

Diese Übersicht für "BASH 3.2" wurde von kg zusammengestellt, ©, October 6, 2007. Aktuelle Version hier:

http://www.oreilly.de/catalog/bashtbger/chapter/bashref_german.pdf

Wichtige Online-Infos sind [bash.1](#) und [bashref.info](#).

Anmerkung: **Optionen** sind so dargestellt (Optionen in Optionen so: `[option]`), **\$Variablen** so. Notwendige Leerzeichen: `_.` "cmd" steht für "command".

Weitere Informationen: "Bash kurz & gut" O'Reilly ISBN 978-3-89721-533-7.

Befehlszeilenoptionen

Die wichtigsten Befehlszeilenoptionen sind:

- `-c` *Befehlszeile* Führt die *Befehlszeile* aus und terminiert.
- `-i` Aktiviert das Verhalten interaktiver Shells.
- `-l`, `--login` Lässt die Bash wie eine Login-Shell starten.
- `-r`, `--restricted` Bash als eingeschränkte Shell (kein cd, absolute Pfade...).
- `-s` Erwartet Befehl(s)zeilen) via Standardeingabekanal.
- `--init-file file`, `--rcfile file` Konfiguration aus angegebener *Datei*.
- `--noprofile`, `--norc` Ignoriert alle Konfigurationsdateien.
- `--verbose` Umfangreichere Ausgaben, zeigt Befehlszeilen vor der Ausführung.
- `--noediting` Deaktiviert Readline-Funktionen in interaktiven Shells.
- `--posix` Schaltet POSIX 1003.2 konformes Verhalten ein.
- `--` Was folgt, sind keine Optionen.
- `+o shopt_option` Aktiviert(-) oder deaktiviert(+) *shopt_option*, siehe unten.
- `--debugger` Aktiviert erweiterten Debugging-Modus.
- `--version` Zeigt Versionsinformationen.
- `--help` Zeigt Kurzhilfe.

STARTUP

Bash liest die erste existierende Datei und führt den Inhalt aus:

Interaktive Login-Shell sucht diese Konfigurationsdateien (in dieser Reihenfolge):
`/etc/profile`, `~/ .bash_profile`, `~/ .bash_login`, `~/ .profile`
Interaktive NICHT-Login-Shell sucht nach: `/etc/bash.bashrc`, `~/ .bashrc`
NICHT-interaktive Shells (in Scripten) verhalten sich so:

```
if [ -n "$BASH_ENV" ]; then . "$BASH_ENV"; fi
```

Im Posix Modus werten interaktive Shells `$ENV` aus und lesen Initfiles.

Remote Shells suchen: `/etc/bash.bashrc`, `~/ .bashrc`

Relevante Befehlszeilenoptionen sind:

- `--norc`, `--noprofile` Verhindern das Einlesen von Konfigurationsdateien.
- `--rcfile file`, `--init-file file` Verwendet alternative Konfigurationsdatei.

Beim Beenden führen Login-Shells `~/ .bash_logout` aus.
Readline wird durch `~/ .inputrc` konfiguriert.

COMPOUND COMMANDS

Mehrere Befehl zusammenfassen:

`cmd1` ; `cmd2` (wird als "Liste" bezeichnet)

Zwei spezielle Varianten ("short circuit test"):

`cmd1 && cmd2` Führt `cmd2` nur aus, wenn `cmd1` Rückgabewert 0 liefert.
`cmd1 || cmd2` Führt `cmd2` nur aus, falls `cmd1` Rückgabewert ungleich 0 (Fehler!).
(*Liste*) Führt die *Liste* in einer Subshell aus.
{ *Liste* ; } Führt die *Liste* in der aktuellen Shell (als "Gruppe") aus.
((*Berechnung*)) Arithmetische *Berechnung*, wie mit `let`.
[[*Berechnung*]] Rückgabewert durch *Berechnung*, wie bei `test`.
(*Berechnung*) Rückgabewert entspricht Ergebnis der *Berechnung*.
! *Berechnung*/Ausdruck Wahr, wenn *Berechnung*/Ausdruck ungleich Null ergibt.
`Ausdruck1 && Ausdruck2` Nur wahr, wenn beide Ausdrücke wahr sind.
`Ausdruck1 || Ausdruck2` Wahr, wenn einer der Ausdrücke wahr ist.

SHELLOPTIONEN

kontrolliert durch `set`:

Syntax: `set --abefhkmnptuvxwBCHP -o Option Arg ...`

Die aktuellen `set`-Optionen enthält `$-`. Viele Optionen lassen sich sowohl in ihrer Kurzform (`-a`) als auch in der Langform (`-o allexport`) angeben. + statt - deaktiviert die entsprechende Option. Diese Option kennt `set`:

- `-a`, `allexport` Markiert neue und modifizierte Variablen und Funktionen automatisch als exportierbar (in das Environment neuer Prozesse).
- `-b`, `notify` Statusänderungen von Hintergrundprozessen sofort melden.
- `-e`, `errexit` Shell terminiert, sobald ein Befehl mit einem Fehler endet.
- `-f`, `noglob` Deaktiviert Dateinamensmuster (z.B. * ist normales Zeichen).
- `-h`, `hashall` Aktiviert Erinnerung für bereits ausgeführte Befehle.
- `-k`, `keyword` Argumente als Zuweisungen in das Environment nehmen.
- `-m`, `monitor` Rückgabewerte von Hintergrundprozessen melden.
- `-n`, `noexec` Befehlszeilen expandieren, aber nicht ausführen.
- `-p`, `privileged` Privileged Modus, in `$ENV` und `$BASH_ENV` angegebene Files werden nicht ausgeführt, Shellfunktionen nicht vererbt, Shelloptionen ignoriert. Abschalten setzt effektive U/GID auf die Werte der realen.
- `-t`, `onecmd` Shell beendet sich nach Ausführen einer Befehlszeile.
- `-u`, `nounset` Wertet ungesetzte Variablen als Fehler bei der Expandierung.
- `-v`, `verbose` Zeigt Befehlszeilen wie eingelesen an, vor Ausführen.
- `-x`, `xtrace` Zeigt nach `$PS4` Befehlszeilen nach Expandierung einfacher Befehle, `for`, `case`, `select`.

- `-B`, `braceexpand` Aktiviert Klammerexpandierungen, voreingestellt.
- `-C`, `noclobber` Verhindert Überschreiben bestehender Files mit `>`, `>&`, und `<>`, nur durch `|`.
- `-E`, `errtrace` Vererbt ERROR-Traps an Funktionen, Substitutionen, Subshells.
- `-H`, `histexpand` !-Historysubstitutionen, voreingestellt in interaktiven Shells.
- `-P`, `physical` Verhindert, dass `cd` symbolischen Links folgt.
- `-T`, `functrace` Vererbt DEBUG-Traps an Funktionen, Substitutionen, Subshells.

