A GNU Emacs Primer

Thomas Ruschival
Thomas@Ruschival.de

30. Januar 2005

Inhaltsverzeichnis

1 Emacs

1.1 Einleitung

Emacs wurde in seiner heutigen Form (allerdings mit weniger Features) 1984 von Richard M. Stallman dem Gründer der Free Software Foundation als erstes GNU-Projekt unter der GPL veröffentlicht. Emacs sollte in jeder GNU/Linux Distribution dabei sein.
Wer will im Jahre 2004, wo es Visual Studio™ , Eclipse™ und Millionen anderer IDEs in denen man alles 'nur' anklicken muss, überhaupt einen Texteditor der mit kryptischen Hotkeys und Befehlen arbeitet. Ich, da ich beim Programmieren meistens die Tastatur benutze. Mit Emacs muß ich ein Wort nochmals nicht tippen, nur weil mir jetzt einfällt es sollte in Großbuchstaben geschrieben sein. In Emacs löst sich dieses Problem mit zwei Tastenkombinationen `M-b` (ein Wort zurück) `M-u` (Wort in Großbuchstaben).
Irgendjemand hat Emacs mal als "thermonuclear texteditor" bezeichnet, was der Funktionalität und Komplexität durchaus gerecht wird.
Diese Kurzreferenz erhebt keinerlei Ansprüche auf Vollständigkeit. Ich habe mich darauf beschränkt wie Emacs das tägliche Leben am Rechner vereinfacht. Funktionen wie Emailclient, Newsclient, Webbrowser, Terminkalender, Tetris, life und Türme von Hanoi habe ich bewusst weggelassen, ich brauche sie nicht.
Wenn was nicht tut, schaut bitte selbst darum warum es nicht tut. Im Internet bei gnu.org [4] oder in Emacs selbst ist eine vortreffliche Hilfe eingebaut.
Diese Kurzreferenz soll nur als Appetitanreger dienen sich mit Emacs weiter zu beschäftigen. Die mehr oder minder aktuelle Version dieses Machwerks findet sich im Internet unter http://doomgarage.de/~ruschi/stuff/ [5] zusammen mit meiner .emacs-Datei 6.1 und ein paar nützlichen 'modes'.

1.2 Bedienung

Kommandos werden in Emacs immer mit `C-` und `M-` eingegeben.
`C-` steht für die Controltaste (Strg)
`M-` steht für die Metataste (Windows, Apfel), man kann auch ESC drücken, finde ich nur nicht ergonomisch.
`C-x C-s` heißt also zuerst `'Control'` und `'x'` , danach `'Control'` und `'s'` gleichzeitig drücken, wobei man auch den kleinen Finger auf `'Control'` lassen kann und `'x'` und `'s'` hintereinander drücken.
Mit die wichtigsten Kommandos in Emacs sind:

abbrechen: `C-g` **undo:** `C-x u`

Mit `M-x` werden im Minibuffer alle Kommandos als Wort eingeben, `tab` zeigt die möglichen Kommandos an. Beispiel: `M-x` sa `tab` zeigt einen neuen Buffer 2.1.1 an:

```
Possible completions are:
save-buffer save-buffers-kill-Emacs
save-some-buffers say-minor-mode
```

Emacs hilft weiterhin beim erlernen von Hotkeys, gibt es bereits einen solchen für einen bestimmten Befehl weist Emacs nach der Ausführung im Minibuffer darauf hin.

1.3 Einstellungen

Wird Emacs in einer graphischen Umgebung gestartet können über den Menüpunkt Options Customize Emacs alle Eigenschaften (und das sind verdammt viele) eingestellt werden. Das Mausrad ist per default inaktiv, in Options->Emacs->Environment->Mouse kann die Unterstützung aktiviert werden. Ich empfehle dringend, wenn man mit den Einstellungen zufrieden ist, eine Kopie seiner .emacs Datei wegzusichern. Die Datei .emacs 6.1 enthält alle Benutzerdefinierten Einstellungen für Emacs. Hier können auch 'Hotkeys' definiert werden.

