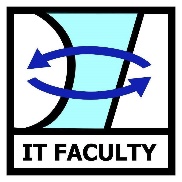
ĐẠI HỌC ĐÀ NẴNG

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA CÔNG NGHỆ THÔNG TIN

Tel. (+84.0236) 3736949, Fax. (84-511) 3842771

Website: http://dut.udn.vn/khoacntt, E-mail: cntt@dut.udn.vn



BÁO CÁO THỰC TẬP TUẦN 5

THỰC TẬP TỐT NGHIỆP

Đề tài:

Nfacct API

|  |  |  |
| --- | --- | --- |
| Họ và tên | Mã số sinh viên | Lớp học phần |
| Thân Nguyên Minh Quân | 102200148 | 20Nh14 |
| Hoàng Huân | 102200131 | 20Nh14 |
| Huỳnh Nguyễn Tấn Sang | 102200230 | 20Nh14 |

CBHD: TS Nguyễn Thế Xuân Ly

Đà Nẵng, 11/2024

MỤC LỤC

[I. Nội dung tuần 5 3](#_Toc186446170)

[**1. Nội dung 3**](#_Toc186446171)

[**2. Yêu cầu triển khai: 4**](#_Toc186446172)

[II. Triển khai các yêu cầu 5](#_Toc186446173)

[**1. Nfacct 5**](#_Toc186446174)

[**2. Iptables 6**](#_Toc186446175)

[**3. Libraries 6**](#_Toc186446176)

[**4. Xây dựng API theo sơ đồ sequence 7**](#_Toc186446177)

[**4.1. Sơ đồ Sequence gồm có: 7**](#_Toc186446178)

[**4.2. Các chức năng: 7**](#_Toc186446179)

[**4.2.1. Thêm tài khoản (Add nfacct): 7**](#_Toc186446180)

[**4.2.2. Xóa tài khoản (Delete nfacct): 9**](#_Toc186446181)

[**4.2.3. List tài khoản (List nfacct): 11**](#_Toc186446182)

[**5. Demo và nhận xét: 12**](#_Toc186446183)

[**5.1. Demo: 12**](#_Toc186446184)

[**5.2. Nhận xét: 16**](#_Toc186446185)

1. Nội dung tuần 5

1. Nội dung

1.1. Concept

nfacct is the command line tool to create/retrieve/delete accounting objects

Main Features

- listing the objects of the nfacct table in plain text/XML

- automatically get and reset objects of the nfacct table

- adding new objects to the nfacct table

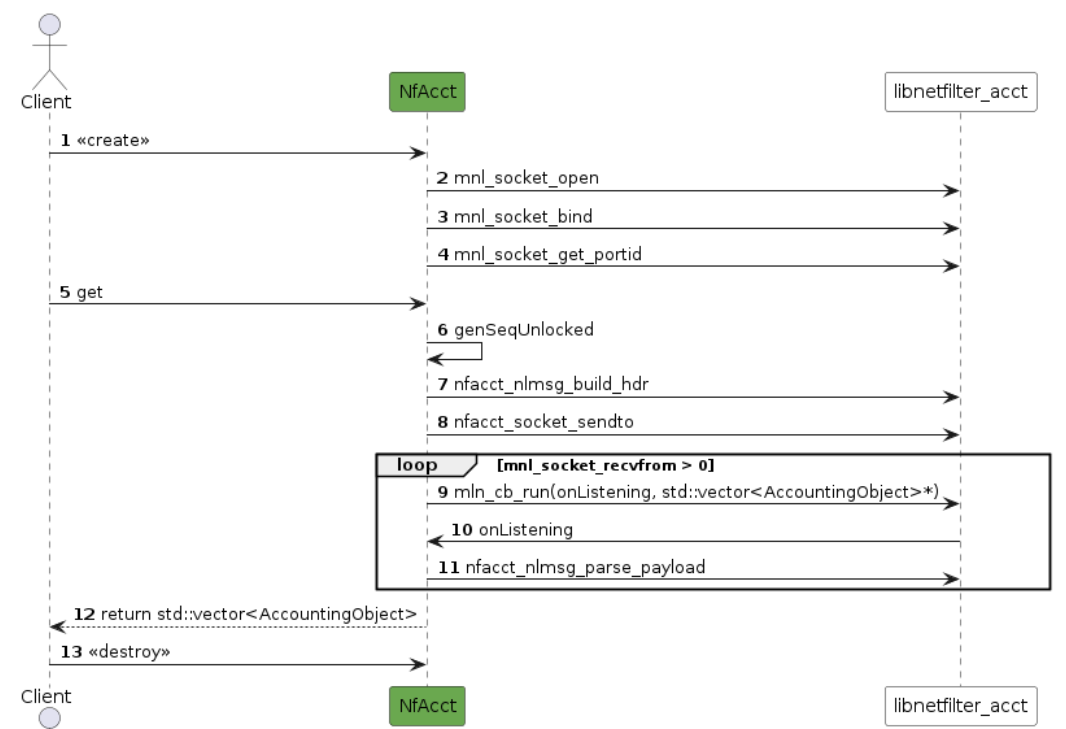
- deleting objects from the nfacct table

