Portable Terminal Control From Scripts

`tput` and `tabs`
Edition 2.0, for version 2.0
December 1995

by David MacKenzie and Junio Hamano

1 tput: Portable Terminal Control

The `tput` command allows shell scripts to do things like clear the screen, underline text, and center text no matter how wide the screen is. To do these things, it translates the terminal-independent name of a terminal capability into its actual value for the terminal type being used.

`tput` takes as an argument the name of a Unix System V terminfo capability, which it translates into the equivalent termcap capability name (see Section 1.4 [Capabilities], page 4, for a list of the equivalencies). Terminfo is a database that is similar to termcap but which has different capability names and is stored in a different format. The GNU `tput` command takes a terminfo name as an argument to make it compatible with the Unix System V `tput` command.

There are three types of terminfo (and termcap) capabilities: string, Boolean, and numeric. String capabilities either cause a special effect on the terminal when they are displayed or are the value sent by a special key on the terminal (the latter type are probably of no use in shell scripts). Numeric and Boolean capabilities give information about the terminal such as how many columns wide it is or whether whether it has a meta key. See Section 1.2 [Output], page 3, for more detailed information on the three types of capabilities.

The format of the `tput` command is illustrated below.

```
tput [options] capability [parameter ...]
tput [options] longname
tput [options] init
tput [options] reset
```

GNU `tput` takes the following options:

```
[-T terminal-type] [--terminal=terminal-type]
[-t] [--termcap]
[-S] [--standard-input]
[-V] [--version]
```

Here is an example of how to clear the terminal screen using `tput`:

```
tput clear
```

1.1 Using tput

The format of the `tput` command is illustrated below.

```
tput [options] capability [parameter ...]
tput [options] longname
tput [options] init
tput [options] reset
```

GNU `tput` takes the following options:

```
[-T terminal-type] [--terminal=terminal-type]
[-t] [--termcap]
[-S] [--standard-input]
[-V] [--version]
```

Some string capabilities accept parameters, such as the number of lines to delete or the column to move to. These parameters are specified on the command line following the capability name. They are always numbers.

'`-T` *termtype*'
'`--terminal=`*termtype*'
: This option indicates the type of terminal. By default, this value is taken from the '`TERM`' environment variable.

'`-t`'
'`--termcap`'
: GNU `tput` by default accepts termcap name if the capability a user specifies is not a terminfo name. This option tells `tput` not to try terminfo names and look up only termcap names.

'`-S`'
'`--standard-input`'
: This option tells `tput` to read a sequence of capabilities and parameters from the standard input. Only string capabilities can be used in this mode.

'`-V`'
'`--version`'
: This option displays the version of `tput`.

When '`longname`' is specified, `tput` displays the long descriptive name for the terminal type.

When '`init`' is specified, `tput` sends the initialization strings for the terminal. If the terminal's tab width is other than 8, and it cannot be reset, the tab expansion in the tty driver is turned on. Otherwise, tab width is reset to 8 and the tab expansion is turned off.

When '`reset`' is specified, `tput` sends the reset strings for the terminal and then follows the same initialization sequence as `tput` '`init`'.

Below are some example uses of `tput`. See Section 1.4 [Capabilities], page 4, for a complete list of the functions that `tput` can cause terminals to perform. Note that not all terminals can perform any given function. See Section 1.3 [More Examples], page 3, for some more complex samples of `tput` use.

The following command moves the cursor to row 10, column 30 of the screen:

```
tput cup 10 30
```

The following command makes the cursor invisible:

```
tput civis
```

The following command makes the cursor visible again:

```
tput cnorm
```

The following command deletes 10 lines below and including the one on which the cursor is positioned:

```
tput dl 10
```

1.2 Output and Exit Status

The `tput` command produces different kinds of output for each of the three types of terminal capabilities: string, numeric, and Boolean.

If the terminfo capability given on the command line is a string capability, `tput` displays its value and exits with a status of 0. If the capability is not defined for the terminal type being used, `tput` produces no output and exits with a status of 1.

