

Terminalverwaltung

Dieser Teil beschreibt Datenein- und -ausgabe über Terminals, also serielle Schnittstellen mit den dazu erforderlichen Terminaltreibern.

Dazu wird das Kommando *stty* und die Mechanismen **termcap** und **terminfo** behandelt.

Inhalt

1. Überblick.....	3
2. Das Kommando stty.....	5
2.1. Umdefinieren von Tasten.....	6
2.2. Ein- und Ausschalten von Flags für stty.....	7
3. termcap und terminfo.....	11
3.1. termcap.....	11
3.1.1. Tuning.....	15
3.2. terminfo.....	15
3.3. Zugriff aus der Shell.....	17
3.3.1. Funktionstasten.....	17

3.3.2. Cursorposition.....18

1. Überblick

Bei der Datenein- und -ausgabe über Terminals (serielle Schnittstellen) ist immer ein **Terminaltreiber** (TTY-Treiber) beteiligt.

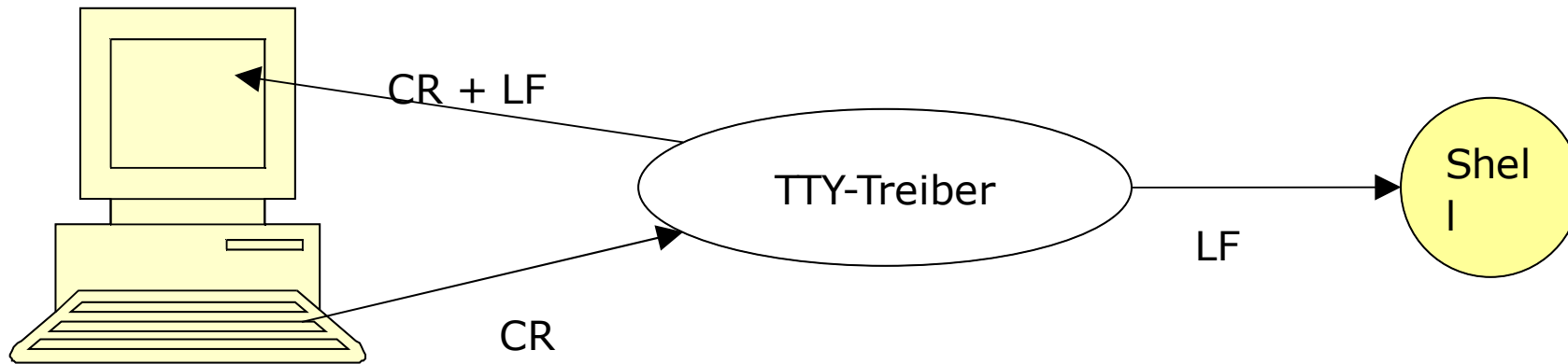
Ein **TTY-Treiber nimmt Bytes** von der Tastatur, **sammelt** sie und gibt sie i.a. **zeilenweise** (RETURN) weiter. Dabei müssen einige **Bytes** als Kommandos **interpretiert** werden, andere müssen **übersetzt** werden.

Beispiel (CR-LF Problem):

Beim Drücken der Taste "Return" wird das Zeichen CR (^M, `'\r'`) erzeugt. Das lesende Programm, z.B. die Korn-Shell akzeptiert aber nur ein LF (^J, `'\n'`) als Zeilentrenner. Dann muss CR in LF übersetzt werden.

Da die eingegebenen Zeichen auf dem Bildschirm reflektiert werden sollen, muss der TTY-Treiber nach "Return" 2 Bytes an den Bildschirm senden:

LF setzt auf Cursor auf gleiche Position eine Spalte tiefer,
CR positioniert ohne Zeilenvorschub an den Zeilenanfang.



Weiterhin muss der TTY-Treiber die Tasten gesondert behandeln, die Aktionen auslösen sollen, etwa die Tasten "Break", "CTR_C" oder "Delete".

