NAME

tset, reset – establish or restore terminal characteristics

SYNOPSIS

tset [–**InQrs**] [–**e***c*] [–**k***c*] [–**m** [*port –ID* [*baudrate*] : *type*] . . .] [*type*]

reset [–] [–**e***c*] [–**I**] [–**k***c*] [–**n**] [–**Q**] [–**r**] [–**s**]
[–**m** [*indent*] [*test baudrate*]: *type*] . . . [*type*]

AVAILABILITY

SUNWscpu

DESCRIPTION

tset sets up your terminal, typically when you first log in. It does terminal dependent processing such as setting erase and kill characters, setting or resetting delays, sending any sequences needed to properly initialized the terminal, and the like. **tset** first determines the *type* of terminal involved, and then does necessary initializations and mode settings. If a port is not wired permanently to a specific terminal (not hardwired) it is given an appropriate generic identifier such as **dialup**.

reset clears the terminal settings by turning off CBREAK and RAW modes, output delays and parity checking, turns on NEWLINE translation, echo and TAB expansion, and restores undefined special characters to their default state. It then sets the modes as usual, based on the terminal type (which will probably override some of the above). See **stty**(1) for more information. All arguments to **tset** may be used with **reset**. **reset** also uses **rs=** and **rf=** to reset the initialization string and file. This is useful after a program dies and leaves the terminal in a funny state. Often in this situation, characters will not echo as you type them. You may have to type LINEFEED **reset** LINEFEED since RETURN may not work.

When no arguments are specified, **tset** reads the terminal type from the **TERM** environment variable and re-initializes the terminal, and performs initialization of mode, environment and other options at login time to determine the terminal type and set up terminal modes.

When used in a startup script (**.profile** for **sh**(1) users or **.login** for **csh**(1) users) it is desirable to give information about the type of terminal you will usually use on ports that are not hardwired. Any of the alternate generic names given in the file **/etc/termcap** are possible identifiers. Refer to the –**m** option below for more information. If no mapping applies and a final *type* option, not preceded by a –**m**, is given on the command line then that type is used.

It is usually desirable to return the terminal type, as finally determined by **tset**, and information about the terminal's capabilities, to a shell's environment. This can be done using the –, –**s**, or –**S** options.

For the Bourne shell, put this command in your **.profile** file:

**eval `tset –s** *options...*`

or using the C shell, put these commands in your **.login** file:

set noglob
**eval `tset –s** *options...*`
unset noglob

With the C shell, it is also convenient to make an alias in your **.cshrc** file:

**alias ts ´eval `tset –s \!***`´

This also allows the command:

ts 2621

to be invoked at any time to set the terminal and environment. It is not possible to get this aliasing effect with a Bourne shell script, because shell scripts cannot set the environment of their parent. If a process could set its parent's environment, none of this nonsense would be necessary in the first place.

Once the terminal type is known, **tset** sets the terminal driver mode. This normally involves sending an initialization sequence to the terminal, setting the single character erase (and optionally the line-kill (full line erase)) characters, and setting special character delays. TAB and NEWLINE expansion are turned off during transmission of the terminal initialization sequence.

On terminals that can backspace but not overstrike (such as a CRT), and when the erase character is '**#**', the erase character is changed as if **–e** had been used.

OPTIONS

– The name of the terminal finally decided upon is output on the standard output. This is intended to be captured by the shell and placed in the **TERM** environment variable.

–e*c* Set the erase character to be the named character *c* on all terminals. Default is the BACKSPACE key on the keyboard, usually **^H** (CTRL-H). The character *c* can either be typed directly, or entered using the circumflex-character notation used here.

–i*c* Set the interrupt character to be the named character *c* on all terminals. Default is **^C** (CTRL-C). The character *c* can either be typed directly, or entered using the circumflex-character notation used here.

–I Suppress transmitting terminal-initialization strings.

–k*c* Set the line kill character to be the named character *c* on all terminals. Default is **^U** (CTRL-U). The kill character is left alone if **–k** is not specified. Control characters can be specified by prefixing the alphabetical character with a circumflex (as in CTRL-U) instead of entering the actual control key itself. This allows you to specify control keys that are currently assigned.

–n Specify that the new tty driver modes should be initialized for this terminal. Probably useless since **stty new** is the default.

−Q Suppress printing the '**Erase set to**' and '**Kill set to**' messages.