1.4 Modes

'Modes' sind Verhaltensvorschriften für EMACS. 'major-modes' werden nach der Dateierweiterung ausgewählt, EMACS unterstützt damit Syntax-highlighting für die entsprechende Sprache, das Menü bekommt neue Unterpunkte und Coding-hilfen werden aktiv. Von Haus aus bringt EMACS schon eine ganze Fülle voll verschiedener 'major-' und 'minor-modes' mit. Ein 'major-mode' ist z.B. der C-mode. Java und C++ verwenden diesen mode per default, denn beide haben ähniche Syntax und Funktionalität. Zu jedem 'major-mode' gibt es 'minor-modes' die das Verhalten detailliert anpassen. Der Benutzer kann sich aber auch eigene Modes installieren, im Internet gibt es für **jede** Sprache ausgereifte Mode-Dateien, die man entweder systemweit oder ins $HOME/.emacs/more-modes/ Verzeichnis ablegen kann. Bei den meisten Gnu/Linux Distributionen sind schon eine Menge Modes und Erweiterungen für EMACS dabei, z.B. eine Java-"IDE" die 'JDE'. Mit der 'JDE' hat man dann auch automatisierte Erstellung von get-/set Methoden, Hotkeys für Zeilen wie "System.out.println()"....
Momentan benutze ich AUCTEX einen erweiterten LaTeX-mode. Bei Mathworks findet sich sogar ein "matlab-mode" , der eine Matlab™ shell in EMACS ermöglicht. Bei Modes, die manuell nachinstalliert werden, muss gegebenenfalls die Datei .emacs 6.1 angepasst werden, um den mode automatisch zu laden. Will man manuell EMACS in einen bestimmten Mode zwingen reicht das Kommando `M-x 'modename'`.

2 Die Oberfläche

2.1 Arbeiten mit 'Buffers' & 'Windows'

Buffers sind Dateien die man editiert. Alle Daten in EMACS sind in Buffers gespeichert
Windows sind Bufferansichten, nicht zu verwechseln mit 'Frames' 2.1.4 die vom Windowmanager verwaltet werden.
Am Anfang grüßt EMACS mit einem seltsamen Text:

```
;; This buffer is for notes you don't want to save, and for Lisp evaluation.
;; If you want to create a file, visit that file with C-x C-f,
;; then enter the text in that file's own buffer.
```

Das ist der 'scratch' Buffer, man kann sich hier Sachen notieren, Abschnitte, zum zwischenspeichern will reinwerfen oder LISP Kommandos absetzen. LISP ist die native Programmiersprache von EMACS.
Darunter ist der 'minibuffer',dort werden mit `M-x` Befehle Eingegeben.
Beispiel: Datei öffnen `C-x C-f`,jetzt fragt EMACS im 'minibuffer' `Find file: ~/`

2.1.1 Buffers

Ein Buffer ist die Kopie einer Datei, erst beim Speichern wird die Datei überschrieben.

Basics	
Hotkey / Befehl	**Bedeutung**
`C-x b` switch-to-buffer	Buffer wechseln
`C-x k` kill-buffer	aktuellen Buffer schliessen
`C-x C-f` find-file	Öffnen einer Datei im aktuellen Buffer
`C-x C-i` insert-file	Datei an Cursorposition einfügen
`C-x C-v` find-alternate-file	Öffnen im aktuellen Buffer alte Datei schliessen
`C-x C-s` save-buffer	Speichern des aktuellen Buffers
`C-x C-w` write-file	Speichern unter einem anderen Namen

2.1.2 Bufferlist

Liste aller gerade offenen Buffer.

Bufferlist `C-x C-b`	
Hotkey / Befehl	**Bedeutung**
`%`	Buffer schützen, nur-lese Status an- & ausschalten
`d`	Buffer zum löschen markieren (wird noch nicht gelöscht)
`x`	markierte Buffer löschen
`s`	Buffer speichern
`m`	Buffer zum laden markieren
`v`	mit `m` markierte buffer in Windows laden
`o`	Buffer in anderem Window laden
`f`	Bufferliste durch diesen Buffer ersetzen
`q`	Bufferliste verlassen (kann auch durch `C-x k` geschlossen werden)

2.1.3 Windows

'Windows' sind SDI-Ansichten in EMACS. 'Buffers' werden in 'Windows' dargestellt.

