大家在学C、C++ 的时候，老师多半会讲过：C语言是面向过程的编程语言，C++是面向对象的编程语言。但归根结底，面向过程还是面向对象，这是编程思想的差异，而不是语言的差异。笔者最近在看baresip源码，C语言也能写成面向对象。

笔者，想了个应用场景，写了个简单的demo。笔者当年的第一份工作，维护BPLA（北洋打印机指令集）SDK，其中重要的一项工作是，通过串口、并口、usb、网络等软硬件接口，发送控制指令。

笔者先创建了一个interface.h文件，相当于C++里的虚类。

typedef void (\* p\_func\_connect) ();  
typedef void (\* p\_func\_send\_cmd) (char \*);  
typedef void (\* p\_func\_disconnect) ();  
  
struct interface  
{  
 char \*name;  
 p\_func\_connect connect;  
 p\_func\_disconnect disconnect;  
 p\_func\_send\_cmd send\_cmd;  
};

然后创建modules文件，并在此文件夹下，创建了usb.c、com.c、driver.c 、network.c，分别对应usb、串口、驱动接口、网络接口的实现。相当于C++中，继承虚类的各个子类实现。usb.c如下：

#include "../interface.h"  
  
static void connect() {  
 printf("connect by usb\n");  
}  
  
static void disconnect() {  
 printf("disconnect by usb\n");  
}  
  
static void send\_cmd(char \*cmd) {  
 printf("send cmd by usb: %s\n", cmd);  
}  
  
const struct interface usb = {  
 "usb",  
 connect,  
 disconnect,  
 send\_cmd  
};

为了让这些“类”用起来更优雅些，笔者在interface.h、interface.c中添加了find\_device函数，添加了static.c。

interface.c文件

#include "interface.h"  
  
extern const struct interface \*interfaces[];  
  
const struct interface \* find\_device(char \*name) {  
 int i = 0;  
 struct interface \*dev = interfaces[0];  
  
 for (; dev; i++) {  
 if (0 == strcmp(name, dev->name)) {  
 break;  
 }  
  
 dev = interfaces[i + 1];  
 }  
  
 return dev;  
}

static.c

#include <stdio.h>  
#include "interface.h"  
  
extern const struct interface com;  
extern const struct interface usb;  
extern const struct interface driver;  
extern const struct interface network;  
  
const struct interface \*interfaces[] = {  
 &com,  
 &usb,  
 &driver,  
 &network,  
 NULL };

static.c进阶版，Makefile文件动态生成：

MODULES:=$(wildcard ./modules/\*.c)  
MODULES:=$(patsubst ./modules/%.c, %, $(MODULES))  
  
static.c:  
 echo "#include <stdio.h>" > $@  
 echo "#include \"interface.h\"\n" >> $@  
 for n in $(MODULES); do \  
 echo "extern const struct interface $${n};" >> $@ ; \  
 done  
 echo "" >> $@  
 echo "const struct interface \*interfaces[] = {" >> $@  
 for n in $(MODULES); do \  
 echo " &$${n}," >> $@ ; \  
 done  
 echo " NULL };" >> $@

也可根据实际情况，动态设置使用哪些module，如，现在使用的打印机只有usb口和网口。

MODULES:=usb network

最后贴下main.c文件和调用

#include <stdio.h>  
#include <stdlib.h>  
#include "interface.h"  
  
int main() {  
 struct interface \*dev = NULL;  
 char name[255];  
  
 while (1) {  
 memset(name, 0, 255);  
 gets(name);  
  
 dev = find\_device(name);  
 if (dev) {  
 dev->connect();  
 dev->send\_cmd("print hello world");  
 dev->disconnect();  
 } else {  
 printf("no such device: %s!\n", name);  
 }  
 }  
  
 return 0;  
}

运行如下：

➜ communication ./main   
warning: this program uses gets(), which is unsafe.  
usb  
connect by usb  
send cmd by usb: print hello world  
disconnect by usb  
network  
connect by network  
send cmd by network: print hello world  
disconnect by network

附 Makefile文件：

# MODULES:=usb network  
  
MODULES:=$(wildcard ./modules/\*.c)  
MODULES:=$(patsubst ./modules/%.c, %, $(MODULES))  
  
# SRC:=$(wildcard \*.c ./interface/\*.c)  
SRC:=$(wildcard \*.c)  
SRC+=$(patsubst %, modules/%.c, $(MODULES))  
  
OBJ:=$(patsubst %.c, %.o, $(SRC))  
CC:=gcc -fPIC  
  
all: static.c $(OBJ)  
 $(CC) $(OBJ) -o main  
  
static.c:  
 echo "#include <stdio.h>" > $@  
 echo "#include \"interface.h\"\n" >> $@  
 for n in $(MODULES); do \  
 echo "extern const struct interface $${n};" >> $@ ; \  
 done  
 echo "" >> $@  
 echo "const struct interface \*interfaces[] = {" >> $@  
 for n in $(MODULES); do \  
 echo " &$${n}," >> $@ ; \  
 done  
 echo " NULL };" >> $@  
  
%o:%c  
 $(CC) $(CFLAGS) -c $< -o $@  
  
clean:  
 rm \*.o  
 rm ./modules/\*.o  
 rm main  
 rm static.c