NAME

gterm – virtual graphics terminal for the SunView environment

SYNOPSIS

gterm [**–C**] [**–T**] [**–B** *boldstyle*] [**–I** *command*] [**–ignore**] [*ttyargs*]
[**–G** [**–Gopen** (**noaction** | **show** | **expose**)] [**–Gclose** (**noaction** | **blank** | **hide**)]
[**–**[**no**]**pause**] [**–**[**no**]**retain**] [**–**[**no**]**clip**] [**–**[**no**]**reverse**] [**–color** | **–mono**]
[**–ginterm** [*char* [*char*]]] [*gioargs*]]
[*program* [*args*]]

GETTING STARTED

gterm is a virtual graphics terminal implemented as a set of windows running within the SunView environment. The graphics terminal consists of two primary windows, the main **text window** and a **graphics subwindow**. The text window is a standard SunView TTY subwindow, identical to that used in *shelltool*. The graphics window is a Tektronix 4012 compatible graphics window. The two windows may be moved and sized independently. Terminal i/o may be directed to either window, but to only one window at a time. When i/o is directed to the text window the terminal is said to be in **text mode**, and when i/o is directed to the graphics window the terminal is said to be in **graphics mode**. Mode switching may be performed either manually or under program control. *gterm* is upwards compatible with *shelltool*.

OPTIONS

–C Redirect system console output to this instance of *gterm*.

–T Run *gterm* with the graphics plane disabled. Any graphics commands embedded in the input data stream will appear as printable characters in the text window. This option is useful when using *gterm* for a nongraphics session over a noisy line, or when debugging a graphics program and switching back and forth between text and graphics mode would interfere with the operation of the debugger.

–B *boldstyle*
Sets the *boldstyle* (rendering of boldface type in the text window) for this instance of *gterm*. The choices are the same as for any tool based on a SunView tty subwindow: see *shelltool*(1) for a description of the *boldstyle* options.

–I *command*
Input the given command to the program or shell run in the *gterm* window, as if it had been typed into the window. Spaces in the command must be escaped, or the command string must be quoted.

–ignore Ignore the suspend control characters (SIGTSTP), *if* the process being run in the *gterm* window is not the default SHELL process. If this option is in effect, typing one of the *stty* suspend-process control characters **susp** or **dsusp** while the terminal is not in raw mode will cause the character to be echoed in the text window, but to be otherwise ignored. This may be desirable if the process being run in the *gterm* window is not capable of restarting itself after it has been suspended (see also the **continue** frame menu option).

ttyargs *gterm* also takes generic tool arguments; see *suntools*(1) for a list of these arguments. The *ttyargs* generic tool arguments affect only the text window.

–G Generic tool arguments preceding this optional placeholder switch affect only the text window; those following the switch affect only the graphics window.

-Gopen (**noaction** | **show** | **expose**)
Specifies the action to be taken when graphics mode is entered, i.e., when terminal i/o is redirected from the text window to the graphics window. **noaction** means do nothing which visibly affects the screen. **show** means display the graphics frame if it is not already visible on the screen, i.e., because the graphics window has been closed. **expose** means move the graphics window, if already displayed, to the top of the stack of open windows.

-Gclose (**noaction** | **blank** | **hide**)

Specifies the action to be taken when graphics mode is exited, restoring terminal i/o to the text window. **noaction** means do nothing which visibly affects the screen. **blank** means close the graphics subwindow, causing the window to disappear from the screen (the closed window can be redisplayed at any time provided the contents are *retained*). **hide** means move the graphics window to the bottom of the stack of open windows.

–[**no**]**pause**

Pause before closing the graphics window and returning to text mode, after opening the graphics frame to display a plot noninteractively (no cursor reads). If **–nopause** is specified and the **–Gclose** option is **blank**, a newly plotted graph will disappear immediately after the plotting operation is completed, assuming no blocking operation such as a cursor read occurs while in graphics mode. If **–pause** is specified *gterm* will detect this condition and pause for a keystroke to be typed before exiting graphics mode.

–[**no**]**retain**

Specifies whether or not the contents of the graphics frame are to be retained. If **–noretain** is specified plotting will be somewhat faster, but any window event which affects the region of the workstation screen occupied by the graphics frame will cause the plot to be lost.