Windows	
Hotkey / Befehl	**Bedeutung**
`C-x 3` split-window-vertically	aktives Window vertikal teilen
`C-x 2` split-window-horizontally	aktives Window horizontal teilen)
`C-x 1` delete-other-window	alle anderen Windows schliessen
`C-x 0` (Null) delete-window	aktives Fenster schließen
`C-x o` other-window	Wechsel in anderes Fenster
`C-x 4 f` find-file-other-window	Datei im anderen window laden
`C-x 4 0` kill-buffer-and-window	Window und Buffer schließen

2.1.4 Frames

Frames sind echte Fenster, die vom Windowmanager verwaltet werden. EMACS kann mit mehreren Fenstern als MDI Programm arbeiten, ohne für jedes Fenster einen neuen Prozess zu starten. Somit kann übersichtlich mit verschiedenen Dateien gearbeitet werden.

Frames	
Hotkey / Befehl	**Bedeutung**
`C-x 5 2` make-frame-command	Neuen Frame erzeugen
`C-x 5 o` other-frame	in anderen Frame wechseln (umständlich, ich benutzte die Maus)
`C-x 5 0` delete-frame	anderen Frame schließen
`C-x 5 f` find-file-other-frame	Datei in anderen Frame laden
`C-x 5 b` switch-to-buffer-other-frame	Buffer in anderen Frame anzeigen

3 Emacs & Text

3.1 Editieren - Normalo

3.1.1 Cursornavigation

Pfeiltasten sind gut, um im Text von Buchstabe zu Buchstabe zu kommen, große Schritte sind schneller. Wortweises springen erfolgt mit `ALT Pfeiltaste`.

Navigation	
Hotkey / Befehl	**Bedeutung**
`C-a` beginning-of-line	Zum Anfang der Zeile springen
`C-e` end-of-line	Zum Ende der Zeile springen
`M-a`	Zum Anfang des Satzes springen
`M-e`	Zum Ende des Satzes springen
`M-f` forward-word	ein Wort vorspringen
`M-b` backward-word	ein Wort zurückspringen
`M-<` beginning-of-buffer	zum Bufferende springen
`M->` end-of-buffer	zum Bufferanfang springen

3.1.2 Kill & Yank - Copy & Paste the Emacs way

Emacs hat theoretisch nahezu unendlich viele Zwischenablagen, den sogenannten 'killring'. Die Größe wird nur vom Speicher und den Benutzereinstellungen begrenzt. Marken und Auswahlen können mit der Tastatur oder mit der Maus gesetzt werden.

Kill & Yank	
Hotkey / Befehl	**Bedeutung**
`C-Space` set-mark-command	Marke im Text setzen
`C-x C-x` exchange-point-and-mark	Marke und Cursor vertauschen
`C-w` kill-region	Text zwischen Marke und Cursor ausschneiden
`C-Insert` kill-ring-save	Text zwischen Marke und Cursor kopieren
`M-d` kill-word	nächstes Wort löschen
`C-DELETE` backward-kill-word	vorheriges Wort löschen
`M-k` kill-sentence	Satz löschen
`C-k` kill-line	Zeile löschen
`C-y` yank	Letzter Eintrag aus dem 'killring' einfügen
`M-y` yank-pop	Killring durchblättern und einfügen , nach `C-y`

