichensatz angeben.

Aufgabe 2:

Legen Sie eine neue TUSTEP-Datei an, tragen Sie in dieser Datei einen beliebigen Inhalt ein (z.B. eine Liste Ihrer fünf Lieblingsbücher und Ihrer fünf liebsten Filmklassiker o.ä.). Exportieren Sie anschließend den Inhalt der TUSTEP-Datei in eine Textdatei beliebigen Namens mit der Zeichencodierung UTF-8 und öffnen Sie das Ergebnis in einem Texteditor (sinnvollerweise nicht den Windows-Texteditor).

## Einstieg in hochwertigen Satz: \*SATZ

Vgl. für diesen Abschnitt einführend das Dokument → `satzmkr.pdf`. Die für den Satz des Handouts notwendigen Satzparameter sind in der Datei `satzpatr.seg` zu finden. Sie ist ggf. in das jeweilige Projektverzeichnis zu kopieren.

Weiterführende Informationen zum Standardmakro \*SATZ inklusive Anwendungsszenarien finden Sie in der Beschreibung zum Standardmakro und im TUSTEP-Wiki.<sup>57</sup>

Die Beschreibung kann aus jeder TUSTEP-Sitzung heraus mit dem Kommando

`#*ZEBE, SATZMAKRO` aufgerufen werden. Alternativ steht sie auch online zur Verfügung.

---

<sup>57</sup> [http://www.tustep.wikispaces.com/TUSTEP+-+EINFÜHRUNG+in+\\*SATZ,spaces.com/TUSTEP+-+Satz+mit+Makro+\\*SATZ](http://www.tustep.wikispaces.com/TUSTEP+-+EINFÜHRUNG+in+*SATZ,spaces.com/TUSTEP+-+Satz+mit+Makro+*SATZ).

<http://www.tustep.wiki>



## Einstieg und einfache Anwendungen in TUSCRIPT

## Skripting

### Was ist Skripting?

A scripting or script language is a programming language that supports scripts: programs written for a special run-time environment that automate the execution of tasks that could alternatively be executed one-by-one by a human operator. Scripting languages are often interpreted (rather than compiled).<sup>58</sup>

Mögliche Ziele, die mit Skripting erreicht werden soll:

- Hilfe im akademischen Alltag (Daten aus fremden oder eigenen (Forschungs-)Kontexten aufbereiten, bereinigen, in die eigenen Systeme einspeisen)<sup>59</sup>
- Datenaufbereitung im Rahmen der Grundlagenforschung<sup>60</sup>
- wissenschaftliche Auswertung bzw. heuristische Hilfsmittel für selbige<sup>61</sup>

Die Alternative zum Skripting ist i.S. der o.g. Definition die manuelle Ausführung der Aufgaben. Dies kann kaum als ernsthafte Option in Betracht gezogen werden, da die manuelle Arbeit fehleranfälliger und vor allen Dingen nicht eindeutig zu reproduzieren sein wird. Ein Flüchtigkeitsfehler, der nach mehreren Stunden manueller Bearbeitung bei einer repetitiven Aufgabe eingestreut wurde, ist wesentlich schwerer zu finden und zu korrigieren als ein Fehler, der durch ein Problem im Skript systematisch entstanden ist und durch eine Korrektur im Skript zu beheben ist. Zudem werden durch eine (richtig angewandte) Skriptlösung die Ausgangsdaten nicht verändert, bleiben somit als Dokumentations- und Archivform erhalten, während für verschiedene Zwecke unterschiedlich bearbeitete Derivate automatisch erstellt werden können.

Wichtig ist das Vorgehen beim Skripting, das grundsätzlich immer in einem Dreischritt gedacht werden kann: Nach einer intensiven Analyse der Ausgangssituation wird das gewünschte Zielformat definiert oder analysiert, um anschließend zu definieren, welche Schritte erforderlich sind, um die notwendigen Veränderungen durchzuführen.

---

<sup>58</sup> [https://en.wikipedia.org/wiki/Scripting\\_language](https://en.wikipedia.org/wiki/Scripting_language).

<sup>59</sup> Beispiele hierfür: Bibliotheksdaten, Normdaten, bibliographische Daten, Forschungstexte, Quellentexte.

<sup>60</sup> Beispiele: Vorbereitung von Transkriptionen mehrerer Abschriften eines Textes, um eine Kollationierung durchführen zu können.

<sup>61</sup> Beispiele: Wortfrequenzanalysen, Konkordanzen, Informationsextraktion.

Bsp.: Ein Text enthält Endnoten, die wie folgt ausgezeichnet sind:

`<en>Endnotentext</en>` Diese sollen in Fußnoten umgebaut werden. Hierfür würde es genügen, wenn die Tagnamen ersetzt würden. »en« müsste also »fn« werden. In einem Skript würde es somit genügen, eine Austausch-Anweisung zu realisieren:

```
$$ MODE TUSCRIPT, {}
SET quelle = "datei.tf"
SET ziel = "ziel.tf"
SET txt = FILE (quelle)
BUILD X_TABLE x_fn1 = *
  DATA |<{0}/en>|<{+2=}fn>|
SET txt = EXCHANGE (txt, x_fn1)
FILE/ERASE/PRINT $ziel = txt
```

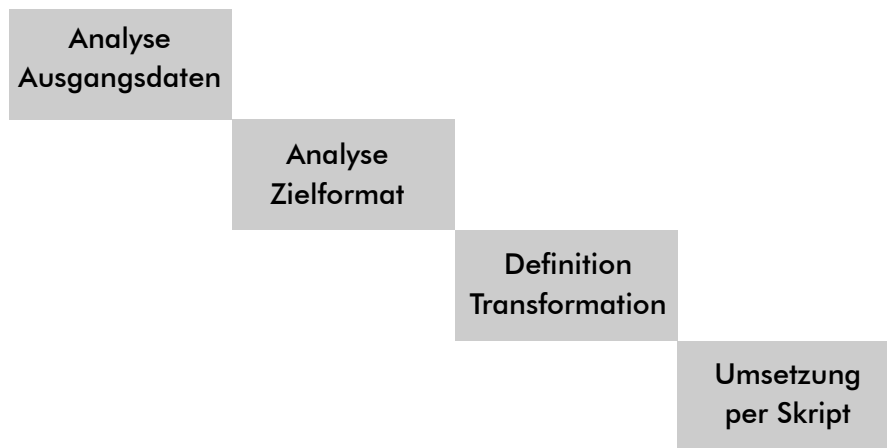


Abb.: Skizze Skriptingprozess

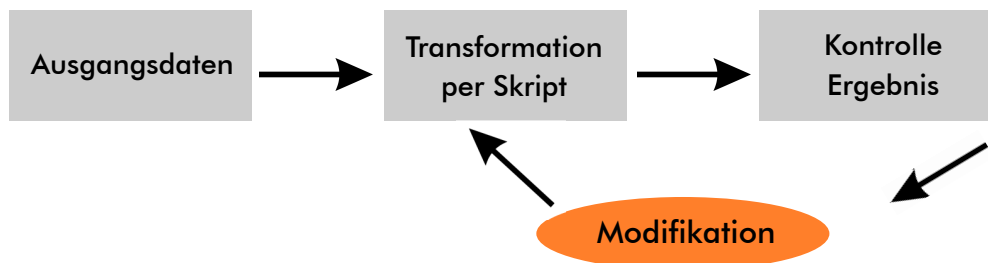


Abb.: Skizze Skriptingprozess II

---

## TUSCRIPT-Grundlagen

---

TUSCRIPT ist die eigene, hochentwickelte, extrem flexible und effiziente Skriptsprache von TUSTEP, speziell für die Verarbeitung von Textdaten, besonders auch SGML/XML.<sup>62</sup>

Exemplarische ➡TUSCRIPT -Snippets (im April 2017: 152 Beispiele) von Thomas Kollatz können im Portal `Rosetta-Code` recherchiert und mit diversen anderen Programmier- und Skriptsprachen verglichen werden.<sup>63</sup>

TUSCRIPT geht auf die ursprünglich sog. *Kommandomakros* zurück. Daher werden die Begriffe TUSCRIPT, Makros und seltener Kommandomakros oft synonym verwendet. Im TUSTEP-Handbuch ist die Beschreibung zu TUSCRIPT im Kapitel »Makros« zu finden.

Mit TUSCRIPT können auch externe Dateiformate (z.B. TXT, XML oder HTML, = sog. »Fremddatenformate« oder »Systemdateien«) direkt, d.h. ohne vorherige Umwandlung ins TUSTEP-interne Format, verarbeitet werden.

---

<sup>62</sup> <http://tustep.wikispaces.com/TUSCRIPT>.

<sup>63</sup> <http://rosettacode.org/wiki/Category:TUSCRIPT>.

TUSCRIPT vs. parametergesteuerte Programme (#KOPIERE, #NUMMERIERE...): Stärken/Schwächen

Eigenschaft	TUSCRIPT	parametergesteuerte Programme
Syntax/ Nomenklatur	eingängige Syntax, sprechende Nomenklatur (EXCHANGE), etwas mehr Schreibaufwand	stark reduzierte, aber konsistente Nomenklatur (X, XX, XXX)
Wartung	einfach zu warten aufgrund der i.d.R. eingängigen Programmstruktur	komplexe Sprungtabellen und Wahlschalter können die Wartung erschweren
Lernkurve	angelehnt an gängige moderne Skriptingsprachen	Orientierung an klassischen Programmiersprachen wie FORTRAN und C
Funktionsvielfalt	bessere Interaktion mit dem Web, Möglichkeit zur Benutzerführung, GUI-Optionen, der Betriebssystemebene, Verarbeitung von Fremddaten und UTF-8-Fähigkeit <sup>64</sup>	sehr leistungsfähige Syntax, aber reduzierter Funktionsumfang
Kombinationsmöglichkeiten	TUSCRIPT und parametergesteuerte Programme ...	... können problemlos gemeinsam genutzt werden

### Vorteile TUSCRIPT

- relativ eingängige Syntax und sprechende Nomenklatur
- i.d.R. einfacher zu warten als z.B. #KOPIERE-Sprungtabellen, allerdings im Vergleich zu parametergesteuerten Programmen mehr Notationsaufwand
- Ähnlichkeit mit modernen Skriptsprachen
- größere Funktionsvielfalt von TUSCRIPT im Vergleich zu parametergesteuerten Programmen:
  - Interaktion mit dem Web
  - Interaktion mit Betriebssystemebene (z.B. via Zwischenablage, Abfrage von Variablen, Ausgabe von Meldungen über Windows-Fenster)

<sup>64</sup> <http://tustep.wikispaces.com/TUSCRIPT+-+Dateneingabe+im+Browser>.

- Verarbeitung von Fremddaten (z.B. XML) ohne vorheriges #UMWANDLE<sup>65</sup>
  - Interaktion mit dem Benutzer während der Laufzeit (z.B. über graphische Benutzeroberflächen)
  - Aufruf von externen Programmen
  - Eingabe von UTF-8-Daten via Webbrowser<sup>66</sup>
- TUSCRIPT und parametergesteuerte Programme können zusammen verwendet werden
  - Möglichkeit zur Strukturierung von TUSCRIPT-Routinen durch Verwendung von SECTION, SUBMACRO

Viele Grundlagen der TUSTEP-Benutzung gelten äquivalent bei der Verwendung von TUSCRIPT, etwa die Regeln zur Benennung von Dateinamen oder das Pattern Matching. Die TUSCRIPT-Syntax folgt jedoch anderen Regeln als die der parametergesteuerten Programme. Die Positionierung der Anweisung (z.B. durch Einrückungen) ist frei wählbar. Häufig finden sich für die aus den parametergesteuerten Programmen gewohnten Operationen (z.B. xx-Parameter in #KOPIERE zum Austauschen von Textteilen) Entsprechungen in TUSCRIPT (EXCHANGE-Funktion). Die TUSCRIPT-Funktionen können unterschiedliche Komplexitätsgrade annehmen und je nach Anwendungszweck passend ausgewählt werden, z.B.:

Definitionen:<sup>67</sup>

```
EXCHANGE (var, xtab, apos, epos, xnum, xanz)
EXCHANGE (var, xtab, apos, epos, xnum, xanz, num, anz, xtabx)
```

exemplarischer Aufruf:

```
SET neu = EXCHANGE (alt, tausch)
SET neu = EXCHANGE (alt, t, a, e, 2, 1, 1, 0)
```

---

<sup>65</sup> #UMWANDLE ist notwendig, weil die parametergesteuerten Programme nur Dateien im TUSTEP-eigenen Speicherformat verarbeiten können.

