Module

Signal

Many operating systems allow signals to be sent to running processes. Some signals have a defined effect on the process, and others may be trapped at the code level and acted upon. For example, your process may trap the USR1 signal and use it to toggle debugging, and it may use TERM to initiate a controlled shutdown.

```
pid = fork do
  Signal.trap("USR1") do
    $debug = !$debug
    puts "Debug now: #$debug"
  end
  Signal.trap(:TERM) do   # symbols work too...
    puts "Terminating..."
    exit
  end
  # . . . do some work . . .
end
Process.detach(pid)
# Controlling program:
Process.kill("USR1", pid)
# ...
Process.kill(:USR1, pid)
# ...
Process.kill("TERM", pid)
```

produces:

```
Debug now: true
Debug now: false
Terminating...
```

The list of available signal names and their interpretation is system dependent. Signal delivery semantics may also vary between systems; in particular, signal delivery may not always be reliable.

Module methods

list

Signal.list → *hash*

Returns a list of signal names mapped to the corresponding underlying signal numbers.

```
Signal.list   # =>   {"ABRT"=>6, "ALRM"=>14, "BUS"=>10, "CHLD"=>20,
                     "CLD"=>20, "CONT"=>19, "EMT"=>7, "EXIT"=>0, "FPE"=>8,
                     "HUP"=>1, "ILL"=>4, "INFO"=>29, "INT"=>2, "IO"=>23,
                     "IOT"=>6, "KILL"=>9, "PIPE"=>13, "PROF"=>27,
                     "QUIT"=>3, "SEGV"=>11, "STOP"=>17, "SYS"=>12,
                     "TERM"=>15, "TRAP"=>5, "TSTP"=>18, "TTIN"=>21,
                     "TTOU"=>22, "URG"=>16, "USR1"=>30, "USR2"=>31,
                     "VTALRM"=>26, "WINCH"=>28, "XCPU"=>24, "XFSZ"=>25}
```

trap Signal.trap(*signal*, *command*) → *obj*
Signal.trap(*signal*) { *block* } → *obj*

1.9

Specifies the handling of signals. The first parameter is a signal name (a string or symbol such as SIGALRM, SIGUSR1, and so on) or a signal number. The characters SIG may be omitted from the signal name. The command or block specifies code to be run when the signal is raised. If the command is nil, the string IGNORE or SIG_IGN, or the empty string, the signal will be ignored. If the command is DEFAULT or SIG_DFL, the operating system's default handler will be invoked. If the command is EXIT, the script will be terminated by the signal. Otherwise, the given command or block will be run.

The special signal name EXIT or signal number zero will be invoked just prior to program termination.

trap returns the previous handler for the given signal.

```
Signal.trap(0, lambda { |signo| puts "exit pid #{$$} with #{signo}" })
Signal.trap("CLD")  { |signo| puts "Child died (#{signo})" }
if fork   # parent
  do_something # ...
else
  puts "In child, PID=#{$$}"
end
```

produces:

```
In child, PID=85867
exit pid 85867 with 0
Child died (20)
exit pid 85866 with 0
```

Note that you must specify a block taking a parameter if you use lambda to create the proc object.