Programação Orientada a Objetos para Redes de Computadores

Prof. Miguel Elias Mitre Campista

http://www.gta.ufrj.br/~miguel

PARTE 2

Programação em C++ - Classes e Objetos

Programa da Captura de Pacotes em Redes

- A ideia é criar um programa para captura de pacotes
  - Requer uso da biblioteca **`libpcap`**
    - Mesma usada nos programas que fazem farejamento de redes ("*packet sniffers*")
      - Ex.: tcpdump, wireshark, etc.
    - Instalação via apt-get:
      - **`apt-get install libpcap-dev`**
    - Documentação disponível em:
      - **`http://www.tcpdump.org/pcap.html`**

Programa de Captura em Execução

```
root@itaqua:/lab/users/Miguel/miguel/disciplinas/progredes/libpcap# ./captura
Device: eth0
*** Press ctrl-c to finish ***
[2017-06-27 10:13:49.901723] Got packet of 86 bytes
[2017-06-27 10:13:49.901739] Got packet of 54 bytes
[2017-06-27 10:13:49.937981] Got packet of 278 bytes
[2017-06-27 10:13:49.938115] Got packet of 138 bytes
[2017-06-27 10:13:49.938127] Got packet of 66 bytes
[2017-06-27 10:13:49.938284] Got packet of 242 bytes
[2017-06-27 10:13:49.938385] Got packet of 138 bytes
[2017-06-27 10:13:49.938553] Got packet of 234 bytes
[2017-06-27 10:13:49.938775] Got packet of 138 bytes
[2017-06-27 10:13:49.938844] Got packet of 206 bytes
[2017-06-27 10:13:49.939000] Got packet of 258 bytes
[2017-06-27 10:13:49.939075] Got packet of 222 bytes
[2017-06-27 10:13:49.939279] Got packet of 8192 bytes
[2017-06-27 10:13:49.939299] Got packet of 66 bytes
[2017-06-27 10:13:49.939374] Got packet of 4410 bytes
[2017-06-27 10:13:49.939379] Got packet of 66 bytes
[2017-06-27 10:13:49.939613] Got packet of 2034 bytes
[2017-06-27 10:13:49.939620] Got packet of 66 bytes
[2017-06-27 10:13:49.939677] Got packet of 206 bytes
[2017-06-27 10:13:49.939892] Got packet of 258 bytes
[2017-06-27 10:13:49.939959] Got packet of 242 bytes
[2017-06-27 10:13:49.940095] Got packet of 138 bytes
[2017-06-27 10:13:49.940272] Got packet of 242 bytes
[2017-06-27 10:13:49.940432] Got packet of 138 bytes
[2017-06-27 10:13:49.940507] Got packet of 242 bytes
[2017-06-27 10:13:49.940624] Got packet of 138 bytes
[2017-06-27 10:13:49.940694] Got packet of 242 bytes
[2017-06-27 10:13:49.940832] Got packet of 138 bytes
[2017-06-27 10:13:49.940890] Got packet of 234 bytes
[2017-06-27 10:13:49.941016] Got packet of 138 bytes
[2017-06-27 10:13:49.941071] Got packet of 234 bytes
[2017-06-27 10:13:49.941183] Got packet of 138 bytes
[2017-06-27 10:13:49.941234] Got packet of 186 bytes
[2017-06-27 10:13:49.941367] Got packet of 114 bytes
[2017-06-27 10:13:49.980465] Got packet of 66 bytes
[2017-06-27 10:13:49.984109] Got packet of 214 bytes
[2017-06-27 10:13:50.94819] Got packet of 60 bytes
[2017-06-27 10:13:50.97228] Got packet of 92 bytes
[2017-06-27 10:13:50.130323] Got packet of 455 bytes
[2017-06-27 10:13:50.130534] Got packet of 455 bytes
[2017-06-27 10:13:50.199395] Got packet of 214 bytes
^C
Finishing the capture... that's all folks!
root@itaqua:/lab/users/Miguel/miguel/disciplinas/progredes/libpcap#
```

Função Principal

```
#include <iostream>

#include "pcapwrapper.h"

using namespace std;

int main (int argc, char *argv[]) {

        PcapWrapper pcapwrapper (argv [1]);

        //pcapwrapper.setFilter ("port 80");
        pcapwrapper.run ();

        return 0;
}
```