<sup>66</sup> <http://tustep.wikispaces.com/TUSCRIPT+-+Dateneingabe+im+Browser>.

<sup>67</sup> TUSTEP-Handbuch, Beschreibung der EXCHANGE-Funktion.

TUSCRIPT ist nicht(!) case-sensitive, d.h. es macht keinen Unterschied, ob eine Variable als »neu«, »Neu«, »NEU«, »NeU« etc. deklariert wird. Es wird zwischen *normalen Variablen* beliebigen Inhalts und *Sternvariablen*,<sup>68</sup> die mehrere Zeilen enthalten, unterschieden.

Bsp.:

```
SET wort = "Once"
SET satz = *
DATA Once more unto the breach,
DATA dear friends,
DATA once more
```

TUSCRIPT erwartet keine Deklaration, ob eine zu initialisierende Variable mit dem Datentyp Character, Integer, String o.ä. belegt werden soll. Automatisch unterschieden wird jedoch bei der Variablenbelegung zwischen numerischen Belegungen einerseits und gemischten bzw. nichtnumerischen Belegungen andererseits. Numerische Variablendefinitionen benötigen keine Anführungszeichen und eignen sich für Berechnungen.

### Aufbau eines TUSCRIPTs

Ein vollständiges TUSCRIPT, das in Form eines Kommandos aufgerufen werden kann, benötigt die folgende Zeile zur Deklaration von optionalen Spezifikationen<sup>71</sup>

```
$$!
```

in der zugleich Variablen initiiert und belegt werden können, z.B.:

```
$$! quelle=textfile.tf, ziel=textausg.xml
```

Zur Festlegung des Interpretationsmodus ist folgende Zeile einzufügen:

```
$$ MODE TUSCRIPT, {}
```

Lässt man die geschweiften Klammern weg, wird aus Kompatibilitätsgründen nach wie vor der Interpretationsmodus der <>-Parameter angewandt.<sup>72</sup>

<sup>68</sup> Hinweis insb. für Testläufe: Sternvariablen können nicht(!) mit PRINT auf den Bildschirm ausgegeben werden. Um dies zu umgehen, kann die Sternvariable mit der Funktion JOIN (s.u. »Datensätze zusammenfassen«) zunächst zusammengefasst, gegebenenfalls modifiziert und anschließend ausgegeben werden.

<sup>69</sup> Zum Umgang mit Segmentdateien s. <http://tustep.wikispaces.com/Grundlagen+Segmentdatei>.

<sup>70</sup> <http://tustep.wikispaces.com/Grundlagen+Makrodatei>.

<sup>71</sup> Die Deklarationszeile muss am Anfang eines TUSCRIPTs stehen. Davor sind nur Kommentarzeilen erlaubt.

<sup>72</sup> S. hierzu im TUSTEP-Handbuch die Kapitel »{}-Parameter« bzw. »<>-Parameter«.

Beispiel für ein vollständiges TUSCRIPT, das einen Satz ins Ablaufprotokoll auf Kommandoebene ausgibt:

```
$$!  
$$ MODE TUSCRIPT, {}  
PRINT "Das ist ein TUSCRIPT-Beispiel."
```

Ausgeführt werden können TUSCRIPT-Makros mit folgenden Kommandos:

```
$$?$$datei.m  
#MA, datei.m
```

Direkt aus dem Editor heraus kann ein TUSCRIPT mit folgender Anweisung gestartet werden. Damit wird die Datei `datei.m` als MARKO ausgeführt:

```
x $$?$$datei.m  oder  
x #MA, datei.m
```

Ein aktuell im Editor geöffnetes TUSCRIPT kann mit folgender Anweisung ausgeführt werden:

```
x $$?$$<editor>
```

bzw.

```
x #MA, <editor>
```

Wurde ein TUSCRIPT-Makro in einer Segmentdatei gespeichert und wurde die Segmentdatei als eine von drei Makrodateien definiert, kann der Aufruf folgendermaßen erfolgen:<sup>73</sup>

```
$makroname
```

---

## Befehlssyntax

---

TUSCRIPT unterscheidet *Anweisungen* und *Funktionen*. Funktionen werden grundsätzlich auf Variablen angewandt, während Anweisungen unabhängig von Variablen genutzt werden können. Im Folgenden werden zunächst grundlegende Notationsprinzipien von TUSCRIPT vorgestellt, die anschließend in praktischen Beispielen Anwendung finden.

Bsp. für Anweisungen:

```
PRINT "Hallo"  
LOOP...END LOOP
```

---

<sup>73</sup> Vgl. im TUSTEP-Handbuch den Abschnitt »Makro/Allgemeines/Aufruf«.



Bsp. für eine Funktion:

```
SET var_neu = EXCHANGE (var_alt, "|a|ä|")
```

Zur Initialisierung und Belegung von Variablen sollte grundsätzlich die Anweisung SET vorangestellt werden:

```
SET txt = "Once more unto the breach, dear friends, once more"
```

Mit dieser Anweisung wird die Variable »txt« initialisiert und mit dem nachfolgend in Anführungszeichen geschriebenen Text belegt.

#### »Hello World« (1)

---

- auf Kommandoebene: Anlegen einer permanenten Datei »hello.m«
- Eintragen der folgenden Zeilen:

```
$$!  
$$ MODE TUSCRIPT, {}  
PRINT "Hello World"
```

- Ausführen auf Kommandoebene: #MA,hello.m

#### »Hello World« (2)

---

- Anlegen einer permanenten Datei »hello2.m«
- Aufgabe: Modifikation des vorherigen Beispiels durch Verwendung einer Variablen für den Text:

```
$$!  
$$ MODE TUSCRIPT, {}  
SET txt = "Hello World"  
PRINT txt
```

- Ausführen auf Kommandoebene: #MA,hello2.m

#### »Hello World« (3)

---

- Anlegen einer permanenten Datei »hello3.m«
- Aufgabe: Modifikation des vorherigen Beispiels durch Verwendung von zwei Variablen für einen zusammensetzenden Text:

```
$$!  
$$ MODE TUSCRIPT, {}  
SET txt = "Hello World"
```

```
SET user = "Max Meier"  
PRINT txt, ", ", user
```

- Ausführen auf Kommandoebene: #MA,hello3.m

## Kommentare

---

Kommentarzeilen können innerhalb von TUSCRIPT mit einem Minus (»–«) eingeleitet werden (Voraussetzung: Einstellung `MODE TUSCRIPT`). Zeilenrestkommentare beginnen mit einer Tilde (~). Diese Kommentare werden bei der Ausführung des Programmcodes nicht ausgewertet und können vom Anwender genutzt werden, um sein Vorgehen zu dokumentieren. Bsp.:

```
– Kommentar: Belegung der Variable var_1 mit Inhalt  
SET var_1 = "Text"  
SET var_2 = var_1 ~ und hier der Zeilenrestkommentar
```

---

## Grundlegende TUSCRIPT-Funktionalitäten

---

### Austauschen innerhalb von Variableninhalten

---

Für die Modifikation von Variableninhalten kann z.B. die Funktion `EXCHANGE` genutzt werden, für die sich in aller Regel die Definition einer Exchange-Table (`X_TABLE`)<sup>74</sup> anbietet, z.B.

```
SET var = *
DATA Es war einmal eine Königstochter,
DATA die ging hinaus in den Wald
BUILD X_TABLE x1 = *
DATA |Königstochter|Waidfrau|
DATA |ging hinaus in den|jagte im|

SET var = EXCHANGE (var, x1)
```

Man definiert also zunächst eine `X_TABLE`, die aus mehreren Zeilen (= mehreren Austausch-Anweisung) bestehen kann, und gibt dort äquivalent zur Notation im Editor bei der Austausch-Anweisung an, welche Suchzeichenfolge in welche Zielzeichenfolge überführt werden soll. Anschließend wird diese `X_TABLE` mit der Funktion `EXCHANGE` auf den Inhalt einer Variable angewendet, d.h. die Austauschvorschriften werden ausgeführt.

Regelmäßig soll eine Austausch-Anweisung nur in einem bestimmten Bereich ausgeführt werden, z.B. nur innerhalb von Fußnoten, im Apparattext oder in Metadatenabschnitten. Hierzu kann eine erweiterte Version der `EXCHANGE`-Funktion eingesetzt werden.

Exemplarische Aufgabenstellung:

In einem Text (Inhalt der Variable `txt`) sollen innerhalb des Fußnotentextes (`<fn>`) und innerhalb von Einschaltungen (`<e>`) alle Sperrungen in Kursivierungen ausgetauscht werden.

```
BUILD X_TABLE x1 = *
DATA |<s>|<i>|
DATA |</s>|</i>

SET txt = EXCHANGE (txt, x1, ":<fn>:<e>:", ":</fn>:</e>:")
```

---

<sup>74</sup> Austausche- oder Suchtabellen dienen zur Definition von Austausch- bzw. Suchzeichenfolgen, auf die in anderen Makroanweisungen Bezug genommen werden kann.

**Alternativer Ansatz mit einer Search-Table (S\_TABLE):**

```
BUILD S_TABLE anf = "|<fn>|<e>|"
BUILD S_TABLE end = "|</fn>|</e>|"
BUILD X_TABLE x1 = *
DATA |<s>|<i>|
DATA |</s>|</i>

SET txt = EXCHANGE (txt, x1, anf, end)
```

## Dateizugriffe und Verarbeitung mit TUSCRIPT

---

## Dateizugriffe und Verarbeitung

---

---

### TUSTEP-Dateien einlesen, verarbeiten und ausgeben

---

Aufgabenstellung: Der Inhalt einer TUSTEP-Datei wird eingelesen, (exemplarisch) verarbeitet und der veränderte Inhalt in eine neue TUSTEP-Datei geschrieben. Hierfür muss – wie auf der Kommandoebene und bei der Benutzung des Editors auch – die Quelldatei zunächst angemeldet (Anweisung `OPEN`) und die Zieldatei eingerichtet werden (Anweisung `CREATE`). `OPEN`, das `#ANMELDE` auf der Kommandoebene entspricht, kann das Argument `READ` oder `WRITE` enthalten und statt `-STD-` als Platzhalter für den Inhalt der Systemvariable `TUSTEP_DSK` (s. Projekt anlegen) auch andere Trägerinformationen enthalten. Diese Möglichkeit besteht genauso bei `CREATE`. Hier kann für das erste Argument zwischen verschiedenen Optionen zum Dateieinrichten gewählt werden:

- `SEQ-P` permanente TUSTEP-Datei
- `SEQ-T` temporäre TUSTEP-Datei
- `SEQ-O` permanente TUSTEP-Datei anmelden oder, falls nicht vorhanden, anlegen
- `SEQ-E` temporäre TUSTEP-Datei löschen oder, falls nicht vorhanden, anlegen
- `FDF` Fremd-Datei (ebenfalls mit den Optionen `P`, `T`, `O`, `E`)

Darüber hinaus können mit der Anweisung auch Verzeichnisse, Banddateien und `RAN`-Dateien im TUSTEP-Format angelegt werden.

Die Anweisungen `FILE/ERASE` bzw. `FILE/ERASE/PRINT` dienen dazu, einen Variableninhalt (der nach dem Gleichheitszeichen steht) in eine Variable zu schreiben, deren Inhalt – falls vorhanden –, überschrieben wird. Ist zudem `PRINT` angegeben, wird eine Meldung ins Ablaufprotokoll gegeben.

Eine Datei soll angemeldet, ihr Inhalt in eine Variable eingelesen, modifiziert und in eine neue Datei geschrieben werden.

---

### Anwendungsbeispiel

---

```
$$!  
$$ MODE TUSCRIPT, {}  
- Bsp.skript "TUSTEPFILE"  
- TUSTEP-Datei einlesen, Inhalt verarbeiten, Ergebnis ausgeben  
SET quelle = "frosch_reg.tf"  
SET ziel = "frosch_reg1.tf"  
- Datei zum Lesen anmelden, andernfalls Fehlermeldung  
ERROR/STOP OPEN (quelle, READ, -STD-)
```

---

```
- Datei anlegen bzw. anmelden, andernfalls Fehlermeldung
ERROR/STOP CREATE (ziel, SEQ-O, -STD-)
- Quelldatei in Variable einlesen
SET txt = FILE (quelle)
- Text modifizieren
BUILD X_TABLE html = *
DATA |<p>|<p style="font-size: 20px;">|
SET txt = EXCHANGE (txt, html)
- modifizierten Text in Zielfeld schreiben
FILE/ERASE $ziel = txt
```

## XML-Auszeichnungen mit TUSCRIPT



---

XML-Dateien einlesen, verarbeiten und als XML ausgeben (I)

---

Im ersten Beispiel wird die XML-Datei mit dem Kommando #UMWANDLE nach TUSTEP konvertiert, bevor die eigentliche Verarbeitung beginnt und anschließend wieder mit #UMWANDLE nach XML konvertiert. Im zweiten Fall wird die XML-Datei direkt verarbeitet, was mit parametergesteuerten Programmen nicht möglich ist (vgl. Unterscheidung oben).

