Katedra Informatyki

Wydział Informatyki i Telekomunikacji
Politechnika Krakowska

Programowanie Usług Sieciowych

mgr inż. Michał Niedźwiecki

Zarządzanie lokalną konfiguracją sieciową

Laboratorium: 05,

system operacyjny: Linux

Kraków, 2021

Spis treści

1. Wiadomości wstępne

Pierwsza część niniejszej instrukcji zawiera podstawowe wiadomości teoretyczne dotyczące zarządzania konfiguracją sieciową z poziomu aplikacji użytkownika. Poznanie tych wiadomości umożliwi prawidłowe zrealizowanie praktycznej części laboratorium.

1.1. Tematyka laboratorium

Tematyką laboratorium jest programowanie aplikacji pozwalających na administrowanie lokalną stacją sieciową, a w szczególności na:

- pobieranie informacji konfiguracyjnych i statystyk,
- zarządzanie interfejsami,
- zarządzanie adresami IP,
- zarządzanie tablicą routingu oraz tablicą sąsiadów (ang. neighbour table).

W systemie Linux dostępne są trzy mechanizmy umożliwiające przeprowadzenie powyższych operacji:

- interfejs ioctl,
- system plików proc,
- interfejs gniazd Netlink.

1.2. Zagadnienia do przygotowania

Przed przystąpieniem do realizacji laboratorium należy zapoznać się z zagadnieniami dotyczącymi:

- obsługi programu ifconfig [5]
- obsługi programu ip z pakietu iproute2 [6, 7]
- systemu plików proc (/proc/net) [8]
- adresowania w protokołach IPv4 i IPv6 [1, 2]
- adresów MAC oraz ich zastosowania w protokole IPv6 [2]
- MTU (ang. Maximum Transmission Unit)
- protokołu ARP [4]
- konfiguracji aliasów interfejsu [5]
- adresów podstawowych (ang. primary) [6, 10]
  i drugorzędnych (ang. secondary)

Należy także zapoznać się z zagadnieniami dotyczącymi:

- stosowania gniazd w systemie Linux [1]
- konwersji adresów IP - inet_ntop(), inet_pton() [1]
- stosowania funkcji ioctl() [1, 9]

Literatura:

[1] W.R. Stevens, „Programowanie Usług Sieciowych”, „API: gniazda i XTI”.

[2] IETF (http://www.ietf.org/), RFC 4291, „IP Version 6 Addressing Architecture”

[3] IETF, RFC 3549, „Linux Netlink as an IP Services Protocol”

[4] IETF, RFC 826, „An Ethernet Address Resolution Protocol”

[5] MAN (8), „ifconfig”, http://net-tools.berlios.de/man/ifconfig.8.html

[6] „IPROUTE2 Utility Suite Howto”, http://www.policyrouting.org/iproute2.doc.html

[7] „Linux Advanced Routing & Traffic Control HOWTO”, http://lartc.org/howto/

[8] MAN (5), „proc”

[9] MAN (2), „ioctl”

[10] Christian Benvenuti, „Understanding Linux Network Internals”, O’Reilly

1.3. Opis laboratorium

1.3.1. Interfejs ioctl

Funkcja ioctl() stanowi systemowy interfejs pozwalający na komunikację pomiędzy procesami użytkownika a komponentami jądra (sterownikami urządzeń, implementacją stosu TCP/IP, itp.). Za pomocą funkcji ioctl() można zarządzać parametrami urządzeń – w tym interfejsów sieciowych - oraz gniazd. Funkcja pozwala również na manipulację tablicą routingu i tablicą ARP.

```
#include <sys/ioctl.h>

int ioctl(int sd, int request, ... /* char *argp */ );
```

Parametr	Opis
sd	deskryptor pliku (gniazda)
request	kod polecenia; określa czy argp jest parametrem wejściowym, czy wyjściowym oraz determinuje typ wskaźnika argp
*argp	trzeci argument jest zawsze wskaźnikiem do obszaru pamięci, ale jego typ zależy od parametru request – tradycyjnie jest to char* argp.

Zazwyczaj, po pomyślnym zakończeniu funkcja ioctl() zwraca wartość zero. W przypadku niektórych kodów poleceń, ioctl() używa zwracanej wartości jako parametru wyjściowego - wówczas zwracana jest pewna wartość nieujemna. W przypadku błędu, funkcja zwraca -1 i odpowiednio ustawiana jest wartość zmiennej errno.