Classe PcapWrapper (*.h)

```
#include <iostream>
#include <stdlib.h>
#include <pcap.h>
#include <signal.h>
#include <unistd.h>

using namespace std;

/*! \class PcapWrapper
    \brief This class encapsulates some functions of libpcap.

    This class can be used as an exercise for a simple packet capture.
*/
class PcapWrapper {

        public:

        PcapWrapper (char * = NULL);
        ~PcapWrapper ();

        void run (int = 0) ;

        void setFilter (char *);

        private:
                char * dev, * filter;
                char errbuf [PCAP_ERRBUF_SIZE];
                bpf_u_int32 mask, net;
                struct bpf_program fp;
                pcap_t * handle;
};
```

```
#include "pcapwrapper.h"

pcap_t * _handle;

void terminate_process (int) {
        pcap_breakloop (_handle);
}

void got_packet(u_char *args, const struct pcap_pkthdr *header, const u_char *packet) {
        time_t nowtime;
        struct tm *nowtm;
        char tmbuf [64];

        nowtime = header->ts.tv_sec;
        nowtm = localtime(&nowtime);
        strftime(tmbuf, sizeof tmbuf, "%Y-%m-%d %T", nowtm);

        cout << "[" << tmbuf << "." << header->ts.tv_usec << "]" << " Got packet of " << header->caplen << " bytes" << endl;
}


PcapWrapper::PcapWrapper (char * filter_) {

        dev = pcap_lookupdev (errbuf);

        if (dev == NULL) {
                cerr << "Couldn't find default device: " << errbuf << endl;
                cerr << "*** You must probably run as root! ***" << endl;
                exit (EXIT_FAILURE);
        }

        cout << "Device: " << dev << endl;

        // Find the properties for the device
        if (pcap_lookupnet (dev, &net, &mask, errbuf) == -1) {
                cerr << "Couldn't get netmask for device" << dev << ":" << errbuf;
                net = 0;
                mask = 0;
        }

        handle = pcap_open_live (dev, BUFSIZ, 1, 0, errbuf);
        if (handle == NULL) {
                cerr << "Couldn't open device " << dev << " : " << errbuf << endl;
                exit (EXIT_FAILURE);
        }

        if (filter_ != NULL)
                setFilter (filter);
}
```

Classe PcapWrapper (*.cpp)

```
PcapWrapper::~PcapWrapper () {
        cout << "\nFinishing the capture... that's all folks!" << endl;

        if (filter != NULL) pcap_freecode(&fp);
        pcap_close (handle);
}

void PcapWrapper::run (int stoptime_) {
        // stop after stoptime seconds
        if (stoptime_ > 0) {
                cout << "*** Finishing after " << stoptime_ << " seconds ***" << endl;
                signal (SIGALRM, terminate_process);
                alarm (stoptime_);
                _handle = handle;
        } else {
                cout << "*** Press ctrl-c to finish ***" << endl;
                signal (SIGINT, terminate_process);
                _handle = handle;
        }

        // now we can set our callback function
        pcap_loop (handle, -1, got_packet, NULL);
}

void PcapWrapper::setFilter (char * filter_) {
        filter = filter_;

        // Compile and apply the filter
        if (pcap_compile (handle, &fp, filter_, 0, net) == -1) {
                cerr << "Couldn't parse filter " << filter_ << ":" << pcap_geterr (handle);
                exit (EXIT_FAILURE);
        }

        if (pcap_setfilter (handle, &fp) == -1) {
                cerr << "Couldn't parse filter " << filter_ << ":" << pcap_geterr (handle);
                exit (EXIT_FAILURE);
        }
}
```

Classe PcapWrapper (*.cpp)

Makefile

```
CPP = g++
CPPFLAGS = -Wall
LD = g++
LIBS = -lpcap

PROGRAM = captura
OBJS = main.o pcapwrapper.o

all: $(PROGRAM)

$(PROGRAM): $(OBJS)
	$(LD) $(OBJS) $(LIBS) -o $@

.cpp.o:
	$(CPP) $(CPPFLAGS) $(LIBS) -c $<

clean:
	rm -f *.o $(PROGRAM)
```

