Lab 2a报告

王颖 1900013016

Link-layer: Packet I/O on Ethernet

> 1. Use libpcap to implement following methods to support network device management.

- See `incl/device.h` and `scr/device.cpp`
- The following functions are implemented:

```
class Device {
  //private:
  public:
    char* name;
    int32_t id;
    unsigned char MAC[8];
    //ipv4_t IP;
    pcap_t* handle;

    Device();
    Device(const char* _name, const int32_t& _id, bool& flag);
    ~Device();
};

class DeviceList {
  public:
    DeviceList* nxt;
    Device* cur;
    DeviceList();
};

//Detect all devices on the host and return their names.
vector <string> detectAllDevice();

//use addDevice to add all available devices to the library
void addAlldevices();

/**
* Add a device to the library for sending/receiving packets.
*
* @param it information of network device to send/receive packet on.
* @return A non-negative _device-ID_ on success, -1 on error.
*/
int addDevice(const pcap_if_t* it);

/**
* Find a device added by `addDevice`.
*
* @param device Name of the network device.
* @return A non-negative _device-ID_ on success, -1 if no such device was found.
*/
int findDevice(const char* device);
```

```
//show the devices added to the library(name and id)
void showDevices();

Device* getDevice(int id);
Device* getDevicefromMAC(const char* mac);
int getDevNum();
DeviceList* getDevlist();
```

> 2. Use libpcap to implement the following methods to support sending/receiving Ethernet II frames.

- See `scr/outputmanagement.cpp` and `scr/packetio.cpp`
- The following functions are implemented:

  ```
  // file scr/packetio.cpp to avoid racing when printing.
  #ifndef _OUTPUTMANEGE_H_
  #define _OUTPUTMANEGE_H_

  void initOutputConsole();

  void senderOutputAsking();
  void senderOutputFinished();

  void receiverOutputAsking();
  void receiverOutputFinished();
  #endif
  ```

  ```
  // file scr/outputmanagement.cpp
  /**
  * @brief Encapsulate some data into an Ethernet II frame and send it.
  *
  * @param buf Pointer to the payload.
  * @param len Length of the payload.
  * @param ethtype EtherType field value of this frame.
  * @param destmac MAC address of the destination.
  * @param id ID of the device(returned by `addDevice`) to send on. @return 0
  on success, -1 on error. @see addDevice
  */
  int sendFrame(const void* buf, int len, int ethtype, const void* destmac,
  int id);

  /**
  * @brief Process a frame upon receiving it.
  * @param buf Pointer to the frame.
  * @param len Length of the frame.
  * @param id ID of the device (returned by `addDevice`) receiving current
  frame. @return 0 on success, -1 on error. @see addDevice
  */
  typedef int (*frameReceiveCallback)(const void*, int, int);
  //The concrete implementation of callback function
  int recvFrame(const void* buf, int len, int id);

  /**
  * @brief Register a callback function to be called each time an Ethernet II
  frame was received.
  ```

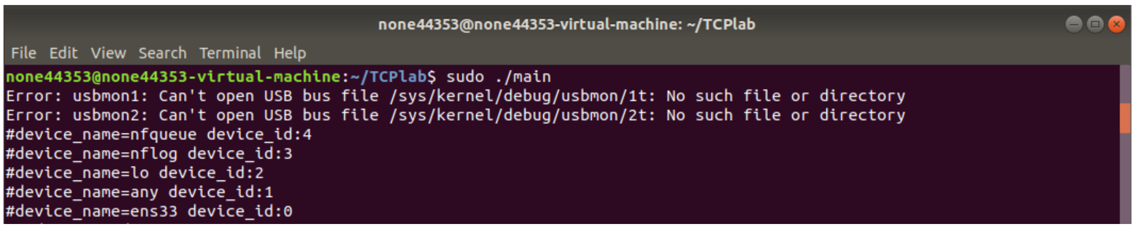
```
 * @param id the device id
 * @param callback the callback function.
 * @return 0 on success, -1 on error.
 * @see frameReceiveCallback
 */
int setFrameReceiveCallback(int id, frameReceiveCallback callback);

void *deviceThread(void *vargp); //deviceThread is a thread routine that
installs the callback function
//setFrameReceiveCallback will call it to install callback
```

> 3. Show that your implementation can detect network interfaces on the host.