- `--` Folgen keine Argumente, werden die Positionsparameter zurückgesetzt.
- `-` Beendet Optionen, alles was folgt, sind Positionsparameter.
- `-o Optionsname` Ohne *Optionsname*, aktuelle Optionen anzeigen; `+o Optionsname` erzeugt wiederverwertbares Format. Diese Optionen sind möglich:
 - `emacs` Emacs-artige Befehlszeileneditierfunktionen.
 - `history` Befehlszeilenhistory, voreingestellt für interaktive Shells.
 - `ignoreeof` Entspricht `IGNOREEOF=10`.
 - `pipefail` Rückgabewert entspricht dem letzten Fehler in einer Pipe (oder Null). Voreingestellt deaktiviert.
 - `posix` Aktiviert POSIX-Modus.
 - `vi` VI-artige Befehlszeileneditierfunktionen.

ERWEITERTE SHELLOPTIONEN

Gesteuert durch `shopt`:

`shopt -pqsu -o Optionsname ...`

- `-s` Aktiviert (set) die angegebene Option.
- `-u` Deaktiviert (unset) die angegebene Option.
- `-q` Unterdrückt Ausgaben (quiet mode); nur der Rückgabewert zeigt an, ob Optionen gesetzt sind.
- `-o` Interpretiert *Optionsname* als `-o`-Argument von `set`.

Diese `shopt` -Optionen werden unterstützt:

- `cdable_vars` Interpretiert `cd`-Argumente als Variablen (nicht Verzeichnisse).
- `cdspell` Erlaubt kleine Fehler in Verzeichnisnamen.
- `checkhash` Überprüft Hash-Einträge auf Existenz vor dem Ausführen.
- `checkwinsize` Überprüft Fenstergröße (aktualisiert `$LINES/$COLUMNS`).
- `cmdhist` Speichert mehrzeilige Befehlszeilen als einen Historyeintrag.
- `dotglob` Berücksichtigt auch mit `."` beginnende Dateinamen bei Expandierungen.
- `execfail` Verhindert das Terminieren nicht-interaktiver Shells, wenn `exec` fehlschlug.
- `expand_aliases` Aktiviert Aliasexpandierung, voreingestellt für interaktive Shells.
- `extdebug` Aktiviert div. Debuggerfunktionen. (`-F` zeigt Dateinamen und Zeilennummern. Falls Fehler in DEBUG-traps auftreten, wird die folgende Zeile übersprungen. Rückgabewert 2 einer Funktion bewirkt `return`).
- `extglob` Aktiviert erweiterte Dateinamensmuster.
- `extquote` Strings der Form `$'String'` und `$(String)` werden expandiert, voreingestellt.

- `failglob` Fehler bei Expandierung von Dateinamen gelten als Fehler.
- `force_ignore` In `$FIGIGNORE` enthaltene Anhänge werden bei Kompletierungen ignoriert, voreingestellt.
- `gnu_errfmt` Fehlermeldungen im Standard GNU Error Format.
- `histappend` Hängt History an die in `$HISTFILE` definierte Datei (statt sie zu überschreiben).
- `histredit` Mit Readline lassen sich Historysubstitutionen editieren.
- `histverify` Lädt mit Readline Zeilen in Editing Buffer für Modifikationen.
- `hostcomplete` Mit Readline sind Kompletierungen für Hostnamen für Wörter mit @ möglich, (voreingestellt).
- `huponexit` Sendet SIGHUP an Jobs, wenn interaktive Login-Shell terminiert.
- `interactive_comments` Ignoriert Text nach #, in interaktiven Shells voreingestellt.
- `lithist` Speichert mit `cmdhist` mehrzeilige Befehle mit eingebetteten NL.
- `login_shell` Shell arbeitet als Login-Shell.
- `mailwarn` Warnt, wenn seit letzter Prüfung auf Mail-Datei zugegriffen wurde.
- `no_empty_cmd_completion` Mit Readline unterbleibt die `$PATH`-Suche für leere Zeilen.
- `nocaseglob` Groß/klein bei Dateinamenkompletierungen ignorieren.
- `nocasematch` Groß/klein bei Mustern in `case` und `[[` ignorieren.
- `nulglob` Muster ohne Treffer werden ignoriert und nicht übernommen.
- `progcomp` Aktiviert programmierbare Kompletierungen (voreingestellt).
- `promptvars` Aktiviert Parameterexpandierungen, Befehls- und arithmetische Expandierungen und Quote Removal für Prompts, voreingestellt.
- `restricted_shell` Versetzt Shell in den eingeschränkten Modus.
- `shift_verbose` Warnt, wenn `shift` zu oft aufgerufen wird.
- `sourcepath` Verwendet `$PATH` für `.` und `source`, voreingestellt.
- `xpg_echo` Builtin `echo` interpretiert Backslash-Escape Sequenzen.

ALIASES Ein Alias ersetzt als erstes Wort in einer Zeile in interaktiven Shells eine Abkürzung durch Text, aber ohne Parameter, was `expand_aliases` verhindert.

`alias -p` Zeigt bestehende Alias-Definitionen.
`alias Name=Definition` Erzeugt "Name" als Abkürzung für *Definition*.
`unalias Name` löscht Alias-Definition *Name*.

SHELL FUNKTIONEN

Erlauben Parameter, lokale Variablen, Rekursion.

Erzeugen: `function Name () {Befehle;} Redirection`
Funktionen zeigt: `set`, `declare`, `typeset` (Option `-f`)
Deklariert lokale Variablen: `local` (Optionen wie `declare`)
Exportieren in Subshells: `export -f`
Return-(exit)-status festlegen: `return Status`

PARAMETER

Folgende spezielle Parameter lassen sich nur referenzieren:

- `$1`, `$2`, `$n...` Positionsparameter Nummer *n*.
- `$0` Name der Shell oder des Shellskripts.
- `$*` Positionsparameter, beginnend bei 1; zwischen "doppelten Quotes" ("`$*`"): als *ein Wort* ("`single word... parameters`").
- `$@` Positionsparameter, beginnend bei 1; zwischen "doppelten Quotes": Parameter in Hochkommata ("`$1`", "`$2`", "`$3`"...).
- `##` Anzahl der Positionsparameter (dezimal).
- `$?` Status der zuletzt ausgeführten (Vordergrund-) Pipeline.
- `$-` Aktuelle Shelloptionen (gesetzt durch `set`).
- `$$` PID der aktuellen Shell.
- `$!` PID des letzten Hintergrundprozesses.
- `$_` Shell-Startup: absoluter Pfad von Shell oder Skript, danach: letztes Argument des vorigen Befehls.