1.2. libraries

- libnetfilter\_acct is the userspace library providing the interface to extended accounting infrastructure.

- libmnl: is a minimalistic user-space library oriented to Netlink developers. There are a lot of common tasks in parsing, validating, and constructing of both the Netlink header and TLVs that are repetitive and easy to get wrong. This library aims to provide simple helpers that allow you to reuse code and avoid re-inventing the wheel

1.3. Sequence Diagram



2. Yêu cầu triển khai:

2.1. Dựa vào gợi ý về khái niệm và tính năng của Nfacct hãy xây dựng các file cấu hình của Nfacct để thêm/sửa/xoá các tài khoản.

Chú ý kết hợp với iptables để kiểm tra có thực hiện được các thao tác trên hay không?

2.2. Xây dựng API thực thi các công việc dựa vào Sequence Diagram trên

1. Triển khai các yêu cầu
2. Nfacct

NNfacct sẽ được sử dụng để tạo và quản lý các tài khoản ghi nhận lưu lượng (accounting objects) trên hệ thống.

* Add account nfacct:

sudo nfacct add my\_acc1

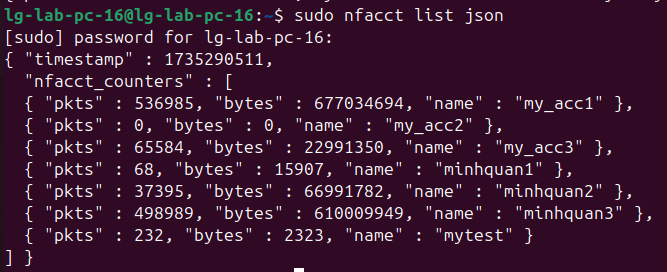
sudo nfacct add my\_acc2

* Delete account nfacct:

sudo nfacct delele my\_acc2

* Show nfacct config:

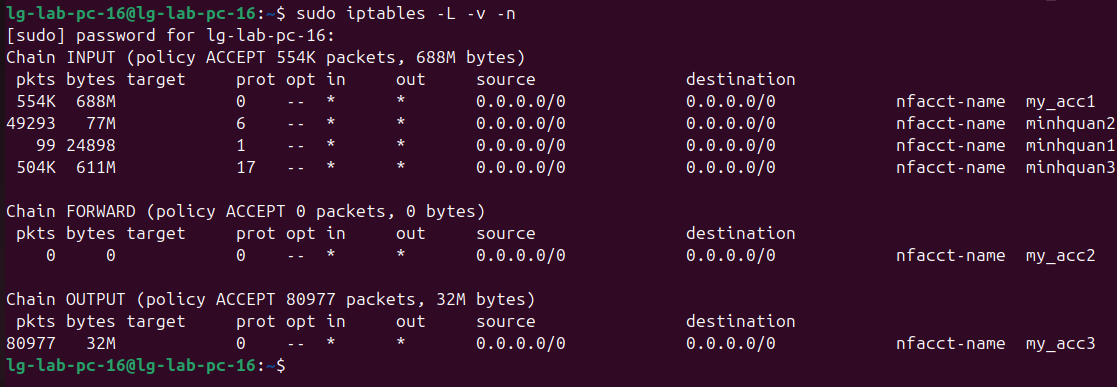
sudo nfacct list json



Mỗi tài khoản trong list nhận các lưu lượng gói tin đi qua khác nhau, phụ thuộc vào cấu hình rules Iptables. Ngoài ra còn các lệnh như Flush, Monitor, Restore... dùng để quản lí các tài khoản cụ thể hơn hay "xóa bỏ" hoặc "đẩy" (flush) dữ liệu hiện tại trong bộ đệm của công cụ.