If the capability is a numeric capability, `tput` displays its value (an integer). If the capability is not defined for the terminal type being used, `tput` displays the value '`-1`'. The exit status is always 0 for numeric capabilities, unless an error occurs (see Section 1.5 [Error Messages], page 11 for a complete list of the possible exit status values).

If the capability is a Boolean capability, `tput` produces no output and exits with status 0 if the capability is defined for the terminal type being used, or status 1 if the capability is not defined. See section "Definitions of the Terminal Capabilities" in *Termcap*, for a more detailed description of termcap capabilities.

The values of numeric capabilities should be saved into shell variables so they can be used later without having to run `tput` again. Here is how it can be done:

For the Bourne, Bourne-again, and Korn shells:

To set an environment variable: `` COLUMNS=`tput cols` export COLUMNS ``

To set a local variable: `` tabwidth=`tput it` ``

For the C shell:

To set an environment variable: `` setenv COLUMNS `tput cols` ``

To set a local variable: `` set tabwidth = `tput it` ``

The values of string capabilities can be saved in shell variables in the same way, then displayed later using the `echo` command. Since `echo` is built into most shells, it runs more quickly than `tput` does. However, using `echo` instead of `tput` to display string values can cause problems for capabilities that use padding, because null padding characters cannot be passed as arguments to commands, including `echo`.

1.3 Yet More Examples

Here are some more advanced examples of using `tput`; most involve some shell programming. Because the C shell's flow control (decision making) constructs differ from those of the other shells, these examples do not work under the C shell.

The following sequence of commands prints '`I am infalible`' and then crosses it out on terminals that can overstrike, and prints '`I am on strike`' on terminals that cannot.

```
if tput os; then
    echo 'I am infalible\r- -- ---------'
else
    echo 'I am on strike'
```

```
fi
```

The following example is a shell script that centers a line of text given as command line arguments. An alternative approach would be to have `tput` send the ‘`rep`’ terminfo capability to print the multiple spaces instead of using the `while` loop.

```
COLUMNS=‘tput cols‘ export COLUMNS # Get screen width.
echo "$@" | awk '
{ spaces = ('$COLUMNS' - length) / 2
  while (spaces-- > 0) printf (" ")
  print
}'
```

The following commands cause the terminal to save the current cursor position, print ‘`Hello, World`’ centered in the screen in reverse video, then return to the original cursor position.

```
COLUMNS=‘tput cols‘
LINES=‘tput lines‘
line=‘expr $LINES / 2‘
column=‘expr \( $COLUMNS - 6 \) / 2‘
tput sc
tput cup $line $column
tput rev
echo 'Hello, World'
tput sgr0
tput rc
```

The middle three lines of the above example can also be written using ‘`--standard-input`’.

```
tput --standard-input <<EOF
sc
cup $line $column
rev
EOF
```

1.4 Capabilities

1.4.1 Boolean Capabilities

Name	Termcap Equiv.	Description
am	am	Has automatic margins
bw	bw	‘cub1’ wraps from column 0 to last column
chts	HC	Cursor is hard to see
da	da	Display may be retained above screen
db	db	Display may be retained below screen
eo	eo	Can erase overstrikes with a blank
eslok	es	Using escape on status line is ok
gn	gn	Generic line type (e.g., ‘dialup’, ‘switch’)
hc	hc	Hardcopy terminal

hs	hs	Has a status line
hz	hz	Hazeltine; cannot print tildes
in	in	Insert mode distinguishes nulls
km	km	Has a meta key (a shift that sets parity bit)
mc5i	5i	Data sent to printer does not echo on screen
mir	mi	Safe to move while in insert mode
msgr	ms	Safe to move in standout modes
npc	NP	No pad character is needed
nrrmc	NR	‘smcup’ does not reverse ‘rmcup’
nxon	nx	Padding does not work; xon/xoff is required
os	os	Overstrikes
ul	ul	Underline character overstrikes
xenl	xn	Newline ignored after 80 columns (Concept)
xhp	xs	Standout is not erased by overwriting (HP)
xon	xo	Uses xon/xoff handshaking
xsb	xb	Beehive (f1=escape, f2=ctrl-c)
xt	xt	Tabs are destructive, magic ‘smso’ (t1061)