Um innerhalb eines TUSCRIPTs Kommandos ausführen zu können, ist die Anweisung EXECUTE notwendig, die im folgenden Bsp. verwendet wird.

```

$$!
$$ MODE TUSCRIPT, {}
SET quelle = "frosch-1.xml"
SET ziel = "frosch-1be.xml"
- Bsp.skript "XMLPROC1"
- Funktionalität:
- XML-Datei umwandeln, einlesen, Inhalt verarbeiten,
- Ergebnis als XML ausgeben
- Quelldatei zum Lesen anmelden, andernfalls Fehlermeldung
ERROR/STOP OPEN (quelle, READ, -STD-)
- Zieldatei anlegen bzw. anmelden, andernfalls Fehlermeldung
ERROR/STOP CREATE (ziel, FDF-O, -STD-)
- temporäre Zwischendatei einrichten, ggf. löschen (ERASE)
SET s1 = "s1"
ERROR/STOP CREATE (s1, SEQ-E, -STD-)
- XML nach TUSTEP importieren (umwandeln) mit
- parametergesteuerten Programmen
EXECUTE #UMW,QU=frosch-1.xml,ZI=s1,CO=utf8,LO=+
- Text in TUSCRIPT-Variable einlesen
SET txt = FILE (s1)
- Text modifizieren
BUILD X_TABLE html = *
DATA |<p>|<p style="font-size: 20px;">|
SET txt = EXCHANGE (txt, html)
SET s2 = "s2"
ERROR/STOP CREATE (s2, SEQ-E, -STD-)
- modifizierten Text aus Variable txt in Zwischendatei schreiben
FILE/ERASE $s2 = txt

```

- Zwischendatei in XML-Datei exportieren (umwandeln)

```
EXECUTE #UMW,QU=s2,ZI=frosch-1be.xml,CO=utf8,LO=+
```

---

XML-Dateien einlesen, verarbeiten und als XML ausgeben (2)

---

In diesem Beispiel sollen XML-Dateien unmittelbar (d.h. ohne vorheriges #UMWANDLE) verarbeitet werden. Hierfür ist die TUSCRIPT-Anweisung ACCESS notwendig.<sup>75</sup>

```

$$!
$$ MODE TUSCRIPT, {}
SET quelle = "frosch-1.xml"
SET ziel = "frosch-1bel.xml"
$$ MODE TUSCRIPT, {}
- Bsp.skript "XMLPROC2"
- Funktionalität:
- XML-Datei direkt einlesen, Inhalt verarbeiten,
- Ergebnis als XML ausgeben
- Quelldatei zum Lesen anmelden, andernfalls Fehlermeldung
ERROR/STOP OPEN (quelle, READ, -STD-)
- Zieldatei anlegen bzw. anmelden, andernfalls Fehlermeldung
ERROR/STOP CREATE (ziel, FDF-O, -STD-)
- Text modifiizieren
BUILD X_TABLE html = *
DATA |<p>|<p style="font-size: 20px;">|
- Kompilieren kann u.U. Geschwindigkeitsvorteile bringen
COMPILE
- Zugriffe auf Quelle und Ziel definieren
- datensatzweise Verarbeitung, UTF8 als Zusatz für Verarbei-
tung der Fremddatei
ACCESS q: READ/RECORDS/UTF8          $quelle s, txt
ACCESS z: WRITE/ERASE/RECORDS/UTF8 $ziel s, txt
- die eigentliche Verarbeitung läuft in einer Schleife ab:
- max. 999999 Durchgänge, Quelle datensatzweise einlesen,
- Variable txt modifizieren, Datensatz nach Ziel schreiben
LOOP/999999
READ/NEXT/EXIT q
txt = EXCHANGE (txt, html)

```

---

<sup>75</sup> Bitte beachten Sie, dass die Anweisung ACCESS neben den im Nachfolgenden gezeigten noch einige weitere Parametrisierungsmöglichkeiten hat. Da diese u.U. umfangreich werden können, kann die Anweisung nach der Angabe des Dateinamens in zwei Zeilen aufgeteilt werden. Dazu muss am Ende der ersten Zeile nach dem Dateinamen »...« eingefügt werden, um kenntlich zu machen, dass die Anweisung in der Folgezeile fortgeführt wird.

```
WRITE/NEXT z
ENDLOOP
- Zugriffe beenden und Meldung ausgeben
ENDACCESS/PRINT q
ENDACCESS/PRINT z
ENDCOMPILE
```

### Bedingte Ausführung von Aktionen

---

Wie in anderen Programmiersprachen üblich, kennt auch TUSCRIPT die bedingte Ausführung von Aktionen mittels der Konstruktion

```
IF bedingung THEN
    aktion
ENDIF
```

sowie erweitert

```
If bedingung_1 THEN
    aktion_1
ELSEIF bedingung_2 THEN
    aktion_2
ELSEIF bedingung_n THEN
    aktion_n
ELSE
    aktion_m
ENDIF
```

Beispielhaft sei folgende Anwendung skizziert:

Aus den Kinder- und Hausmärchen (KHM) der Brüder Grimm sollen verschiedene Auflagen verarbeitet werden. In Abhängigkeit von der Auflage, die aus dem Dateinhalt ersichtlich wird, soll eine Ausgabe ins Ablaufprotokoll ausgegeben werden, damit der Bearbeiter eine Rückmeldung bekommt, ob er gerade die richtige Datei verarbeitet.

```

$$!
$$ MODE TUSCRIPT, {}
- Bsp.skript "IFTHENBSP"
- Beispiel für die Anwendung einer IF-THEN-ELSE-Abfrage:
- Ausgabe der KHM-Auflage auf den Bildschirm
SET quelle = "frosch_reg.tf"
ERROR/STOP OPEN (quelle, READ, -STD-)
SET txt = FILE (quelle)
- der Einfachheit halber hier Zusammenfassung
- der Datensätze in einen einzigen String
SET txt = JOIN (txt)
IF (txt .CT. ":KHM, 1. Aufl.:") THEN
  PRINT "Textausgabe: Kinder- und Hausmärchen, 1. Auflage"
ELSEIF (txt .CT. ":KHM, 2. Aufl.:") THEN
  PRINT "Textausgabe: Kinder- und Hausmärchen, 2. Auflage"
ELSEIF (txt .CT. ":KHM, 3. Aufl.:") THEN
  PRINT "Textausgabe: Kinder- und Hausmärchen, 3. Auflage"
ELSE
  PRINT/ERROR "Es scheint sich um keine der ersten drei Auflagen"
  PRINT/ERROR "der Kinder- und Hausmärchen zu handeln."
ENDIF

```

### Übungsaufgaben IF-THEN-ELSE

Schreiben Sie für die nachfolgenden Aufgaben jeweils ein eigenes TUSCRIPT. Evt. benötigte Funktionen und Anweisungen, die Sie noch nicht kennen, finden Sie im Handbuch, Kapitel »Makro« oder durch Ihre Dozenten:

- Prüfen Sie, ob Ihr Projektverzeichnis eine Datei namens »bech7.tf« enthält. Falls ja, soll ins Ablaufprotokoll eine Bestätigung ausgegeben werden, falls nein, legen Sie eine Datei namens »bech7\_f.tf« an und geben eine entsprechende Meldung aus.
- Übertragen Sie den Inhalt der Datei »bech7.tf« in eine neue Datei »bech7\_be.tf«. Fragen Sie dabei ab, wie oft der Begriff »Menschenfresser« im Text vorkommt, und lassen Sie sich das Ergebnis in eine Datei »bech7\_res.tf« schreiben.

- 
- Schreiben Sie ein TUSCRIPT, das abprüft, ob in der Datei »frosch\_reg.tf« in einem Datensatz sowohl der »Prinz« als auch der »Menschenfresser« erwähnt werden. Falls die Prüfung positiv ausfällt, geben Sie eine Meldung ins Ablaufprotokoll aus und stoppen Sie die Verarbeitung. Falls nicht, überprüfen Sie, ob der »Prinz« und »Heinrich« in einem Datensatz vorkommen. Falls ja, geben Sie bitte eine Meldung ins Ablaufprotokoll aus und stoppen die Verarbeitung, falls nein, schreiben Sie ein entsprechendes Ergebnis in eine Protokolldatei »frosch\_prot.tf«.

---

### Datensätze zusammenfassen

---

Mit der Funktion `JOIN` können Datensätze zusammengefasst werden. Dabei wird per default ein Hochkomma (') an die Stelle gesetzt, an der zwei Datensätze zusammengefügt wurden. Alternativ können andere Trennzeichen angegeben werden.

```
SET gedicht=*
DATA Königstochter, jüngste,
DATA mach mir auf,
DATA weißt du nicht was gestern
DATA du zu mir gesagt

SET gedicht_neu1 = JOIN (gedicht)
SET gedicht_neu2 = JOIN (gedicht, "$")
```

---

### Datensätze aufspalten

---

Mit der Funktion `SPLIT` können Datensätze aufgeteilt werden. Dabei wird per default bei Hochkommata (') die Unterteilung vorgenommen, was u.U. – wie unten beschrieben – zu unerwünschten Folgen für die Datencodierung führen kann. Es empfiehlt sich daher eine explizite Definition, an welcher Stelle/an welchen Stellen die Datensätze unterteilt werden sollen.

```
SET text = *
DATA Absatzende. $ Hier startet ein neuer Absatz.

SET text = SPLIT (text, |":$:")
ERROR/STOP CREATE ("ziel_split", SEQ-O, -STD-)
FILE/ERASE "ziel_split" = text
```

Variante: Wenn man den senkrechten Strich vor der Definition des Zeichens, an welchem die Datensätze gesplittet werden sollen, weglässt, wird dieses Zeichen beim Aufteilen entfernt (= default-Einstellung).

---

## Inhalte aus Variablen extrahieren

---

Eine sehr nützliche TUSCRIPT-Funktion ist `EXTRACT`, um aus Variablen bestimmte Inhalte auszuschneiden und anderweitig weiterzuverarbeiten. Dabei kann mithilfe der Anfangs- und Endposition sowie weiterer Auswahlkriterien, etwa der Angabe, ab dem wievielten Bereich ausgewählt werden soll, die Extraktion gesteuert werden. Wie bei den o.g. Funktionen kann bei der Verwendung von `EXTRACT` gesteuert werden, ob die Anfangs- und Endposition mit ausgeschnitten werden soll oder nicht.

```
SET text = "bald darauf klopfte es auch an die Thüre und rief: <q>Königstochter, jüngste, mach mir auf!</q> Sie lief hin"

BUILD S_TABLE anf = *
  DATA |<q>|
BUILD S_TABLE end = *
  DATA |</q>|
SET zit = EXTRACT (text, |anf, end|)
PRINT zit
```

---

## Levenshtein-Distanz berechnen<sup>76</sup>

---

Die Levenshtein-Distanz ist ein Maß zur Angabe der Unterschiede zwischen zwei Strings beliebiger Länge. Hierbei wird gezählt, wie viele Operationen (Einfügung, Löschung oder Austausch eines Zeichens) notwendig sind, um aus einem String den Vergleichsstring herstellen zu können. Bsp.:

$LD(\text{baum}, \text{haus}) = 2$  ( $b \rightarrow h, m \rightarrow s$ )

Für die Berechnung der Levenshtein-Distanz zweier Strings können die TUSCRIPT-Funktionen `DISTANCE` und `DISTANCE_EXACT` genutzt werden. Letztere unterscheidet im Gegensatz zur ersteren auch Unterschied in der Groß- und Kleinschreibung.

```
string1 = "Baum"
string2 = "Haus"
lev1 = DISTANCE (string1, string2)
string1 = "Baum"
string2 = "haus"
lev2 = DISTANCE_EXACT (string1, string2)
```

---

<sup>76</sup> <http://tustep.wikispaces.com/Levenshtein-Distanz>.