–[**no**]**clip**

Specifies whether or not clipping of graphics vectors is to be performed. If **–noclip** is specified plotting will be somewhat faster, but vectors may be drawn outside the graphics window if the screen changes while a graph is being drawn.

–[**no**]**reverse**

Specifies whether the graphics frame is to be displayed in normal or reverse video. What constitutes normal video depends upon whether color or monochrome is specified, and upon what has been specified for the foreground and background colors. Reverse video swaps the foreground and background colors within the graphics window.

–color Display the graphics frame in two colors, if *gterm* is run on a color workstation.

–mono Display the graphics frame in monochrome, using the foreground and background colors specified when *suntools*(1) was started.

–ginterm [*char* [*char*]]

Set the GIN terminator character or characters (used to delimit the value returned for a cursor read) to the indicated octal values *char*. If a *char* value is omitted the corresponding GIN mode terminator will be omitted when a cursor value is returned. For example, "–ginterm" causes both GIN delimiter characters to be omitted, and "–ginterm 015" causes the single GIN terminator character CR (carriage return) to be transmitted.

gioargs *gterm* also takes generic tool arguments; see *suntools*(1) for a list of these arguments. The *gioargs* generic tool arguments affect only the graphics subwindow, and are recognized only after the **–G** placeholder argument.

[*program* [*args*]]

If a *program* argument is present, the named program is run in the *gterm* window without spawning an intermediate shell. If no *program* is specified, *gterm* runs the shell specified by your SHELL environment variable. If this environment variable is not defined, *gterm* runs **/bin/sh**.

Minimum match abbreviations are permitted for all *gterm* arguments except the generic tool arguments, which have two character aliases. Use of the full names is however recommended, as the number of characters required for minimum matching a specific argument may change in a future release of *gterm* when new command line arguments are added.

DESCRIPTION

Virtual Graphics Terminal

gterm is a virtual graphics terminal implemented as a set of windows running within the SunView environment. The terminal consists of two primary windows, a **text window** and a **graphics window**, and several special purpose subwindows, the most important of which is the **setup panel**, used to dynamically change *gterm* options at runtime, or to reset the terminal to a known state. The virtual terminal is interfaced to applications programs via the UNIX pseudoterminal or *pty*(4) interface, providing an applications interface identical to that provided for ordinary terminals, allowing applications running either on the local machine or on a remote node communicating via a network interface (*telnet*, *rlogin*, modem etc.) to communicate transparently with the *gterm* virtual terminal.

Text Window

The text window is a standard SunView TTY subwindow, as used in *shelltool*, *gfxtool*, and so on. Documentation for tty subwindows, including both user documentation and a description of the special escape sequences recognized by tty subwindows, is given in *shelltool*(1). In particular, note that a **.ttyswrc** file may be placed in one's login directory to set tty subwindow parameters, and to map function keys to control sequences to be sent either to the tty subwindow or to the program running in the tty subwindow. The **selection service** can be useful for passing text to programs via the mouse instead of the keyboard, and numerous terminal **escape sequences** are defined for resizing the text window, setting the frame label, and so on.

Graphics Window

The graphics window emulates a Tektronix 4012 terminal with minor differences, e.g., the screen size is 35x80 rather than 35x75, and some significant extensions, e.g., graphics and text can be selectively erased, and support is provided for a **status line** at the bottom of the screen, in which text can be dynamically read and written without affecting the contents of the graphics plane. Although nonstandard, these features are very useful when designing interactive user interfaces, and they can be emulated on most modern graphics terminals (hence programs which use these features need not be device dependent).

The position and size of the graphics window may be set at startup time via the generic tool arguments *gioargs*. A number of preprogrammed sizes may also be selected via the setup panel at runtime, or the mouse may be used to directly resize the window to produce a window of any size and aspect ratio. The standard graphics window sizes range from very small to the full screen and all share the same standard landscape mode aspect ratio; users should note that manually adjusting the window size usually results in a window with a nonstandard aspect ratio, which may cause graphics programs which assume the standard aspect ratio to misbehave, producing oddly shaped objects, or poorly centered text strings.

gterm will automatically detect any changes in the size of the graphics window, adjusting the transformation from 4012 coordinates (780x1024) to screen coordinates so that subsequent graphics and text will be drawn at the correct relative position within the window. The best text font for the new window size is also selected; if the window is of an arbitrary size it will not in general be possible to select a font which provides exactly 35x80 characters on the screen without overlap, since there are only a limited number of fixed size fonts to choose from. If the window is especially wide and short text lines may overlap vertically. After resizing the graphics window, the previously displayed graph *must* be redrawn under program control to adjust the graph to fit the new window.