Durch das zeilenweise Weiterreichen der Daten, ist der TTY-Treiber eine rudimentärer Zeileneditor:

"#", "Backspace" Zeichen löschen

"CTR_U", "CTR_X" Zeile löschen

Das Verhalten des TTY-**Treibers** lässt sich **beeinflussen** durch:

C-Programme mittels des Systemaufrufs "**ioctl()**" (input output control)

das Unix Kommando **stty**

2. Das Kommando stty

Das Kommando stty ist auf allen Unix Systemen verfügbar und funktioniert überall gleich (bis auf herstellerspezifische Zusätze).

stty ohne Argumente zeigt die aktuelle Terminaleinstellung an; durch die Option -a erhält man zusätzliche Informationen:

```
$ stty
speed 38400 baud; line = 0;
-brkint -imaxbel
$
$ stty -a
speed 38400 baud; rows 27; columns 79; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undefiniert>;
eol2 = <undefiniert>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R;
...
opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0
ff0
isig icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop -echoprt
echoctl echoke
$
```

2.1.Umdefinieren von Tasten

Die allgemeine Form des Kommandos zum Umdefinieren von Tasten ist:

```
stty Taste Kode
```

Beispiel:

```
$ stty intr '^g'  
$
```

Innerhalb dieser Session ist nun CTR_G das Interrupt-Zeichen.

Der Kode kann, wie oben im Beispiel symbolisch angegeben werden oder durch die Taste auf der Tastatur.

Auf diese Weise lassen sich viele Tasten belegen, die gebräuchlichsten sind in der nachfolgenden Tabelle aufgelistet (CRT_S und CTR_Q lassen sich **nicht** umdefinieren):

Name	Default	Bedeutung
eof	^D	End of File
eol	^@	alternativer Zeilentrenner
erase	^H ("backspace")	letztes Zeichen löschen (früher #)
intr	^? ("delete")	Abbruchtaste
kill	^U	Eingabezeile verwerfen (früher @)
quit	^\	Abbruch mit coredump
switch	^Z	Wechsel zu Hintergrundprozess

2.2. Ein- und Ausschalten von Flags für stty

Mit Hilfe von Flags kann der TTY-Treiber konfiguriert werden. Ein Flag wird gesetzt (eingeschaltet), indem man den Namen beim Aufruf von `stty` angibt; es wird ausgeschaltet, indem man ein Minuszeichen voranstellt.

```
$ stty -ixon  
$
```

Durch das o.a. Kommando wird das XON/XOFF Protokoll abgeschaltet – jetzt kann die Terminalausgabe nicht mehr angehalten werden.

Mehrere Einstellungen können auf einmal verändert werden, z.B. durch das Flag `echo`.

```
$ stty -echo  
$
```

Das letzte Kommando bewirkt, dass die Eingabe nicht mehr auf dem Bildschirm reflektiert wird – man schreibt blind.

Es existieren einige Optionen, die nicht nur einzelne Flags repräsentieren, sondern eine Menge von Flags verändern.

```
$ stty raw  
$
```

Durch `stty raw` wird das Terminal in den **raw-Modus** versetzt, d.h. Buchstaben werden uninterpretiert weitergeleitet:

"Del" bzw. CTR_C bricht nicht mehr ab

Eingaben werden nicht korrekt angezeigt

CTR_D ist nicht mehr EOF

In diesen Modus gelangt man auch bei Kommandos, die bei Abbruch den Modus nicht wieder umsetzen. In diesem Fall muss das **Terminal restauriert** werden:

```
stty sane CTRL_J
```

Probleme mit stty sane

Manchmal funktioniert nach dem Sanieren die Taste "Backspace" nicht mehr, die Eingabe wird gelöscht, aber nicht auf dem Bildschirm.

Grund: Einige Terminals verwenden zu Löschen von Zeichen die Folge "Bachspace" "Space" "Backspace".

Beeinflusst wird das Verhalten von dem Flag `echoe`.