`set -- Liste aus Wörtern` Weist Positionsparameter neu zu: `$1` (Liste), `$2` (aus), `$3` (Wörtern), getrennt durch das erste Zeichen in `$IFS`.
 `${n}` Referenziert Positionsparameter *n* oder Variable *Name*.
Indirekte Referenz: `${ ! Variablename }` oder `${ ! Positionsparameter }`

SHELL VARIABLEN Durch die Shell automatisch gesetzt:

\$BASH Dateiname und Pfad der aktuellen Bash.
\$BASH_ARGC Array, Anzahl von Parametern im Frame des Call Stacks.
\$BASH_ARGV Array, alle Parameter des aktuellen Call Stacks.
\$BASH_COMMAND Aktueller/nächster auszuführender Befehl.
\$BASH_EXECUTION_STRING Argument von Befehlszeilenoption `-c`.
\$BASH_LINENO Array mit Zeilennummern für die Elemente in **\$FUNCNAME**.
\$BASH_REMATCH Array durch Zuordnung von `=~` in `[[]]`.
\$BASH_SOURCE Array-Namen von Quelltexten gemäß **\$FUNCNAME**.
\$BASH_SHELL Level der aktuellen Subshell.
\$BASH_VERSION Array mit Bash Versionsinformationen.
\$BASH_VERSION Bash Version String.
\$COMP_CWORD Index in **{ \$COMP_WORDS }** des Wortes mit dem Cursor.
\$COMP_LINE Aktuelle Befehlszeile (nur in Komplettierungen).
\$COMP_POINT Cursorposition relativ zur aktuellen Befehlszeile.
\$COMP_WORDSBREAKS Von Readline verwendete Wort-Trennzeichen.
\$COMP_WORDS Array mit der aktuellen Befehlszeile.
\$DIRSTACK Array des Directory-Stacks.
\$EUID Effektive User-ID.
\$FUNCNAME Array der Shellfunktionen im Call Stack.
\$GROUPS Array aller Gruppen des Users.
\$HISTCMD Historyzeilennummer.
\$HOSTNAME Name des aktuellen Hosts.
\$HOSTTYPE Typ des aktuellen Hosts.
\$LINENO Aktuelle Zeilennummer im Skript oder Funktion.
\$MACHTYPE Beschreibt den System-Type.
\$OLDPWD Vorheriges Arbeitsverzeichnis (gesetzt durch **cd**).
\$OPTARG Letzte mit **getopts** bearbeitete Option.
\$OPTIND Indexnummer des nächsten Arguments für **getopts**.
\$OSTYPE Beschreibt das Betriebssystem.
\$PIPESTATUS Array mit Rückgabewerten der letzten (Vordergrund-) Pipeline.
\$PPID Process ID des die Shell aufrufenden Prozesses.
\$PWD Aktuelles Arbeitsverzeichnis (gesetzt mit **cd**).
\$RANDOM Zufällige Ganzzahl zwischen 0 und 32767.
\$REPLY Letzte Eingabe von **read** oder **select**.
\$SECONDS Sekunden seit dem Start der Shell.
\$SHELLOPTS Mit Doppelpunkten getrennte Liste aktiver Shelloptionen.
\$SHLVL Level der aktuellen Shell.
\$UID Aktuelle User ID.

SHELL VARIABLEN Von der Bash ausgewertet:

\$BASH_ENV Konfigurationsdatei zur Auswertung von Skripten.
\$CDPATH Suchpfad für **cd** (":"-Liste von Verzeichnissen).
\$COLUMNS Terminalbreite in Zeichen.
\$COMPREPLY Array mit möglichen Komplettierungen.
\$EMACS Falls der Inhalt mit "t" beginnt, wird Readline deaktiviert.
\$FCEDIT Voreingestellter Editor für **fc**.
\$FIGIGNORE ":"-getrennte Liste von Extensionen für Komplettierungen.
\$GLOBIGNORE ":"-Musterliste, bei Dateinamenkomplettierungen ignoriert.
\$HISTCONTROL ":"-getrennte Liste von Optionen für die History.
\$HISTFILE Historydatei (voreingestellt: `~/.bash_history`).
\$HISTFILESIZE Maximale Anzahl von Zeilen in der Historydatei.
\$HISTIGNORE ":"-getrennte Liste von Mustern für gespeicherte Zeilen.
\$HISTSIZE Maximale Anzahl von Historyzeilen (voreingestellt: 500).
\$HISTTIMEFORMAT Formatstring Historyzeitangaben.
\$HOME Homeverzeichnis des aktuellen Users.
\$HOSTFILE Datei für Hostnamenkomplettierungen.
\$IFS Drei Trennzeichen für Wörter und Zeilen (default): **space, tab, newline**.
\$IGNOREEOF Anzahl von EOFs, bevor die Shell terminiert (voreingestellt: 10).
\$INPUTRC Readline-Konfigurationsdatei (voreingestellt: `~/.inputrc`).

\$LANG Lokale Voreinstellung, wenn es **\$LC_*** Variable gibt.
\$LC_ALL Übersteuert **\$LANG** und alle **\$LC_***-Variable.
\$LC_COLLATE Definiert Sortierreihenfolge.
\$LC_CTYPE Interpretation/Verhalten von Zeichen(klassen) in Mustern.
\$LC_MESSAGES Bestimmt Sprache für **'String'** und **"String"**.
\$LC_NUMERIC Bestimmt Formatierung von Zahlen.
\$LINES Zeilenanzahl des aktuellen Terminals.
\$MAIL Datei, die die Bash für Email überwacht.
\$MAILCHECK Zeit (in Sekunden, voreingestellt 60) zwischen Mailchecks.
\$MAILPATH ":"-getrennte Liste von Datei für Mailchecks.
\$OPTERR 1 (Voreinstellung) meldet Fehler von **getopts**.
\$PATH Suchpfad für Programme (":"-getrennte Liste von Verzeichnissen).
\$POSIIXLY_CORRECT Aktiviert den POSIX-Modus.
\$PROMPT_COMMAND Befehle vor Ausgabe von **PS1**.
\$PS1 Primärer Prompt (voreingestellt: `\s-\v\$_`).
\$PS2 Sekundärer Prompt (voreingestellt: `'>_'`).
\$PS3 Prompt für **select** (voreingestellt: `'#?_'`).
\$PS4 Prompt für Tracing (voreingestellt: `'+_'`).
\$SHELL Pfad der aktuellen Shell (voreingestellt: der Login-Shell).
\$TIMEFORMAT Format für Zeitangaben.
\$TMOUT Voreingestellte Wartezeit für **read** und **select**.
\$TMPDIR Verzeichnis für temporäre Dateien der Bash.
\$auto_resume Steuert Jobcontrol: **exact** oder **substring**.
\$command_not_found_handle Funktion, für nicht gefundene Befehle.
\$histchars Zeichen zur Feinsteuerung der History.

ARRAYS Index beginnt mit Null

Erzeugen: **Name[Index]=Wert** oder **declare -a Name[Index]**
Zweisen: **Name[Index]=Wert** oder **Name=(Wert1. . Wertn)**
Referenzieren: **{Name[Index]}**
Zwischen doppelten Quotes **\${name[*]}** expandiert ein Array zu einem Wort mit allen Elementen (getrennt durch das erste Zeichen in **\$IFS**) und **\$Name[@]** expandiert zu einzelnen Wörtern.
unset Zerstört Arrays; **unset Name[Index]** Löscht einzelne Elemente.