By default, the graphics window overlaps the text window, with eight or ten characters of text visible to the left of the graphics window. If the graphics window is left in its default position and the text window is repositioned on the workstation screen, the graphics window will "track" the text window, i.e., retain its position relative to the text window (this is desirable when there are multiple instances of *gterm* in use at one time to avoid losing track of which graphics frame goes with which text window). If the graphics window is manually repositioned, however, then thereafter the positions of the two windows are completely independent.

Although the graphics window is normally used only for graphics (plotting), one should not forget that the graphics window emulates an (80 column wide) Tektronix 4012 terminal, and hence may be used as a terminal for ordinary text i/o, as well as for graphics. The TTY subwindow will normally be superior for

terminal graphics, but the extra page of text and the larger, brighter font typically used in the graphics window may occasionally be useful for some applications. Text i/o may easily be switched back and forth between the text and graphics windows via the function keys described below, transparently to most applications software.

Active Window

At any one time, terminal output is always directed to either the text window or the graphics window, but never to both at the same time. When output is directed to the text window the terminal is said to be in **text mode**, and when output is directed to the graphics window the terminal is said to be in **graphics mode**, although either text or graphics may be written into the graphics plane. Mode switching is normally in response to control codes embedded in the input data stream from the applications program being run, but keyboard function keys may be used to manually perform mode switching if desired.

Keyboard *input* may be directed to either window at any time, regardless of which window is currently active, i.e., text may be typed into one window but echoed in the other window. This can be confusing if the window in which text is echoed is not currently displayed; one types and nothing seems to be happening, but in fact commands are being input and executed normally. For example, if the **–Gclose** option is set to **noaction** and the graphics window is adjusted to fill the full screen, then when graphics mode is exited terminal output will be directed to the text window, but the text window will be covered by the graphics window, and subsequent commands and textual output will not be visible without manually redisplaying the text window (or undisplaying the graphics window) with a function key.

The Frame Menu

gterm uses a special frame menu which may be called up by the mouse at any time, by placing the cursor on the border of the *text window* and holding down the right mouse button. The items in the *gterm* frame menu are as follows:

Frame	Access the standard SunView frame menu.
Setup	Display the setup panel.
Continue	Send the SIGCONT signal to the process group attached to the *gterm* window, e.g., after accidentally suspending a process which cannot otherwise restart itself. See also the **–ignore** command line option.
Show graph	Display the graphics window.
Textcopy	Make a hardcopy of the text window.
Graphcopy	Make a hardcopy of the graphics window. (IRAF users should use the cursor mode *snap* function instead).
Screencopy	Make a hardcopy of the full screen.

The textcopy, graphcopy, and screencopy selections are all entry points to the general screen capture utility, discussed in the next section.

Hardcopy Output

The hardcopy functions produce a "what you see is what you get" bitmap of the rectangular region of the screen occupied by the indicated object of interest. If the region of interest is partially covered by another window, then the hardcopy will be a picture of a partially covered window. If the window is displayed in reverse video, the hardcopy will also be rendered in reverse video.

The screen capture software reads out the full memory of the workstation in the region of interest, and in the case of a color workstation, processes the screen pixels through the colortable to produce an image corresponding to what appears on the screen. No full color output options are currently provided, hence the average of the red, green, and blue color values is next computed. If a bitmap output image is desired a simple thresholding algorithm is used to produce the final bitmap image, otherwise a greyscale image is produced. If rasterfile output is being generated, the raw pixel values and RGB color table entries are saved directly in the rasterfile, rather than applying the tables in software to produce a monochrome or

bitmap image.