W systemie Linux polecenie man 2 ioctl_list udostępnia listę kodów poleceń funkcji ioctl(). Lista określa nazwę kodu (parametr request), typ wskaźnika argp dla danego kodu oraz to, czy jest on parametrem wejściowym, czy wyjściowym (może być zarówno wejściowym jak i wyjściowym).

Funkcja ioctl() wykorzystywana jest przez pakiet net-tools do przeprowadzania większości operacji konfiguracyjnych. Na przykładzie programów z pakietu net-tools przedstawione zostaną najczęściej stosowane kody poleceń ioctl wraz z ich zastosowaniem.

Kody poleceń ioctl w pakiecie net-tools

- Program ifconfig (ifconfig.c, lib/interface.c)

Nazwa kodu	Zastosowanie
SIOCGIFHWADDR	pobranie adresu sprzętowego (MAC dla Ethernet) interfejsu
SIOCSIFHWADDR	ustawienie adresu sprzętowego interfejsu
SIOCGIFFLAGS	pobranie flag interfejsu
SIOCSIFFLAGS	ustawienie flag interfejsu
SIOCGIFMTU	pobranie MTU (ang. *Maximum Transmission Unit*) interfejsu
SIOCSIFMTU	ustawienie MTU interfejsu
SIOCGIFADDR	pobranie adresu interfejsu (tylko IPv4); adresy IPv6 pobierane są za pomocą systemu plików *proc*
SIOCSIFADDR	ustawienie adresu interfejsu (IPv4 lub IPv6)
SIOCDIFADDR	usunięcie adresu IPv6; adresy IPv4 są usuwane przez wyłączenie interfejsu
SIOCGIFNETMASK	pobranie maski sieciowej (tylko IPv4); dla protokołu IPv6 pobranie długości prefiksu odbywa się za pomocą systemu plików *proc*
SIOCSIFNETMASK	ustawienie maski sieciowej (tylko IPv4); dla protokołu IPv6 ustawienie długości prefiksu odbywa się jednocześnie z ustawieniem adresu - za pomocą SIOCSIFADDR
SIOCGIFBRDADDR	pobranie adresu rozgłoszeniowego (tylko IPv4); protokół IPv6 nie definiuje adresu rozgłoszeniowego
SIOCSIFBRDADDR	ustawienie adresu rozgłoszeniowego (tylko IPv4)
SIOCGIFDSTADDR	pobranie zdalnego adresu dla interfejsu point-to-point (tylko IPv4)
SIOCSIFDSTADDR	ustawienie zdalnego adresu dla interfejsu point-to-point (tylko IPv4)

Wszystkie przedstawione kody poleceń wymagają wskaźnika na strukturę ifreq jako ostatniego argumentu funkcji ioctl(). Część operacji dla protokołu IPv6 jest wykonywana za pomocą systemu plików proc (rozdział 1.3.2).

Poniżej przedstawiona jest struktura ifreq.

```
#include <net/if.h>

#define IFNAMESIZE 16

struct ifreq {
```

```
 #define IFNAMSIZ IFNAMESIZE
 union {
 char ifrn_name[IFNAMSIZ]; /* Nazwa interfejsu */
 } ifr_ifrn;

 union {
 struct sockaddr ifru_addr; /* Adres */

 /* Adres zdalny point-to-point: */
 struct sockaddr ifru_dstaddr;
 struct sockaddr ifru_broadaddr; /* Adres rozgłoszeniowy */
 struct sockaddr ifru_netmask; /* Maska */
 struct sockaddr ifru_hwaddr; /* Adres MAC */
 short int ifru_flags; /* Flagi interfejsu */
 int ifru_ivalue; /* Metryka */
 int ifru_mtu; /* MTU interfejsu */

 /* Parametry sprzętowe: IRQ, DMA, ...: */
 struct ifmap ifru_map;

 /* Urządzenie slave w przypadku równoważenia obciążenia
 *(ang. load balancing): */
 char ifru_slave[IFNAMSIZ];

 /* Nowa nazwa interfejsu: */
 char ifru_newname[IFNAMSIZ];
 __caddr_t ifru_data;
 } ifr_ifru;
};

/* Do pól unii należy odwoływać się za pomocą następujących nazw: */
#define ifr_name ifr_ifrn.ifrn_name
#define ifr_hwaddr ifr_ifru.ifru_hwaddr
#define ifr_addr ifr_ifru.ifru_addr
#define ifr_dstaddr ifr_ifru.ifru_dstaddr
#define ifr_broadaddr ifr_ifru.ifru_broadaddr

/* Pozostałe dyrektywy define zostały pominięte. */
```