REDIRECTION (Umleitungen); neue Descriptoren: exec Descriptor Datei

< Standardeingabekanal (File Descriptor 0).
> Standardausgabekanal (File Descriptor 1).
&> Datei Standardausgabe in Datei.
2> Standardfehlerkanal (File Descriptor 2).
>&Datei Standardfehlerkanal in Datei.
2>&1 Standardausgabe und Standardfehlerkanal.
n<Datei Eingaben n aus Datei.
n>Datei Leitet Ausgabe n in Datei (Überschreiben, falls nicht noclobber).
>|Datei Überschreiben auch bei noclobber.
n>>Datei Leitet n in Datei (anhängen).
n<&Datei Dupliziert Eingabe n aus Datei.
n>&Wort Dupliziert Ausgabe n in Datei.
n<&Nummer- Leitet EingabeDatei von Nummer zum Deskriptor n.
n>&Nummer- Leitet AusgabeDatei Deskriptor Nummer zu n.
n<>Datei Öffnet Deskriptor n zum Lesen und Schreiben.
<<- Wort "here document".
<<< Wort "here strings".

Spezielle Umleitungen:

/dev/fd/fd Mit *fd* als Ganzzahl, dupliziert dies Deskriptor *fd*.
/dev/stdin Dupliziert Kanal 0.
/dev/stdout Dupliziert Kanal 1.
/dev/stderr Dupliziert Kanal 2.
/dev/tcp/Host/Port Öffnet TCP-Verbindung.
/dev/udp/Host/Port Öffnet UDP-Verbindung.

QUOTING Verhindert Auswertung von Metazeichen durch Bash:

\ (Escape) nächstes Zeichen wird nicht verändert.
'Eingeschlossene Zeichen werden nicht verändert'.
"Eingeschlossene Zeichen bis auf \$, ', , ! unverändert"
\$' String' Erlaubt diese Escape-Sequenzen:
\a Alert (Bell)
\b Backspace
\e Escape
\f FormFeed
\n NewLine
\r CarriageReturn
\t horizontaler Tabulator
\v vertikaler Tabulator
\ Backslash
\ ' einfacher Quote
\nnn 8-Bit Zeichen (oktal angegeben)
\xHH 8-Bit Zeichen (hexadezimal)
\cx Control-X Zeichen

Zwischen doppelten Quotes expandiert "\$ " gemäß der Locale-Einstellung; falls diese C oder POSIX ist, wird \$ ignoriert. Übersetzungen stehen auch zwischen doppelten Quotes.

EXPANDIERUNGEN Substitutionen spezieller Zeichen {, }, ~ und Parametern oder Variablen in Befehlszeilen erfolgt vor deren Ausführung.

Parameter Expansion **\${Parameter}** Ersetzt *Parameter* durch Werte. Geschweifte Klammern sind bei Positionsparametern mit mehreren Ziffern und zur Abgrenzung erforderlich. Operatoren ohne Doppelpunkt (= statt :=) wirken *nicht* bei existierenden, aber bei leeren Variablen und Parametern. Variablenamen bestehen aus Buchstaben, Ziffern und _.

! als erstes Zeichen (z. B. **{!var}**) führen zur indirekten Adressierung.
{parameter:-voreingestellt} Entspricht *voreingestellt* falls *parameter* ungesetzt oder leer ist; verändert *parameter* nicht.
{parameter:=voreingestellt} Entspricht *voreingestellt*, falls *parameter* ungesetzt oder leer ist, weist *parameter voreingestellt* zu.
{parameter:?message} Zeigt *message*, falls *parameter* ungesetzt oder leer.
{parameter:+alt} Falls ungesetzt oder leer: *alt*; nur Test, keine Ersetzung!
{parameter:offset:length} Expandiert *parameter* ab *offset* (voreingestellt: 0) um *length* Zeichen (voreingestellt: alle).
{!prefix*}, **{!prefix@}** Expandiert Variablen, deren Namen mit *prefix* beginnen, getrennt durch erstes **\$IFS**-Zeichen.
{!Name[@]}, **{!Name[*]}** Bei Arrays: expandiert alle Elemente von *Name*, 0, falls *Name* leer ist, sonst Null.
{#parameter} Expandiert zur Zeichenanzahl. Mit * oder @ expandiert *parameter* zur Anzahl der Positionsparameter, in Arrays die Anzahl der Elemente.
{parameter#Muster}, **{parameter##Muster}** Entfernt *Muster* von links (#: Minimum, ## Maximum) aus *parameter*, gibt den Rest aus.
{parameter%Muster}, **{parameter%%Muster}** Entfernt *Muster* von rechts (%: Minimum, %% Maximum) aus *parameter*, gibt den Rest aus.
{parameter/Muster/Ersatz}, **{parameter//Muster/Ersatz}**
Ersetzt *Muster* durch *Ersatz* (nur einmal: /, alle Muster: //), gibt Result aus.

Tilden Expansion Wörter, die mit einer (nicht maskierten) Tilde (~) beginnen, betrachtet die Bash bis zu ersten, nicht maskierten / (oder alle Zeichen) als *Tilde-Prefix* und expandiert dies:

~*Tilde-Prefix*/ Expandiert *Tilde-Prefix* als Login-Namen.
~+n Expandiert zu **\$PWD**; mit einer Nummer *n*: **\$DIRSTACK**-Eintrag *n*.
~-n Expandiert zu **\$OLDPWD**; mit Nummer *n*: **\$DIRSTACK**-Eintrag *n*.

Klammer Expandierungen Auswertung von links nach rechts:
`a{d,c,b}e` expandiert zu `ade ace abe`. Bereiche sind so möglich: `{x.y}`, mit `x` und `y` vom gleichen Typ: Ganzzahlen oder Buchstaben. Die Bash substituiert alle ausgelassenen Elemente, bei Buchstaben in der lokalen lexikalischen Ordnung: `mkdir ./{old,new}` oder `ls ./{ucb/{ex,ed},lib/{ex?.?* ,hw}}`

ARITHMETISCHE EXPANSION Fügt das Ergebnis der Ganzzahlberechnung in die Befehlszeile; statt `$(Ausdruck)` besser `$((Ausdruck))` verwenden; es lassen sich Operatoren und Variablen nutzen.