1. Iptables

Khi sử dụng iptables kết hợp với nfacct, có thể phân loại lưu lượng theo các yếu tố như giao thức (TCP, ICMP, UDP), địa chỉ IP nguồn hoặc đích, hoặc cổng (port). Mỗi loại lưu lượng sẽ được chuyển hướng vào một tài khoản tương ứng trong nfacct.



Mỗi tài khoản lưu trữ gói tin qua các chain khác nhau, giao thức, ip hoặc các hành động khác nhau.

1. Libraries

**Sử dụng 2 thư viện:**

Tương tác với hệ thống Netfilter của Linux để quản lý và theo dõi các account.

* **libnetfilter\_acct**: cho phép nfacct thực hiện các thao tác như truy xuất và thay đổi dữ liệu trong bảng nfacct.

Sử dụng các hàm như nfacct\_alloc, nfacct\_attr\_set\_\*, và nfacct\_nlmsg\_build\_\* để thao tác với các account.

* **libmnl**: tương tác với Netlink, cung cấp các công cụ xử lý giao thức Netlink.

Các hàm như mnl\_socket\_open, mnl\_socket\_bind, và mnl\_socket\_sendto được sử dụng để tương tác với Netlink.

1. Xây dựng API theo sơ đồ sequence

### Sơ đồ Sequence gồm có:

**Khởi tạo (Create)**:

* + Mở socket (mnl\_socket\_open).
  + Ràng buộc socket (mnl\_socket\_bind).
  + Lấy portid từ socket (mnl\_socket\_get\_portid).

**Lấy dữ liệu (Get)**:

* + Tạo một thông điệp Netlink (nfacct\_nlmsg\_build\_hdr).
  + Gửi thông điệp (nfacct\_socket\_sendto).
  + Nhận dữ liệu (mnl\_socket\_recvfrom trong vòng lặp).
  + Gọi callback (mln\_cb\_run).

**Hủy kết nối (Destroy)**:

* + Đóng socket.

### Các chức năng:

#### **Thêm tài khoản (Add nfacct):**

Code: addNfacct

void addAccountingObject(std::vector<nfacct\*>& objects, const std::string& name, uint64\_t packets = 0, uint64\_t bytes = 0, uint32\_t flags = 0, uint64\_t quota = 0) {

struct nfacct \*acct = nfacct\_alloc();

if (!acct) {

std::cerr << "Failed to allocate memory for accounting object!" << std::endl;

return;

}

nfacct\_attr\_set\_str(acct, NFACCT\_ATTR\_NAME, name.c\_str());

nfacct\_attr\_set\_u64(acct, NFACCT\_ATTR\_PKTS, packets);

nfacct\_attr\_set\_u64(acct, NFACCT\_ATTR\_BYTES, bytes);

if (flags) {

nfacct\_attr\_set(acct, NFACCT\_ATTR\_FLAGS, &flags);

nfacct\_attr\_set\_u64(acct, NFACCT\_ATTR\_QUOTA, quota);

}

objects.push\_back(acct);

try {

struct mnl\_socket \*nl;

char buf[MNL\_SOCKET\_BUFFER\_SIZE];

struct nlmsghdr \*nlh;

uint32\_t portid, seq;

seq = time(NULL);

nlh = nfacct\_nlmsg\_build\_hdr(buf, NFNL\_MSG\_ACCT\_NEW, NLM\_F\_CREATE | NLM\_F\_ACK, seq);

nfacct\_nlmsg\_build\_payload(nlh, acct);

nl = mnl\_socket\_open(NETLINK\_NETFILTER);

if (nl == NULL) {

throw std::system\_error(errno, std::generic\_category());

}

if (mnl\_socket\_bind(nl, 0, MNL\_SOCKET\_AUTOPID) < 0) {

mnl\_socket\_close(nl);

throw std::system\_error(errno, std::generic\_category());

}

portid = mnl\_socket\_get\_portid(nl);

if (mnl\_socket\_sendto(nl, nlh, nlh->nlmsg\_len) < 0) {

mnl\_socket\_close(nl);

throw std::system\_error(errno, std::generic\_category());