3.1.3 Suchen und Ersetzen

Hier können Reguläre Ausdrücke 3.2.1 eingesetzt werden

Basics	
Hotkey / Befehl	**Bedeutung**
`C-s` isearch-forward	inkrementell Vorwärts suchen
`C-r` isearch-backward	Rückwärts suchen
`M-%` query-replace	ersetzen
`n`	nächster Treffer
`^`	einen Treffer zurück
`.`	diesen Treffer ersetzen und beenden
`!`	alle Treffer ab hier ersetzen
`SPACE`	diesen Treffer ersetzen
`C-s` isearch-forward-regexp	Nach Regulärem Ausruck vorwärts suchen
`C-r` isearch-backward-regexp	Nach Regulärem Ausruck rückwärts suchen
`F4`[1] query-replace-regexp	Regulären ausdruck ersetzen
replace-regexp	Globales ersetzen ohne Rückfrage

[1] Hotkey wurde in .emacs 6.1 definiert

3.2 Editieren - Poweruser

Allen Befehlen kann das 'universal argument' `C-u 'x'` vorangestellt werden, wobei `'x'` eine Zahl ist. EMACS führt dann das nachfolgende Kommando x-mal aus.
Beispiel: `C-u 5 M-d` löscht die nächstem 5 Wörter im Text.
Wer größere Kommandos im Minibuffer eingegeben hat, wie z.B. suchen und ersetzen mit einem komplizierten regulären Ausdruck will den Befehl beim nächsten Gebrauch nicht nochmals eingeben. Hier bietet EMACS eine Befehlshistorie, die man sich mit `M-x list-command-history` anzeigen kann. `C-x ESC ESC` (`M-x` repeat-complex-command) dient zum wiederholen des letzten Befehls. Gibt man zusätzlich eine Nummer ein, wird der entsprechende Befehl aus der Historie wiederholt.

3.2.1 Reguläere Ausdrücke

Reguläre Ausdrücke [2] wie sie beim Suchen & Ersetzen eingesetzt werden können. EMACS-Regexps unterscheiden sich allerdings leicht von grep oder awk [1]. `C-h s` wird die Syntax angezeigt.

Regular Expressions	
Hotkey / Befehl	**Bedeutung**
`C-x C-j`	newline
`C-x TAB`	tabulator
`_`	alle Zeichen außer newline
`|`	logisches Oder
`\`	escape für Sonderzeichen ($,.)
`?`	0-1 mal zutreffend (greedy)
`+`	1-n mal zutreffend (greedy)
`*`	0-n mal zutreffend (greedy)
`^`	Zeilenanfang
`$`	Zeilenende
`[ ]`	Klasse von Zeichen (^ am Anfang negiert die Klasse)
`\( \)`	Gruppe von Zeichen
`<\i>`	Referenz auf Gruppe i

3.2.2 Korrigieren

Funktionen für tippfaule Menschen.

Korrigieren	
Hotkey / Befehl	**Bedeutung**
`M-u` upcase-word	Nächstes Wort in GROSSBUCHSTABEN
`M-c` capitalize-word	Nächstes Wort mit Grossbuchstaben beginnen
`M-l` lowcase-word	Nächstes Wort in kleinbuchstaben
`C-x C-u` upcase-region	Region in GROSSBUCHSTABEN
`M-t` transpose-words	Worte vertauschen
`C-t` transpose-chars	Buchstaben vertauschen
`C-x C-t` transpose-lines	Zeilen vertauschen
`C-x C-u` upcase-region	Region in GROSSBUCHSTABEN
`C-x C-l` lowcase-region	Region in kleinbuchstaben

3.2.3 Abbrev mode - Abkürzungen

Noch mehr Funktionen für tippfaule Menschen. EMACS unterstützt dynamische Abkürzungen und Abkürzungen aus Wordlists.
Dynamische Abkürzungen funktionieren immer. Mit `M-/` werden der Reihe nach alle bisherigen Worte an der Stelle ausgeschrieben, die auf die Buchstabenkombination passen. EMACS schägt zuerst die letzte Möglichkeit vor, bei nochmaligem `M-/` die vorletzte.
Globale Abkürzungen werden automatisch bei einem Leerzeichen nach der Abkürzung ausgeschrieben. Deshalb ist es sinnvoll Buchstabenkombinationen zu verwenden, die nicht als Wort vorkommen.
Der 'abbrev-mode' muß erst gestartet werden, dies geschieht mit `M-x abbrev-mode`. Um den Mode schon zu Beginn einzuschalten, genügt es folgende Zeilen in die Datei .emacs 6.1 aufzunehmen:

```
(setq-default abbrev-mode t)
(read-abbrev-file ‘‘~/.abbrev_defs’’)
(setq save-abbrevs t)
```

