

NAME

`pcap_setnonblock`, `pcap_getnonblock` - set or get the state of non-blocking mode on a capture device

SYNOPSIS

```
#include <pcap/pcap.h>
```

```
char errbuf[PCAP_ERRBUF_SIZE];
```

```
int pcap_setnonblock(pcap_t *p, int nonblock, char *errbuf);
```

```
int pcap_getnonblock(pcap_t *p, char *errbuf);
```

DESCRIPTION

`pcap_setnonblock()` puts a capture handle into “non-blocking” mode, or takes it out of “non-blocking” mode, depending on whether the *nonblock* argument is non-zero or zero. It has no effect on “savefiles”. If there is an error, **PCAP_ERROR** is returned and *errbuf* is filled in with an appropriate error message; otherwise, **0** is returned.

In “non-blocking” mode, an attempt to read from the capture descriptor with `pcap_dispatch(3)` and `pcap_next_ex(3)` will, if no packets are currently available to be read, return **0** immediately rather than blocking waiting for packets to arrive.

`pcap_loop(3)` will loop forever, consuming CPU time when no packets are currently available; `pcap_dispatch()` should be used instead. `pcap_next(3)` will return **NULL** if there are no packets currently available to read; this is indistinguishable from an error, so `pcap_next_ex()` should be used instead.

When first activated with `pcap_activate(3)` or opened with `pcap_open_live(3)`, a capture handle is not in “non-blocking mode”; a call to `pcap_setnonblock()` is required in order to put it into “non-blocking” mode.

RETURN VALUE

`pcap_getnonblock()` returns the current “non-blocking” state of the capture descriptor; it always returns **0** on “savefiles”. If called on a capture handle that has been created but not activated, **PCAP_ERROR_NOT_ACTIVATED** is returned. If there is another error, **PCAP_ERROR** is returned and *errbuf* is filled in with an appropriate error message.

errbuf is assumed to be able to hold at least **PCAP_ERRBUF_SIZE** chars.

SEE ALSO

`pcap(3)`, `pcap_next_ex(3)`, `pcap_geterr(3)`