:mod:`termios` --- POSIX style tty control

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]termios.rst`, **line 1**); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]termios.rst`, **line 4**)

Unknown directive type "module".

```
.. module:: termios
   :platform: Unix
   :synopsis: POSIX style tty control.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]termios.rst`, **line 8**)

Unknown directive type "index".

```
.. index::
   pair: POSIX; I/O control
   pair: tty; I/O control
```

This module provides an interface to the POSIX calls for tty I/O control. For a complete description of these calls, see :manpage:`termios(3)` Unix manual page. It is only available for those Unix versions that support POSIX *termios* style tty I/O control configured during installation.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]termios.rst`, **line 14**); *backlink*

Unknown interpreted text role "manpage".

All functions in this module take a file descriptor *fd* as their first argument. This can be an integer file descriptor, such as returned by `sys.stdin.fileno()`, or a :term:`file object`, such as `sys.stdin` itself.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]termios.rst`, **line 19**); *backlink*

Unknown interpreted text role "term".

This module also defines all the constants needed to work with the functions provided here; these have the same name as their counterparts in C. Please refer to your system documentation for more information on using these terminal control interfaces.

The module defines the following functions:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]termios.rst`, **line 31**)

Unknown directive type "function".

```
.. function:: tcgetattr(fd)

   Return a list containing the tty attributes for file descriptor *fd*, as
   follows: ``[iflag, oflag, cflag, lflag, ispeed, ospeed, cc]`` where *cc* is a
   list of the tty special characters (each a string of length 1, except the
   items with indices :const:`VMIN` and :const:`VTIME`, which are integers when
   these fields are defined).  The interpretation of the flags and the speeds as
   well as the indexing in the *cc* array must be done using the symbolic
   constants defined in the :mod:`termios` module.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]termios.rst`, **line 42**)

Unknown directive type "function".

```
.. function:: tcsetattr(fd, when, attributes)

   Set the tty attributes for file descriptor *fd* from the *attributes*, which is
   a list like the one returned by :func:`tcgetattr`.  The *when* argument
   determines when the attributes are changed: :const:`TCSANOW` to change
   immediately, :const:`TCSADRAIN` to change after transmitting all queued output,
   or :const:`TCSAFLUSH` to change after transmitting all queued output and
   discarding all queued input.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-`**
`main\Doc\library\[cpython-main][Doc][library]termios.rst,` line 52)**

Unknown directive type "function".

```
.. function:: tcsendbreak(fd, duration)

   Send a break on file descriptor *fd*.  A zero *duration* sends a break for
   0.25--0.5 seconds; a nonzero *duration* has a system dependent meaning.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-`**
`main\Doc\library\[cpython-main][Doc][library]termios.rst,` line 58)**

Unknown directive type "function".

```
.. function:: tcdrain(fd)

   Wait until all output written to file descriptor *fd* has been transmitted.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-`**
`main\Doc\library\[cpython-main][Doc][library]termios.rst,` line 63)**

Unknown directive type "function".

```
.. function:: tcflush(fd, queue)

   Discard queued data on file descriptor *fd*.  The *queue* selector specifies
   which queue: :const:`TCIFLUSH` for the input queue, :const:`TCOFLUSH` for the
   output queue, or :const:`TCIOFLUSH` for both queues.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-`**
`main\Doc\library\[cpython-main][Doc][library]termios.rst,` line 70)**

Unknown directive type "function".

```
.. function:: tcflow(fd, action)

   Suspend or resume input or output on file descriptor *fd*.  The *action*
   argument can be :const:`TCOOFF` to suspend output, :const:`TCOON` to restart
   output, :const:`TCIOFF` to suspend input, or :const:`TCION` to restart input.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-`**
`main\Doc\library\[cpython-main][Doc][library]termios.rst,` line 77)**

Unknown directive type "function".

```
.. function:: tcgetwinsize(fd)

   Return a tuple ``(ws_row, ws_col)`` containing the tty window size for file
   descriptor *fd*. Requires :const:`termios.TIOCGWINSZ` or
   :const:`termios.TIOCGSIZE`.

   .. versionadded:: 3.11
```

```
.. function:: tcsetwinsize(fd, winsize)

   Set the tty window size for file descriptor *fd* from *winsize*, which is
   a two-item tuple ``(ws_row, ws_col)`` like the one returned by
   :func:`tcgetwinsize`. Requires at least one of the pairs
   (:const:`termios.TIOCGWINSZ`, :const:`termios.TIOCSWINSZ`);
   (:const:`termios.TIOCGSIZE`, :const:`termios.TIOCSSIZE`) to be defined.

   .. versionadded:: 3.11
```

```
.. seealso::

   Module :mod:`tty`
      Convenience functions for common terminal control operations.
```

Example

Here's a function that prompts for a password with echoing turned off. Note the technique using a separate :func:`tcgetattr` call and a
:keyword:`try` ... :keyword:`finally` statement to ensure that the old tty attributes are restored exactly no matter what happens:

```
def getpass(prompt="Password: "):
    import termios, sys
    fd = sys.stdin.fileno()
    old = termios.tcgetattr(fd)
    new = termios.tcgetattr(fd)
    new[3] = new[3] & ~termios.ECHO          # lflags
    try:
        termios.tcsetattr(fd, termios.TCSADRAIN, new)
        passwd = input(prompt)
    finally:
        termios.tcsetattr(fd, termios.TCSADRAIN, old)
    return passwd
```