---

## Mögliche Fehlerquellen

---

Bei der Nutzung von TUSCRIPT werden sehr häufig Tabellen definiert, die an einer späteren Stelle genutzt werden sollen. Ein häufig auftretender Fehler ist die Definition von Tabellen, die anschließend nicht aufgerufen werden und damit wirkungslos bleiben (z.B. Definition einer `X_TABLE`, die nachher nicht in einer `EXCHANGE-Funktion` eingesetzt wird), was zu Irritationen beim Anwender führen kann, wenn eine erwartete Aktion offenkundig nicht ausgeführt wird.

Ein weiterer, häufig auftretender Benutzerfehler ist der Versuch, eine bereits existierende Tabelle erneut zu belegen. Dies ist problemlos möglich, allerdings erst dann, wenn die betreffende Tabelle zuvor mit einer `RELEASE-Anweisung` freigegeben wurde.

Beim Einsatz der Makro-Funktionen ist auf die default-Einstellung zu achten. So kann der Einsatz der Funktion `SPLIT` dazu führen, dass Datensätze an unerwünschten Stellen, z.B. innerhalb eines Wortes, das einen Apostroph enthält, unterteilt werden, weil das Apostroph standardmäßig genutzt wird, um die Stellen zu kennzeichnen, an denen Datensätze aufgeteilt werden sollen. Um unerwünschte Nebeneffekte zu vermeiden, ist eine benutzerseitige Anpassung notwendig.

## Praxistipps zur Anwendung von TUSCRIPT

## einfache Methoden zum Zählen und Rechnen

Wie viele Sprechakte und Bühnenanweisungen enthält Goethes Faust 1 in der Fassung von 1808 (Quelle: Deutsches Textarchiv)?

\$\$!

\$\$ MODE TUSCRIPT, {}

```
-----
- Einfaches Skript zum Zählen der Sprechakte und
- Bühnenanweisungen in einem TEI-codierten Drama.
- Ggf. ist der Zählalgorithmus anzupassen, um
- bspw. mehrere Sprechakte in einer Zeile
- berücksichtigen zu können.
-----
```

```
- Quelldatei definieren, Leserechte anmelden
```

```
SET quelle = "faust1.xml"
```

```
ERROR/STOP OPEN (quelle, READ, -STD-)
```

```
COMPILE
```

```
- gezählt wird nur innerhalb von <body>...</body>
```

```
BUILD S_TABLE atab = *
```

```
DATA |<body>|
```

```
BUILD S_TABLE etab = *
```

```
DATA |</body>|
```

```
- Definition der Zugriffe auf Quelldatei
```

```
ACCESS q: READ/STREAM/RECORDS/UTF8 $quelle sn, a/atab+txt+e/etab
```

```
- gesucht wird nach <sp und <stage [ggf. jeweils mit Attributen]
```

```
BUILD S_TABLE sp = *
```

```
DATA |<sp|
```

```
BUILD S_TABLE st = *
```

```
DATA |<stage|
```

```
SET i1 = 0
```

```
SET i2 = 0
```

```
- i1 und i2 enthalten die Anzahl der Sprechakte bzw. Bühnenanweisungen
```

```
LOOP/99999999
```

```
    READ/NEXT/EXIT q
```

```
IF (txt .CT. sp) THEN
```

```
    i1 = i1 + 1
```

```
ENDIF
```

---

```
IF (txt .CT. st) THEN
    i2 = i2 + 1
ENDIF
ENDLOOP
PRINT ""
PRINT "Ihr Text enthält ", i1, " Sprechakte."
PRINT "Ihr Text enthält ", i2, " Bühnenanweisungen."
PRINT ""
ENDCOMPILE
ENDACCESS q
```

## Zählen in der Vulgata

Fragestellung: Wie viele Bücher enthält die Vulgata?

```

$$!
$$ MODE TUSCRIPT, {}
SET quelle = "vulgata-u.tf"
ERROR/STOP OPEN (quelle, READ, -STD-)
SET ziel = "stat_vulg.tf"
ERROR/STOP CREATE (ziel, SEQ-O, -STD-)
COMPILE
ACCESS q: READ/RECORDS          $quelle, sn.zn/un, txt
ACCESS z: WRITE/ERASE/RECORDS $ziel, sn.zn/un, txt
BUILD S_TABLE ref_anf = *
DATA |<ref n="|
BUILD S_TABLE ref_end = *
DATA | |
- Die Bibel beginnt mit dem ersten Buch Mose Genesis.
- Daher als Startwert für buch2 = Gn.
- Prüfung in jeder Verszeile: Entspricht das aktuelle
- Buch der Zeile (= buch1) dem aktuellen Wert von buch2.
- Falls nicht > neues Buch. buch2 neu belegen
- und Zähler i0 um 1 erhöhen.
SET i0 = 1
SET buch2 = "Gn"
LOOP/999999
READ/IGNORE/EXIT q
IF (txt .CT. "<ref n:") THEN
    SET buch1 = EXTRACT (txt, ref_anf|, |ref_end)
    IF (buch1 .NE. buch2) THEN
        SET buch2 = buch1
        SET i0 = i0 + 1
    ENDIF
ENDIF
ENDLOOP
PRINT/COLOR:F0 ""
PRINT/COLOR:70 " Die Bibel hat 66 offizielle Bücher."
PRINT/COLOR:70 " Die {quelle} enthält: ", i0," Bücher."

```

```
PRINT/COLOR:70 "  Das letzte ist ", buch2, "."
PRINT/COLOR:F0 ""
ENDACCESS q
ENDACCESS z
ENDCOMPILE
```

---

Aufruf von Fremddateien unter Windows mit TUSCRIPT<sup>77</sup>

---

```
#- Satzroutine
Verarbeitung von Fußnoten, Apparaten....
#SA,.....
#- Aufruf des Satzergebnisses (Postscript-Datei)
#MAKRO
$$ MODE TUSCRIPT, {}
SET erg = "satzerg.ps"
BROWSE $erg
*EOF
```

Bsp.: Am Ende einer umfangreichen Satzroutine wird das Ergebnis des Satzlaufes (eine Postscript-Datei) über die Makroanweisung `BROWSE` aufgerufen. Die Anweisung `BROWSE` greift auf die betriebssystemseitige Registry-Funktion zurückgegriffen wird, die bestimmte Dateierweiterungen einem spezifischen Programm zuordnet und öffnet die angegebene Datei mit dem dafür vorgesehenen Programm. Mit der Anweisung `BROWSE` können neben lokalen Dateien auch URLs im Standardbrowser aufgerufen werden.

---

Nutzen anderer Programmiersprachen oder Programme

---

Um das Anwendungsspektrum von TUSTEP zu erweitern, kann man Funktionalität auslagern und die Stärken externer Programme, Programmiersprachen und Markup-sprachen nutzen. Möglich sind unter anderem:

- Einsatz von HTML und CSS, um plattformunabhängige Formulare zur Steuerung von Programmabläufen zu erstellen<sup>78</sup>
- Nutzung einer in Java oder Tcl/Tk geschriebenen GUI zur Benutzerinteraktion
- Auslagerung von mathematischen Operationen auf eine andere Programmiersprache wie C++, Python oder Java
- Nachnutzung bestehender Programme durch Aufruf im Batch-Betrieb
- Ausführen von Backup-Routinen an bestimmten Punkten des Arbeitsablaufes

---

<sup>77</sup> TUSTEP-Handbuch, Kapitel »Makros | Makroanweisungen | Starten und Beenden von Anwendungen«.

<sup>78</sup> Dieselbe Funktion kann mit TUSCRIPT-Fenstern realisiert werden. Allerdings sind die graphischen Gestaltungsmöglichkeiten mit HTML/CSS wesentlich größer, auch wenn sie zumindest in Teilen abhängig vom verwendeten Browser und Betriebssystem sind, insbesondere wenn fortgeschrittene Elemente genutzt werden sollen. Unabkömmlich und gleichzeitig elegant möglich ist die Nutzung von HTML/CSS aus TUSTEP heraus per Webserver, so dass ein Webseitenbesucher Funktionen aufrufen kann, die im Hintergrund wiederum von TUSTEP ausgeführt werden, z.B. die Erstellung von Kollationen.

Für die beschriebenen Möglichkeiten ist in erster Linie die Skriptingsprache TUSCRIPT zu verwenden. Hier kann mittels der Makroanweisung EXECUTE mit externen Werkzeugen kommuniziert werden.

### TUSCRIPT als Teil einer *Toolchain*

#### Ausgangssituation:

Im Rahmen einer TUSCRIPT-Verarbeitung soll ein externes Tool für eine bestimmte Aufgabe genutzt werden (hier: ein Python-Skript zum Berechnen des dekadischen Logarithmus). Zuvor sollen die Daten mit TUSCRIPT vorbereitet, dann an das externe Programm übergeben und anschließend mit TUSCRIPT weiterverarbeitet werden.<sup>79</sup>

```

$$ MODE TUSCRIPT, {}
- ### Vorverarbeitung ###
- =====
- Python-Skript zur Berechnung des dekadischen Logarithmus
- ausführen. Als Argument wird exemplarisch
- der zu berechnende Bruch "10/3" übergeben.
- =====
SET python = "log10_fi.py" ~ Python-Skript zur Berechnung
ERROR/STOP OPEN (python, READ, -STD-)
SET pfad = FULL_NAME (TUSTEP, "log10_fi.py")
EXECUTE "{pfad} 10/3"
- =====
- nach Ausführung des Skriptes wird das Ergebnis aus
- der Ergebnisdatei eingelesen
- =====
- Ergebnis aus Datei einlesen
ERROR/STOP OPEN ("log10out.txt", READ, -STD-)
SET log_erg = FILE ("log10out.txt")
SET log_erg = JOIN (log_erg)
PRINT "Das Ergebnis (aus Datei) lautet: ", log_erg

```

<sup>79</sup> Vgl. <http://tustep.wikispaces.com/Externe+Programme+aufrufen>.



## Nutzung von TUSCRIPT zur ereignisabhängigen Ablaufsteuerung

- Ablaufsteuerung durch die Kombination von Makros und parametergesteuerten Programmen

```

$$! quelle = "quelldatei.rtf", ziel = "zieldatei.xml", iso-
lier = "ja"
$$ MODE TUSCRIPT, {}<code>
-...
-...
IF (isolier .EQ. "ja") THEN
    ERROR/STOP OPEN ("isoturb.p", READ, -STD-)
    SET anw = "#T,isoturb.p,,tmp3"
    FILE/ERASE "komm" = anw
ELSE IF (isolier .EQ. "nein") THEN
    SET anw = ""
    FILE/ERASE "komm" = anw
ELSE
    ERROR/STOP "Bitte für Variable >isolier< ja oder nein angeben."
END IF
DATA #T,komm

```

- ereignisabhängige Steuerung von Einstellungsdateien

```

$$!
$$ MODE TUSCRIPT, {}
- Datei festlegen, welche die einzustellenden Definitionen enthält
SET einstell = "einstell.tf"
SET p_einstell = "p*einstell.tf"
IF ( (einstell .NE. 'FILE') .AND. (p_einstell .NE. 'FILE') ) THEN
    ERROR/STOP "Die angegebene Datei scheint nicht zu existieren."
END IF
IF (einstell .EQ. 'FILE') THEN
    EXECUTE #E,DE={einstell}
    PRINT "Einstellungsdatei {einstell} liegt im aktuellen Projekt"
    PRINT "und wurde definiert."
ELSEIF (p_einstell .EQ. 'FILE') THEN
    EXECUTE #E,DE={p_einstell}

```

```
PRINT "Einstellungsdatei {p_einstell} liegt im Projekt p"
PRINT "und wurde definiert."
END IF
```

---

E-Mail versenden<sup>80</sup>

---

```
$$!  
$$ MODE TUSCRIPT, { }  
- TUSTEP-Datei für Mailtext anlegen  
- Daten in der Datei löschen für den Fall, dass die Prozedur mehrfach  
- ausgeführt wird  
SET datei = "textdatei"  
ERROR/STOP CREATE (datei, SEQ-O, -STD-)  
ERROR/STOP ERASE (datei)  
  
- Datum und Uhrzeit abfragen  
SET datum = DATE(2)  
SET uhrzeit = TIME(1)  
  
- Definition des Mail-Textes als Sternvariable  
SET mailtext = *  
DATA Hier steht ein Probetext:  
DATA Für die Projektdatenbank wurde ein Update  
DATA eingespielt:  
DATA Datum und Uhrzeit des Updates:  
DATA {datum}, {uhrzeit} Uhr  
DATA  
DATA TUSTEPianische Grüße  
DATA Ihr Admin  
  
- Variablen werden belegt: Empfänger, Betreff  
SET an = "arbeitsgruppe@uni-xy.de"  
SET wg = "E-Mail-Versand mit TUSCRIPT"  
  
- Makrofunktion zum Mail-Versand  
SET status = SEND_MAIL (an, -, wg, mailtext, -)
```

---

<sup>80</sup> TUSTEP-Handbuch, Kapitel »Makros | Makrofunktionen | Makrofunktion zum Abschicken einer E-Mail«. Diese Makrofunktion ist derzeit nur für Windows implementiert.  
<http://tustep.wikispaces.com/E-Mail+versenden>.