`id++`, `id--` Variablen Post-Increment bzw. Post-Decrement.
`++id`, `--id` Variablen Pre-Increment bzw. Pre-Decrement.
`-`, `+` Unadisches Minus bzw. Plus.
`!`, `~` logische bzw. bitweise Negation.
`**` Potenzieren: `$((links ** rechts))` bedeutet: *links^{rechts}*.
`*`, `/`, `%` Multiplikation, Ganzzahldivision, Ganzzahlrest.
`+`, `-` Addition, Subtraktion.
`<<`, `>>` Linkes und rechtes bitweises Verschieben.
`<=`, `>=`, `<`, `>` Vergleiche; z. B. `$((3 <= 5))`: 1 für WAHR, sonst 0.
`==`, `!=` Test auf Gleichheit, Ungleichheit: 1 für WAHR, sonst 0.
`&` Bitweise UND.
`^` Bitweise exklusive ODER.
`|` Bitweise ODER.
`&&` Logisches UND.
`||` Logisches ODER.
`expr?expr : expr` Bedingungsoperatoren.
`=`, `*=`, `/=`, `%=`, `+=`, `-=`, `<<=`, `>>=`, `&=`, `^=`, `|=` Zuweisungen.
`expr1`, `expr2` Sequenzielle Bearbeitung.

Konstanten mit führender `0` interpretiert die Bash oktal, mit `0x` oder `0X` hexadezimal, sonst dezimal. Zahlen lassen sich in der Form `Basis#n` angeben, mit Basen im Bereich von `2` und `64` (voreingestellt: `10`). Ziffern größer als `9` bezeichnen Kleinbuchstaben, Großbuchstaben und `@`, `_` (in dieser Reihenfolge).

WORT-SPLITTUNG `$IFS` enthält drei Trennzeichen für Wörter, voreingestellt: `space`, `tab`, `newline` (in dieser Reihenfolge, ohne Trennzeichen). Die Bash nutzt diese bei Parameterexpansionen, Wort-Splittung in Befehlszeilen, beim Einlesen, bei Array-Zuweisungen und in programmierbaren Komplettierungen.

PFADNAMEN-EXPANSION Die Sortierreihenfolge wird durch die Variable `$LC_COLLATE` gesteuert, `$GLOBIGNORE` enthält Ausschlussmuster, die Option `dotglob` schließt mit `.` beginnende Dateinamen ein.

PATTERN MATCHING (Mustervergleiche)

`*` steht für eine beliebige, auch leere (nicht existierende) Zeichenkette. `?` repräsentiert genau ein Zeichen, ausgenommen den `"`.
`[...]` repräsentiert die enthaltenen Zeichen, mit einem `"` einen Bereich (alle Zeichen zwischen den Grenzen, einschließlich); `!` oder `^` (als erstes Zeichen) negiert die Auswahl (alle bis auf die angegebenen Zeichen). Die Option `extglob` ermöglicht erweiterte Muster:

- `?(Musterliste)` : Null- oder einmaliges Auftreten der Muster.
- `*(Musterliste)` : Null- oder mehrmaliges Auftreten der Muster.
- `+(Musterliste)` : Ein- oder mehrmaliges Auftreten der Muster.
- `@(Musterliste)` : Eines der Muster aus der Liste.
- `!(Musterliste)` : Alles bis auf die angegebenen Muster.

COMMAND SUBSTITUTION (Befehlersetzungen) Fügt Befehlsausgabe in die aktuelle Befehlszeile. `$(Befehlszeile)` (neue, bevorzugte Form) oder mit Backticks: ``Befehlszeile``.

PROZESSERSETZUNG Fügt Ausgabe der Prozesse (`cmd list`) in die Befehlszeile. Lesen: `<(cmd list)` Schreiben: `>(cmd list)`.

CONDITIONAL EXPRESSIONS (Bedingungen, Test) Eingesetzt bei Schleifen und Verzweigungen. `[[Ausdruck]]` (bevorzugte Form) oder `[Ausdruck]` oder `test Ausdruck`. Wahre `Ausdrücke` haben den Wert `0`, falsche alles andere, oft `1`. `!` invertiert das Ergebnis. (Wort-Splittung, Expansion von Dateinamen, Tildern, Variablen, Parametern, arithmetischen Ausdrücke unterbleiben, ebenso wie Befehls- und Prozessersetzungen, Quote Removal).

Operatoren für Dateien:

- `-a file` Wahr, wenn `file` existiert.
- `-b file` Wahr, wenn `file` ein Block-orientiertes Gerät ist.
- `-c file` Wahr, wenn `file` ein Zeichen-orientiertes Gerät ist.
- `-d file` Wahr, wenn `file` ein Verzeichnis ist.
- `-e file` Wahr, wenn `file` existiert.
- `-f file` Wahr, wenn `file` eine normale Datei ist (und existiert).
- `-g file` Wahr, wenn für `file` set-group-id gesetzt ist.
- `-h file` Wahr, wenn `file` ein symbolischer Link ist.
- `-k file` Wahr, wenn `file` das "sticky" Bit gesetzt hat.
- `-p file` Wahr, wenn `file` eine benannte Pipe (Fifo) ist.
- `-r file` Wahr, wenn `file` lesbar ist.
- `-s file` Wahr, wenn `file` existiert und nicht leer ist.
- `-S file` Wahr, wenn `file` ein Socket ist.
- `-t fd` Wahr, wenn `File Descriptor` ein Terminal ist.
- `-u file` Wahr, wenn `file` das set-user-id-Bit gesetzt hat.
- `-w file` Wahr, wenn `file` schreibbar ist.
- `-x file` Wahr, wenn `file` ausführbar ist.
- `-o file` Wahr, wenn `file` der effektiven UID gehört.
- `-g file` Wahr, wenn `file` zur effektiven GID gehört.
- `-l file` Wahr, wenn `file` ein symbolischer Link ist.
- `-n file` Wahr, wenn `file` seit dem letzten Lesen verändert wurde.
- `-s file` Wahr, wenn `file` existiert und ein Socket ist.
- `file1 -nt file2` Wahr, wenn `file1` neuer als `file2` ist (oder `file2` nicht existiert).
- `file1 -ot file2` Wahr, wenn `file1` älter als `file2` ist (oder `file2` nicht existiert).
- `file1 -ef file2` Wahr, wenn `file1` und `file2` gleiche Inode haben.

Stringoperatoren:

- `-z string` Wahr, wenn `string` leer ist.
- `string` Wahr, wenn `string` nicht leer ist.
- `-n string` Wahr, wenn `string` nicht leer ist.
- `string1 = string2` Wahr, wenn `string` gleich `string2`.
- `string1 == string2` Wahr, wenn `string1` gleich `string2`.
- `string1 != string2` Wahr, wenn `string1` ungleich `string2`.
- `string1 < string2` Wahr, wenn `string1` vor `string2` sortiert wird.
- `string1 > string2` Wahr, wenn `string1` nach `string2` sortiert wird.

Andere Operatoren:

- `-o optname` Wahr, wenn die Shelloption `optname` (durch `set`) gesetzt ist.
- Zahlenvergleiche: `Nummer1 OP Nummer2`
- `OP` ist: `-eq (=)`, `-ne (!=)`, `-lt (<)`, `-le (<=)`, `-gt (>)`, `-ge (>=)`.

BUILTINS (Eingebaute Befehle) Optionale Argumente werden so dargestellt: `option`. `help builtin` gibt eine kurze Online-Hilfe.