1.4.2 Numeric Capabilities

Name	Termcap Equiv.	Description
cols	co	Number of columns in a line
it	it	Width of initial tab settings
lh	lh	Number of rows in each label
lines	li	Number of lines on screen or page
lm	lm	Lines of memory if > ‘lines’; 0 means varies
lw	lw	Number of columns in each label
nlab	Nl	Number of labels on screen (start at 1)
pb	pb	Lowest baud rate where padding is needed
vt	vt	Virtual terminal number (CB/Unix)
wsl	ws	Number of columns in status line
xmc	sg	Number of blanks left by ‘smso’ or ‘rmso’

1.4.3 String Capabilities

In the following table, ‘(P)’ following an explanation means that the capability takes one or more parameters (and is evaluated by the `tparam` function, or in the case of ‘`cup`’, `tgoto`); ‘(*)’ means that padding may be based on the number of lines affected; and ‘`#n`’ refers to the ‘`n`’th parameter.

Name	Termcap Equiv.	Description
acsc	ac	Graphic character set pairs aAbBcC - default vt100
bel	bl	Ring bell (beep)
blink	mb	Begin blinking mode
bold	md	Begin double intensity mode

cbt	bt	Back tab
civis	vi	Make cursor invisible
clear	cl	Clear screen (*)
cmdch	CC	Settable command character in prototype
cnorm	ve	Make cursor normal (undo ‘cvvis’ & ‘civis)’
cr	cr	Carriage return (*)
csr	cs	Change scrolling region to lines #1 through #2 (P)
cub	LE	Move cursor left #1 spaces (P)
cub1	le	Move cursor left one space
cud	DO	Move cursor down #1 lines (P*)
cud1	do	Move cursor down one line
cuf	RI	Move cursor right #1 spaces (P*)
cuf1	nd	Move cursor right one space
cup	cm	Move cursor to row #1, column #2 of screen (P)
cuu	UP	Move cursor up #1 lines (P*)
cuu1	up	Move cursor up one line
cvvis	vs	Make cursor very visible
dch	DC	Delete #1 characters (P*)
dch1	dc	Delete one character (*)
dim	mh	Begin half intensity mode
dl	DL	Delete #1 lines (P*)
dl1	dl	Delete one line (*)
dsl	ds	Disable status line
ech	ec	Erase #1 characters (P)
ed	cd	Clear to end of display (*)
el	ce	Clear to end of line
el1	cb	Clear to beginning of line, inclusive
enacs	eA	Enable alternate character set
ff	ff	Form feed for hardcopy terminal (*)
flash	vb	Visible bell (must not move cursor)
fsl	fs	Return from status line
hd	hd	Move cursor down one-half line
home	ho	Home cursor (if no ‘cup’)
hpa	ch	Move cursor to column #1 (P)
ht	ta	Tab to next 8 space hardware tab stop
hts	st	Set a tab in all rows, current column
hu	hu	Move cursor up one-half line
ich	IC	Insert #1 blank characters (P*)
ich1	ic	Insert one blank character
if	if	Name of file containing initialization string
il	AL	Add #1 new blank lines (P*)
il1	al	Add one new blank line (*)
ind	sf	Scroll forward (up) one line
indn	SF	Scroll forward #1 lines (P)
invis	mk	Begin invisible text mode
ip	ip	Insert pad after character inserted (*)
iprog	iP	Path of program for initialization
is1	i1	Terminal initialization string