Lösung:

```
stty sane echoe CTRL_J
```

Wenn man **Shell-Programme** schreibt, sollten die **Terminaleinstellungen** vor und nach der Ausführung identisch sein. Dies kann mit der Option `-g` folgt erreicht werden:

1. Erste Anweisung im Programm: Speichern der aktuellen Einstellung durch `einstellung=`stty -g``
2. Kommandos, die Einstellung verändern
3. `stty $einstellung`

Beispiel (Ausgabe von `stty -g`)

```
$ stty -g
2102:5:bf:8a3b:3:1c:7f:15:4:0:1:0:11:13:1a:0:12:f:17:16:0:0:2f:0:0:0:0:0:0:0:0
:0:0:0:0
$
```

Das `stty` Kommando wirkt immer auf das Terminal, das mit dem Standard-Input verbunden ist. Somit kann durch Eingabeumlenkung die Einstellung eines anderen Terminals beeinflusst werden; z.B:

```
$ tty
/dev/pts/11
$
$ $ stty raw < /dev/console
$
```

Das o.a Kommando ändert die Einstellungen der Konsole (normalerweise hat ein normaler Benutzer dazu keine Berechtigung).

Die man Pages "verraten" die möglichen Flags.

Hörsaalübung

Definieren Sie die Standardeinstellungen um, sodass Das Zeichen intr CTR_B ist.

Versetzen Sie Ihr Terminal in den raw-Modus und restaurieren Sie es.

3.termcap und terminfo

termcap und Terminfo sind Mechanismen, um Terminaleigenschaften allgemein zu Beschreiben und zugänglich zu machen, um beim Programmieren unabhängig von Terminalimplementierungen zu sein.

3.1.termcap

Will man portable Programme schreiben, die Bildschirmattribute ausnutzen (Fettdruck, unterstreichen, blinken, u.ä.m.), so müsste man für jedes Gerät einen speziellen Gerätetreiber entwickeln.

In Unix wird ein eleganterer Weg gewählt: **allgemeine Terminalinterfaces**.

Historie:

Ende der 70er Jahre implementierte **Bill Joy** in Berkley den Editor **vi**, um den zeilenorientierten **ed** zu ersetzen. Er entwickelte dazu ein Ausgabeinterface für sein Terminal.

Da der vi sehr beliebt wurde, wurde Bill Joy nach Anpassungen für verschiedene Terminals gefragt. Er entschied sich, nicht mehrere Gerätetreiber zu schreiben, sondern er verpasste dem vi ein allgemeines Terminalinterface mit generischen Kommandos.

Das Interface hatte 2 Teile:

1. Eine **Beschreibung** beliebig vieler **Terminals** in **normierter** Form und
2. eine genau definierte **Zugriffsmethode** in Form einer Sammlung von **C-Routinen**.

Damit konnte jedes Programm (so auch der vi) relativ terminalunabhängig realisiert werden.

Dieser nunmehr Standard heißt termcap-Mechanismus (terminal capabilities). Die ASCII-Datei, in der die Terminalbeschreibungen stehen ist "/etc/termcap".

Einträge in der Datei haben die Form:

```
#Kommentar  
Name | weitere Namen | langer Name:\  
:cap:cap:cap:...
```

Beispiel:

```

$ cat /etc/termcap
...
xterm-basic|xterm terminal emulator - common (XFree86):\
    :5i:am:km:mi:ms:ut:xn:\
    :Co#8:co#80:it#8:li#24:pa#64:\
    :AB=\E[4%dm:AF=\E[3%dm:AL=\E[%dL:DC=\E[%dP:DL=\E[%dM:\
    :DO=\E[%dB:IC=\E[%d@:LE=\E[%dD:RA=\E[?7l:RI=\E[%dC:\
    :SA=\E[?7h:UP=\E[%dA:\
    :ac=`aaffggiijjkkllmmnnooppqqrrssttuuvvwxxyzz{||}~::~\
    :ae:^O:al=\E[L:as:^N:bl:^G:bt=\E[Z:cb=\E[1K:cd=\E[J:ce=\E[K:\
    :ch=\E[%i%dG:cl=\E[H\E[2J:cm=\E[%i%d;%dH:cr:^M:\
    :cs=\E[%i%d;%dr:ct=\E[3g:cv=\E[%i%dd:dc=\E[P:dl=\E[M:\
    :do:^J:eA=\E(B\E)O:ec=\E[%dX:ei=\E[4l:ho=\E[H:im=\E[4h:\
    :is=\E[!p\E[?3;4l\E[4l\E>:kD=\E[3~:kb:^H:ke=\E[?1l\E>:\
    :ks=\E[?1h\E=:le:^H:mb=\E[5m:md=\E[1m:me=\E[m\017:\
    :mk=\E[8m:ml=\E1:mr=\E[7m:mu=\Em:nd=\E[C:op=\E[39;49m:\
    :pf=\E[4i:po=\E[5i:ps=\E[i:rl=\Ec:\
    :r2=\E[!p\E[?3;4l\E[4l\E>:rc=\E8:sc=\E7:se=\E[27m:sf:^J:\
    :so=\E[7m:sr=\EM:st=\EH:ta:^I:te=\E[?1049l:ti=\E[?1049h:\
    :u6=\E[%i%d;%dR:u7=\E[6n:u8=\E[?1;2c:u9=\E[c:ue=\E[24m:\
    :up=\E[A:us=\E[4m:vb=\E[?5h\E[?5l:ve=\E[?25h:vi=\E[?25l:\
    :vs=\E[?25h:
...
$