Abbrev mode	
Hotkey / Befehl	**Bedeutung**
`C-x a i g` inverse-add-global-abbrev	Globale Abkürzung definieren (Abk. ist das letzte getippte Wort) Vorteil: Ganze Sätze mit Leerzeichen
`C-x a g` add-global-abbrev	Globale Abkürzung definieren (Ausschreibung ist das letzte Wort) Nachteil: Auschreibung ist maximal ein Wort
write-abbrev-file	Wordlist speichern
read-abbrev-file	Wordlist laden
unexpand-abbrev	letzte Auschreibung rückgängig machen
`M-/`	dynamische Abkürzung vervollständigen

3.2.4 Rechteck editieren

Eine absolute Powerfunktion, besonders nützlich, wenn man strukturierten Text mit Datenspalten oder Variabledeklarationen editieren will. Ein Rechteck wird von Zeile/Spalte der Marke bis Zeile/Spalte des Cursors definiert.

Rectangle	
Hotkey / Befehl	**Bedeutung**
`C-x r t` string-rectangle	String in Rechteck einfügen
`C-x r k` kill-rectangle	Rechteck ausschneiden
`C-x r y` yank-rectangle	Rechteck einfügen
`C-x r d` delete-rectangle	Rechteck löschen
`C-x r c` clear-rectangle	Leerzeichen in Rechteck einfügen

3.2.5 Ausblenden

Mit 'Narrowing' kann die Ansicht auf eine ausgewählte Region beschränkt werden, um die Übersichtlichkeit zu erhöhen. 'Narrowing ist per default dektiviert, da es den 'Normalouser' irritiert, wenn plötzlich der Text weg ist. Der Text ist natürlich noch da, man sieht ihn nur nicht. Beim ersten aufruf von `C-x n n` fragt EMACS nach ob 'Narrowing' aktiviert werden soll.

Selectives Editieren	
Hotkey / Befehl	**Bedeutung**
`C-x n n` narrow-to-region	Ansicht auf markierte Region beschränken
`C-x n w` widen	Ansicht wieder auf ganzen Buffer setzen
`C-x n p` narrow-to-page	Ansicht aktuelle Seite setzen
`C-x =`	Info wo man sich befindet, wird auch in der Statusleiste gezeigt

3.2.6 Bookmarks

Ist ein Dokument sehr lang (wie dieses hier) ist scrollen einfach nicht effizient. Besser man sagt direkt wo man hin will.

Bookmarks	
Hotkey / Befehl	**Bedeutung**
`C-x r m` bookmark-set	Bookmark der aktuellen Cursorposition anlegen
`C-x r b` bookmark-jump	Zu einem bookmark springen
`C-x r l` bookmark-bmenu-list	Alle Bookmarks anzeigen
Bookmarks-list `C-x r b`	
`d`	Bookmark zum löschen markieren
`x`	markierte Bookmarks löschen
`s`	alle Bookmarks speichern
`f` oder `ENTER`	Textstelle anzeigen
`r`	Bookmark umbenennen

3.2.7 Registers

Der Killring ist manchmal nicht genug, um ganze Textpassagen oder Sprungmarken zu speichern dienen 'Register'. Ein Speicher wie ein Array, in das man alles mögliche kopieren kann und über einen Index erreicht. In der folgenden Tabelle steht 'n' für eine Ganze Zahl, die Adresse des Registers.