Avaliação com o Valgrind

```
root@itaqua:/lab/users/Miguel/miguel/disciplinas/progredes/libpcap# valgrind ./captura
==16234== Memcheck, a memory error detector
==16234== Copyright (C) 2002-2011, and GNU GPL'd, by Julian Seward et al.
==16234== Using Valgrind-3.7.0 and LibVEX; rerun with -h for copyright info
==16234== Command: ./captura
==16234==
==16234== Warning: noted but unhandled ioctl 0x8946 with no size/direction hints
==16234==    This could cause spurious value errors to appear.
==16234==    See README_MISSING_SYSCALL_OR_IOCTL for guidance on writing a proper wrapper.
==16234== Conditional jump or move depends on uninitialised value(s)
==16234==    at 0x40574B7: ??? (in /usr/lib/x86_64-linux-gnu/libpcap.so.1.3.0)
==16234==    by 0x405B4E7: pcap_activate (in /usr/lib/x86_64-linux-gnu/libpcap.so.1.3.0)
==16234==    by 0x405BD2C: pcap_open_live (in /usr/lib/x86_64-linux-gnu/libpcap.so.1.3.0)
==16234==    by 0x405C015: ??? (in /usr/lib/x86_64-linux-gnu/libpcap.so.1.3.0)
==16234==    by 0x405C23E: ??? (in /usr/lib/x86_64-linux-gnu/libpcap.so.1.3.0)
==16234==    by 0x405A8B6: pcap_findalldevs (in /usr/lib/x86_64-linux-gnu/libpcap.so.1.3.0)
==16234==    by 0x405C625: pcap_lookupdev (in /usr/lib/x86_64-linux-gnu/libpcap.so.1.3.0)
==16234==    by 0x4011A0: PcapWrapper::PcapWrapper(char*) (in /net/tijuca/lab/users/Miguel/miguel/disciplinas/progredes/libpcap/captura)
==16234==    by 0x400FD4: main (in /net/tijuca/lab/users/Miguel/miguel/disciplinas/progredes/libpcap/captura)
```

■ ■ ■

```
[2017-06-27 10:21:59.934320] Got packet of 54 bytes
[2017-06-27 10:21:59.934322] Got packet of 84 bytes
[2017-06-27 10:21:59.934325] Got packet of 54 bytes
[2017-06-27 10:22:00.531097] Got packet of 64 bytes
^C
Finishing the capture... that's all folks!
==16234== Conditional jump or move depends on uninitialised value(s)
==16234==    at 0x4013C7: PcapWrapper::~PcapWrapper() (in /net/tijuca/lab/users/Miguel/miguel/disciplinas/progredes/libpcap/captura)
==16234==    by 0x400FFC: main (in /net/tijuca/lab/users/Miguel/miguel/disciplinas/progredes/libpcap/captura)
==16234==
==16234== Conditional jump or move depends on uninitialised value(s)
==16234==    at 0x405DBC1: pcap_freecode (in /usr/lib/x86_64-linux-gnu/libpcap.so.1.3.0)
==16234==    by 0x4013DA: PcapWrapper::~PcapWrapper() (in /net/tijuca/lab/users/Miguel/miguel/disciplinas/progredes/libpcap/captura)
==16234==    by 0x400FFC: main (in /net/tijuca/lab/users/Miguel/miguel/disciplinas/progredes/libpcap/captura)
==16234==
==16234==
==16234== HEAP SUMMARY:
==16234==     in use at exit: 0 bytes in 0 blocks
==16234==   total heap usage: 295 allocs, 295 frees, 435,480 bytes allocated
==16234==
==16234== All heap blocks were freed -- no leaks are possible
==16234==
==16234== For counts of detected and suppressed errors, rerun with: -v
==16234== Use --track-origins=yes to see where uninitialised values come from
==16234== ERROR SUMMARY: 12 errors from 10 contexts (suppressed: 4 from 4)
```

Documentação

- Uso do **Doxygen**:
  - Programa que facilita a documentação do código com vários níveis de detalhes
    - Instalação via apt-get:
      - **`apt-get install doxygen`**
    - Documentação disponível em:
      - http://www.stack.nl/~dimitri/doxygen/
    - Entre outros formatos, pode gerar documentos em:
      - HTML [captura docs html] : Abrir o arquivo html/index.html
      - PDF: [captura docs pdf]: Abrir o arquivo refman.pdf

Uso Básico do Doxygen

- Criação do arquivo de configuração (**Doxyfile**):

```
$ doxygen -g
```

- Exemplo de configuração no **Doxyfile**:

```
PROJECT_NAME        = "Capturing with libpcap"
EXTRACT_ALL         = YES
EXTRACT_PRIVATE     = YES
HAVE_DOT            = YES
UML_LOOK            = YES
GENERATE_TREEVIEW   = YES
INPUT       = main.cpp pcapwrapper.h pcapwrapper.cpp
```

- Geração da documentação:

```
$ doxygen Doxyfile
```

- Geração do PDF:

```
$ cd latex; make
```

Leitura Recomendada

- Tim Carstens, "Programming with pcap", disponível em: http://www.tcpdump.org/pcap.html

- Doxygen, Disponível em http://www.stack.nl/~dimitri/doxygen/