NAME

pcap_breakloop − force a pcap_dispatch() or pcap_loop() call to return

SYNOPSIS

#include <pcap/pcap.h>

void pcap_breakloop(pcap_t *);

DESCRIPTION

pcap_breakloop() sets a flag that will force **pcap_dispatch**(3PCAP) or **pcap_loop**(3PCAP) to return rather than looping; they will return the number of packets that have been processed so far, or **PCAP_ERROR_BREAK** if no packets have been processed so far.

This routine is safe to use inside a signal handler on UNIX or a console control handler on Windows, as it merely sets a flag that is checked within the loop.

The flag is checked in loops reading packets from the OS - a signal by itself will not necessarily terminate those loops - as well as in loops processing a set of packets returned by the OS. **Note that if you are catching signals on UNIX systems that support restarting system calls after a signal, and calling pcap_breakloop() in the signal handler, you must specify, when catching those signals, that system calls should NOT be restarted by that signal. Otherwise, if the signal interrupted a call reading packets in a live capture, when your signal handler returns after calling pcap_breakloop(), the call will be restarted, and the loop will not terminate until more packets arrive and the call completes.**

Note also that, in a multi-threaded application, if one thread is blocked in pcap_dispatch(), pcap_loop(), pcap_next(3PCAP), or pcap_next_ex(3PCAP), a call to pcap_breakloop() in a different thread will not unblock that thread. You will need to use whatever mechanism the OS provides for breaking a thread out of blocking calls in order to unblock the thread, such as thread cancellation or thread signalling in systems that support POSIX threads, or **SetEvent**() on the result of **pcap_getevent**() on a **pcap_t** on which the thread is blocked on Windows. Asynchronous procedure calls will not work on Windows, as a thread blocked on a **pcap_t** will not be in an alertable state.

Note that **pcap_next**() and **pcap_next_ex**() will, on some platforms, loop reading packets from the OS; that loop will not necessarily be terminated by a signal, so **pcap_breakloop**() should be used to terminate packet processing even if **pcap_next**() or **pcap_next_ex**() is being used.

pcap_breakloop() does not guarantee that no further packets will be processed by **pcap_dispatch**() or **pcap_loop**() after it is called; at most one more packet might be processed.

If **PCAP_ERROR_BREAK** is returned from **pcap_dispatch**() or **pcap_loop**(), the flag is cleared, so a subsequent call will resume reading packets. If a positive number is returned, the flag is not cleared, so a subsequent call will return **PCAP_ERROR_BREAK** and clear the flag.

SEE ALSO

pcap(3PCAP)