'Registers'	
Hotkey / Befehl	**Bedeutung**
`C-x r s 'n'` copy-to-register	Auswahl in Register 'n' kopieren
`C-x r i 'n'` insert-register	Auswahl aus Register 'n' einfügen
append-register	Auswahl an Register 'n' anhängen
prepend-register	Auswahl vorne an Register 'n' anhängen
`C-x r r 'n'` copy-rectangle-to-register	Rechteck in Register 'n' kopieren

3.2.8 Schönerer Code

Um Programmcode zu formatieren bietet EMACS auch einige Funktionen. Allerdings ist das mit vorsicht zu geniessen. Oft wird der User hier bevormundet. EMACS entscheidet dann in einigen modes selbst wann er einen TAB einfügt. Um diese Freiheit zurückzuerlangen sollte irgendwo in der Datei .emacs 6.1 die Zeile

```
(global-set-key "\C-i" ‘self-insert-command)
```

stehen.
Alternativ kann man auch die Tastenkombination `M-i` benutzen ,um Tabstops einzufügen, das funktioniert in allen modes und hat manchmal noch Extras wie relatives einrücken.

Formatierungshilfen	
Hotkey / Befehl	**Bedeutung**
`F6` [1] indent-region	Region einrücken
`F7` [1] comment-region	Region auskommentieren
`F8` [1] uncomment-region	Kommentare entfernen
`M-;` [1] uncomment-region	Kommentar einfügen

[1] Hotkey wurde in .emacs 6.1 definiert

4 Features in C/C++ & Java Mode

Java und C sehen ähnich aus. Daher ist der Java-mode nur ein 'minor-mode' vom C-Mode. Zwar hat jeder einen eigenen Geschmack wie Code formatiert sein sollte, aber alle sinnvollen Möglichkeiten sind schon in Emacs eingebettet. Mit dem Befehl `M-x c-set-style` wird die Art wie Klammern gesetzt werden definiert. Wenn man auf die Frage " `Which` style?" mit `'?'` antwortet bekommt man folgende Liste:

```
bsd cc-mode ellemtel gnu java k&r linux python stroustrup user whitesmith
```

Um zu sehen sehen wie die einzelnen 'styles' die Klammern setzen und den Code einrücken nimmt man am besten eine kleinen C/Java Code-schnipsel, markiert ihn und wendet den Befehl `M-x` indent-region an.
Ferner ist es leicht möglich Emacs zu sagen wie weit Substatements und andere Codeelemente eingerückt werden soll, mit dem Befehl `M-x` c-set-offset kann die Anzahl der der Leerzeichen für Substatements und andere Codeelemente festgelegt. Um die Liste aller Syntaxsymbole zu sehen gibt man wieder `'?'` ein.

5 Goodies

5.1 'Dired' - Mode

In EMACS ist `dired` ist ein eingebauter Filemanager.

Dired - mode `C-x D` oder `M-x dired`	
Hotkey / Befehl	**Bedeutung**
`f` dired-find-file	Datei öffnen (anlegen falls noch nicht da)
`R` dired-do-rename	Datei umbenennen
`M` dired-do-chmod	Zugriffsberechtigungen ändern
`Q` dired-do-query-replace	Ersetzen in markierten dateien
`G` dired-do-chgrp	Gruppe ändern
`g` dired-revert-buffer	Verzeichnisansicht aktualisieren
`d` dired-flag-file-deletion	Datei zum löschen markieren (wird noch nicht gelöscht)
`x` dired-flag-file-delete	markierte Dateien löschen
`D` dired-do-delete	Datei direkt löschen
`+` dired-create-directory	Verzeichnis anlegen
`=` dired-diff	Diff zwischen markierter und ausgewählter Datei
`~` / `C-u` dired-flag-backup-files	Backups zum löschen markieren / Markierung entfernen
`#` / `C-u #` dired-flag-autosave-files	Autosaves zum löschen markieren / Markierung entfernen
`!` dired-do-shell-command	Shellbefehl auf Datei anwenden
`q`	'Dired' mode beenden

5.2 'Shell' - Mode

EMACS kann Shell Befehle direkt ausführen.