Das Makro kann unter Windows mit folgendem Kommando aufgerufen werden  
`$?$makroname`.

Das vorstehende Beispielmakro versendet eine Mail mit automatisierten Angaben an eine Mailing-Liste. Sinnvolle Verwendungsmöglichkeiten ergeben sich beispielsweise bei Projekten, in denen mehrere Mitarbeiter an einer Datenbank arbeiten und Informationen über Veränderungen an der Datenbank an die Beteiligten versandt werden sollen. Wird TUSTEP als Serveranwendung betrieben, ist zudem die Generierung von automatischen Nachrichten oder Dokumentauslieferungen denkbar.

Vollständige Makrofunktion `SEND_MAIL (mailadr, ccadr, subject, text, files)`

Als Argumente der Makrofunktion können die Mail-Empfänger, CC-Empfänger, Betreff, die Textdatei sowie anzuhängende Dateien (Anlagen) angegeben werden.

## Makrofenster als GUI zur Anwendungssteuerung

Zur Vereinfachung der Bedienung von TUSTEP-/TUSCRIPT-Routinen können mittels TUSCRIPT Makrofenster erstellt werden, die eine graphische Benutzerführung zulassen und den Nutzer von der manuellen Eingabe von Befehlen entbinden. Die einfachste Möglichkeit besteht in der Abfrage von Variablen (z.B. Quelldatei, Zieldatei, Zeichencode usw.) mittels #\*M.

TUSTEP-Datei-Manager | Aufrufen eines Makros | Beenden

Name BBAWKOLL01 Auswählen \* << >> X

QUELLE	=		
ZIEL	=		
AUSG	=		
LO	=		

Keine Beschreibung vorhanden

Datei anmelden Beschreibung ausblenden Makro aufrufen merken

Abb.: Ergebnis von #\*M für eine Routine mit zwei obligatorischen Spezifikationen

Neben dieser automatischen Lösung für eine einfache Eingabemaske können komplexere TUSCRIPT-Benutzeroberflächen mit mehreren Schichten, Benutzerinteraktion, Überprüfung von Eingabefehlern u.ä.m. geschrieben werden.

**\*\*\* SSB \*\*\***  
**Michael Trauth 2005-2015**

**Satz-Vorlage:**

☐ SFB2: 9,5/12 StempelGaramond, 300×477 [B×H Didot]

**Wählbare Dateien:**

- ☐ CHEAT\_DA.TF
- ☐ FROSCH\_REG.TF
- ☐ FROSCH\_REG1.TF
- ☐ FROSCH1.TF
- ☐ FROSCH1\_BE.TF
- ☐ HANDOUT.TF
- ☐ IFTHENZI.TF

**Gewählte Optionen:**

Zu setzende Datei = -

Postscript-Datei = -

Postscript-Fehlerprotokoll = -

Trennliste(n) = -

Schriftendefinitionen = p\*schriftn.mtr

Grafik-Container = -

eigene Parameterdatei? = -

Auszugebende Seiten = 1(9999)

Seiten-Höhenausgleich = 1

Trennregeln nach den (neuen) deutschen RSR von 1996? ☐ 0

Registersatz zweispaltig (m. kleinerem Schriftgrad)? ☐ 0

Eingriffe in die Typographie colorieren? ☐ 0

Ausgabe von: Grafiken? ☒ \* PS-Kopftext? ☒ \*

Ausgabe im Buchformat (Bounding Box)? ☐ 0

Hilfslinien um den Satzspiegel? ☐ 0

**SPEZIALITÄTEN:**

Paginierung römisch? ☐ 0

Korrekturwert Seitenreferenz =

Fn-Block unten nicht vom linken SSpRand einziehen? ☐ 0

Fn-Nr. in Alternativziffern oben? ☐ 0 unten? ☐ 0

Fußnoten-Nr. unten halbfett & links anschlagend? ☐ 0

Hilfe
Abbruch
Start

Hier bitte die Datei auswählen, die gesetzt werden soll!

[2015-05-28/MTr]

Abb.: GUI zur Steuerung der Routine SSB.P von Dr. Michael Trauth (Trier)

---

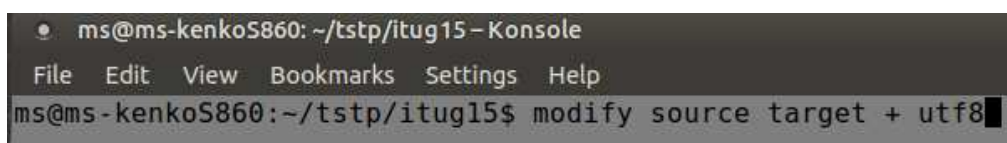
TUSTEP als System-Kommando ausführen

---

TUSTEP kann auf Systemebene mithilfe der Windows-Eingabeaufforderung (`cmd.exe`) bzw. mit dem Terminal unter Linux (z.B. KDE-Konsole) oder macOS gestartet werden:

TUSTEP kann auch als Programm für ein System-Kommando verwendet werden, bei dem keine Interaktion mit dem Benutzer stattfindet. Dabei wird automatisch ein CMD [Command, MS]-Makro aus der INI-Datei [...] ausgeführt und danach TUSTEP wieder beendet.<sup>81</sup>

Nach Einrichtung einer Batch-Datei, die als Konfigurationsdatei fungiert, kann auf der Systemebene der Programmaufruf erfolgen, etwa wie folgt:



```
ms@ms-kenkoS860: ~/tstp/itug15 – Konsole
File Edit View Bookmarks Settings Help
ms@ms-kenkoS860:~/tstp/itug15$ modify source target + utf8
```

Abb.: Aufruf von TUSTEP-Funktionen via Konsole

Nach dem Namen des Batch-Skripts werden Quelle und Ziel genannt, für die Spezifikation »Loeschen« der Wert »+« und für »Zeichencode« der Wert »utf8« angegeben. Durch diese Art der Benutzung ist eine Nutzung von TUSTEP ohne (sichtbaren) Aufruf des eigentlichen Programmpakets möglich.

Dies ermöglicht z.B. die Einbindung von TUSTEP-Verarbeitungen im Rahmen einer *tool chain*, ohne den interaktiven Modus von TUSTEP (Nutzung des Editors) zu gebrauchen.

---

<sup>81</sup> TUSTEP-Handbuch, Kapitel »TUSTEP-Aufruf«.

## Webservice

Abschließend sei auf die Möglichkeit hingewiesen, TUSTEP auf einem Webserver zu installieren und hierüber TUSCRIPT-Funktionalitäten bereitzustellen. Denkbare Einsatzmöglichkeiten sind z.B. die automatische Konvertierung von Textdaten des Nutzers, die einem definierten Schema entsprechen, auf dem Server verarbeitet und an den Nutzer zurückgeliefert werden oder auch die Abfrage einer Datenbank:

**epidat - epigraphische Datenbank**

Startmenu   EDV   Epigraphik   Steinheim-Institut   Kontakt

Suche:


wie?   wann?   wo?   ?

☒ und   ☒ Gesamtzeitraum   ☒ Edition   epidat-suche

☐ oder   ☐ von   ☐ Edition   X

     Übersetzung

Kontakt



<http://steinheim-institut.de/cgi-bin/epidat>

ist lizenziert unter einer

**Creative Commons Namensnennung 4.0 International Lizenz**

**Valid XHTML 1.0 Strict**

Abb.: Recherche in epidat-Datenbank des Steinheim-Instituts Essen, umgesetzt mit TUSCRIPT via CGI,

<http://www.steinheim-institut.de/cgi-bin/epidat>.



## Konvertierung bibliographischer Datenbankexporte

## Konvertierung bibliographischer Datenbankexporte

Aufgabe: Der Export einer Citavi-Literaturdatenbank<sup>82</sup> soll nach TUSTEP importiert, aufbereitet und mit `#*SATZ` ausgegeben werden. Welche Arbeitsschritte sind notwendig?

```

$$!
$$ MODE TUSCRIPT, {}
-----
- Skript zum Import und zur Aufbereitung eines
- Citavi-Datenbankexports (Beispielprojekt im
- Citavi-Basisstil).
- Ziel: Ausgabe mithilfe von #*SATZ
-----

SET quelle = "citbsp.rtf"
ERROR/STOP OPEN (quelle, READ, -STD-)
SET imp = "citbsp1.ttf"
ERROR/STOP CREATE (imp, SEQ-O, -STD-)
SET ziel = "citbsp2.ttf"
ERROR/STOP CREATE (ziel, SEQ-O, -STD-)
- Import nach TUSTEP
EXECUTE #*IMPORT,QU={quelle},ZI={imp},IGN=+,LO=+
- Bereinigungen
BUILD S_GROUP tz = *
DATA ?{-}<>{+}
BUILD X_TABLE x_berl = *
DATA |<document{00}{C:tz}>||</document>||
DATA |<{0}/body>||
DATA |<{0}/section{00}{C:tz}>||
- Überführen in Satztags
BUILD X_TABLE x_satz1 = *
DATA |Überschrift-1>|h1>|
DATA |Citavi-Literaturverzeichnis>|bi>|
- URLs anklickbar machen
BUILD C_GROUP sz = ". , ; ; ) \ ] \ } '"
BUILD C_GROUP nb = "?{-} {+}"
BUILD X_TABLE x_satz2 = *

```

<sup>82</sup> <http://citavi.com/>.

DA-

TA |http{00}{C:nb}{C:sz}|<link><pdfLink>{=0=}</pdfLink>{=0=}</link>|

- Text nach txt einlesen

SET txt = FILE (imp)

SET txt = EXCHANGE (txt, x\_ber1)

SET txt = EXCHANGE (txt, x\_satz1)

SET txt = EXCHANGE (txt, x\_satz2)

- aufbereiteten Text in Zieldatei schreiben

FILE/ERASE/PRINT \$ziel = txt

- Aus Sicherheitsgründen dürfen aus einem TUSCRIPT

- heraus keine Prozeduren mit #MAKRO, #\$\$ oder #TUE

- mithilfe von EXECUTE aufgerufen werden, die selbst

- Kommandos zusammenstellen.

- Standardmakros (#\*makroname) und Makros, die

- keine Kommandofolgen zusammenstellen (#\$makroname)

- können hingegen gestartet werden:

EXECUTE #\*SATZ,QU=<ziel>,MO=SATZ,AUSF=S'D

Herstellung von Korrekturfassungen: Export nach  
TXT, RTF, HTML, EPUB, PDF

---

Von TUSTEP nach RTF: `#*EXPORT`

---

Für den `#*EXPORT` nach RTF ist programmseitig ein XML-Tagset vorgesehen. Dieses Tagset ist in der Beschreibung zum Standardmakro `#*EXPORT` mit zahlreichen Beispielen versehen dokumentiert. Außerdem kann der Benutzer eigene Dokument- und Formatvorlagen definiert werden (im nachfolgenden Bsp. in der Datei `exportmtr01.def`). Diese Definitionen werden in der Regel in eine eigene Datei geschrieben, auf die beim Export verwiesen werden muss.

Texte, die entsprechend ausgezeichnet sind, können dann von TUSTEP aus nach RTF exportiert werden, bspw. mit dem folgenden Aufruf, der zugleich eine benutzereigene Definitionsdatei referenziert:

```
#*EXPORT,eval.tf,eval.rtf,LO=+,DE=exportmtr01.def
```

Eine exemplarische Definitionsdatei für den Export ist die mitgelieferte Datei `exportmtr01_n.def`. Sie enthält Formatierungsvorschriften auf verschiedenen Ebenen (Dokument-, Absatz-, Zeichenebene ...) und kann beliebig an eigene Bedürfnisse angepasst werden.