`:` `arguments` Keine Aktion, Argumente und Umleitungen werden ausgeführt.
`.` `Skript Argumente` oder `source Skript Argumente` Führt `Skript` im aktuellen Environment aus; Rückgabewert durch letzten Befehl im Skript.
`alias -p Name [=Befehle]` ... Zeigt `(-p)` oder definiert Alias-Konstrukte.
`unalias -a Name` ... Löscht `(-a: alle)` Alias-Konstrukte.
`builtin Shell-Builtin Argumente` Startet den angegebenen eingebauten Befehl.
`command -pVv Befehl Argumente` ... Führt angegebenen Befehl (aber keine gleichnamige Funktion) aus; `-p` berücksichtigt `$PATH`; `-v (-V)` zeigt ihn.
`caller expr` Zeigt Kontext der aktuellen Subroutine (Shell/Funktion/Skript).

`enable -adnps -f Datei Builtin ...` (De-)aktiviert `(-n) Builtin`. `-f Datei` lädt neue `Builtins` (Object File), `-d` entfernt geladene `Builtins`, `-p` zeigt sie an, `(-a` mit Indikator). `-n` zeigt deaktivierte `Builtins`. `-s` nutzt POSIX-Format.
`eval arg ...` Wertet `args` aus und führt die entstandene Zeile aus.
`exec -cl -a name Befehl [Argumente]` Ersetzt aktuelle Shell durch `Befehl` mit den `Argumenten`, führt Umleitungen durch. `-l` als erste Argument setzt `"` in `$0` wie `login`. `-c` löscht Environment. Berücksichtigt Shelloption `execfail`.
`exit n` Definiert Rückgabewert `n` (für Skript, Funktion).
`export -fn Name [=Wert]` ... Exportiert Variable/Funktion `Name (-f)` ins Environment neuer Prozesse. `-n` entfernt `Name`, `-p` zeigt sie an. `--` beendet Optionen.
`getopts Optstring Name Args` Bearbeitete Positionsparameter. `:` im `Optstring` markiert Optionen mit Argumenten; als erstes Zeichen: unterdrückt Meldungen.
`hash -lr -p Pfad -dt Name` Zeigt/Entfernt gespeicherte Pfade. `-d` löscht `Name`. `-r` löscht alles. `-p` ergänzt `Pfad`. `-t` zeigt `Namen`, mit `-l` wie Eingaben.
`help -s Muster` Online-Hilfe. `-s:` zeigt nur "Synopsis".
`logout` Beendet Login-Shell.

`let Ausdruck Arg ...` Berechnet Ausdruck, Ergebnis steuert Rückgabewert.
`read -ers -ufd -pPrompt -aArray -tTimeout ...` Liest (Standard-) Eingabe, oder vom `fd`. `-s` für Terminals ohne Echo. `-e` nutzt Readline zum Einlesen, `-r` liest Rohdaten. `-a` definiert Antwortarray (voreingestellt: `$REPLY`), `-u` Filedescriptor, `-t` legt Timeout fest (voreingestellt: unendlich, berücksichtigt `$TMOUT`).
`return n` Beendet Funktion/Skript mit Rückgabewert `n`.
`shift n` Rotiert Positionsparameter `n`mal nach links.
`suspend -f` Suspendiert Shell bis `SIGCONT` eintrifft. `-f` für Login-Shells.
`times` Zeigt User-/System-Zeit für Shell und Prozesse.
`trap -lp [Befehl] Signal ...` Reagiert auf `Signal` mit `Befehl`. Ohne `Befehl` oder mit `-` werden Voreinstellungen restauriert. `-l` zeigt Liste von Signalnamen und Nummern. `-p` zeigt zugewiesene `Befehle`.
`type -aftp Name Name ...` Nennt Type von `Name`. `-t:` kurz; `-p` mit Pfaden. `-P` bewirkt Suche gemäß `$PATH`, `-a` zeigt alle Pfade. `-f` unterdrückt Funktionssuche.
`unset -fv Name ...` Löscht Variablen oder Funktion. `-f:` nur Funktion; `-v` nur Variablen.
`echo -neE Ausgabe ...` Zeigt durch Leerzeichen getrennt `Ausgaben`. `-n:` unterdrückt folgenden Zeilenumbruch, `-e` erlaubt Steuerzeichen, `-E` deaktiviert `-e`.
`printf -v Variable` Erweitertes `echo` mit `printf(1)` Format. `-v` ermittelt Ausgabekanal aus `Variable` statt `stdout`.
`test Ausdruck` oder `[Ausdruck]` Rückgabewert als Ergebnis der Auswertung von `Ausdruck` `0` oder `1`.

`Jobs` (Hintergrundprozesse z. B. mit `&` am Ende der Befehlszeile). Voreingestellt (ohne `Jobspec`) wird der letzte Job verwendet.

`disown -ar -h Jobspec ...` Entfernt `Jobspec` aus der Tabelle. `-h:` lässt Job `$SIGHUP` ignorieren. `-a` löscht alle Jobs aus der Tabelle, `-r` nur laufende.
`bg Jobspec ...` Verlagert angehaltenen Job in den Hintergrund (wie `&`).
`fg Jobspec` Verlagert Hintergrundprozess in den Vordergrund.
`jobs -lnprs Jobspec ..` oder `jobs -x Befehl Jobspec` Zeigt/Ersetzt aktive Jobs. `-l` ergänzt PIDs, `-p` zeigt nur PIDs. `-r:` nur laufende Jobs, `-s:` nur angehaltene. `-n:` Jobs mit verändertem Status. `jobs -x` startet `Befehl`, nachdem `Jobspecs` durch PID des Process Group Leaders ersetzt wurde.
`kill -l -[s] Sigspec|-nSignalum PID|Job ...` Sendet Signal(e) an `PID|Job`.
`-s` definiert Signal mit Namen, `-n` mit Nummer. `-l` zeigt verfügbare Signale.
`wait n ...` Wartet auf das Ende von Prozessen und gibt `n` zurück.

Schleifen, Tests, Verzweigungen Kriterium ist der Rückgabewert; `:` für WAHR.
`break n` Verlässt die Schleifen (`for`, `while`, `until`, `select`) in Level `n`.
`continue n` Setzt Schleifen in Level `n` fort.
`if cmds1; then cmds2; elif cmds3; then cmds4; ... else cmds5; fi` `cmds2` werden nur ausgeführt, wenn `cmds1` den Rückgabewert `0` erzeugt, sonst wird (wenn vorhanden) `cmds5` ausgeführt. `elif` erlaubt weitere Tests (`cmds3...`).
`case Wort in Muster [[Muster]...] Befehle ;; ... esac` Testet `Wort` anhand des `Muster`, führt bei Treffern `Befehle` aus.

for *Variable* in *Liste* ... ;**do** *Befehle*; **done** Führt *Befehle* für jedes Element aus *Liste* (Voreinstellung: "&@") in der *Variablen* aus. Oder: **for** ((*exp1*; *exp2*; *exp3*)); **do** *Befehle*; **done** Entspricht: ((*exp1*)) ; **while** ((*exp2*)); **do** *cmds*; ((*exp3*)); **done** Führt *Befehle* abhängig von drei arithmetischen Ausdrücken aus (fehlende Ausdrücke gleich 1). **until** *cmds*; **do** ...; **done** Führt ... aus, solange *cmds* Rückgabewert Null liefert. **while** *cmds*; **do** ...; **done** Führt ... aus, bis *cmds* Rückgabewert Null liefern.