W przypadku operacji pobierania informacji konfiguracyjnych należy zdefiniować nazwę interfejsu (ifr_name w strukturze ifreq), a następnie wywołać funkcję ioctl() z odpowiednim kodem SIOCGIF*. śądany parametr konfiguracyjny zostanie zwrócony w polu unii ifr_ifru.

W przypadku operacji konfigurowania parametru (SIOCSIF*) należy zdefiniować nazwę interfejsu (ifr_name) oraz odpowiednie pole w unii ifr_ifru, po czym wywołać funkcję ioctl(). Operacje konfiguracyjne związane z protokołem IPv4 wymagają wywołania funkcji ioctl() na deskryptorze gniazda utworzonego dla domeny komunikacyjnej AF_INET.

Podobnie operacje konfiguracyjne dla protokołu IPv6 wymagają deskryptora gniazda utworzonego dla domeny komunikacyjnej AF_INET6. Ograniczenie to nie występuje w przypadku operacji niezależnych od wersji protokołu IP, np.: SIOCSIFMTU i SIOCGIFHWADDR.

W przypadku korzystania z funkcji ioctl() konwencją jest stosowanie gniazd typu SOCK_DGRAM.

Przykład pobrania informacji na temat MTU interfejsu eth0:

```
int sockfd;
struct ifreq ifr;

sockfd = socket(AF_INET, SOCK_DGRAM, 0);

strcpy(ifr.ifr_name, “eth0”);
ioctl(sockfd, SIOCGIFMTU, &ifr);

fprintf(stdout, “MTU: %d\n”, ifr.ifr_mtu);
```

Polecenia o kodach SIOCGIFFLAGS i SIOCSIFFLAGS wykorzystują flagi interfejsu, które zostały zdefiniowane w pliku nagłówkowym <net/if.h>. Najczęściej wykorzystywane flagi to:

Nazwa flagi	Znaczenie
IFF_UP	interfejs uruchomiony (w stanie UP)
IFF_RUNNING	interfejs gotowy do działania (zasoby zostały zaalokowane)
IFF_PROMISC	interfejs w trybie *promiscuous* – odbiera cały ruch docierający do karty sieciowej
IFF_MULTICAST	interfejs wspiera transmisję typu *multicast*
IFF_LOOPBACK	interfejs jest interfejsem *loopback*
IFF_NOARP	wsparcie dla protokołu ARP wyłączone

Przykład uruchomienia interfejsu eth0:

```
int sockfd;
struct ifreq ifr;

sockfd = socket(AF_INET, SOCK_DGRAM, 0);

strcpy(ifr.ifr_name, “eth0”);
ioctl(sockfd, SIOCGIFFLAGS, &ifr); /* Pobranie aktualnych flag */

strcpy(ifr.ifr_name, “eth0”);
ifr.ifr_flags |= IFF_UP | IFF_RUNNING; /* Dodanie nowych flag */
ioctl(sockfd, SIOCSIFFLAGS, &ifr);
```

Uwaga:

Polecenia o kodach SIOCSIF* i SIOCDIF* wymagają uprawnień roota.

- Program arp (arp.c)

Nazwa kodu	Zastosowanie
SIOCSARP	Dodanie wpisu do tablicy ARP (*ARP cache*)
SIOCDARP	Usunięcie wpisu z tablicy ARP

Dodawanie i usuwanie wpisów z tablicy ARP wymaga wykorzystania struktury arpreq:

```
#include <net/if_arp.h>

struct arpreq {
 struct sockaddr arp_pa; /* Ptotocol address */
 struct sockaddr arp_ha; /* Hardware address */

 /* Opcjonalne flagi.
 * Opis - man 8 arp, flagi pub (ATM_PUBL) oraz temp (ATM_PERM) */
 int arp_flags;

 /* Dla ProxyARP (RFC 1027). Opis – man 8 arp, opcja netmask */
 struct sockaddr arp_netmask;
 char arp_dev[16]; /* Nazwa interfejsu */
};
```

W celu usunięcia wpisu z tablicy ARP należy podać adres IP (pole arp_pa) i opcjonalnie nazwę interfejsu (pole arp_dev).