}

int ret = mnl\_socket\_recvfrom(nl, buf, sizeof(buf));

while (ret > 0) {

ret = mnl\_cb\_run(buf, ret, seq, portid, nullptr, nullptr);

if (ret <= 0) {

break;

}

ret = mnl\_socket\_recvfrom(nl, buf, sizeof(buf));

}

if (ret == -1) {

mnl\_socket\_close(nl);

throw std::system\_error(errno, std::generic\_category());

}

mnl\_socket\_close(nl);

} catch (const std::exception& e) {

std::cerr << "Error while adding accounting object to system: " << e.what() << std::endl;

}

}

Cấp phát đối tượng : struct nfacct \*acct = nfacct\_alloc();

Thiết lập các thuộc tính chính: Name, Byte, Flag, Pkt, Quato

Thêm đối tượng vào object: objects.push\_back(acct);

Thiết lập thông điệp Netlink:

**seq = time(NULL); nlh = nfacct\_nlmsg\_build\_hdr(buf, NFNL\_MSG\_ACCT\_NEW, NLM\_F\_CREATE | NLM\_F\_ACK, seq); nfacct\_nlmsg\_build\_payload(nlh, acct);**

Tạo socket Netlink -> Gửi thông điệp đến kernel( chứa tài khoản)

Nhận phản hồi từ kernel -> Lặp để xử lí phản hồi bằng mnl\_socket\_recvfrom.

Đóng socket Netlink: mnl\_socket\_close(nl);

#### **Xóa tài khoản (Delete nfacct):**

Code: delNfacct

void removeAccountingObject(std::vector<nfacct\*>& objects, const std::string& name) {

bool objectFound = false;

for (auto it = objects.begin(); it != objects.end();) {

nfacct\* acct = \*it;

if (nfacct\_attr\_get\_str(acct, NFACCT\_ATTR\_NAME) == name) {

objectFound = true;

std::cout << "Attempting to remove accounting object: " << name << std::endl;

try {

struct mnl\_socket\* nl = nullptr;

char buf[MNL\_SOCKET\_BUFFER\_SIZE];

struct nlmsghdr\* nlh = nullptr;

uint32\_t portid = 0, seq = 0;

seq = time(nullptr);

nlh = nfacct\_nlmsg\_build\_hdr(buf, NFNL\_MSG\_ACCT\_DEL, NLM\_F\_ACK, seq);

nfacct\_nlmsg\_build\_payload(nlh, acct);

nl = mnl\_socket\_open(NETLINK\_NETFILTER);

if (nl == nullptr) {

throw std::runtime\_error("Failed to open Netlink socket");

}

if (mnl\_socket\_bind(nl, 0, MNL\_SOCKET\_AUTOPID) < 0) {

mnl\_socket\_close(nl);

throw std::runtime\_error("Failed to bind Netlink socket");

}

portid = mnl\_socket\_get\_portid(nl);

if (mnl\_socket\_sendto(nl, nlh, nlh->nlmsg\_len) < 0) {

mnl\_socket\_close(nl);

throw std::runtime\_error("Failed to send Netlink message");

}

int ret = mnl\_socket\_recvfrom(nl, buf, sizeof(buf));

while (ret > 0) {

ret = mnl\_cb\_run(buf, ret, seq, portid, nullptr, nullptr);

if (ret <= 0) {

break;

}

ret = mnl\_socket\_recvfrom(nl, buf, sizeof(buf));

}

if (ret == -1) {

mnl\_socket\_close(nl);

throw std::runtime\_error("Error receiving Netlink response");

}

mnl\_socket\_close(nl);

std::cout << "Successfully removed accounting object: " << name << std::endl;

} catch (const std::exception& e) {

std::cerr << "Error removing accounting object: " << name << " - " << e.what() << std::endl;

}

nfacct\_free(acct);

it = objects.erase(it);

} else {

++it;

}

}

if (!objectFound) {

std::cerr << "No accounting object found with name: " << name << std::endl;

}

}

Giống với Add Nfacct, Remove Nfacct cũng xử lí như vậy nhưng khác là xóa account. Các bước bao gồm:

Tìm và xác định đối tượng cần xóa: **bool objectFound = false;**

Duyệt qua danh sách objects để tìm đối tượng theo tên

Thiết lập thông điệp Netlink để yêu cầu xóa tài khoản:

**seq = time(nullptr);**

**nlh = nfacct\_nlmsg\_build\_hdr(buf, NFNL\_MSG\_ACCT\_DEL, NLM\_F\_ACK, seq);**

**nfacct\_nlmsg\_build\_payload(nlh, acct);**

Mở socket Netlink -> Gửi thông điệp đến kernel để xóa tài khoản:

**if (mnl\_socket\_sendto(nl, nlh, nlh->nlmsg\_len) < 0) {**

**mnl\_socket\_close(nl);**

**throw std::runtime\_error("Failed to send Netlink message");**

**}**

Nhận và xử lý phản hồi từ kernel, tương tự như trong hàm thêm (addAccountingObject)

Xóa đối tượng khỏi danh sách và dọn dẹp tài nguyên: Tài khoản sẽ được xóa khỏi danh sách objects và bộ nhớ sẽ được giải phóng.

#### **List tài khoản (List nfacct):**

Code: GetNfacct

void NfAcct::get(std::vector<nfacct\*>& objects, bool reset) {

char buf[MNL\_SOCKET\_BUFFER\_SIZE \* 2];

struct nlmsghdr \*nlh;

std::lock\_guard<std::mutex> lock(nlMutex);

uint32\_t seq = genSeqUnlocked();

nlh = nfacct\_nlmsg\_build\_hdr(buf, reset ? NFNL\_MSG\_ACCT\_GET\_CTRZERO : NFNL\_MSG\_ACCT\_GET, NLM\_F\_DUMP, seq);

if (mnl\_socket\_sendto(nl, nlh, nlh->nlmsg\_len) == -1)

throw std::system\_error(errno, std::generic\_category());

int ret;

while ((ret = mnl\_socket\_recvfrom(nl, buf, sizeof(buf))) > 0) {

if ((ret = mnl\_cb\_run(buf, ret, seq, portId, onListening, &objects)) <= 0)

break;

}

if (ret == -1)

throw std::system\_error(errno, std::generic\_category());

}

Hàm get lấy các đối tượng kế toán từ hệ thống và lưu chúng vào vector objects

Tạo một thông điệp Netlink (nlmsghdr) để yêu cầu các đối tượng kế toán từ hệ thống -> Sau khi gửi yêu cầu, hàm get sẽ nhận dữ liệu từ socket Netlink.

Hàm sử dụng hàm callback onListening để xử lý thông điệp Netlink nhận được. Nếu một đối tượng kế toán được tìm thấy, nó sẽ được thêm vào vector objects.

Sau đó, sử dụng print để in các đối tượng ra màn hình

void printAccountingObjects(const std::vector<nfacct\*>& objects) {

for (const auto& obj : objects) {

if (obj) {

const char\* name = nfacct\_attr\_get\_str(obj, NFACCT\_ATTR\_NAME);

uint64\_t packets = nfacct\_attr\_get\_u64(obj, NFACCT\_ATTR\_PKTS);

uint64\_t bytes = nfacct\_attr\_get\_u64(obj, NFACCT\_ATTR\_BYTES);

uint32\_t flags = nfacct\_attr\_get\_u64(obj, NFACCT\_ATTR\_FLAGS);

uint64\_t quota = nfacct\_attr\_get\_u64(obj, NFACCT\_ATTR\_QUOTA);

std::cout << "Accounting Object:\n";

std::cout << " Name: " << name << "\n";

std::cout << " Packets: " << packets << "\n";

std::cout << " Bytes: " << bytes << "\n";

std::cout << " Flags: " << flags << "\n";

std::cout << " Quota: " << quota << "\n";

std::cout << "------------------------------\n";

