五、libpcap接口表

时间：2015/8/13

char \*pcap\_lookupdev(char \*);

：[支持]

int pcap\_lookupnet(const char \*, bpf\_u\_int32 \*, bpf\_u\_int32 \*, char \*);

：不支持，设备板卡没有IP地址；

pcap\_t \*pcap\_create(const char \*, char \*);

：不支持，不支持pcap\_create && pcap\_active 只能够使用Open\_live接口获得句柄；

int pcap\_set\_snaplen(pcap\_t \*, int);

：支持，但是板卡在Open\_live的时候就已经处于actived，所以使用set\_snaplen之前将会检查板卡是否actived，也就是说对于板卡该句柄没有实际意义；

int pcap\_set\_promisc(pcap\_t \*, int);

：板卡一直处于混杂模式；

int pcap\_can\_set\_rfmon(pcap\_t \*);

：支持该接口，但是板卡一开始已经处于actived模式，没有实际意义；

int pcap\_set\_rfmon(pcap\_t \*, int);

：支持该接口，但是板卡一开始已经处于actived模式；

int pcap\_set\_timeout(pcap\_t \*, int);

：不支持，板卡默认设置了该值，该值被无视；

int pcap\_set\_tstamp\_type(pcap\_t \*, int);

：不支持，默认由主机提供高精度时间戳；

int pcap\_set\_immediate\_mode(pcap\_t \*, int);

：不支持，但是板卡一开始已经处于actived模式；

int pcap\_set\_buffer\_size(pcap\_t \*, int);

：不支持，缓存区是有驱动申请的，后期没有办法改变；

int pcap\_set\_tstamp\_precision(pcap\_t \*, int);

：不支持；默认精度为us秒级别；

int pcap\_get\_tstamp\_precision(pcap\_t \*);

：支持；

int pcap\_activate(pcap\_t \*);

：不支持，默认open\_live之后设置actived；

int pcap\_list\_tstamp\_types(pcap\_t \*, int \*\*);

：支持，板卡时间默认使用host us时间级；

void pcap\_free\_tstamp\_types(int \*);

：支持

int pcap\_tstamp\_type\_name\_to\_val(const char \*);

：支持

const char \*pcap\_tstamp\_type\_val\_to\_name(int);

：支持

const char \*pcap\_tstamp\_type\_val\_to\_description(int);

：支持

pcap\_t \*pcap\_open\_live(const char \*, int, int, int, char \*);

：支持，主要功能函数，用来获取接口句柄

pcap\_t \*pcap\_open\_dead(int, int);

：支持，通用功能函数

pcap\_t \*pcap\_open\_dead\_with\_tstamp\_precision(int, int, u\_int);

：支持，通用功能函数

pcap\_t \*pcap\_open\_offline\_with\_tstamp\_precision(const char \*, u\_int, char \*);

：支持，通用功能函数

pcap\_t \*pcap\_open\_offline(const char \*, char \*);

：支持，通用功能函数

void pcap\_close(pcap\_t \*);

：支持，主要功能函数，主要是调用platform\_cleanup

int pcap\_loop(pcap\_t \*, int, pcap\_handler, u\_char \*);

:支持，主要功能函数，主要调用qnf\_read

int pcap\_dispatch(pcap\_t \*, int, pcap\_handler, u\_char \*);

：支持，主要功能函数，调用底层qnf\_read接口

const u\_char\* pcap\_next(pcap\_t \*, struct pcap\_pkthdr \*);

：支持，调用底层函数接口qnf\_read；

int pcap\_next\_ex(pcap\_t \*, struct pcap\_pkthdr \*\*, const u\_char \*\*);

：支持

void pcap\_breakloop(pcap\_t \*);

：支持，设置break位；

int pcap\_stats(pcap\_t \*, struct pcap\_stat \*);

：支持，对照wireshark

int pcap\_setfilter(pcap\_t \*, struct bpf\_program \*);

：支持，软件bpf过滤；

int pcap\_setdirection(pcap\_t \*, pcap\_direction\_t);

：不支持，只能够支持进来的数据包抓包；如果需要设置强制兼容，需要重新考虑

int pcap\_getnonblock(pcap\_t \*, char \*);

：不支持，高速率性能考虑，默认设置50ms阻塞机制；

int pcap\_setnonblock(pcap\_t \*, int, char \*);

：不支持

int pcap\_inject(pcap\_t \*, const void \*, size\_t);

：支持，主要功能函数，发送raw数据包；

int pcap\_sendpacket(pcap\_t \*, const u\_char \*, int);

：支持，依赖于inject函数；

const char \*pcap\_statustostr(int);

：支持，通用调试接口

const char \*pcap\_strerror(int);

：支持，通用调试接口

char \*pcap\_geterr(pcap\_t \*);

：支持，通用调试接口，p->errbuf

void pcap\_perror(pcap\_t \*, char \*);

：支持，通用调试接口

int pcap\_compile(pcap\_t \*, struct bpf\_program \*, const char \*, int,

bpf\_u\_int32);

：支持，软件过滤

int pcap\_compile\_nopcap(int, int, struct bpf\_program \*,

const char \*, int, bpf\_u\_int32);

：支持，通用接口；

void pcap\_freecode(struct bpf\_program \*);

：支持，释放过滤规则内存空间；

int pcap\_offline\_filter(const struct bpf\_program \*,

const struct pcap\_pkthdr \*, const u\_char \*);

：支持，通用过滤接口

int pcap\_datalink(pcap\_t \*);

：支持，板卡默认连接链路为DLT\_EN10MB；

int pcap\_datalink\_ext(pcap\_t \*);

：支持；

int pcap\_list\_datalinks(pcap\_t \*, int \*\*);

：支持；

int pcap\_set\_datalink(pcap\_t \*, int);

：不支持，固定连接类型DLT\_EN10MB；

void pcap\_free\_datalinks(int \*);

：支持；

int pcap\_datalink\_name\_to\_val(const char \*);

：支持；

const char \*pcap\_datalink\_val\_to\_name(int);

：支持

const char \*pcap\_datalink\_val\_to\_description(int);

：支持

int pcap\_snapshot(pcap\_t \*);

：支持，返回snaplen

int pcap\_is\_swapped(pcap\_t \*);

：不支持；

int pcap\_major\_version(pcap\_t \*);

：支持，通用

int pcap\_minor\_version(pcap\_t \*);

：支持，通用

FILE \*pcap\_file(pcap\_t \*);

：支持，通用接口；需要测试

int pcap\_fileno(pcap\_t \*);

：支持；

pcap\_dumper\_t \*pcap\_dump\_open(pcap\_t \*, const char \*);

：支持；

pcap\_dumper\_t \*pcap\_dump\_fopen(pcap\_t \*, FILE \*fp);

：支持；

FILE \*pcap\_dump\_file(pcap\_dumper\_t \*);

：支持；

long pcap\_dump\_ftell(pcap\_dumper\_t \*);

：支持；

int pcap\_dump\_flush(pcap\_dumper\_t \*);

：支持；

void pcap\_dump\_close(pcap\_dumper\_t \*);

：支持

void pcap\_dump(u\_char \*, const struct pcap\_pkthdr \*, const u\_char \*);

：支持

int pcap\_findalldevs(pcap\_if\_t \*\*, char \*);

：支持；

void pcap\_freealldevs(pcap\_if\_t \*);

：支持

const char \*pcap\_lib\_version(void);

：支持