Dodawanie wpisu do tablicy ARP wymaga podania:

- adresu IP (pole arp_pa) oraz adresu MAC (pole arp_ha),
- opcjonalnie nazwy interfejsu (pole arp_dev),
- opcjonalnie maski arp_netmask dla techniki ProxyARP (gdy ustawiona jest flaga ATM_PUBL).

Ponadto, wymogiem jest ustawienie flagi ATF_COM (co oznacza, że podany adres sprzętowy jest kompletny i poprawny).

Wpis z ustawioną flagą ATM_PERM uznawany jest za permanentny, a bez tej flagi – za tymczasowy. Adres MAC należy zapisać bezpośrednio do pola sa_data struktury sockaddr, a pole sa_family ma przyjąć wartość identyfikującą protokół Ethernet – ARPHRD_ETHER.

Pobieranie zawartości tablicy ARP w programie arp odbywa się za pomocą systemu plików proc (rozdział 1.3.2).

- Program route (route.c, lib/inet_sr.c, lib/inet6_sr.c, lib/inet_gr.c, lib/inet6_gr.c)

Nazwa kodu	Zastosowanie
SIOCADDRT	Dodanie wpisu do tablicy routingu
SIOCDELRT	Usunięcie wpisu z tablicy routingu

Dodawanie i usuwanie wpisów z tablicy routingu wymaga wykorzystania struktury rtentry (<net/route.h>).

Pobieranie wpisów z tablicy routingu w programie route odbywa się za pomocą systemu plików proc.

Pobieranie adresów IP wszystkich interfejsów za pomocą funkcji ioctl

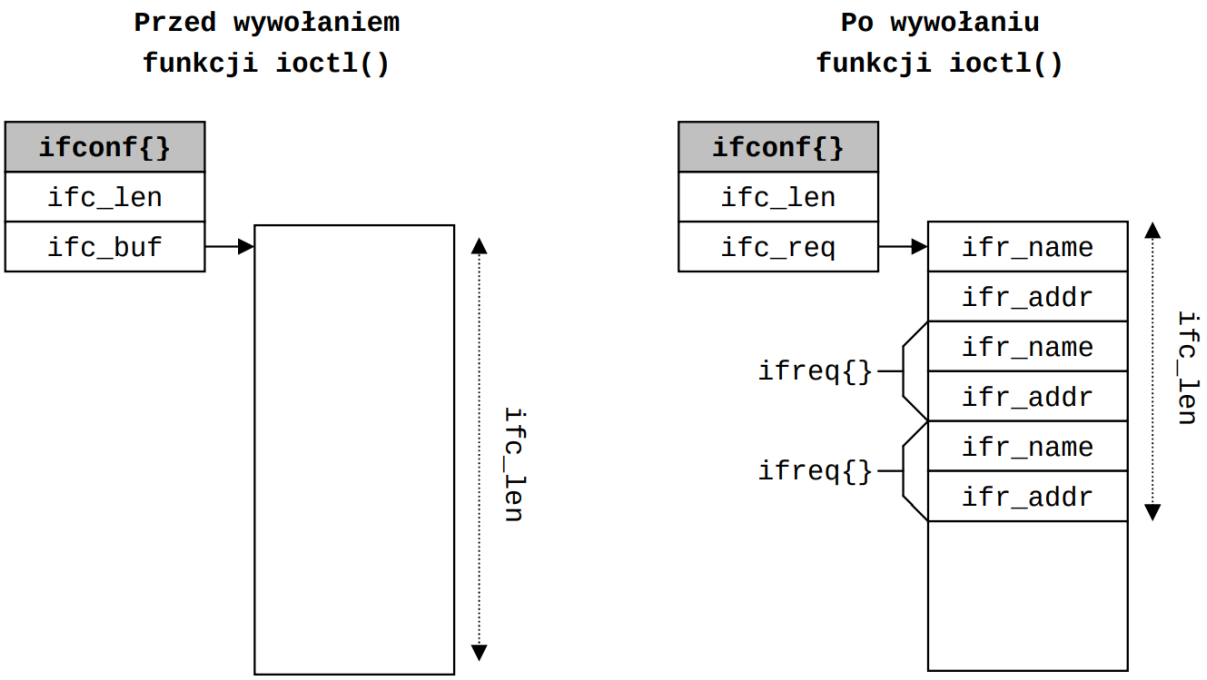
Wywołanie funkcji ioctl() z kodem polecenia SIOCGIFCONF umożliwia pobranie adresów IPv4 wszystkich interfejsów w stanie UP. Argumentem funkcji ioctl() dla operacji SIOCGIFCONF powinien być wskaźnik na strukturę ifconf.

```
#include <net/if.h>

struct ifconf {
 int ifc_len; /* Rozmiar bufora */
 union {
  char *ifc_buf; /* Bufor na zwracane informacje */
  struct ifreq *ifc_req; /* Tablica struktur */
 };
};
```