}

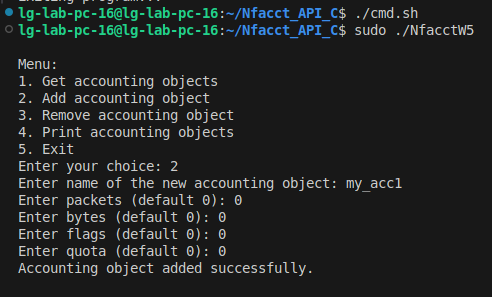
}

}

1. Demo và nhận xét:

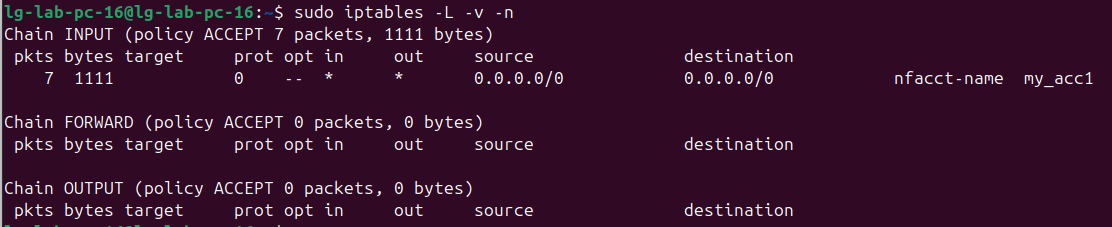
### Demo:

Sử dụng chức năng add các acccount vào hệ thống

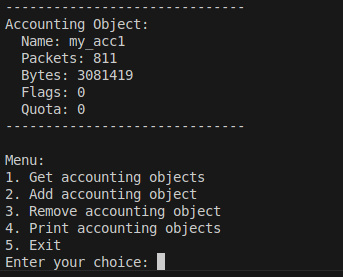


Dùng Iptables thêm my\_acc1 vào để thống kê lưu lượng của INPUT

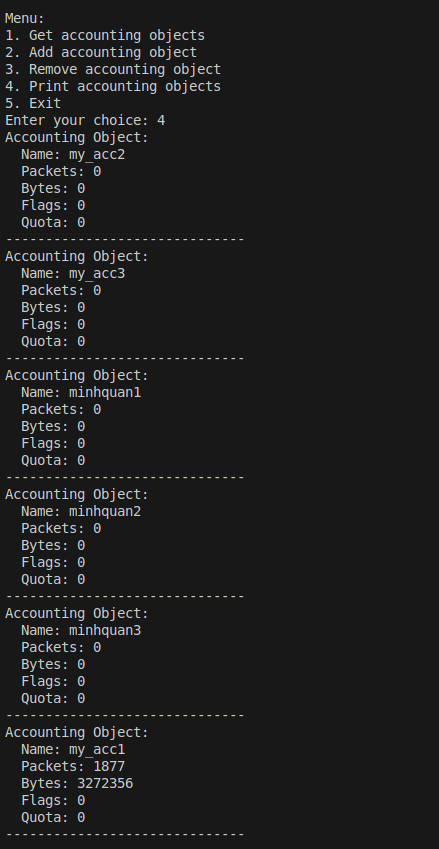
sudo iptables -A INPUT -m nfacct --nfacct-name my\_acc1



Xem lại lưu lượng bằng code

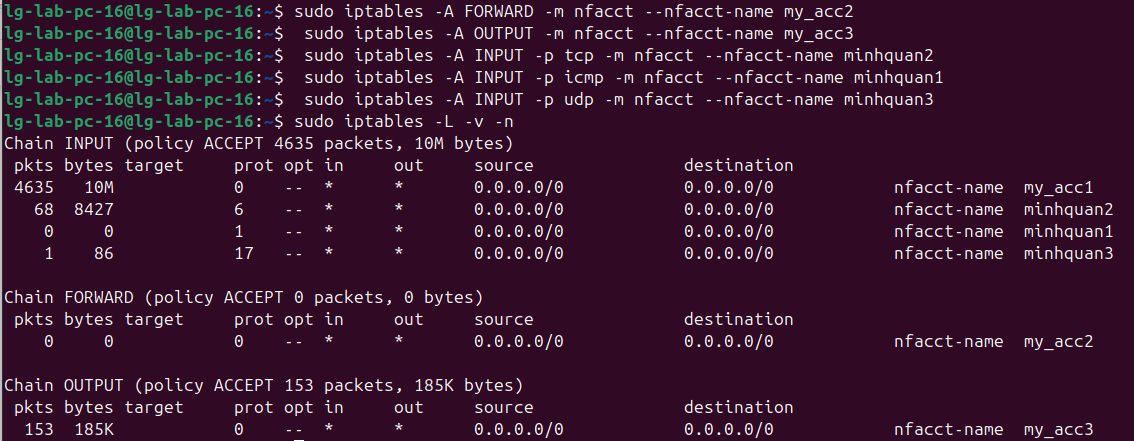


Bây giờ, thêm hàng loạt tài khoản và thiết lập Iptables để thống kê lưu lượng mạng cho mỗi chain, protocol ...