is2	is	Terminal initialization string
is3	i3	Terminal initialization string
kBEG	&9	Shifted beginning key
kCAN	&0	Shifted cancel key
kCMD	*1	Shifted command key
kCPY	*2	Shifted copy key
kCRT	*3	Shifted create key
kDC	*4	Shifted delete char key
kDL	*5	Shifted delete line key
kEND	*7	Shifted end key
kEOL	*8	Shifted clear line key
kEXT	*9	Shifted exit key
kFND	*0	Shifted find key
kHLP	#1	Shifted help key
kHOM	#2	Shifted home key
kIC	#3	Shifted input key
kLFT	#4	Shifted left arrow key
kMOV	%b	Shifted move key
kMSG	%a	Shifted message key
kNXT	%c	Shifted next key
kOPT	%d	Shifted options key
kPRT	%f	Shifted print key
kPRV	%e	Shifted prev key
kRDO	%g	Shifted redo key
kRES	%j	Shifted resume key
kRIT	%i	Shifted right arrow
kRPL	%h	Shifted replace key
kSAV	!1	Shifted save key
kSPD	!2	Shifted suspend key
kUND	!3	Shifted undo key
ka1	K1	Upper left of keypad
ka3	K3	Upper right of keypad
kb2	K2	Center of keypad
kbeg	@1	Beginning key
kbs	kb	Backspace key
kc1	K4	Lower left of keypad
kc3	K5	Lower right of keypad
kcan	@2	Cancel key
kcbt	kB	Back tab key
kclo	@3	Close key
kclr	kC	Clear screen or erase key
kcmd	@4	Command key
kcpy	@5	Copy key
kcrt	@6	Create key
kctab	kt	Clear tab key
kcub1	kl	Left arrow key
kcud1	kd	Down arrow key
kcuf1	kr	Right arrow key

