

**NAME**

signal – catch or ignore signals

**SYNOPSIS**

(signal = 48.)

**sys signal; sig; value**

**signal(sig, func)**

**int (\*func)();**

**DESCRIPTION**

When the signal defined by *sig* is sent to the current process, it is to be treated according to *value*. The following is the list of signals:

- 1 hangup
- 2 interrupt
- 3\* quit
- 4\* illegal instruction
- 5\* trace trap
- 6\* IOT instruction
- 7\* EMT instruction
- 8\* floating point exception
- 9 kill (cannot be caught or ignored)
- 10\* bus error
- 11\* segmentation violation
- 12\* bad argument to sys call

If *value* is 0, the default system action applies to the signal. This is processes termination with or without a core dump. If *value* is odd, the signal is ignored. Any other even *value* specifies an address in the process where an interrupt is simulated. An RTI instruction will return from the interrupt. As a signal is caught, it is reset to 0. Thus if it is desired to catch every such signal, the catching routine must issue another *signal* call.

The starred signals in the list above cause core images if not caught and not ignored. In C, if *func* is 0 or 1, the action is as described above. If *func* is even, it is assumed to be the address of a function entry point. When the signal occurs, the function will be called. A return from the function will simulate the RTI.

After a *fork*, the child inherits all signals. The *exec* call resets all caught signals to default action.

**SEE ALSO**

kill (I, II)

**DIAGNOSTICS**

The error bit (c-bit) is set if the given signal is out of range. In C, a -1 indicates an error; 0 indicates success.