```

Für jedes Terminal (wie oben ein xterm) existiert ein Eintrag aus einer Zeile ("\\" ist Zeilenfortsetzungszeichen) mit Namen und mehreren Terminaleigenschaften.

Namen:

2 Zeichen, erster Buchstabe Hersteller (kann wegfallen)

Weitere Namen

Eine Folge von weiteren Namen, wobei der erste dieser Namen als Suchkriterium verwendet wird und in der Variablen TERM abgelegt wird.

Langer Namen

Dient der Angabe des Herstellers

Cap

Es gibt drei Arten von Capabilities:

- boolean
- numeric (am #-Zeichen erkennbar)
- string (am =-Zeichen erkennbar)

Beispiel:

- am automatischer Umbruch
- li#24 24 Zeilen

- `cl=50\E[;H\E2J` Escape Sequenz für clear, d.h. Bildschirm löschen

Die Manual Seiten verraten mehr!

3.1.1.Tuning

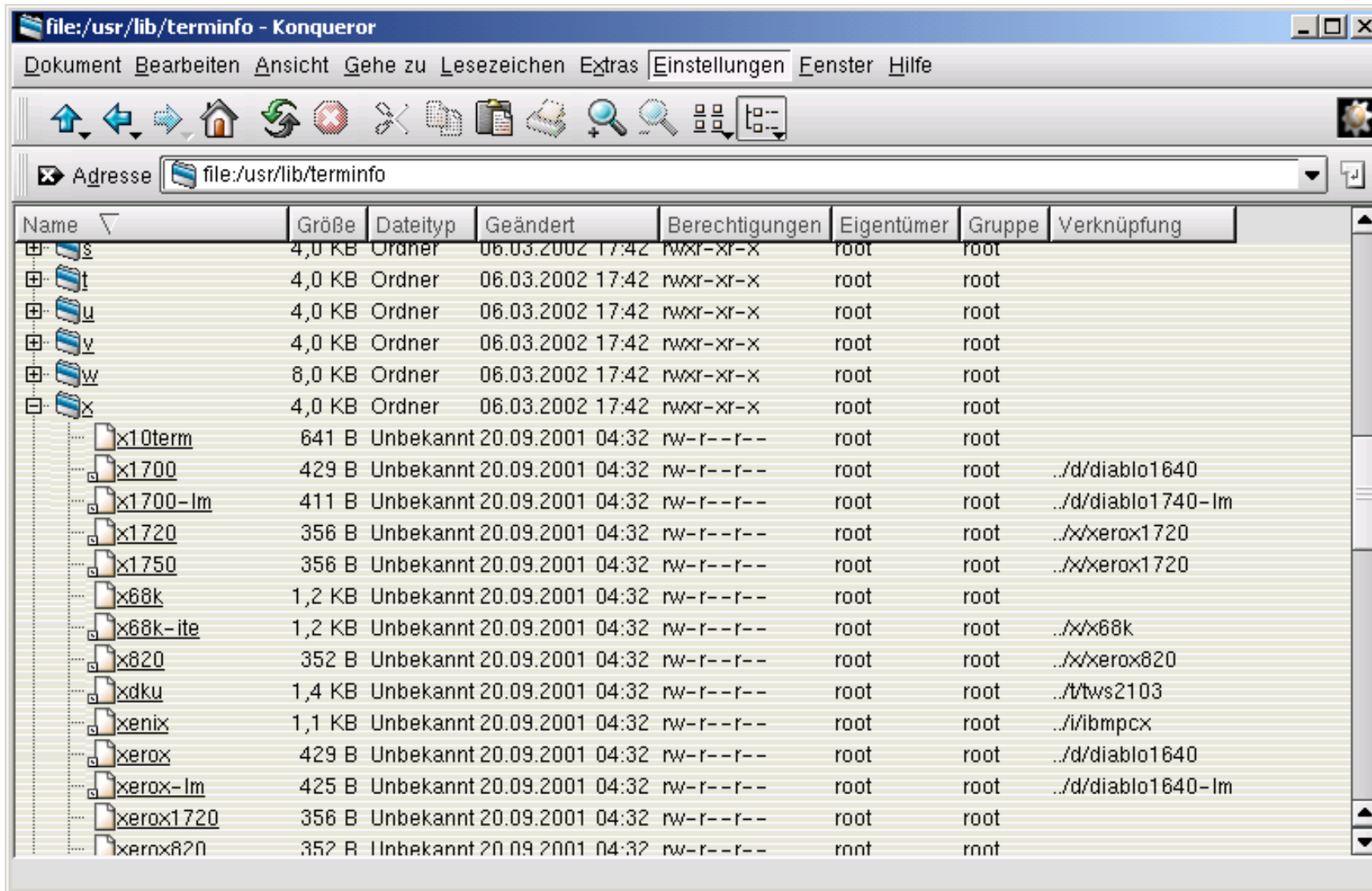
Da mittlerweile mehrere mehr als 10.000 Einträge in der Datei sind, kann das sequentielle Suchen einige Zeit kosten. Der Systemadministrator sollte die gebräuchlichsten an den Anfang kopieren.