kcuu1	ku	Up arrow key
kdch1	kD	Delete character key
kdl1	kL	Delete line key
ked	kS	Clear to end of screen key
kel	kE	Clear to end of line key
kend	@7	End key
kent	@8	Enter/send key (unreliable)
kext	@9	Exit key
kf0	k0	Function key f0
kf1	k1	Function key f1
kf10	k;	Function key f10
kf11	F1	Function key f11
kf12	F2	Function key f12
kf13	F3	Function key f13
kf14	F4	Function key f14
kf15	F5	Function key f15
kf16	F6	Function key f16
kf17	F7	Function key f17
kf18	F8	Function key f18
kf19	F9	Function key f19
kf2	k2	Function key f2
kf20	FA	Function key f20
kf21	FB	Function key f21
kf22	FC	Function key f22
kf23	FD	Function key f23
kf24	FE	Function key f24
kf25	FF	Function key f25
kf26	FG	Function key f26
kf27	FH	Function key f27
kf28	FI	Function key f28
kf29	FJ	Function key f29
kf3	k3	Function key f3
kf30	FK	Function key f30
kf31	FL	Function key f31
kf32	FM	Function key f32
kf33	FN	Function key f13
kf34	FO	Function key f34
kf35	FP	Function key f35
kf36	FQ	Function key f36
kf37	FR	Function key f37
kf38	FS	Function key f38
kf39	FT	Function key f39
kf4	k4	Function key f4
kf40	FU	Function key f40
kf41	FV	Function key f41
kf42	FW	Function key f42
kf43	FX	Function key f43
kf44	FY	Function key f44

```
kf45    FZ     Function key f45
kf46    Fa     Function key f46
kf47    Fb     Function key f47
kf48    Fc     Function key f48
kf49    Fd     Function key f49
kf5     k5     Function key f5
kf50    Fe     Function key f50
kf51    Ff     Function key f51
kf52    Fg     Function key f52
kf53    Fh     Function key f53
kf54    Fi     Function key f54
kf55    Fj     Function key f55
kf56    Fk     Function key f56
kf57    Fl     Function key f57
kf58    Fm     Function key f58
kf59    Fn     Function key f59
kf6     k6     Function key f6
kf60    Fo     Function key f60
kf61    Fp     Function key f61
kf62    Fq     Function key f62
kf63    Fr     Function key f63
kf7     k7     Function key f7
kf8     k8     Function key f8
kf9     k9     Function key f9
kfnd    @0     Find key
khlp    %1     Help key
khome   kh     Home key
khts    kT     Set tab key
kich1   kI     Ins char/enter ins mode key
kil1    kA     Insert line key
kind    kF     Scroll forward/down key
kll     kH     Home down key
kmov    %4     Move key
kmrk    %2     Mark key
kmsg    %3     Message key
knp     kN     Next page key
knxt    %5     Next object key
kopn    %6     Open key
kopt    %7     Options key
kpp     kP     Previous page key
kprt    %9     Print or copy key
kprv    %8     Previous object key
krdo    %0     Redo key
kref    &1     Reference key
kres    &5     Resume key
krfr    &2     Refresh key
kri     kR     Scroll backward/up key
krmir   kM     rmir or smir in insert mode
```

```
krpl    &3     Replace key
krst    &4     Restart key
ksav    &6     Save key
kslt    *6     Select key
kspd    &7     Suspend key
ktbc    ka     Clear all tabs key
kund    &8     Undo key
lf0     l0     Label on function key f0 if not ‘f0’
lf1     l1     Label on function key f1 if not ‘f1’
lf10    la     Label on function key f10 if not ‘f10’
lf2     l2     Label on function key f2 if not ‘f2’
lf3     l3     Label on function key f3 if not ‘f3’
lf4     l4     Label on function key f4 if not ‘f4’
lf5     l5     Label on function key f5 if not ‘f5’
lf6     l6     Label on function key f6 if not ‘f6’
lf7     l7     Label on function key f7 if not ‘f7’
lf8     l8     Label on function key f8 if not ‘f8’
lf9     l9     Label on function key f9 if not ‘f9’
ll      ll     Go to last line, first column (if no ‘cup’)
mc0     ps     Print screen contents
mc4     pf     Turn printer off
mc5     po     Turn printer on
mc5p    pO     Turn printer on for #1 bytes (P)
mgc     MC     Clear left and right soft margins
mrcup   CM     Move cursor to row #1, column #2 of memory (P)
nel     nw     Newline (like cr followed by lf)
pad     pc     Pad character (rather than nul)
pfkey   pk     Program function key #1 to type string #2 (P)
pfloc   pl     Program function key #1 to execute string #2 (P)
pfx     px     Program function key #1 to transmit string #2 (P)
pln     pn     Program label #1 to show string #2 (P)
prot    mp     Begin protected mode
rc      rc     Restore cursor to position of last ‘sc’
rep     rp     Repeat character #1, #2 times (P*)
rev     mr     Begin reverse video mode
rf      rf     Name of file containing reset string
rfi     RF     Send next input character (for ptys)
ri      sr     Scroll backward (down) one line
rin     SR     Scroll backward #1 lines (P)
rmacs   ae     End alternate character set
rmam    RA     Turn off automatic margins
rmcup   te     String to end programs that use ‘cup’
rmdc    ed     End delete mode
rmir    ei     End insert mode
rmkx    ke     End keypad transmit mode
rmln    LF     Turn off soft labels
rmm     mo     End meta mode
rmp     rP     Like ‘ip’ but when in replace mode
```

`rmso`	`se`	`End standout mode`
`rmul`	`ue`	`End underscore mode`
`rmxon`	`RX`	`Turn off xon/xoff handshaking`
`rs1`	`r1`	`Reset terminal to sane modes`
`rs2`	`r2`	`Reset terminal to sane modes`
`rs3`	`r3`	`Reset terminal to sane modes`
`sc`	`sc`	`Save cursor position`
`sgr`	`sa`	`Define video attributes #1 through #9 (P)`
`sgr0`	`me`	`Turn off all attributes`
`smacs`	`as`	`Begin alternate character set`
`smam`	`SA`	`Turn on automatic margins`
`smcup`	`ti`	`String to begin programs that use ‘cup’`
`smdc`	`dm`	`Begin delete mode`
`smgl`	`ML`	`Set soft left margin to #1 (P)`
`smgr`	`MR`	`Set soft right margin to #1 (P)`
`smir`	`im`	`Begin insert mode`
`smkx`	`ks`	`Begin keypad transmit mode`
`smln`	`LO`	`Turn on soft labels`
`smm`	`mm`	`Begin meta mode (8th bit set)`
`smso`	`so`	`Begin standout mode`
`smul`	`us`	`Begin underscore mode`
`smxon`	`SX`	`Turn on xon/xoff handshaking`
`tbc`	`ct`	`Clear all tab stops`
`tsl`	`ts`	`Go to status line, column #1 (P)`
`uc`	`uc`	`Underscore one character and move past it`
`vpa`	`cv`	`Move cursor to row #1 (P)`
`wind`	`wi`	`Set window to lines #1-#2, columns #3-#4 (P)`
`xoffc`	`XF`	`xoff character`
`xonc`	`XN`	`xon character`