Two output options are currently provided, i.e., **Postscript** output suitable for output directly to a laser writer to produce the final graphics hardcopy, or **Sun rasterfile** output. The default action is to output a Postscript program to the device "lw", e.g., the Apple Laserwriter (any 300 dpi Postscript device should do). These defaults may be changed by defining the following environment variables:

R_RASTERFILE
: If this variable is defined a Sun rasterfile will be generated, otherwise a Postscript plotfile is generated. The string value of the variable is a *printf* style format string to be used to generate the filename of the rasterfile. If multiple rasterfiles are to be generated, the format string may contain a decimal integer field (e.g., "frame.%d") to be replaced by the *file number* of the current rasterfile. The first file generated will be number zero, with the file number being incremented once for every rasterfile produced. If Postscript plotfile output is desired, the plotfile will be a uniquely named temporary file in **/tmp**.

R_DISPOSE
: The string value of this variable is a *printf* style format string with one string valued subfield to be replaced by the plotfile or rasterfile name, to be used to generate the command used to dispose of the output file. If this variable is not defined and the output file is a Postscript plotfile, the default format string "lpr -Plw -r -s %s" will be used. If the variable is not defined and the output file is a rasterfile, no action is taken. It is the responsibility of the dispose command to delete the output file.

It should only take several seconds to capture the screen and produce the output rasterfile or queue the Postscript job to the printer. The screen is flashed to indicated when the operation has completed. The Postscript processing time may take up to several minutes (of laserwriter time) in the worst case, i.e., a hardcopy of the full workstation screen.

The Setup Panel

The setup panel is used to dynamically change terminal options while the terminal is in use. In general, nearly any terminal option which can be set on the command line when **gterm** is started can also be set via the setup panel, and vice versa. The setup panel can also be used to reset the terminal to the startup or "power on" state. The setup panel may be called up at any time via the frame menu; it is normally closed after the desired setup action has been performed. By default the setup panel is located within the text window, but it may be moved anywhere else on the workstation screen if desired.

There are two types of items in the setup panel: multiple choice options and "push buttons". To see what the choices are in a multiple choice option, position the mouse to the area where the current choice is displayed and depress the right mouse button. At this point a selection may be made by moving the mouse to the desired selection and releasing the mouse button. Alternatively, the left mouse button may be used to cycle through the choices. To perform the action indicated on a push button, place the mouse cursor on the button and press the left mouse button.

The multiple choice options in the setup panel are the following:

Graphics plane
: These options determine what the terminal does when graphics data and control instructions are encountered in the input stream. **Disable** means disable the graphics plane, causing the terminal to output graphics control codes and data as printable characters in the text window. **Enable** means enable the graphics plane for normal mixed text and graphics operation. **Discard Graphics Data** means discard all graphics data, effectively disabling the graphics plane.

Open workstation action
: These options determine the action taken by the terminal when graphics mode is entered. Some visible action is generally desirable to render the graphics window fully visible, and to indicate that a mode switch has occurred. **No action** means do nothing which visibly affects the workstation. **Show graphics** means open the graphics window, e.g., if the graphics window is to be

closed (not displayed) when the terminal is in text mode. **Expose graphics** means move the graphics window to the top of the stack of open windows, displaying any portions of the graphics window which may have been covered by other windows (such as the text window).

Close workstation action

These options determine the action taken by the terminal when graphics mode is exited, returning the terminal to text mode. **No action** means do nothing. **Blank graphics** means close the graphics window, i.e., remove the window from the screen. **Hide graphics** means move the graphics window to the bottom of the stack of open windows, allowing any overlapping windows to cover the graphics window.

Pause on close workstation

This boolean option determines whether or not the terminal displays the **pause panel**, waiting for a key to be typed, before exiting graphics mode following a noninteractive graphics session (no cursor input).

Retain graphics frame

This boolean option determines whether or not the contents of the graphics window are *retained*. Graphics drawing will be somewhat faster if the graphics plane is not retained, but almost any event which affects the region of the screen occupied by the graphics window will cause the contents of an unretained window to be lost.

Clip graphics

This boolean option determines whether or not graphics vectors are clipped to the boundaries of the visible portions of the graphics window. Graphics drawing will be somewhat faster if clipping is disabled, but vectors may be drawn in nearby, unrelated windows, especially if the graphics window is partially covered by other windows.