−r In addition to other actions, reports the terminal type.

−s Output commands to set and export **TERM**. This can be used with

```
set noglob
eval `tset −s . . .`
unset noglob
```

to bring the terminal information into the environment. Doing so makes programs such as **vi**(1) start up faster. If the **SHELL** environment variable ends with **csh**, C shell commands are output, otherwise Bourne shell commands are output.

−m [*port-ID* [*baudrate*] **:** *type*] . . .

Specify (map) a terminal type when connected to a generic port (such as *dialup* or *plugboard*) identified by *port-ID*. The *baudrate* argument can be used to check the baudrate of the port and set the terminal type accordingly. The target rate is prefixed by any combination of the following operators to specify the conditions under which the mapping is made:

- **>** Greater than
- **@** Equals or ''at''
- **<** Less than
- **!** It is not the case that (negates the above operators)
- **?** Prompt for the terminal type. If no response is given, then *type* is selected by default.

In the following example, the terminal type is set to **adm3a** if the port is a dialup with a speed of greater than 300 or to **dw2** if the port is a dialup at 300 baud or less. In the third case, the question mark preceding the terminal type indicates that the user is to verify the type desired. A **NULL** response indicates that the named type is correct. Otherwise, the user's response is taken to be the type desired.

```
tset −m 'dialup>300:adm3a' −m 'dialup:dw2' −m \
    'plugboard:?adm3a'
```

To prevent interpretation as metacharacters, the entire argument to −**m** should be enclosed in single quotes. When using the C shell, exclamation points should be preceded by a backslash (\).

EXAMPLES

These examples all use the '−' option. A typical use of **tset** in a **.profile** or **.login** will also use the −**e** and −**k** options, and often the −**n** or −**Q** options as well. These options have been omitted here to keep the examples short.

To select a 2621, you might put the following sequence of commands in your **.login** file (or **.profile** for Bourne shell users).

```
set noglob
eval `tset −s 2621`
unset noglob
```

If you have a switch which connects to various ports (making it impractical to identify which port you may be connected to), and use various terminals from time to time, you can select from among those terminals according to the *speed* or baud rate. In the example below, **tset** will prompt you for a terminal type if the baud rate is greater than 1200 (say, 9600 for a terminal connected by an RS-232 line), and use a Wyse® 50 by default. If the baud rate is less than or equal to 1200, it will select a 2621. Note the placement of the question mark, and the quotes to protect the **>** and **?** from interpretation by the shell.

```
set noglob
eval `tset –s –m 'switch>1200:?wy' –m 'switch<=1200:2621'`
unset noglob
```

The following entry is appropriate if you always dial up, always at the same baud rate, on many different kinds of terminals, and the terminal you use most often is an **adm3a**.

```
set noglob
eval `tset –s ?adm3a`
unset noglob
```

If you want to make the selection based only on the baud rate, you might use the following:

```
set noglob
eval `tset –s –m '>1200:wy' 2621`
unset noglob
```

The following example quietly sets the erase character to BACKSPACE, and kill to CTRL-U. If the port is switched, it selects a Concept™ 100 for speeds less than or equal to 1200, and asks for the terminal type otherwise (the default in this case is a Wyse 50). If the port is a direct dialup, it selects Concept 100 as the terminal type. If logging in over the ARPANET, the terminal type selected is a Datamedia® 2500 terminal or emulator. Note the backslash escaping the NEWLINE at the end of the first line in the example.

```
set noglob
eval `tset –e –k^U –Q –s –m 'switch<=1200:concept100' –m \
        'switch:?wy' –m dialup:concept100 –m arpanet:dm2500`
unset noglob
```

FILES

.login
.profile
/etc/termcap

SEE ALSO

csh(1), **sh**(1), **stty**(1), **vi**(1), **environ**(5)

NOTES

The **tset** command is one of the first commands a user must master when getting started on a UNIX system. Unfortunately, it is one of the most complex, largely because of the extra effort the user must go through to get the environment of the login shell set. Something needs to be done to make all this simpler, either the **login** program should do this stuff, or a default shell alias should be made, or a way to set the environment of the parent should exist.

This program cannot intuit personal choices for erase, interrupt and line kill characters, so it leaves these set to the local system standards.

It could well be argued that the shell should be responsible for ensuring that the terminal remains in a sane state; this would eliminate the need for the **reset** program.