Shell - mode `M-x shell`	
Hotkey / Befehl	**Bedeutung**
`C-c C-c` comint-interrupt-subjob	Subprozess abschiessen
`C-c C-d` comint-send-eof	EOF senden

6 Anhang

6.1 .emacs

Hier ist meine .emacs Datei mit eigenen 'Hotkey'-Definitionen:

```
(custom-set-variables
  ;; custom-set-variables was added by Custom -- don't edit or cut/paste it!
  ;; Your init file should contain only one such instance.
 '(case-fold-search t)
 '(column-number-mode t)
 '(compilation-scroll-output t)
 '(current-language-environment "German")
 '(default-input-method "german-postfix")
 '(display-time-24hr-format t)
 '(display-time-day-and-date t t)
 '(global-font-lock-mode t nil (font-lock))
 '(jde-auto-parse-buffer-interval 80)
 '(jde-compile-option-deprecation t)
 '(jde-compile-option-directory "../classes")
 '(jde-compiler (quote ("javac" "")))
 '(jde-complete-function (quote jde-complete-in-line))
 '(jde-db-initial-step-p nil)
 '(jde-db-query-missing-source-files nil)
 '(jde-enable-abbrev-mode nil)
 '(jde-expand-classpath-p nil)
 '(jde-global-classpath (quote ("." "./lib" "./classes" "../classes" "../lib" "/files/classes")))
 '(jde-import-collapse-imports-threshold 3)
 '(jde-javadoc-gen-destination-directory "../doc")
 '(jde-read-compile-args nil)
 '(jde-run-working-directory "../classes")
 '(lpr-add-switches t)
 '(lpr-page-header-program "pr")
 '(lpr-page-header-switches (quote ("--number-lines=3" "-F")))
 '(mouse-wheel-mode t nil (mwheel))
 '(normal-erase-is-backspace t)
 '(ps-font-size (quote (8 . 10)))
 '(ps-footer-lines 1)
 '(ps-header-lines 2)
 '(ps-header-offset 14.346456692913385)
 '(ps-line-number t)
 '(ps-lpr-switches nil)
 '(ps-paper-type (quote a4))
 '(ps-print-footer-frame nil)
 '(ps-print-only-one-header nil)
 '(ps-printer-name nil)
 '(ps-right-header (quote ("/pagenumberstring load" ps-time-stamp-locale-default ps-time-stamp-hh:mm:ss
 '(show-paren-mode t nil (paren))
 '(speedbar-sort-tags nil)
 '(speedbar-use-images t)
 '(tab-always-indent t)
 '(tab-stop-list (quote (4 8 12 16 20 24 28 32 36 40 44 48 52 56 60 64 68 72 76 80 88)))
 '(tab-width 4)
 '(transient-mark-mode t)
 '(words-include-escapes t)
 '(x-select-enable-clipboard t))
(custom-set-faces
  ;; custom-set-faces was added by Custom -- don't edit or cut/paste it!
  ;; Your init file should contain only one such instance.
 '(default ((t (:stipple nil :background "black" :foreground "white" :inverse-video nil :box nil :strik
 '(cursor ((t (:background "#AAAAAA"))))
 '(custom-button-face ((((type x w32 mac) (class color)) (:background "slategray" :foreground "black" :
```

```
 '(custom-changed-face ((((class color)) (:background "steelblue" :foreground "white"))))
 '(custom-rogue-face ((((class color)) (:background "black" :foreground "pink"))))
 '(font-lock-comment-face ((((class color) (background dark)) (:foreground "olivedrab3"))))
 '(font-lock-doc-face ((t (:inherit font-lock-comment-face))))
 '(header-line ((((class color grayscale) (background light)) (:inherit mode-line :foreground "grey20"
 '(menu ((((type x-toolkit)) (:background "slategray" :foreground "black"))))
 '(mode-line ((((type x w32 mac) (class color)) (:background "slategrey" :foreground "black" :box (:lin
 '(region ((((class color) (background dark)) (:background "dodgerblue"))))
 '(scroll-bar ((t (:background "slategray" :foreground "black"))))