Przed wywołaniem funkcji ioctl() należy zaalokować obszar pamięci dla bufora ifc_buf oraz ustawić wartość pola ifc_len na rozmiar tego bufora. W wyniku wywołania funkcji, w zaalokowanym obszarze pamięci zwrócona zostanie tablica struktur ifreq. Każdy element tablicy będzie zawierał nazwę interfejsu oraz adres IPv4 odpowiadający danej nazwie. Do tablicy można odwołać się za pomocą wskaźnika ifc_req. Wartość pola ifc_len jest modyfikowana przez funkcję ioctl() i po jej wywołaniu określa rozmiar zwróconej tablicy w bajtach.

Rys. 1 prezentuje postać struktury ifconf przed oraz po wywołaniu funkcji ioctl().



Rys. 1. Postać struktury ifconf przed i po wywołaniu funkcji ioctl().

Wadą funkcji ioctl() jest to, że w przypadku gdy bufor podawany na wejściu jest za mały, funkcja przycina zwracaną tablicę i nie informuje o wystąpieniu błędu. Aby ustrzec się przed tego typu sytuacją należy wywoływać funkcję ioctl() w pętli, za każdym razem zwiększając rozmiar bufora o pewną wartość. Pętla powinna wykonywać się dopóki rozmiar zwróconej tablicy między dwoma kolejnymi wywołaniami nie ulegnie zmianie. Będzie to oznaczać, że tablica nie została przycięta i zawiera wszystkie elementy.

1.3.2. System plików „proc”

System plików proc stanowi interfejs pomiędzy jądrem systemu Linux a przestrzenią użytkownika (ang. userspace). Został zaprojektowany z myślą o dostarczaniu informacji na temat procesów, ale jego przydatność sprawiła, że obecnie wiele komponentów jądra używa go do raportowania informacji lub umożliwia dynamiczną rekonfigurację systemu (tzn. zmianę jego parametrów bez konieczności ponownego uruchomienia). Proc jest systemem wirtualnym. Zawartość plików jest generowana na podstawie odpowiednich struktur danych jądra i reprezentuje stan w chwili odczytu pliku.

System plików jest montowany w katalogu /proc. Składają się na niego katalogi (jako sposób organizowania informacji) oraz pliki. Większość plików w katalogu /proc posiada prawa tylko do odczytu, jednak istnieją również pliki, które umożliwiają zmianę stanu jądra systemu (z prawami do zapisu). Format danych do odczytania lub zapisania może być różny, ale najczęściej jest to tekst o kodowaniu znaków ASCII.

Format tekstowy umożliwia odczytanie zawartości plików za pomocą polecenia cat.Pliki w katalogu /proc/net umożliwiają odczyt informacji konfiguracyjnych związanych z wieloma aspektami stosu sieciowego:

- /proc/net/dev

Zawiera informacje na temat interfejsów sieciowych (nazwę, liczbę wysłanych i otrzymanych pakietów, liczbę błędów i kolizji, itp.). Za pomocą tego pseudopliku można odczytać listę wszystkich interfejsów, nawet administracyjnie wyłączonych.

- /proc/net/if_inet6

Zawiera informacje o adresach IPv6 interfejsów w stanie UP. Każda linia posiada następującą formę: adres IPv6, długość prefiksu, zakres adresu (ang. scope), flagi, nazwa interfejsu.

