

**NAME**

`pcap_findalldevs`, `pcap_freealldevs` - get a list of capture devices, and free that list

**SYNOPSIS**

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

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

```
int pcap_findalldevs(pcap_if_t **alldevsp, char *errbuf);
```

```
void pcap_freealldevs(pcap_if_t *alldevs);
```

**DESCRIPTION**

`pcap_findalldevs()` constructs a list of network devices that can be opened with `pcap_create(3)` and `pcap_activate(3)` or with `pcap_open_live(3)`. (Note that there may be network devices that cannot be opened by the process calling `pcap_findalldevs()`, because, for example, that process does not have sufficient privileges to open them for capturing; if so, those devices will not appear on the list.) If `pcap_findalldevs()` succeeds, the pointer pointed to by *alldevsp* is set to point to the first element of the list, or to `NULL` if no devices were found (this is considered success). Each element of the list is of type `pcap_if_t`, and has the following members:

**next**

if not `NULL`, a pointer to the next element in the list; `NULL` for the last element of the list

**name**

a pointer to a string giving a name for the device to pass to `pcap_open_live()`

**description**

if not `NULL`, a pointer to a string giving a human-readable description of the device

**addresses**

a pointer to the first element of a list of network addresses for the device, or `NULL` if the device has no addresses

**flags**

device flags:

**PCAP\_IF\_LOOPBACK**

set if the device is a loopback interface

**PCAP\_IF\_UP**

set if the device is up

**PCAP\_IF\_RUNNING**

set if the device is running

**PCAP\_IF\_WIRELESS**

set if the device is a wireless interface; this includes IrDA as well as radio-based networks such as IEEE 802.15.4 and IEEE 802.11, so it doesn't just mean Wi-Fi

**PCAP\_IF\_CONNECTION\_STATUS**

a bitmask for an indication of whether the adapter is connected or not; for wireless interfaces, "connected" means "associated with a network"

The possible values for the connection status bits are:

**PCAP\_IF\_CONNECTION\_STATUS\_UNKNOWN**

it's unknown whether the adapter is connected or not

**PCAP\_IF\_CONNECTION\_STATUS\_CONNECTED**

the adapter is connected

**PCAP\_IF\_CONNECTION\_STATUS\_DISCONNECTED**

the adapter is disconnected

**PCAP\_IF\_CONNECTION\_STATUS\_NOT\_APPLICABLE**

the notion of "connected" and "disconnected" don't apply to this interface; for example, it doesn't apply to a loopback device

Each element of the list of addresses is of type **pcap\_addr\_t**, and has the following members:

**next**

if not **NULL**, a pointer to the next element in the list; **NULL** for the last element of the list

**addr**

a pointer to a **struct sockaddr** containing an address

**netmask**

if not **NULL**, a pointer to a **struct sockaddr** that contains the netmask corresponding to the address pointed to by **addr**

**broadaddr**

if not **NULL**, a pointer to a **struct sockaddr** that contains the broadcast address corresponding to the address pointed to by **addr**; may be null if the device doesn't support broadcasts

**dstaddr**

if not **NULL**, a pointer to a **struct sockaddr** that contains the destination address corresponding to the address pointed to by **addr**; may be null if the device isn't a point-to-point interface

Note that the addresses in the list of addresses might be IPv4 addresses, IPv6 addresses, or some other type of addresses, so you must check the **sa\_family** member of the **struct sockaddr** before interpreting the contents of the address; do not assume that the addresses are all IPv4 addresses, or even all IPv4 or IPv6 addresses. IPv4 addresses have the value **AF\_INET**, IPv6 addresses have the value **AF\_INET6** (which older operating systems that don't support IPv6 might not define), and other addresses have other values. Whether other addresses are returned, and what types they might have is platform-dependent. For IPv4 addresses, the **struct sockaddr** pointer can be interpreted as if it pointed to a **struct sockaddr\_in**; for IPv6 addresses, it can be interpreted as if it pointed to a **struct sockaddr\_in6**.

The list of devices must be freed with **pcap\_freealldevs(3)**, which frees the list pointed to by *alldevs*.

**RETURN VALUE**

**pcap\_findalldevs()** returns **0** on success and **PCAP\_ERROR** on failure; as indicated, finding no devices is considered success, rather than failure, so **0** will be returned in that case. If **PCAP\_ERROR** is returned, *errbuf* is filled in with an appropriate error message. *errbuf* is assumed to be able to hold at least **PCAP\_ERRBUF\_SIZE** chars.

**BACKWARD COMPATIBILITY**

The **PCAP\_IF\_UP** and **PCAP\_IF\_RUNNING** constants became available in libpcap release 1.6.1. The **PCAP\_IF\_WIRELESS**, **PCAP\_IF\_CONNECTION\_STATUS**, **PCAP\_IF\_CONNECTION\_STATUS\_UNKNOWN**, **PCAP\_IF\_CONNECTION\_STATUS\_CONNECTED**, **PCAP\_IF\_CONNECTION\_STATUS\_DISCONNECTED**, and **PCAP\_IF\_CONNECTION\_STATUS\_NOT\_APPLICABLE** constants became available in libpcap release 1.9.0.

**SEE ALSO**

**pcap(3)**