 '(show-paren-match-face ((((class color)) (:background "darkturquoise"))))
 '(widget-field-face ((((class grayscale color) (background light)) (:background "white")))))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; My personal Customization modified 5.9.2004
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(setq load-path (append load-path (list "~/.emacs.d/moremodes/" )))  ;; local mod path
;(global-set-key "\C-i" 'self-insert-command)  ;; take control over the tabs myself
(setq make-backup-files nil)                  ;; no backups
(which-func-mode t)
(fset 'yes-or-no-p 'y-or-n-p)
(set-default-font "-misc-fixed-medium-r-semicondensed--13-120-75-75-c-60-iso8859-15")
(put 'upcase-region 'disabled nil)            ;; n00b-saftey disabled
(put 'downcase-region 'disabled nil)
(put 'narrow-to-region 'disabled nil)
(tool-bar-mode 0) ;; turn off the tool-bar

;;
;; Hotkeys
;;
(global-set-key "\C-x\C-d" 'dired) ;; override list-directory
(global-set-key [f1] 'goto-line)
(global-set-key [f2] 'kill-rectangle)
(global-set-key [f3] 'yank-rectangle)
(global-set-key [f4] 'query-replace-regexp)
(global-set-key [f5] 'revert-buffer)
(global-set-key [f6] 'indent-region)
(global-set-key [f7] 'comment-region)
(global-set-key [f8] 'uncomment-region)

;; got a Sun® Type 6 Keyboard - more keys to bind!!
(global-set-key [f19] 'repeat-complex-command)
(global-set-key [f21] 'undo)
(global-set-key [f23] 'kill-ring-save)
(global-set-key [f25] 'yank-pop)
(global-set-key [f27] 'kill-region)

;;;
;; MATLAB
;;;
(autoload 'matlab-mode "matlab" "Enter MATLAB mode." t)
(setq auto-mode-alist (cons '("\\.m\\'" . matlab-mode) auto-mode-alist))
(autoload 'matlab-shell "matlab" "!!! MATLAB ® rules !!!" t)
(setq matlab-indent-function t)         ; if you want function bodies indented
(setq matlab-verify-on-save-flag nil) ; turn off auto-verify on save
(defun my-matlab-mode-hook ()
  (setq fill-column 90)) ; where auto-fill should wrap
(add-hook 'matlab-mode-hook 'my-matlab-mode-hook)
(defun my-matlab-shell-mode-hook ()
  '())
(add-hook 'matlab-shell-mode-hook 'my-matlab-shell-mode-hook)
```

```
(setq matlab-shell-command-switches '("-nodesktop"))

;;;
;; PHP
;;;
(require 'php-mode)
(add-hook 'php-mode-user-hook 'turn-on-font-lock)
(add-hook 'php-mode-user-hook
'(lambda () (define-abbrev php-mode-abbrev-table "ex" "extends")))

;;;
;; Maple
;;;
(autoload 'maple-mode "maple" "Enter MAPLE mode." t)
   (setq auto-mode-alist (cons '("\\.mws\\'" . maple-mode) auto-mode-alist))

;;;
;; C/C++ & Java Mode
;;;
;;(c-set-style "k&r") ;; my favourite code-look
(setq c-basic-offset 4
indent-tabs-mode nil)
(c-set-offset 'case-label '++)     ;; + is c-basic-offset
(c-set-offset 'substatement '++)
(c-set-offset 'statement-cont '++)
(c-set-offset 'func-decl-cont '++)
(c-set-offset 'arglist-cont-nonempty '++)
(c-set-offset 'inher-intro '++)
(c-set-offset 'inher-cont '++)
(c-set-offset 'arglist-intro '++)
(c-set-offset 'topmost-intro-cont '++)
```

Literatur

[1] Dale Dougherty & Arnold Robbins - *sed & awk* - O'Reilly & Associates 1997

[2] Jeffrey E.F. Friedel - *Reguläre Ausdrücke* - O'Reilly & Associates 1998

[3] Richard M. Stallman - *Gnu Emacs Manual* - GNU Press 2002

[4] http://www.gnu.org/software/emacs/

[5] http://www.ruschival.de/stuff/