Konfiguration Siehe auch STARTUP.

bind -m *Keymap* -f *file* -q *Name* -u *Name* -lpsvPsrv oder **bind** *Readline-Funktion* oder **bind** -x*Bindung* Zeigt aktuelle Readline Bindungen, Tasten, Funktionen, Makros. -m legt Keymap fest: **emacs**, **emacs-standard**, **emacs-meta**, **emacs-ctlx**, **vi**, **vi-move**, **vi-command**, und **vi-insert**. -l listet Funktionen. -P gibt Bindungen aus, -p in wiederverwertbaren Format. -S zeigt Makros, -s in wiederverwertbarem Format. -V zeigt Variablen, -v in wiederverwertbarem Format. -f liest Bindungen aus *file*. -q Bindungen an Tasten. -u löst Bindungen. -r löscht Bindungen.

bind -x ' "*Tastenfolge* ": *Befehle* ' bindet *Befehle* an *Tastenfolge*. **compgen** *Option Wort* Erzeugt durch *Option* gesteuert Kompletterungen. **complete** -abcdefgjklsuv -o *Comp-Option* -A *Aktion* -G *Muster* -W *Wörter* -P *Prefix* -S *Suffix* -X *Filter* -A *Aktion*. -a Alias, -b Builtin, -c Befehl, -d Verzeichnis, -e Export, -f Datei, -g Gruppe, -j Job, -k Schlüsselwort, -s Service, -u User (Name), -v Variable. -F *Funktion* -C *Befehl Name* *Name* ... -G definiert Dateinamenmuster, -W Wortliste, -C Befehl, -F Funktion. -X Filter, -P Prefix, -S Suffix.

complete -pr *Name* ... Komplettiert *Name*. p zeigt mögliche Kompletterungen, r löscht sie.

set --abefhkmnptuvxBCHP -o *Option Argument* ... Ohne Option: Zeigt Name und Werte von Variable/Funktionen, weist *Argumente* Positionsparametern \$0..\$n zu.. Mit Option: Setzt/Löscht Shelloptionen, siehe SHELLOPTIEN. **shopt** -pqsu -o *Option* ... Zeigt/Setzt/Löscht erweiterte Shelloptionen. p zeigt alle Optionen, q unterdrückt Ausgaben. s setzt Option, u löscht Option. o erlaubt nur set-Optionen.

ulimit -SHacdfilmnpqstuvx [*Limit*] Zeigt/Setzt Shelllimits. -S: Softlimits, -H: Hardlimits, -a zeigt aktuelle Limits. -c: maximaler Coresize, -d maximales Processdatasegment, -e maximaler Nicevalue, -f Dateigröße für Shell und ihre Prozesse, -i maximale Anzahl von Signalen, -l maximale Größe Processmemory, -m maximaler Residentize, -n maximale Anzahl öffentlicher Filedescriptoren, -p Pipebuffergröße, -q maximale Größe von POSIX Message Queues, -r maximale Real-Time Priorität, -s maximaler Stacksize, -t maximale CPU-Zeit in Sekunden, -u maximale Anzahl von Userprozessen, -v maximaler virtueller Speicher, -x maximale Anzahl gesperrter Dateien.

umask -p -s *Modus* Zeigt/setzt Maske für Zugriffsrechte. -p zeigt aktuelle Werte im wiederverwertbarem Format, -S symbolisch.

Directories

cd -L|-P *Verzeichnis* Wechselt das aktuelle *Verzeichnis*. -L: folgt symbolischen Links, -P nutzt physikalische Struktur. \$HOME: voreingestelltes Homeverzeichnis. - vorheriges Verzeichnis, .. eine Ebene nach oben, . aktuelles Verzeichnis.

dirs -clpv +n -n Listet gespeicherte Verzeichnisse. -c löscht Speicher, -l zeigt ganze Pfade, voreingestellt nur relativ zum Homeverzeichnis. -p zeilenweise, -v mit Nummern. +/-n: n Einträge vor/zurück.

popd -n +n -n Restauriert Verzeichniseintrag n (voreingestellt: 1) im Speicher. -n: entfernt Eintrag ohne Verzeichniswechsel.

pushd +-n *Verzeichnis* Speichert *Verzeichnis* und wechselt dahin oder rotiert den Speicher nmal. -n: nur speichern, kein Wechsel.

pwd -LP Zeigt aktuelles Verzeichnis, -L: mit symbolischen Links, -P: in der physikalischen Struktur.

Variables

declare -afFirtx -p *Name* [=Wert] ... -a erzeugt Array, -f: nur Funktionen, -F zeigt nur Funktionsnamen, -i erzeugt Integervariablen, -r aktiviert Schreibschutz, -t: Traceattribut, -x exportiert *Name*. + statt - deaktiviert Attribute.

typeset -afFirtx -p *Name* [=Wert] ... Erzeugt/Zeigt Variables und/oder gibt ihnen Attribute. -a erzeugt Array, -f: nur Funktionen, -F zeigt nur Funktionsnamen, -i Integervariablen, -r Schreibschutz, -t: Traceattribut, -x exportiert *Name*. + statt - deaktiviert Attribute.

local *Option Name* [=Wert] ... Definiert lokale Variablen.

readonly -apf *Name* [=Wert] ... Schreibschutz für Variablen/Funktionen. Ohne Option: zeigt schreibgeschützte. -a: Array. -p zeigt sie. -f: nur Funktionen.

History Relevante Variablen: \$HIST* und \$histchars.

fc -e *eName* -nlr *erster letzter* oder **fc** -s *Muster=ErsetzungBefehl* Editiert Befehlszeilen; erste Form: alle Zeilen einschließlich der angegebenen. -e Editor, voreingestellt ist \$FCEDIT oder \$EDITOR. -n ohne Zeilennummern, -l zeigt Zeilen nur, -r umgekehrte Reihenfolge.

history -cawrn -d *Offset n* Zeigt Historyzeilen mit Nummern, "" markiert veränderte Zeilen; -c löscht History. -d löscht Historyzeile *Offset* \$HISTTIMEFORMAT kann strftime(3)-kompatible Zeitangaben enthalten, die dann in die History aufgenommen werden.

history -awrn *Datei* Liest (-r), schreibt (-w) Historydatei (voreingestellt ~/ .bash_history). -a: hängt Zeilen an. -w schreibt aktuelle History in Datei. -r liest Historydatei ein. -n liest nur bisher nicht vorhandene Historyzeilen ein.

history -ps *Argument* ... Expandiert *Argument*. -p zeigt Ergebnis an. -s hängt Argumente an.

