Libpgen Manual version 0.01

slank

2015 年 8 月 3 日

1　はじめに

libpgen は Slank Develop Network (https://slankdev.net) で情報公開をしている C/C++ のパケットジェネレータのライブラリです。このライブラリはネットワークプログラミング、ネットワークの勉強などに最適だと開発者は考えています。

libpgen はまだ完成段階にはありません。様々なエラーを含んでいるかもしれません。ですが少しでも多くの人に使用してもらい、開発の強力をしていただけたら、と考えています。

開発者のメールアドレスは slank.dev@gmail.com で、twitter アカウントは@slankdev です。
また libpgen のソースコードは GitHub で公開をしています。(https://github.com/slankdev/libpgen)

エラー報告などがあれば開発者まで連絡をお願いします。

2 パケット生成の流れ

2.1 パケットクラスのインスタンスを生成

libpgen ではパケットインスタンスを生成して、それを設定、送信という流れで、パケットを送信します。対応プロトコルは Ethernet、ARP、IP、ICMP、TCP、UDP、DNS、AR Drone で、パケットクラスの名前はそれぞれ pgen_eth、pgen_arp、pgen_ip、pgen_icmp、pgen_tcp、pgen_udp、pgen_dns、pgen_ardrone です。例えば TCP パケットのインスタンスを生成するときは以下の様になります。

```
pgen_tcp packet;
```

2.2 パケットを設定

パケットの設定は、以下で説明するパケットの要素に値を代入していくだけです。例えばパケットの送信先 IP アドレスを 192.168.110.4 にするときは以下の様になります。

```
packet.IP.dst = "192.168.110.4";
```

2.3 パケット送信、情報表示など

パケットの送信や情報表示は各パケットクラスのメンバ関数を使用します。パケットクラスは以下のようなメンバ関数を持ちます。

`void SEND(const char*);` **引数で指定したいんたーふぇーすでパケットを送信します。**
`void INFO();` **パケットの情報を詳細表示します。**
`void SUMMARY();` **パケットの情報を一行で簡単に表示します。**
`bool CAST(const char*, int);` **バイト列からパケットクラスにキャストします。パケット解析などに使用します。キャスト失敗時は** `false` **を返します。**

バイト列からパケットにキャストし、送信する場合以下の様になります。

```
if(packet.CAST(paket,len) == false){
    fprintf(stderr, "cast miss\n");
    return;
}
packet.SEND("wlan0");
```

3 IP アドレス、MAC アドレスの操作について

libpgen はアドレス操作を ipaddr クラスと macaddr クラスで行います。

3.1 ipaddr クラス

ipaddr クラスは IP アドレスを管理するクラスです。このクラスは以下のコンストラクタと、メンバ関数を持ちます。また ipaddr クラスは<, >,==,!=の比較演算子が使用できます。IP アドレスの代入は char 配列文字列、ipaddr クラスの代入に対応しています。

`ipaddr(ipaddr);` **引数と同じアドレスを入力します。**

`ipaddr(const char*);` **引数の文字列から入力します。**

`bool setipbydev(const char*);` **引数のデバイスの** `IP` **アドレスをセットします。失敗時は** `false` **を返します。**

`bool setmascbydev(const char*);` **引数のデバイスのネットマスクをセットします。失敗時は** `false` **を返します。**

`bool isEmpty();` **アドレスが空 (**`0.0.0.0`**) である時に** `true` **を返します。**

`char* c_str();` **アドレスの** `NULL` **ポインタ文字列を返します。**

3.2 macaddr クラス

macaddr クラスも ipaddr クラス同様に<, >,==,!=の比較演算子が使用できます。代入も同様に char 配列文字列、ipaddr クラスの代入に対応しています。

`macaddr(macaddr);` **引数と同じアドレスを入力します。**

`macaddr(const char*);` **引数の文字列から入力します。**

`bool setmacbydev(const char*);` **引数のデバイスの** `MAC` **アドレスをセットします。失敗時は** `false` **を返します。**

`bool isEmpty();` `ipaddr` **クラスと同様です。**

`char* c_str();` **アドレスの** `NULL` **ポインタ文字列を返します。**

`char* bender();` **アドレスの上位３バイトからベンダ名を** `NULL` **ポインタ文字列で返します。**

4 対応プロトコル

Ethernet、ARP、IP、ICMP、TCP、UDP、DNS、AR Dro ne に対応しています。
パケットのプロトコル要素のは以下の様にして設定します。

プロトコル. 要素名 = 123

各プロトコルの要素は以下の様に libpgen 内で定義されています。

```
Ethernet
src MAC address      : macaddr ETH.src
dst MAC address      : macaddr ETH.dst
ether type           : int ETH.type

ARP
src hardware address : macaddr ARP.srcEth
src protocol address : ipaddr ARP.srcIp
dst hardware address : macaddr ARP.dstEth
dst protocol address : ipaddr ARP.dstIp
arp operation        : int ARP.operation

IP
src IP address       : ipaddr IP.src
dst IP address       : ipaddr IP.dst
identification       : int IP.id
type of service      : int IP.tos
time to leave        : int IP.ttl
protocol             : int IP.protocol
```

```
ICMP
icmp option          : int ICMP.option
icmp code            : int ICMP.code
identification       : int ICMP.id
sequence number      : int ICMP.seq

TCP
src port             : int TCP.src
dst port             : int TCP.dst
flag  -FIN           : char TCP.flags.fin
flag  -SYN           : char TCP.flags.syn
flag  -RST           : char TCP.flags.rst
flag  -PSH           : char TCP.flags.psh
flag  -ACK           : char TCP.flags.ack
flag  -URG           : char TCP.flags.urg
window size          : int TCP.window
sequence number      : int TCP.seq
acknowledge number   : int TCP.ack

UDP
src port             : int UDP.src
dst port             : int UDP.dst
```

```
typedef u_int16_t bit16;
typedef u_int8_t  bit8;

DNS
Header
identification       : bit16 DNS.id
flag -qr             : bit8 DNS.flags.qr
flag -opcode         : bit8 DNS.flags.opcode
flag -aa             : bit8 DNS.flags.aa
flag -tc             : bit8 DNS.flags.tc
flag -rd             : bit8 DNS.flags.rd
flag -ra             : bit8 DNS.flags.ra
flag -nouse          : bit8 DNS.flags.nouse
flag -rcode          : bit8 DNS.flags.rcode
question count       : bit16 DNS.qdcnt
answer count         : bit16 DNS.ancnt
auth count           : bit16 DNS.nscnt
additional count     : bit16 DNS.arcnt
Query
name                 : std::string DNS.query.name;
type                 : bit16 DNS.query.type;
class                : bit16 DNS.query.cls;
Answer
name                 : bit16 DNS.answer.name
type                 : bit16 DNS.answer.type
class                : bit16 DNS.answer.cls
time to leave        : bit32 DNS.answer.ttl
length               : bit16 DNS.answer.len
address              : ipaddr DNS.answer.addr
```

```
AR Drone
PCMD
sequence number      : long ar_drone.pcmd.seq
flag                 : long ar_drone.pcmd.flag
roll                 : long ar_drone.pcmd.roll
pitch                : long ar_drone.pcmd.pitch
gazzer               : long ar_drone.pcmd.gaz
yaw -x               : long ar_drone.pcmd.yaw.x
yaw -y               : long ar_drone.pcmd.yaw.y
yaw -z               : long ar_drone.pcmd.yaw.z
REF
sequence number      : long ar_drone.ref.seq
command              : long ar_drone.ref.command
```