Graphics screen type

This option determines whether the graphics plane is to be displayed in color or monochrome on a color workstation. If **mono** is selected the foreground and background colors specified when *suntools* was started are used. If **color** is specified graphics will be rendered in color, with the colors used being specified by the **–Wb** and **–Wf** generic tool arguments in *gioargs*. On a monochrome workstation the only option displayed will be **mono only**, indicating that color is not available.

Graphics video

This option specifies whether graphics are to be rendered in **normal** or **reverse** video. Specifying reverse video causes the foreground and background colors of the graphics window to be reversed. This option may not work on some monochrome workstations.

Graphics font and screen sizes

This option is used to select at runtime the size of graphics window to be used. The graphics window may be resized at any time, including while graphics is being drawn or during a cursor read, but any displayed graphics should always be redrawn following a window resize to ensure that the graph reflects the new coordinate system. The graphics window configurations currently available are listed below in the form ''*pointsize*:[*width*x*height*]'', where *width* and *height* are in pixels. The size of the full screen is workstation dependent, the most common size currently being 1152x900. The exact set of fonts and screen sizes may change in the future as new fonts become available and workstations increase in resolution.

10:[560x420] 12:[640x490] 14:[720x560] 16:[800x630] 18:[880x665] 24:fullscreen

All choices represent 35x80 windows with the standard landscape mode aspect ratio. The size of the graphics window is the size in pixels of a character of the fixed width font used, scaled by 35 vertically and by 80 horizontally. Arbitrary sized windows may also be created by manually sizing the window with the mouse, but this is bound to result in windows with a nonstandard number of lines or columns of text, or a nonstandard aspect ratio.

GIN mode terminators

Set the GIN (graphics or cursor mode input) terminator characters to the indicated octal values. When a key is hit to terminate a cursor read, the terminal transmits a 5 character cursor value sequence to the applications program, following by one or two GIN mode terminator characters. The required GIN mode terminator(s) will in general depend upon the applications program being run. Some programs require no terminators, others require a single CR (octal 015), and so on. The default GIN mode terminator is a single CR.

To enter a new value, select the value box with the left mouse button, rubout the old value, and type in the new value as a string, with zero, one, or two octal values denoting the desired terminator characters, then hit return to establish the new value. Entering a blank string disables both terminators.

The following "push buttons" are also provided in the setup panel:

Reset Reset the terminal to the "power on" state, preserving the values of any options set on the command line, but cancelling any options selected via the setup panel. A **setup reset** is indicated if the terminal does not seem to be behaving correctly. Resetting the internal state of the terminal has no effect on the operation of any applications program being run from the terminal.

Clear Clear the text window (the F9 function key performs the same function).

Gclear Clear the graphics window, leaving the terminal in graphics mode (the F8 function key performs the same function).

Show graphics Open (display) or close (undisplay) the graphics frame. The contents of the graphics frame are not affected.

Quit Close the setup panel.

Closing and opening either the text or graphics frame has no effect on the state of the terminal or on the applications program running within it, even while a cursor read is in progress.

Function Keys

The following function keys have special significance to *gterm*:

F8 In text mode, causes a switch to graphics mode. When already in graphics mode, causes the graphics frame to be cleared.

F9 In graphics mode, causes a switch to text mode. When already in text mode, causes the text frame to be cleared.

To momentarily view the graphics frame while in text mode, one can type F8 followed by F9, without affecting the contents of either window. Commands may be entered in either window, hence to direct the output of a command to the graphics window, one could hit F8, execute the command, and then hit F9 to return to the text window. The standard SunView L7 key, used to close a window, is also detected by *gterm*, hence closing the graphics window with L7 while in graphics mode will automatically cause the terminal to revert to text mode.

Manual control of the terminal mode is sometimes necessary when running naive graphics programs in a *gterm* window. When running a graphics program which uses only standard 4012 instructions, it may be necessary to manually put the terminal into graphics mode with the F8 function key before running the program, or part of the program output may be "lost" (directed to the text window and discarded). Similarly, naive programs will not return the terminal to text mode after generating a plot, hence it will be necessary for the user to hit the F9 key to return to text mode.