Bei der Benutzung gibt es ein paar Dinge zu beachten: Wenn Word exportierte Daten lädt, erfolgt keine Interpretation der Anführungszeichen, die sonst nach der Eingabe über die Tastatur in Word je nach Kontext in die typographischen Varianten umgesetzt werden. Anführungszeichen, die in TUSTEP als doppeltes Hochkomma eingefügt wurden, sollten also vor dem Export in die typographisch korrekten Zeichen („Text“ oder »Text«) ausgetauscht werden. Hierfür bietet sich ein `TUSCRIPT` oder `#KOPIERE` an. Das gleiche gilt für die Eingabe des Apostrophs für das einfache Anführungszeichen. Auch »Striche« wie Binde- vs. Gedankenstrich sollten vor dem Export mit einem kleinen Austauschvorgang eindeutig umgesetzt werden, um Interpretationsprobleme zu vermeiden.

---

Sammlung von Export-Skripten in Segmentdatei `export_segm`

---

Die Segmentdatei `export_segm` enthält verschiedene TUSCRIPTs, um TUSTEP-Textdateien in andere Formate zu überführen. Die Funktionen im Einzelnen sind:

- Ausgabe als HTML-Datei mit Abfrage für benutzerspezifisches Syntax-Highlighting
- Ausgabe als HTML-Datei für die Weiterverarbeitung als EPUB
- Ausgabe als PS und PDF-Datei direkt von der XML-Datei ausgehend
- Ausgabe als RTF (z.B. für *round-tripping*)
- Ausgabe als plain text (z.B. für statistische Berechnungen)

notwendige Vorarbeit vor der Nutzung der `export_segm`:

Anmelden der Makrodatei `export_segm`

Dateimanager öffnen: `#*D`

`export_segm` auswählen und Lesen auswählen

Dateimanager verlassen via ESC oder BEENDEN

Datei `export_segm` als Makrodatei definieren:<sup>83</sup>

`#DE,MAKRO=1:export_segm`

definiert die Datei `export_segm` als erste Makrodatei.

Wie oben für das Standardmakro `#*SATZ` gezeigt, können auch selbst geschriebene und definierte Makros über den Dateimanager aufgerufen werden. Hierzu muss die Makrodatei zunächst wie beschrieben definiert werden. Anschließend kann beim Aufruf

`#*M`

über den Dialog Auswählen aus den Standardmakros und eigenen Makrodateien ausgewählt werden. Wählt man die Datei `export_segm` aus, so stehen die hierin enthaltenen Makros zur Verfügung und können mit Angabe der jeweils obligatorischen Variablen aufgerufen werden. Alternativ können die Prozeduren über die nachstehenden Kommandos aufgerufen werden.

---

<sup>83</sup> Es können in einem Projekt bis zu drei Makrodateien definiert werden. Die Makrodateien enthalten unterschiedliche Segmente, in denen Programme abgelegt werden können. In einer normalen Segmentdatei können bis zu 9 999 Segmente enthalten sein. Eine Erweiterung stellen »Mega-Segment-Dateien« mit bis zu 999 999 Segmentplätzen dar.

---

### Aufruf der Ausgabe als HTML mit Abfrage von Patterns für Syntax-Highlighting

---

```
$html, frosch-1.xml, frosch-1.html
```

Anschließend wird abgefragt, ob bestimmte Patterns in der HTML-Darstellung hervorgehoben werden sollen. Hierbei kann das TUSTEP-Pattern Matching benutzt werden, z.B.:

```
Frosch{00}{&a}'digital'
```

 Nach der Eingabe durch den Benutzer öffnet sich der Standardbrowser mit dem Ergebnis. Im Projektverzeichnis findet sich zudem die Ziel-datei `frosch-1.html`.

---

### Aufruf der Ausgabe als modifiziertes HTML als Grundlage zur Konvertierung in ein EPUB mit Calibre<sup>84</sup>

---

```
$html4EPUB, frosch-1.xml, frosch-1a.html
```

Bei der Herstellung ist ggf. darauf zu achten, dass die Einstellungen angepasst werden. In der CSS-Vorschrift ist etwa ein statischer Verweis auf den Windows-Schriftenordner (C:\Windows\Fonts) enthalten, um den Font Ubuntu einbinden zu können. Ein fertig zusammengestelltes EPUB wird bei den Workshop-Unterlagen mitgeliefert: ›Frosch-könig (1. Auflage), TUSTEP – Grimm, Jacob; Grimm, Wilhelm.epub‹.

---

### Aufruf der Ausgabe als PS-/PDF-Datei direkt von der XML-Datei ausgehend

---

```
$satz_fr, frosch-1.xml
```

Anschließend liegen im Projektverzeichnis die Zieldateien `fr_satz.ps` und `fr_satz.pdf` vor. Unter Mac und Windows wird zudem die Postscript-Datei geöffnet, unter Linux die PDF-Datei.

---

### Aufruf der Ausgabe als Plain Text (TXT)

---

```
$plaintext, frosch-1.xml, fr_1.txt, fr_2.txt
```

In die Datei `fr_1.txt` wird der Inhalt der XML-Datei ohne Markup, aber unter Beibehaltung von formatierenden Leerzeichen und Leerzeilen exportiert. In der Datei `fr_2.txt` sind diese Formatierungselemente entfernt.

---

<sup>84</sup> <http://calibre-ebook.com/>.

Aufruf der Ausgabe als RTF mithilfe des Modus ›Export‹ von `#*SATZ`

---

```
$satz2rtf,frosch-1.xml
```

Aus der TEI-XML-Datei wird eine TUSTEP-Textdatei mit den für das Standardmakro `#*SATZ` vorgesehenen Tags erstellt, die anschließend als RTF-Datei ausgegeben wird.



---

**Export von TEI-XML-Daten als RTF-File mit #\*SATZ und TUSCRIPT**


---

Das nachfolgende Skript konvertiert eine TEI-XML-Datei in die vom Standardmakro #\*SATZ erwarteten XML-Auszeichnungen und exportiert das Ergebnis als RTF-File.

```

$$! quelle
$$ MODE TUSCRIPT, {}
- Bsp.skript "SATZ2RTF"
- TUSCRIPT zur Ausgabe von FROSCHE-1.XML als RTF mittels #*SATZ im
- Modus Export. Die Zieldatei wird automatisch erstellt.
- Quelldatei zur Bearbeitung anmelden
ERROR/STOP OPEN (quelle, READ, -STD-)
- Zwischendatei anlegen (hier werden die Umcodierungen gespeichert)
SET fr_umcod = "fr_umcod.tf"
ERROR/STOP CREATE (fr_umcod, SEQ-O, -STD-)
EXECUTE #UM,{quelle},{fr_umcod},CO=UTF8,LO=+
- Zieldatei für den Text anlegen
SET fr_satz = "fr_satz.tf"
ERROR/STOP CREATE (fr_satz, SEQ-O, -STD-)
- Quelldatei in Variable einlesen
SET txt = FILE (fr_umcod)
- Quelldatei von überflüssigem TEI-Markup bereinigen
BUILD S_GROUP tz = *
DATA ?{-}<>{+}
- die aufgezählten Elemente werden in Nichts ausgetauscht "||", d.h. gelöscht
BUILD X_TABLE del = *
DATA |<\?xml{0-0}{C:tz}>||
DATA |<!DOCTYPE{0-0}{C:tz}>||
DATA |<{0}/TEI{0-0}{C:tz}>||
DATA |<{0}/fileDesc>||
DATA |<{0}/title{0-0}{C:tz}>||
DATA |<{0}/publicationStmnt>||
DATA |<{0}/sourceDesc>||
DATA |<p/>||
DATA |<{0}/text{0-0}{C:tz}>||
DATA |#\[0009]||
DATA |#\[2014]|-|

```

```
DATA |<{0}/body>||
DATA |#.,|"|#.'|"|
SET txt = EXCHANGE (txt, del)
- Umbenennung von TEI-Elementen in Satz-Nomenklatur
BUILD X_TABLE um_1 = *
DATA |<{0}/head>|<{+2=}h2>|
DATA |<p>{|}KHM, 1. Aufl. (1812/15) I: 1-5.{|}</p>|<h2>{=2=}</h2>|
DATA |<lg>|<p left="30" size="9">|
DATA |</lg>|</p>|
DATA |<l>*</l>|{+4=}<br/>|
SET txt = EXCHANGE (txt, um_1)
FILE/ERASE/PRINT $fr_satz = txt
DEFINE/TUSTEP fr_satz = fr_satz
- Ausgabe als RTF-Datei und Öffnen
DATA #*SATZ,<fr_satz>, MO=export
```

---

Transformation XML → TXT

---

- mit TUSCRIPT können Fremddateien wie XML und TXT direkt ohne #UMWANDLE (= Konvertierung nach TUSTEP) verarbeitet werden
- das nachfolgende Skript liest eine XML-Datei datensatzweise ein und entfernt alle Tags hieraus, um die Daten anschließend als UTF-8 Textdatei abzuspeichern
- zu ergänzen ist ggf. der Umgang mit Zeichencodes wie &#x2019; oder &amp;

```
$$!  
$$ MODE TUSCRIPT, {}  
SET source = "xmldatei.xml"  
SET target = "ziel.txt"  
ERROR/STOP "Quell-Datei: ", OPEN (source, READ, -STD-)  
ERROR/STOP "Quell-Datei: ", CREATE (target, FDF-P, -STD-)  
BUILD X_TABLE xx1 = "|<*>|"  
ACCESS q: READ/RECORDS/UTF8 $source s, txt  
ACCESS z: WRITE/ERASE/RECORDS/UTF8 $target s, a+txt+e  
COMPILE  
LOOP/999999  
    READ/NEXT/EXIT q  
    SET txt = EXCHANGE (txt, xx1)  
    WRITE/NEXT z  
ENDLOOP  
END COMPILE  
ENDACCESS/PRINT q  
ENDACCESS/PRINT z
```

---

 Stapelverarbeitung, Bsp. TUSTEP → TXT<sup>85</sup>


---

```

#- Kurzes Skript zur Konvertierung mehrerer TUSTEP-Files in
#- einzelne Textdateien
#- Äquivalent können Textdateien in TUSTEP-Dateien
#- konvertiert werden.
#- temporäre Datei "liste" anlegen
#DA,liste,FR=-
#- Dateinamen in die Standard-Textdatei schreiben, welche
#- die Zeichenfolge ".tf" als Extension aufweisen.
#LI,DA,-STD-,POS=|.tf{||}
#- Dateiextension wird für die nachfolgende automatische Verarbei-
tung entfernt
#-
#KO,-STD-,liste,LO=+,PA=*
XX      |.tf||
*EOF

#- das, was zwischen #TUE und EOF steht, soll auf die Dateien
#- angewandt werden, die in der Datei "liste" enthalten sind
#TUE,*,SCHL=liste
#- (1) die Dateien werden zum Lesen angemeldet, (2) es werden
#- Textdateien als Ziel angelegt sowie die (3) die
#- TUSTEP-Dateien mit der Zeichencodierung ANSI in die
#- gleichnamige Textdatei exportiert.
#AN,?0.tf
#DA,?0.txt,FDF-AP
#U,?0.tf,?0.txt,CO=ANSI,LO=+
*EOF

```

---

<sup>85</sup> Unter Stapelverarbeitung wird hier die einheitliche, automatische und sequentielle Verarbeitung mehrerer Dateien mit den gleichen Parametern verstanden. Hiervon ist der Betrieb von TUSTEP im Batch-Modus abzugrenzen. S. hierzu im TUSTEP-Handbuch das Kapitel »TUSTEP-Aufruf | Starten von TUSTEP als System-Kommando«.

