event

options:	**keyboard**	select the keyboard event queue
	buttons	select the buttons/switches event queue
	pointer	select the pointer event queue
	break	select the break event queue (only with ***open*** and ***close***)
	open/close	open or close the selected queue
	count	return the number of entries in the selected queue(s)
	read	return the first entry from the selected queue(s)
	flush	empty the selected queue
	status	obtain status information about the selected queue(s)
	verify	(with ***status***) print the queue status information

The **event** command provides command level access to the internal Semper event queues. This command allows Semper programs to open, close, check, read and flush the keyboard, pointer and button queues and to open and close the break/abandon queue.

Examples

```
event open pointer
```

This command opens the pointer queue.

```
event count keyboard
```

This command returns the number of outstanding key presses in the variable *nk*.

```
event read flush keyboard
```

This command reads the first key press into the variable *n* and flushes the rest.

event

Description

The **event** command provides access to most of the facilities available in the Semper event queue mechanism. There are four internal event queues:

- keyboard — records key presses from the keyboard
- buttons — records mouse button presses
- pointer — records the movements of the mouse
- break — records any abandon or break requests

Use the **keyboard**, **buttons**, **pointer** and **break** options to select these queues. Note that operations on the break queue are limited to using the **open** and **close** options to switch on and off respectively the event handling on the queue. For example, an **event break close** command can be used to protect sensitive program sections from abandon requests.

If you do not specify a queue, by default all queues are selected, but if you use the **open**, **read** and **close** options you need to specify one or more queues explicitly.

The options **open** and **close** are used to switch queues on and off.

The **count** option returns the number of entries in the specified queue(s). The value is returned in the variables *nk* (keyboard queue), *np* (pointer queue) and *nb* (button queue).

The **read** option is used to extract an entry from the head of the specified queue(s). The values returned depend on the queue:

- the **keyboard** queue returns the key code in the variable *n*. If the queue is empty the value returned is –1. Refer to *Appendix J, ASCII Key Codes* for a translation of the key codes.

- The **pointer** queue returns the incremental *x* and *y* changes in *dx* and *dy*. If the queue is empty the values returned are 0,0.

- The **buttons** queue returns the number of the button closed (or zero if it is a button open event) in the variable *nbc*, the number of the button opened (or zero if it is a button close event) in the variable *nbo* and the bit packed button state in the variable *nbb*. If the queue is empty *nbc* and *nbo* are both returned as zero.

The **flush** option removes all the entries from the specified queue.

The **status** option examines the state of the specified queues, returning the value 0 if the queue is closed, 1 otherwise. The values are returned in the variables *nkq* (keyboard queue), *npq* (pointer queue) and *nbq* (button queue). In addition, if you specify the **verify** option with the **status** option, the current queue length and maximum length are written to the console output.

event

An example program, using the **event** command, is given below. In this program, Semper commands are used to invert the current look-up table every half second until the user presses a mouse button or a key on the keyboard.

```
! Read the current status of the queues
event status keyboard buttons

! Open those queues that are required
unless nkq event open keyboard; unless nbq event open buttons

! Loop with test
m: lut.invert; wait 0.5; event count keyboard buttons;if (nb+nk)=0 jump m

! Flush the relevant queue
if nk>0 event flush keyboard; if nb>0 event flush buttons

! Reset the queue states
unless nkq event close keyboard; unless nbq event close buttons
```

Note that opening and closing queues should be done with care, as it is possible to hang a session if all input queues are closed. To be safe, the above script should either use **trap** statements to catch abandon requests, or should be bracketed with **event break close...event break open**.

Notes

variables set:

n (keycode with ***event read keyboard***)
nbo, nbc, nbb (button status with ***event read buttons***)
dx, dy (pointer moves with ***event read pointer***)
nkq, nbq, npq (queue open/close state with ***event status...***)
nk, nb, np (current queue length with ***event count...***)