Additional function keys may be defined in the user **~/.ttyswrc** file. For example, the function key definitions

```
mapo R1 ^[[8;24;80t
mapo R2 ^[[8;34;80t
mapo R3 ^[[8;40;80t
```

will program the R1, R2, and R3 function keys to set the size of the text window to 24, 34, or 40 lines by 80 columns when the corresponding function key is typed. These definitions are handy for rapidly resizing the text window to one of the "standard" terminal sizes; this is especially useful when executing programs remotely over the network, as most such programs assume some standard size terminal screen.

Mouse Buttons

The significance of the mouse buttons depends upon which window the mouse is in, and upon whether or not the terminal is in GIN mode, i.e., in the process of reading the graphics cursor. When the terminal is in text mode and the mouse is in the text window, the mouse buttons are used only for the **selection service**, as described in *shelltool*(1). The functions of the mouse buttons while the mouse is in the graphics window are outlined below.

Left button	Ignored except in GIN mode, when it may be aliased to a keyboard key and used to terminate a cursor read.
Middle button	Ignored except in GIN mode, when it may be aliased to a keyboard key and used to terminate a cursor read.
Right button	In GIN mode, may be aliased to a keyboard key and used to terminate a cursor read. When not in GIN mode, causes the cursor crosshairs to be displayed while the button is depressed.

The ability to **alias** mouse buttons to keyboard keys is a very important one as it allows arbitrary graphics programs which are driven via an interactive graphics cursor loop to be controlled completely from the mouse, rather than having to position the mouse and then hit a key on the keyboard to terminate each cursor read. For example, to alias *key* to the left mouse button, one would depress the control key and tap the left mouse button twice, immediately after hitting *key* to terminate a normal cursor read. Thereafter, either *key* or the left mouse button may be used equivalently to terminate a cursor read. The alias remains in effect until the terminal is *reset* or the alias is reassigned to a different key.

The Terminal Emulator

The normal function of the terminal is to simultaneously listen for input (program output) on the pseudoterminal file descriptor, while servicing asynchronous keyboard and mouse events generated by the user. The input data stream from the applications program consists of a mixture of text and graphics data transmitted as an ASCII byte stream with no record boundaries. Null bytes in the input data stream are ignored, and no programmed delays are needed for proper terminal operation. As input data is received asynchronously it is copied into a circular buffer and a synchronous event is queued to call a routine which subsequently processes the input characters onto the screen. If input data arrives faster than it can be processed onto the screen **<ctrl/s>** is transmitted to the *pty* terminal driver, followed by **<ctrl/q>** once the circular buffer empties. Characters typed by the user are transmitted directly to the terminal driver, which in normal operation will echo the characters back to the terminal as ordinary data.

The initial state of the terminal is text mode. Transition to graphics mode occurs when the GS character is encountered in the input data stream. Transition back to text mode occurs when the CAN character is encountered in the input stream. While text mode is in effect all input is passed on to the TTY subwindow; while graphics mode is in effect all input is passed on to the graphics subwindow. The behavior of the ANSI standard TTY subwindow is documented elsewhere (e.g., *shelltool*(1), *cons*(4s), and Chapter 10 of the *SunView Programmer's Guide*) hence will not be discussed further here. Likewise, the basic Tektronix 4012 protocol is a well known standard and need not be documented in detail here.

The control codes and escape sequences recognized by the *gterm* graphics window are summarized below. Sequences marked with a ** at the right are nonstandard extensions, although all except the status line feature are fairly common extensions.

GS (035)	**open workstation**, start normal vector drawing sequence	
CAN (030)	**close workstation**	**
FS (034)	start pointmode vector	
US (037)	set alpha mode	

CR (015)	set alpha mode and execute carriage return	
BEL (007)	ring bell and/or flash screen	
ESC CR	set status line mode (ESC = 033)	**
ESC ENQ	inquire graphics state and cursor position	
ESC SUB	initiate a cursor read (SUB = 032)	
ESC FF	clear screen, home alpha cursor (FF = 014)	
ESC / f	set cursor position to current drawing coordinates	**
ESC 0	set character size 0	
ESC 1	set character size 1 [not implemented]	
ESC 2	set character size 2 [not implemented]	
ESC 3	set character size 3 [not implemented]	
ESC / 0 d	set data level 0 (clear bits)	**
ESC / 1 d	set data level 1 (set bits)	**
ESC / 2 d	set data level 2 (toggle bits)	**
ESC ‘	set line style 0 (solid)	
ESC a	set line style 1 (dashed)	
ESC b	set line style 2 (dotted)	
ESC c	set line style 3 (dashdot)	
ESC d	set line style 4 (dash3dot)	
ESC / 0 w	set line width 0 (1 pixel)	**
ESC / 1 w	set line width 1 (2 pixels)	**
ESC / 2 w	set line width 2 (3 pixels)	**