3.2.terminfo

Zusätzlich zum termcap-Mechanismus existiert ein weiterer Mechanismus: **terminfo**.

Unter Ausnutzung der Unix Dateihierarchie wird für jeden Terminaltyp eine eigene Datei (nicht lesbar) angelegt. Mit dem terminfo-Compiler `tic` wird eine ASCII-Quelldatei mit der Terminalbeschreibung übersetzt; dann kann die übersetzte terminfo Datei in das Verzeichnis `/usr/share/terminfo/...` abgelegt werden. Für synonyme Terminalbezeichnungen wird einfach ein Link erzeugt.

Die folgende Abbildung zeigt einen Teil des terminfo Pakets:



Anwendungsprogramme können auf die Variable auf die Variable TERMINFO zurückgreifen, um die aktuelle Einstellung abzufragen.

Wenn Änderungen an einer terminfo-Datei vorgenommen werden sollen und die Quelldatei nicht verfügbar ist, so kann das (public domain) Programm `infocmp` terminfo-Ziele dekompilieren.

3.3.Zugriff aus der Shell

Das Kommando `tput` kann verwendet werden, um Terminalbeschreibungen zu lesen, die in einer terminfo Datei abgelegt sind.

Das Kommando wird mit dem Namen einer Terminaleigenschaft (cap) aufgerufen.

Beispiel (Löschen des Bildschirminhalts):

```
$ tput clear
```

3.3.1.Funktionstasten

Will man Funktionstasten in einem Shellskript (Menü) verwenden (z.B. F1), so kann man den Kode der Taste mit `tput kf1` ermitteln:

```
$ cat key.ksh
F1=`tput kf1`

echo -n "Eingabe: "
stty -echo
read key
case $key
    in
        $F1)    echo F1 gedrueckt
                ;;
        *)      echo $key wurde gedrueckt
    esac
stty echo
$
```