- I set the following check code in `main.cpp` to show the network interfaces detected on the host.

```
//Check Point1 show opened device
    showDevices();
    fflush(stderr);
```

- the result is as follow:



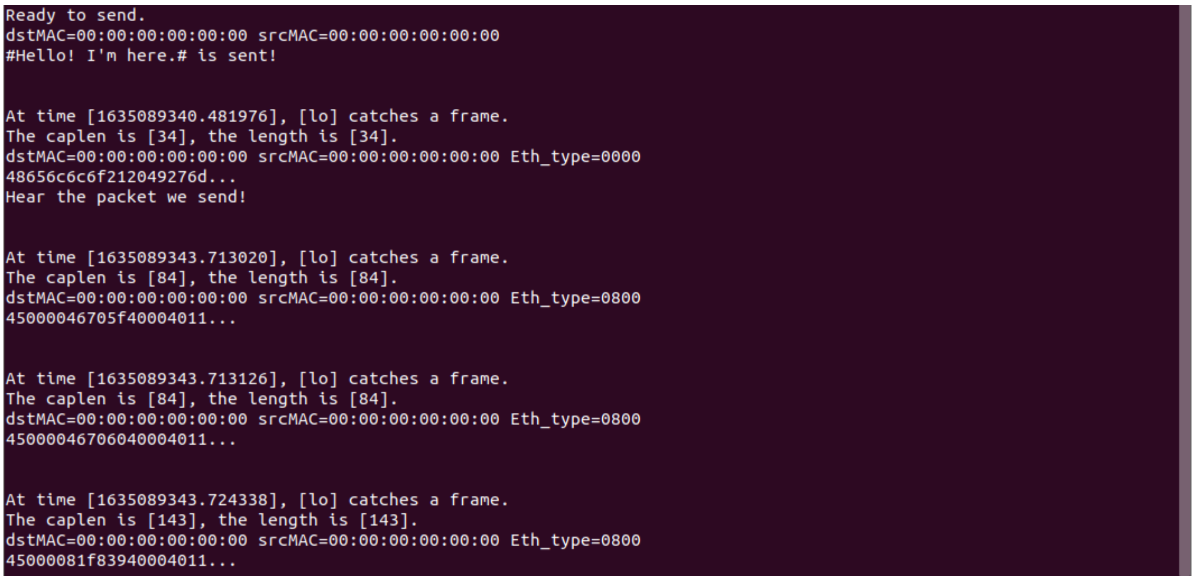
```
none44353@none44353-virtual-machine:~/TCPlab$ sudo ./main
Error: usbmon1: Can't open USB bus file /sys/kernel/debug/usbmon/1t: No such file or directory
Error: usbmon2: Can't open USB bus file /sys/kernel/debug/usbmon/2t: No such file or directory
#device_name=nfqueue device_id:4
#device_name=nflog device_id:3
#device_name=lo device_id:2
#device_name=any device_id:1
#device_name=ens33 device_id:0
```

none44353@none44353-virtual-machine: ~/TCPlab
File Edit View Search Terminal Help

  - Two errors are raised by addAlldevices() called by the main().

> 4. Show that your implementation can capture frames from a device and inject frames to a device using libpcap.

- the rest code in `main.cpp` setFrameReceiveCallback on 'lo' device and show us the packets received.
- I also send a string "Hello! I'm here." out.
- the result is as follow:



```
Ready to send.
dstMAC=00:00:00:00:00:00 srcMAC=00:00:00:00:00:00
#Hello! I'm here.# is sent!


At time [1635089340.481976], [lo] catches a frame.
The caplen is [34], the length is [34].
dstMAC=00:00:00:00:00:00 srcMAC=00:00:00:00:00:00 Eth_type=0000
48656c6c6f212049276d...
Hear the packet we send!


At time [1635089343.713020], [lo] catches a frame.
The caplen is [84], the length is [84].
dstMAC=00:00:00:00:00:00 srcMAC=00:00:00:00:00:00 Eth_type=0800
45000046705f40004011...


At time [1635089343.713126], [lo] catches a frame.
The caplen is [84], the length is [84].
dstMAC=00:00:00:00:00:00 srcMAC=00:00:00:00:00:00 Eth_type=0800
45000046706040004011...


At time [1635089343.724338], [lo] catches a frame.
The caplen is [143], the length is [143].
dstMAC=00:00:00:00:00:00 srcMAC=00:00:00:00:00:00 Eth_type=0800
45000081f83940004011...
```

This complete part A.