Both text and vectors may be erased by setting the data level to 0 and redrawing the objects to be erased. Erasing points which are common to more than one object will cause gaps in other objects sharing the erased point.

Setting **status line** mode causes the region of the graphics frame occupied by the status line to be saved in a memory pixrect, after which the status line is cleared and the status line alpha cursor positioned to the start of the line (the status line is a single 80 character line of text at the bottom of the graphics window). While output is directed to the status line, data characters are output in the status line as for a terminal. BS and DEL behave as expected, allowing characters to be erased. Lines longer than 80 characters are truncated at the right margin. LF (newline) is treated the same as CR, causing the entire line to be erased, and if multiple lines of text are rapidly written to the status line they will scroll as on a one-line terminal. Status line mode is terminated by any control character in the input data stream, e.g., GS, FS, CAN, ESC, and so on. Note that terminating status line mode does not in itself erase the status line, restoring the saved region of the graphics frame to the screen; this is done by transmitting newline or CR to the terminal.

SEE ALSO

suntools(1), shelltool(1), tektool(1), cmdtool(1), pty(4), cons(4s)
SunView Programmer’s Guide, Chapter 10 – TTY Subwindows
Windows and Window-Based Tools: Beginner’s Guide

FILES

~/.ttyswrc
/usr/bin/suntools
/usr/lib/rootmenu
/usr/lib/fonts/fixedwidthfonts/*
$iraf/local/sun/gterm.c

BUGS

(1) *gterm* is a complex program operating in an extremely dynamic environment. The program has been thoroughly tested and is quite reliable, but it is nonetheless possible for the program to get

into peculiar states where it does not behave as expected. Should this happen, a *setup reset* should restore the terminal to a known state.

(2) If more than 256 characters are input to a terminal emulator subwindow without an intervening newline, the terminal emulator may hang (to demonstrate this, hold any key down until the autorepeat generates sufficient characters). If this occurs, display the tty subwindow menu and select the **flush input** item to correct the problem.

(3) When using a terminal emulator to execute a program on a remote node via a network interface (rlogin, telnet, etc.), and the remote program continuously outputs a large amount of data, the terminal will occasionally hang up for several seconds, after which normal output will resume. Typing any character will cause output to resume immediately, but the character will later be delivered to the remote program as normal input hence should be selected with care (**<ctrl/q>** is always harmless). The origin of this bug is not clear, but since all terminal emulators are equally affected, it must be something in the terminal driver, or elsewhere in the SunOS kernel.

(4) The hardcopy functions assume a 1 or 8 bit frame buffer and will not work properly on a Sun-3/110, 3/60, etc., unless the **–8bit_color_only** option is specified to *suntools* [*fixed in Gterm V1.2]*.

(5) Reverse video does not work on a monochrome workstation as there is no colortable and no way to exchange the foreground and background colortable entries; try the **–i** option to *suntools* instead.

(6) *gterm* modifies the keyboard translation table entries for the arrow keys while a cursor read is in progress, restoring the translation table entries when done (this is necessary to allow the arrow keys to be used to terminate cursor reads). If something should happen to *gterm* while it is waiting for cursor input, it is possible that the arrow key translation table entries may not be restored. If this should happen, executing **setkeys reset** will fix things. Note also that changes to the keyboard translation tables are global, i.e., all windows are affected, hence while a cursor read is in progress in a gterm window, the arrow keys may not be usable with a program running in a different window.

(7) When plotting with clipping disabled there are cases where it is possible for *gterm* to coredump with a segmentation violation, killing any interactive subprocesses running within the terminal.

AUTHOR

Doug Tody, National Optical Astronomy Observatories (NOAO), IRAF project.