1.5 Error Messages

`tput` displays various error messages if problems occur. In addition, it exits with one of the following status values:

0 Normal status; the given capability is present.

1 The given Boolean or string capability is not present.

2 Usage error; `tput` was given invalid arguments.

3 The terminal type given (either in the ‘`TERM`’ environment variable or by the ‘`-T`’ switch) is unknown, or the termcap database can not be read.

4 The given capability is unknown.

2 `tabs`: Setting Terminal Tabs

The `tabs` command is used to specify and set hardware tab stops on terminals that have remotely-settable tab stops.

The `tabs` command is used to specify and set hardware tab stops on terminals that have remotely-settable tab stops.

`tabs` [*options*] *tab-specification*

2.1 Options

The options supported by GNU `tabs` are:

`-T` *termtype*
`--terminal=`*termtype*
: Specify the terminal type, overriding the environment variable `TERM`.

`-V`
`--version`
: Show the version number of GNU `tabs` and exit.

`-h`
`--help`
: Show the command line usage of GNU `tabs` and exit.

2.2 Tab Specification

GNU `tabs` accepts the following types of tab specification:

n1`[,`*n2*`,`...`]`
: Set tab stops at columns *n1*, *n2*, ... The leftmost column is the column 1. *n2* and the rest of the list can be of the form ‘`+`*num*’, in which case it specifies the increment from the tab stop last specified by the list.

`-`*n*
: Set tab stops at every *n* columns. For most of the terminals, the standard tab stop setting is the equivalent of ‘`-8`’.

`-`*code*
`--code=`*code*
`-C` *code*
: Set tab stops using one of the “canned” specification. The second and the third formats are useful if the name of the canned specification conflicts with one of the long options supportred by GNU `tabs`.

`--`*filename*
`--file=`*filename*
`--F` *filename*
: Read the first line of *filename*, find parameters separated by blanks enclosed by `<:` and `:>` on that line, find a parameter that begins with `t`, and then use it as tab specification. If any of the above steps fails, set tab stops to every 8 columns. The second and the third formats are useful if the name of the file conflicts with one of the long options supportred by GNU `tabs`.

2.3 Canned tab specifications

GNU `tabs` supports the following "canned" specifications:

'a' Assembler, IBM S/370, first format (1,10,16,36,72)

'a2' Assembler, IBM S/370, second format (1,10,16,40,72)

'c' COBOL, normal format (1,8,12,16,20,55)

'c2' COBOL, compact format (1,6,10,14,49)

'c3' COBOL, compact format with more tab stops (1,6,10,14,18,22,26,30,34,38,42,46,50,54,58,62,67)

'f' FORTRAN (1,7,11,15,19,23)

'p' PL/I (1,5,9,13,17,21,25,29,33,37,41,45,49,53,57,61)

's' SNOBOL (1,10,55)

'u' UNIVAC 1100 Assembler (1,12,20,44)

2.4 Examples

To set tab stops for editing FORTRAN programs on an `xterm`, use the following `tabs` command:

```
tabs -f --terminal=xterm
```

Another example: a source file of FORTRAN program may contain its own tab specification on its first line:

```
C <: t1,7,11,15,19,23,72 :>
```

Before editing this file, the `tabs` command can be used to set tab stops to columns this file expects:

```
tabs --Sample.f
```

Table of Contents