# Anlagen

## Cheat-Sheet für die Benutzung von TUSTEP<sup>86</sup>

### Ebenen unterscheiden

---

- Kommandoebene (#-Ebene, »Gib Kommando«) vs. Anweisungsebene (Editor, »Gib Anweisung«)
- Programmieren mit TUSTEP: #KOPIERE, Makro/TUSCRIPT, TXSTEP

### Kommandoebene (Gib Kommando>)

---

- [F4], [ESC] – Sitzung unterbrechen
- [F6], \*D – Aufruf des Dateimanagers
- [F9] – Blättern in der History (rückwärts)
- [F10] – Blättern in der History (vorwärts)

### Editor (Gib Anweisung>)

---

#### Funktionstasten

---

- [F1] – Sprung an den Dateianfang
- [F2] – Sprung an das Dateiende
- [F3] – Nächste Zeichenfolge suchen
- [F4], [ESC] – Editor beenden
- [F5] – Zeige Umgebung des aktuellen Datensatzes
- [F6] – Liste der zuletzt geöffneten Dateien
- [F7] – Zeige aktuellen Datensatz und Zeilen davor
- [F8] – Zeige aktuellen Datensatz und Zeilen danach<sup>87</sup>
- [F9] – Anzeigen der letzten Editor-Anweisungen (rückwärts)
- [F10] – Anzeigen der letzten Editor-Anweisungen (vorwärts)
- [F11] – Bildschirm teilen, aktivieren des linken bzw.  
oberen Fensters
- [F12] – Bildschirm teilen, aktivieren des rechten bzw.  
unteren Fensters

---

<sup>86</sup> Konzept ursprünglich: Ute Recker-Hamm, Thomas Kollatz, Matthias Schneider (02/2015). Modifikation: Matthias Schneider.

<sup>87</sup> In der TUSTEP.INI für den Workshop ist die F8 mit der Anweisung u! zum Umnummerieren der Datensätze vorbelegt.

## Zeige-Anweisungen

---

Zeige-Anweisungen bestehen aus einem Anweisungsnamen (ZN/ZU), gefolgt von drei Kommata, zwischen denen weitere Angaben gemacht werden können, um die Suche zu spezifizieren, und zwei Begrenzungszeichen (= Separatoren), zwischen denen eine oder mehrere Suchzeichenfolgen geschrieben werden. Soll mit einer Zeige-Anweisung nach mehreren Suchzeichenfolgen gesucht werden, werden diese jeweils durch einen Separator voneinander abgegrenzt. Dieses Zeichen kann frei gewählt werden. Es darf allerdings nicht gleichzeitig Separator und Teil der Suchzeichenfolge sein. Bsp.:

richtig:

```
ZN,,,|Frosch|
```

```
ZN,,, -Frosch-
```

```
ZN,,, $Frosch$
```

falsch:<sup>88</sup>

```
ZN,,, $50$$
```

```
ZN,,, -technisch-methodisch-
```

```
Gib Anweisung> ZN,,,|| - zeigeNur
```

```
Gib Anweisung> ZA,,,|| - zeigeAb
```

```
Gib Anweisung> ZB,,,|| - zeigeBis
```

```
Gib Anweisung> ZU,,,|| - zeigeUm
```

```
Gib Anweisung> GGZ - Liste der letzten Zeige-Anweisungen
```

## Austauschanweisungen

---

```
Gib Anweisung> A,,,|suche|ersetze|
```

```
Gib Anweisung> GGA - zeige Liste der letzten Austausche-
anweisungen
```

## Sonstiges

---

```
Gib Anweisung> [STRG]-[K]-[BLANK] - Anzeige einer Übersicht zum
Pattern Matching
```

```
Gib Anweisung> [ALT]-[Q] - Auswahlmene für Eingabe von
>Sonderzeichen<
```

---

<sup>88</sup> Die erste »falsche« Zeige-Anweisung findet nicht den offenbar gesuchten String »50\$«, sondern liefert eine leere Menge zurück, da »\$\$« als Umschalten zu Ausnahme-Zeichenfolgen gewertet wird. Besser wäre also: ZN,,, -50\$-. Die zweite Zeige-Anweisung findet »technisch« und »methodisch«, nicht aber das Kuppelwort »technisch-methodisch«, nachdem gesucht werden sollte, weil der Bindestrich hier zugleich als Separator eingesetzt wurde.

---

## Editor-Anweisungen für Tags

---

Gib Anweisung> [ALT]-[a] - Ergänze Anfangstag

Gib Anweisung> [ALT]-[e] - Ergänze Endetag

Gib Anweisung> [ALT]-[v] - Nächstes offenes Tag Richtung  
Dateiende

Gib Anweisung> [ALT]-[r] - Nächstes offenes Tag Richtung  
Dateianfang

Gib Anweisung> TL - Liste der verwendeten Tags

Gib Anweisung> TLH - Liste der verwendeten Tags hierarchisch

Gib Anweisung> TP - Tagprüfung (Wohlgeformtheit)

Gib Anweisung> TPV - Tagprüfung (Wohlgeformtheit)  
Richtung Dateiende

Gib Anweisung> TPR - Tagprüfung (Wohlgeformtheit)  
Richtung Dateianfang

---

## Definitionen · Pattern Matching

---



---

### Einzelzeichen inklusive Escapezeichen/Maskierung

---

\?	Fragezeichen
\*	Stern
\[	eckige Klammer auf
\]	eckige Klammer zu
\{	geschweifte Klammer auf
\}	geschweifte Klammer zu
\a	Kleinbuchstabe a
\A	Großbuchstabe A
\\	Backslash

---

### Zeichen- und Stringgruppen

---

?	ein beliebiges TUSTEP-Zeichen
*	null bis beliebig viele beliebige TUSTEP-Zeichen
[...]	lokale Zeichengruppe, z.B. m[ae][iy]er
{C:xy}	Charactergruppe xy
{S:xy}	Stringgruppe xy
{-}	nachfolgende Zeichen aus der Zeichengruppe entfernen
{+}	nachfolgende Zeichen in die Zeichengruppe aufnehmen



## Vordefinierte Zeichengruppen

<code>{\a}</code>	Kleinbuchstaben
<code>{\A}</code>	Großbuchstaben
<code>{&amp;a}</code>	Kleinbuchstaben & Großbuchstaben
<code>{\0}</code>	normale Ziffern ohne hochgestellte Ziffern
<code>{&amp;0}</code>	normale Ziffern und hochgestellte Ziffern
<code>{!}</code>	alle Zeichen des ASCII-Zeichensatzes
<code>{@}</code>	alle Sonderzeichen des TUSTEP-Zeichensatzes
<code>{%}</code>	Zeichen, die zur Akzentcodierung nach % stehen können
<code>{;}</code>	TUSTEP-Zeichen mit Ausnahme der Zeichen aus dem ASCII-Zeichensatz

## Häufigkeitsbedingungen in Suchzeichenfolgen

Häufigkeitsbedingungen stehen im TUSTEP Pattern Matching immer vor dem Element, auf das sie sich beziehen:

`|{2}t|`

ist also die Codierung für 2 hintereinander stehende »t«.

<code>{n}</code>	genau n Elemente
<code>{n-m}</code>	n bis m Elemente, möglichst wenig
<code>{n--m}</code>	n bis m Elemente, möglichst viele
<code>{0}</code>	0 oder 1 Element, gleichbedeutend mit <code>{0-1}</code>
<code>{00}</code>	1 bis beliebig viele Elemente, gleichbed. mit <code>{1-0}</code>
<code>{0-0}</code>	0 bis beliebig viele Elemente
<code>{n-0}</code>	n bis beliebig viele Elemente

Bsp. in Anlehnung an das TUSTEP-Handbuch<sup>89</sup>:

<code> A{0}\*B </code>	Suchzeichenfolge bestehend aus einem A (groß- oder kleingeschrieben), gefolgt von einem Asterisk (= Stern), der vorkommen oder fehlen kann (= also fakultativ ist) und einem B (groß- oder kleingeschrieben)
<code> A{00}\*B </code>	Suchzeichenfolge bestehend aus einem A (groß- oder kleingeschrieben), gefolgt von keinem oder beliebig vielen Asterisken und einem B (groß- oder kleingeschrieben)
<code> {2-4}{\0} </code>	Suchzeichenfolge bestehend aus zwei bis vier Ziffern. Diese Ziffern würden allerdings auch in einer größeren Zahl, die aus

<sup>89</sup> S. das Kapitel »{}-Parameter | Parameterart IX | Häufigkeitsbedingungen«.

fünf oder mehr Stellen besteht, gefunden. Dies kann mit folgender Definition ausgeschlossen werden:

```
|{2-4}{\0}||{5-0}{\0}||
```

Es stehe in einem Text die Zeichenfolge »12300000«.

```
|{1-5}{\0}0|
```

 würde die Zeichenfolge »1230« finden.

```
|{1--5}{\0}0|
```

 würde hingegen die Zeichenfolge »12300000« finden.

## Zahlwertbedingungen

---

{#}	Zahl mit beliebigem Wert
{#n}	Zahl mit dem Wert n
{!n}	Zahl mit einem Wert ungleich n
{#n-m}	Zahl mit einem Wert von n bis m
{!n-m}	Zahl mit einem Wert kleiner n oder größer m

## Verweise in Suchzeichenfolgen

---

{+n=}	n-tes Element von links gezählt, a != A
{-n=}	n-tes Element von rechts gezählt, a != A
{+n:}	n-tes Element von links gezählt, a == A
{-n:}	n-tes Element von rechts gezählt, a == A

## Verweise in Ersatzzeichenfolgen

---

{+n=}	n-tes Element von links gezählt
{-n=}	n-tes Element von rechts gezählt
{=n=}	alle Elemente des n-ten Elementbereichs
{=0=}	alle Elemente der Kernzeichenfolge
{+n-m=}	n-tes bis m-tes Element von links gezählt
{+n-0=}	n-tes bis letztes Element von links gezählt
{-n-m=}	n-tes bis m-tes Element von rechts gezählt
{-0-m=}	erstes bis m-tes Element von rechts gezählt
{=n-m=}	alle Elemente des n-tes bis m-ten Elementbereichs
{...+}	... aus Kleinbuchstaben werden Großbuchstaben
{...-}	... aus Großbuchstaben werden Kleinbuchstaben

---

### Sonstiges in Suchzeichenfolgen

---

{ [] }	Wechsel vom linken Rand zur Kernzeichenfolge
{ } ] }	Wechsel von der Kernzeichenfolge zum rechten Rand
{   }	Wechsel von einem Elementbereich in den nächsten


---

TUSTEP-Beschreibungen

---

TUSTEP liefert bei der Installation die aktuellen PDF-Fassungen der Beschreibungen im Installationsverzeichnis mit:

- handbuch.pdf – allgemeine Beschreibung
- satzmakro.pdf – Standardmakro `#*SATZ`
- config.pdf – Installation & Konfiguration
- importexport.pdf – Standardmakros `#*IMPORT`, `#*EXPORT`

Die TUSTEP-Beschreibungen können mit dem  Standardmakro `#*ZEBE` aus TUSTEP heraus als PDF (s.o.) aufgerufen werden. Hierzu sind folgende Kommandos notwendig:

```
#*ZEBE - zeigt die allgemeine Beschreibung an
#*ZEBE,satzmakro - Beschreibung von #*SATZ
#*ZEBE,config - Installation und Konfiguration
#*ZEBE,import oder
#*ZEBE,export - Beschreibung von #*IMPORT/EXPORT
```

Den Dokumentationsstand der aktuell verwendeten TUSTEP-Version erhält man in der *Online-Hilfe*. Diese kann mit dem Kommando

```
#HILFE
```

bzw. im Editor mit der Anweisung

```
hilfe
```

oder `STRG+O` aufgerufen und durchblättert werden. Das Nachschlagen in der Online-Hilfe eignet sich v.a. für erfahrenere Nutzer, die wissen, an welchen Stellen sie im Handbuch nachschlagen müssen.

---

#SUCHE

---

Ein schnellerer Zugriff auf die Dokumentation ist mit dem Kommando

```
#SUCHE
```

möglich. Von mehreren Benutzungsarten und Auswahloptionen der Suche wird hier nur eine besonders praxisrelevante Zugriffsart vorgestellt:

Nach dem Wechsel in die Suche wird per `ALT+I` (oder per Klick auf »Index« in der Fußzeile) der erste Index ausgewählt und der Suchbegriff (oder auch mehrere) aus dem Index ausgewählt. Anschließend kann man sich die Treffer inklusive Umgebung in der Beschreibung mit der Tastenkombination `ALT+U`. Zur Vollansicht gelangt man entweder durch Anklicken der betreffenden Trefferabschnittsüberschriftszeile mit der Maus oder mit `ALT+V` ( $\hat{=}$  Volltext), nachdem man den Cursor im Treffer positioniert hat. Mit `ALT+I`

gelangt man zurück zum Index und kann neue Suchbegriffe eingeben. Die Suche kann über ESC oder die Schaltfläche Gehe zu → Programm beenden verlassen werden.