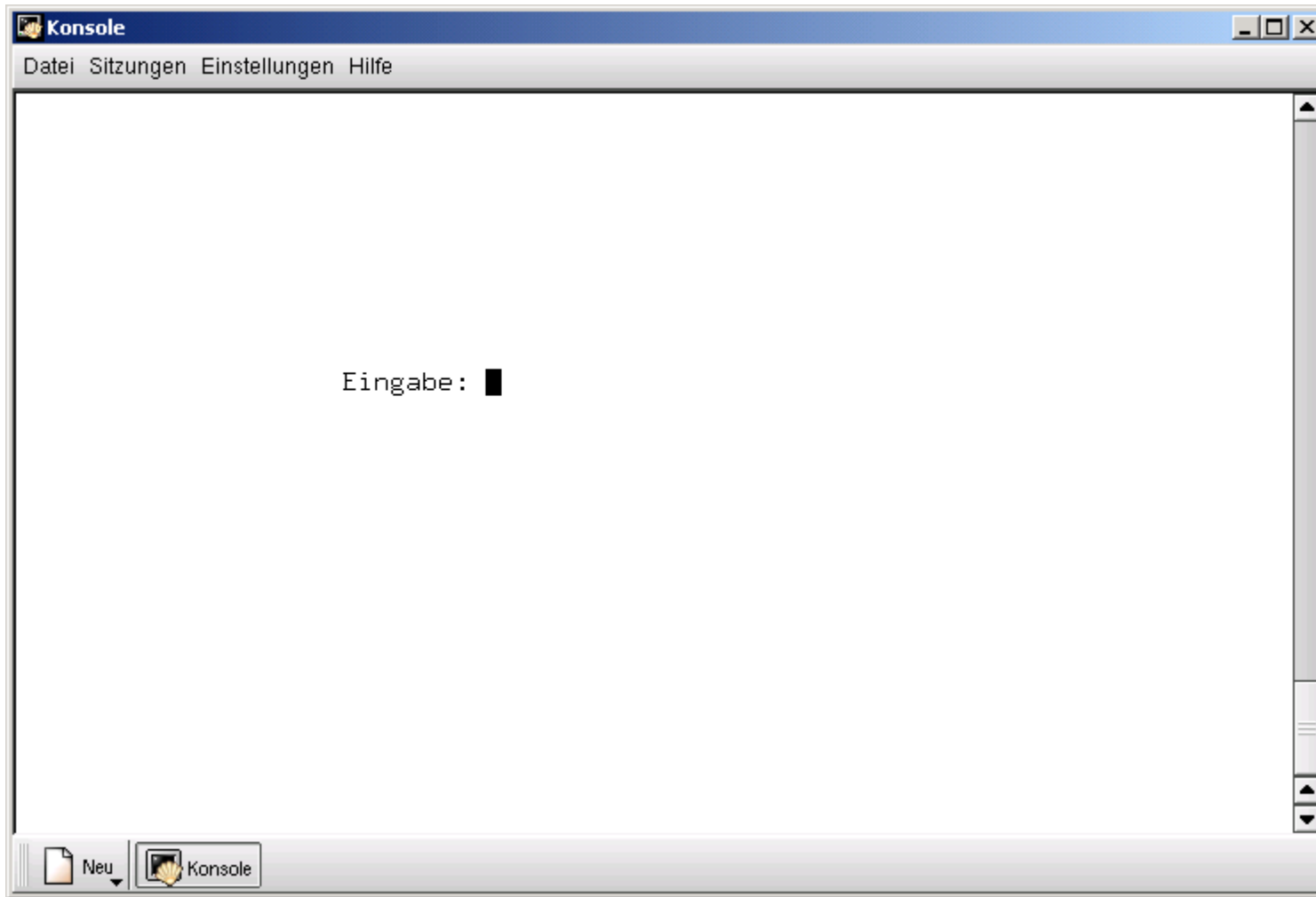
3.3.2. Cursorposition

Die Position des Cursors kann man in Shell-Programmen (Menüs) setzen durch: `tput cup x y`

```
$ cat cursor.ksh
F1=`tput kf1`

tput clear
tput cup 10 20
echo -n "Eingabe: "
stty -echo
read key
case $key
    in
        $F1)    echo F1 gedrueckt
                ;;
        *)      echo $key wurde gedrueckt
    esac
stty echo
$
```

Das Programm bewirkt:



Die Manuseiten von `tput` verraten nützliche Terminaleigenschaften, die mittels `tput` verwendbar sind, um z.B. Menüs zu realisieren.

Hörsaalübung:

Ändern Sie Ihr Menü-Programm so ab, dass das Menü mittig auf dem Bildschirm angezeigt wird.