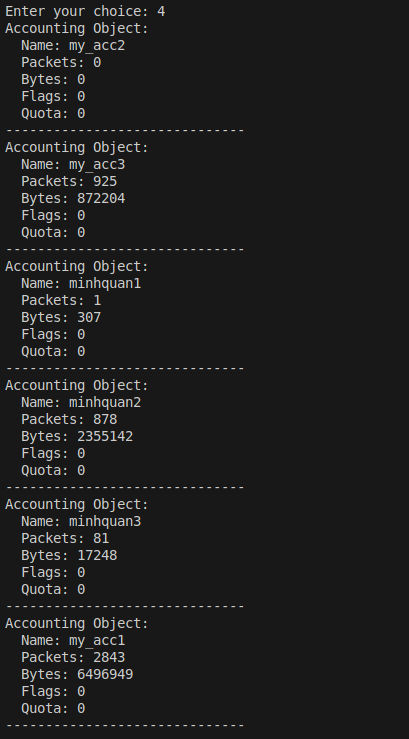


**Rules IPtables:**

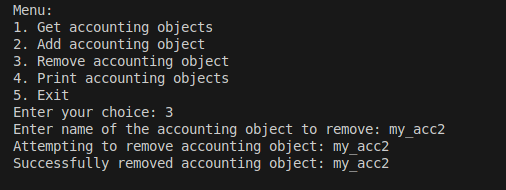
sudo iptables -A INPUT -m nfacct --nfacct-name my\_acc1  
 sudo iptables -A INPUT -p tcp -m nfacct --nfacct-name minhquan2  
 sudo iptables -A INPUT -p icmp -m nfacct --nfacct-name minhquan1  
 sudo iptables -A INPUT -p udp -m nfacct --nfacct-name minhquan3  
 sudo iptables -A FORWARD -m nfacct --nfacct-name my\_acc2  
 sudo iptables -A OUTPUT -m nfacct --nfacct-name my\_acc3



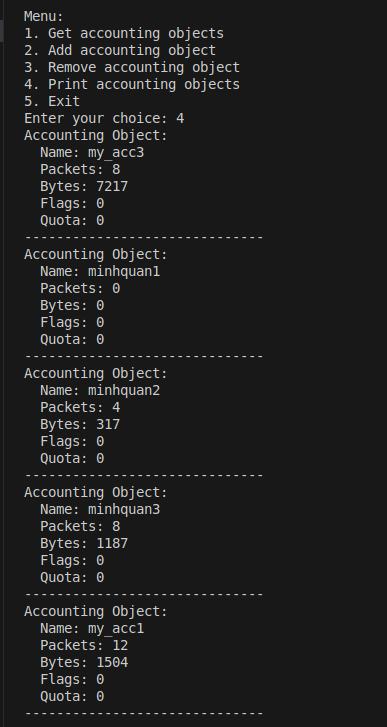
Kết quả thu được của các tài khoản:



Nhận thấy tài khoản my\_acc2 không thu thập được gì, không hiệu quả từ chain Forward => xóa tài khoản.



Xóa thành công:



### Nhận xét:

Nhóm đã triển khai code và xây dựng API theo Sequence Diagram, code hoạt động tốt, các chức năng cho ra kết quả chính xác rõ ràng, liên kết với netlink. Thống kê của mỗi tài khoản đưa ra là đầy đủ phù hợp với các rules Iptables đã thêm.

Với chức năng sửa tài khoản, nhóm chưa hiểu là nên xóa luôn tài khoản đó nếu sai hay không (chưa triển khai), ngoài ra thì chưa sử dụng flush config của nfacct */usr/sbin/nfacct flush.*

Qua nội dung tuần nhóm đã hiểu cách sử dụng Nfacct, quản lí các account để thống kê lưu lượng mạng, code C++ để tạo các chức năng đã yêu cầu.

***Tham Khảo***

*SourceCode:* [*https://git.netfilter.org/nfacct/tree/src/nfacct.c*](https://git.netfilter.org/nfacct/tree/src/nfacct.c)

libnetfilter\_acct: <https://netfilter.org/projects/libnetfilter_acct/doxygen/html/group__nfacct.html>