### Online-Informationsquellen

---

- TUSTEP-Wiki: <http://tustep.wikispaces.com/>.
  - *Informationssammlung mit Kurzeinführungen, Programmbeispielen sowie Tipps & Tricks.*
- Homepage von TUSTEP bei der Universität Tübingen: <http://www.tustep.uni-tuebingen.de>.
  - *Hier kann TUSTEP kostenlos als Open Source für Linux, macOS und Windows bezogen werden.*
- Homepage der International TUSTEP User Group: <http://www.itug.de/>.
  - *Hier kann u.a. die TUSTEP-Mailingliste subskribiert werden, über die Anfragen zu konkreten Problemstellungen sowie TUSTEP- und allgemeine DH-Nachrichten und Veranstaltungshinweise kommuniziert werden. Des Weiteren sind hier Informationen zu TUSTEP-bezogenen Veranstaltungen und Kursen zu finden.*

---

## Hinweise zur Benutzung des TUSTEP-Handbuchs

---

Das TUSTEP-Handbuch ist eine erstklassige Anlaufstelle für Informationen zur Benutzung, Einsatzmöglichkeiten und Steuerungsoptionen von TUSTEP. Nach eigener Beschreibung ist es ein »Nachschlagewerk« und nicht zum »Selbststudium durch Anfänger gedacht«. Aufgrund der Vielzahl von technischen Details, aus denen der Benutzer dieses Buchs schöpfen kann, der Breite an Steuer- und Parametrisierungsoptionen und nicht zuletzt der historisch gewachsenen Strukturen ist das Handbuch nicht für jeden TUSTEP-Einsteiger intuitiv zu benutzen. Daher wird im Folgenden versucht, die wichtigsten Anlaufstellen innerhalb des Handbuchs in Verbindung mit weitergehenden Erschließungsmöglichkeiten zu porträtieren.

---

## Die wichtigsten Anlaufstellen

---

- **Kommandos:** Allgemeines zur Kommando-Syntax und zum Aufruf der Kommandos sowie eine Beschreibung aller TUSTEP-Kommandos. Einige von diesen Kommandos rufen Programme für Grundfunktionen der Textdatenverarbeitung auf, deren Leistung über Parameter spezifiziert werden kann. Diese sind – samt den dort vorgesehenen Parametern – im Kapitel »Parametergesteuerte Programme« ausführlich beschrieben.
- **Makros:** Im Kapitel »Makro« wird die Skriptingsprache `TUSCRIPT` mit sämtlichen Funktionen, Anweisungen und Benutzungshinweisen beschrieben.
- **parametergesteuerte Programme:** Erläuterung von verschiedenen erweiterten Kommandos, die mittels Parameterangabe zu komplexen Problemlösungen kombiniert werden können. Die Ausgangskommandos werden im Kapitel »Kommandos« kurz umrissen.
- **Zeichenvorrat:** Hier findet man die Dokumentation, wie Buchstaben in lateinischen und nicht-lateinischen Alphabeten und Sonderzeichen codiert werden müssen, damit sie in TUSTEP korrekt verarbeitet und bei der Satzausgabe oder bei der Umwandlung in Fremddatenformate (z.B. TXT, RTF oder XML) korrekt wiedergegeben werden können.

## Glossar

Im Folgenden sollen einige wichtige, in dieser Beschreibung genutzte Begriffe aus dem TUSTEP-Umfeld erläutert werden.<sup>90</sup>

**Anweisung:** (1) TUSTEP-Befehl, der im Editor gegeben wird (*Gib Anweisung* >). (2) TUSCRIPT-Befehl.

**Banddatei:** Banddateien sind Dateien, die ein elektronisches Abbild von Magnetbändern darstellen und bspw. für die Versionierung von Texten oder Programmquellcode genutzt werden können.<sup>91</sup>

**Editormakros:** »Im Editor ist ein Makro eine Folge von Steuerbefehlen und/oder Zeichenfolgen, die mit einer Funktionstaste [...], einer Tastenkombination [...] oder der Maus [...] aufgerufen werden kann.«<sup>92</sup>

**ENTER-Taste:** Im allgemeinen Sprachgebrauch werden die RETURN-Taste (= Eingabetaste; unterhalb der Backspace- und oberhalb der Shift-Taste) und die ENTER-Taste (am rechten Rand des NUM-Blocks) häufig synonym verwendet. Es empfiehlt sich jedoch, zwischen beiden Tasten zu differenzieren, da sie in TUSTEP teilweise unterschiedliche Funktionen haben und auch bei benutzerdefinierten Makros unterschiedliche Funktionen annehmen können.

**Fremddatenformat:** TUSTEP verwendet grundsätzlich ein eigenes Datenformat, das für verschiedene Verarbeitungsmöglichkeiten spezifische Eigenschaften aufweist; alle anderen Datenformate, die eingelesen, ausgegeben oder verarbeitet werden können, werden als Fremddatenformat bezeichnet.

**Funktion:** TUSCRIPT-Befehl und Bestandteil der Editor-Funktionalität.

**Kommando:** TUSTEP-Befehl, der auf »Kommandoebene« gegeben wird (*Gib Kommando* >). Kommandos beginnen mit einem Nummernzeichen (#) und können meist bis zum zweiten Buchstaben abgekürzt werden (#KOPIERE = #KOP = #KO).<sup>93</sup> Auf den Kommandonamen folgen in der Regel noch Werte für die einzelnen Spezifikationen.


<sup>90</sup> Die Erklärungen orientieren sich an der jeweiligen Beschreibung im TUSTEP-Handbuch und an den weiteren TUSTEP-Beschreibungen zu #IMPORT/EXPORT, #\*SATZ und Installation/Konfiguration.

<sup>91</sup> »Während der Frühzeit von TUSTEP war das Magnetband das einzige Speichermedium, auf dem größere Datenmengen gesichert werden konnten. Zusätzliche Dateien wurden immer hinter die letzte auf dem Magnetband stehende Datei geschrieben. Dabei durfte eine Datei auch mehrfach mit dem gleichen Namen auf einem Magnetband gesichert werden. Wurde eine Datei auf dem Magnetband überschrieben, waren damit (zumindest logisch) auch alle auf dem Magnetband folgenden Dateien überschrieben. Ein Ersetzen einer Datei auf dem Magnetband war also (wenn es nicht die letzte auf dem Band war) praktisch nicht möglich. Diesem Nachteil stand aber der große Vorteil gegenüber, dass eine Datei wiederholt (z. B. nach jeder größeren Änderung) mit dem gleichen Dateinamen auf ein Magnetband gesichert werden konnte. Auf diese Weise konnte im Bedarfsfall bequem auf die Daten in den einzelnen Entwicklungsstadien der Datei zurückgegriffen werden.« TUSTEP-Handbuch, Kap. »Datensicherung mit Band-Dateien«. Die Banddateien in TUSTEP entsprechen in ihren Eigenschaften den o.g. der Magnetbänder.


<sup>92</sup> TUSTEP-Handbuch, Kapitel »Editor | Organisatorische Anweisungen | Makros aufrufen/definieren/löschen/abfragen«. Vgl. auch »Editor | Tastenkombinationen für Makroaufrufe« und »Editor | Steuerbefehle im Editor«.


<sup>93</sup> Ausnahme: #MBA, #MBE, #MBI, #MBK, #MBL, #MBT. Schon nach dem ersten Buchstaben können abgekürzt werden: #B, #E, #I, #T, #U, #Z).

**Makro:** mehrere Bedeutungen: (1) vom Nutzer zusammengestellte Kombination von Befehlsfolgen. Hier sind zu unterscheiden Kommandomakros (s. Eintrag »TUSCRIPT«), Satzmakros und Editormakros. Satzmakros sind Tags, die vom Benutzer definiert werden können und der Steuerung des Satzvorgangs dienen, Editormakros wiederum sind Kombinationen von Steueranweisungen, Textbausteinen u.ä.m. zur Konfiguration und individuellen Anpassung des Editor. (2) von TUSTEP mitgelieferte, vordefinierte Kombinationen von Befehlsfolgen (»Standardmakros« wie #\*SATZ). (3) Darüber hinaus werden TUSCRIPTs auch als Makros bezeichnet, weil TUSCRIPT auf die sogenannten Kommandomakros zurückgeht.

**Mausmakros:** s.  Editormakros

**Parametergesteuerte Programme:** Programmmodule von TUSTEP, deren Leistung über weitere Steueranweisungen (= Parametern) spezifiziert werden können. Die Notation von Skripten, die auf parametergesteuerte Programmen basieren, weist die Besonderheit auf, dass die horizontale Positionierung der Parameter festgelegt ist. So wird der Name des Parameters (z.B. XX in #KOPIERE) linksbündig auf die Positionen 1–3 geschrieben. Auf der Position 6–7 kann eine optionale Durchgangsnummer ergänzt werden. Weitere Angaben zum Parameter sind ab Position 11 möglich. Eine XX-Parameterkarte (häufig auch XX-Anweisung genannt) darf also nicht beliebig eingerückt werden, sondern muss auf den genannten Positionen zu stehen kommen. Im Unterschied dazu ist bei einem TUSCRIPT-Makro eine horizontale Positionierung nach Gusto des Programmiers möglich.

**Punktmakros:** sind  Editormakros, die mit einem Punkt am Ende des Makronamens gekennzeichnet werden.


**RETURN-Taste:** s.  ENTER-Taste

**Scratch-Dateien:** TUSTEP unterscheidet zwischen permanenten Dateien und temporären Dateien (= Scratch-Dateien). Permanente Dateien bleiben solange auf der Festplatte gespeichert, bis sie vom Benutzer gelöscht werden. Temporäre Dateien dagegen werden von TUSTEP gelöscht, wenn der Benutzer das Projekt mit dem Kommando #BEENDE beendet. Scratch-Dateien eignen sich zum Abspeichern von Zwischenergebnissen oder als digitale »Schmierzettel«, sie stehen nur im jeweiligen TUSTEP-Projekt zur Verfügung. Zu erkennen sind die Dateien an der Form mmmnnnnn.tsf (mmm = TUSTEP-Sitzungsnummer, nnnnnn = laufende Nummer). Mit dieser rein numerischen Ansetzung des Dateinamens werden Namenskonflikte vermieden.

**Segmentdateien:** TUSTEP-Dateien mit einem spezifischen Aufbau in Form von Segmenten, die z.B. einzelne Kapitel eines größeren Dokuments oder Codesnippets enthalten können. Segmente können beispielsweise sehr nützlich sein für die Verwaltung großer Programmbibliotheken. Für die Bearbeitung (und den Aufruf, die Benutzung) der Segmentdateien bzw. Segmente gibt es feste Regeln

**Sprungtabelle:** Tabellenartige Auflistung von Steueranweisungen zur Modifikation des Programmablaufs im Modul #KOPIERE; z.B. Steuerung des Programmverhaltens für den Fall, dass eine definierte Kennung gefunden wird.

**Standardmakro:** Zusammenstellung von Kommandos, die von TUSTEP bei der Installation mitgeliefert werden. Eindeutig zu erkennen sind Standardmakros am Stern in ihrem Aufrufbefehl, z.B. #\*IMPORT oder #\*SATZ. Standardmakros können im Gegensatz zu Kommandos nicht abgekürzt werden.<sup>94</sup>

**Tastaturmakros:** s.  Editormakros

**TUSCRIPT:** eine eigene Skriptingsprache von TUSTEP, die speziell der Verarbeitung von Textdaten dient, aber auch für die Programmierung von GUIs und Interaktionen mit dem Betriebssystem bzw. dem Internet genutzt werden kann. TUSCRIPT geht zurück auf die sog.

<sup>94</sup> Eingabeversuche wie #\*IMP führen folgerichtig zu einer Fehlermeldung.



Kommandomakros, weshalb die Dokumentation im Handbuch nach wie vor mit »Makros« überschrieben ist und im täglichen Sprachgebrauch »TUSCRIPT«, »Kommandomakros« und »Makros« z.T. synonym verwendet werden.

## Lösungen zu Übungsaufgaben

### Übungsaufgaben »Dateiverwaltung«

---

1. Anmeldung zum Lesen auf Kommandoebene: `#AN,LE=bech7.tf` oder `#AN,bech7.tf`.

2. `#DA,ueb.tf,SEQ-P`

3. `#KO,bech7.tf,ueb.tf`

Falls bereits Inhalte in der Datei `ueb.tf` stehen, die überschrieben werden sollen:

`#KO,bech7.tf,ueb.tf,LO=+`

4. `#E,ueb.tf`

5. `#LO,DATEI=ueb.tf` oder `#LO,,ueb.tf`