PROMPT In interaktiven Shells lassen sich \$PS1, \$PS2, \$PS3 und \$PS4 mit Steuerzeichen konfigurieren, siehe auch \$PROMPT_COMMAND:

\a Alert (Bell)
\d Datum in der Form "Wochentag Monat Tag"
\D{*Format*} Datum im strftime(3) *Format*
\e Escape-Zeichen
\h Hostname bis zum ersten "
\H Gesamter Hostname
\j Basename des Shellterminals
\n NewLine
\r CarriageReturn
\s Shellname, wie \$0
\t Uhrzeit im 24-Stunden-Format HH:MM:SS
\T Uhrzeit im 12-Stunden-Format HH:MM:SS
\@ Uhrzeit im 12-Stunden-Format mit am/pm
\A Uhrzeit im 24-Stunden-Format HH:MM
\u Username
\v Bash Major Version
\V Bash Release
\w aktuelles Arbeitsverzeichnis
\W Basename des aktuellen Arbeitsverzeichnisses
\! Historynummer der aktuellen Befehlszeile
\# Nummer der aktuellen Befehlszeile
\! Historynummer der aktuellen Befehlszeile
\\$ abhängig von effektiver UID: # bei #, sonst \$
\nnn 8-Bit-Zeichen (oktal)
\ Backslash
\ [Beginnt Steuerzeichensequenz
\] Beendet Steuerzeichensequenz

READLINE Die Library enthält Befehlszeileneditierfunktionen. Die Bashoption --noediting deaktiviert dies. Zwei Modi sind vorhanden: +o emacs ist voreingestellt; +o vi aktiviert einen alternativen Modus. bind bindet Funktionen an Tasten. M- steht für die Meta- (linke Alt- oder ESC-) und eine weitere Taste, C- steht für Control (Strg). Eingaben lassen sich mit numerischen Argumenten (Wiederholungen) versehen: M-0, M-1, ... M- für negative Argumente. Konfigurationsdatei ist in \$INPUTRC (voreingestellt: ~/.inputrc). Wörter bestehen aus alphanumerischen Zeichen. Wichtige Bindungen:

M-# (emacs) # (vi) comment-begin Fügt # vor Kommentare ein.

C-a beginning-of-line Sprung an Zeilenanfang.

C-e end-of-line Sprung ans Zeilenende.

C-f forward-char Ein Zeichen nach rechts.

C-b backward-char Ein Zeichen nach links.

M-f forward-word Sprung ans Ende des Wortes.

M-b backward-word Sprung an den Anfang des aktuellen/vorigen Wortes.

C-l clear-screen Aktualisiert Ausgabe, aktuelle Zeile als erste zeigen.

C-d delete-char Löscht Zeichen am Point (unter Cursor).

Rubout (Backspace) backward-delete-char Löscht Zeichen vor Point/Cursor; mit numerischen Argument speichern im Kill-Ring.

C-q, C-v quoted-insert Nächstes Zeichen unverändert eingeben.

C-v TAB tab-insert Tabulator eingeben.

C-t transpose-chars Vertauscht Zeichen links/rechts vom Point.

M-t transpose-words Vertauscht Wörter links/rechts vom Point.

M-u upcase-word Großbuchstaben ab Point.

M-l downcase-word Kleinbuchstaben ab Point.

M-c capitalize-word Großbuchstaben am Wortanfang.

Kill (Löschen und Speichern im Kill-Ring)/ (Einfügen aus dem Kill-Ring) Einfügen

C-k kill-line Killt Text vom Point zum Zeilenende.

C-x Rubout backward-kill-line Killt bis zum Zeilenanfang.

C-u unix-line-discard Killt rückwärts bis zum Zeilenanfang.

M-d kill-word Killt vom Point zum Ende des aktuellen/nächsten Wortes.

M-R about backward-kill-word Killt Wort vor Point.

C-w unix-word-rubout Killt Wort vor Point bis Leerzeichen.

M-\ delete-horizontal-space Entfernt Leerzeichen um Point.

C-y yank Fügt aktuellen Kill-Ring-Eintrag am Point ein.

M-y yank-pop Rotiert Kill-Ring eine Position.

TAB complete Versucht Text am Point zu kompletterieren.

M-? possible-completions Zeigt mögliche Kompletterungen.

M-* insert-completions Fügt alle möglichen Kompletterungen ein.

M-/ complete-filename Komplettiert Dateinamen.

C-x / possible-filename-completions Zeigt mögliche Kompletterungen.

M-! complete-command Versucht Text als Alias, reserviertes Wort, Funktion, Builtin, Befehl zu kompletterieren (in dieser Reihenfolge).

C-x ! possible-command-completions Zeigt mögliche Befehlskompletterungen.

M-{ complete-into-braces Dateinamenkompletterungen in Klammern.

M-TAB dynamic-complete-history Kompletterungen anhand Historyzeilen. History

M-< beginning-of-history Sprung zur ersten Historyzeile.

M-> end-of-history Sprung zur letzten Historyzeile.

C-r reverse-search-history Inkrementelle Historysuche rückwärts.

C-s forward-search-history Inkrementelle Historysuche vorwärts.

M-p non-incremental-reverse-search-history Historysuche rückwärts.

M-n non-incremental-forward-search-history Historysuche vorwärts.

M-C-y yank-nth-arg Fügt angegebenes Element (beginnend mit "0" als Befehl) aus der vorherigen Zeile am Point ein.

M- . M- _ yank-last-arg Fügt letztes Element voriger Zeile ein.

M-C-e shell-expand-line Expandiert die Zeile wie die Shell.

M-^ history-expand-line Historyexpandierung der aktuellen Zeile.

C-x C-e edit-and-execute-command Ruft Editor mit Zeile auf.

C-g, C-x C-g, M-C-g abort Bricht Aktion ab.

C-_, C-x C-u undo Rücknahme aller Änderungen in der Zeile.

C-x C-r re-read-init-file Liest inputrc erneut ein.

C-@, M- _ set-mark Setzt Mark am Point.

C-x C-x exchange-point-and-mark Vertauscht Point und Mark.

C-] character-search Sprung zum nächsten Auftreten des Zeichens.

M-C-] character-search-backward Sprung zum letzten Auftreten des Zeichens.

C-x (start-kbd-macro Beginnt Definition eines Keyboardmacros.

C-x) end-kbd-macro Beendet Definition des Keyboardmacros.

C-x e call-last-kbd-macro Führt letztes Keyboardmacro aus.

TAB-TAB Versucht die Eingabe kontextabhängig zu kompletterieren: \$ für Variablen, / für Verzeichnisse (entschließlich versteckter), * ohne versteckter, @ für Hostnamen, und ~ Usernamen.