- /proc/net/arp

Tablica ARP – przedstawia mapowania pomiędzy adresami IPv4 a adresami warstwy łącza danych (MAC). W pliku prezentowane są zarówno wpisy dynamiczne, jak i statyczne (wprowadzone administracyjnie).

- /proc/net/router

Tablica routingu dla protokołu IPv4.

- /proc/net/ipv6_route

Tablica routingu dla protokołu IPv6.

- /proc/net/udp, /proc/net/tcp, /proc/net/raw

Pliki zawierają informacje o gniazdach UDP i TCP oraz o gniazdach typu SOCK_RAW. Program netstat opiera swoje działanie na informacjach pobieranych z tych plików.

- /proc/sys/net

Katalog /proc/sys/net grupuje pliki, które pozwalają na modyfikację zmiennych jądra systemu. Przykładowo plik /proc/sys/net/ipv4/ip_forward pozwala na skonfigurowanie usługi forwardingu pakietów, a plik /proc/sys/net/ip_default_ttl pozwala określić domyślną wartość pola TTL w wysyłanych datagramach IP. Modyfikacja zmiennych może odbywać się za pomocą polecenia echo. Poniższe polecenie uaktywnia forwarding pakietów dla protokołu IPv4:

```
echo "1" > /proc/sys/net/ipv4/ip_forward.
```

Dostęp do plików systemu proc z poziomu aplikacji użytkownika można uzyskać za pomocą funkcji fopen().

```
#include <stdio>

FILE *fopen(const char *path, const char *mode);
```

Parametr	Opis
*path	ścieżka do pliku
*mode	tryb dostępu do pliku, np.: ”r” – tylko do odczytu, ”r+” – do odczytu i zapisu

Funkcja fopen() zwraca wskaźnik do FILE lub NULL w przypadku błędu. Przykład zastosowania funkcji fopen() do konfiguracji usługi forwardingu:

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>

int main(int argc, char** argv) {
int ip_forward;
FILE *file;
/* Otworzenie pliku do odczytu i zapisu.
 * Tryb zapisu do pliku wymaga uprawnień roota. */
file = fopen("/proc/sys/net/ipv4/ip_forward", "r+");
if (file == NULL) {
 perror("fopen()"); exit(EXIT_FAILURE);
}

/* Pobranie aktualnej wartości zmiennej (0 lub 1): */
fscanf(file, "%d", &ip_forward);
printf("Wartosc ip_forward przed zmiana: %d\n", ip_forward);

/* Ustawienie pozycji wskaźnika na początek pliku: */
fseek(file, 0L, SEEK_SET);
/* Zmiana stanu usługi: */
fprintf(file, "%d", (ip_forward == 0) ? 1 : 0);

fclose(file);
exit(EXIT_SUCCESS);
}
```

1.3.3. Nazwa i indeks interfejsu

Interfejsy sieciowe w systemie Linux posiadają nazwę (np.: „eth0”) oraz indeks – dodatnią liczbę całkowitą przypisaną przez jądro systemu. Liczba zero nigdy nie jest używana w celu identyfikacji interfejsu sieciowego. Należy podkreślić, że numeracja interfejsów nie musi być ciągła.

RFC 3493 definiuje dwie funkcje odpowiedzialne za mapowanie między nazwą a indeksem interfejsu, funkcję umożliwiającą pobranie listy interfejsów (nazw i indeksów) oraz funkcję odpowiedzialną za zwolnienie zasobów zaalokowanych na potrzeby utworzenia tej listy.

Funkcja if_nametoindex() odwzorowuje nazwę interfejsu na jego indeks:

```
#include <net/if.h>

unsigned int if_nametoindex(const char *ifname);
```

Parametr	Opis
*ifname	nazwa interfejsu, np.: “eth0”

Funkcja zwraca indeks interfejsu o nazwie określonej przez parametr ifname lub zero, jeżeli podanej nazwie nie odpowiada żaden indeks.

Funkcja if_indextoname() odwzorowuje indeks interfejsu na jego nazwę:

```
#include <net/if.h>

char *if_indextoname(unsigned int ifindex, char *ifname);
```

Parametr	Opis
ifindex	indeks interfejsu
*ifname	bufor przeznaczony na nazwę interfejsu; rozmiar bufora powinien wynosić co najmniej IF_NAMESIZE

Jeżeli ifindex jest poprawnym indeksem interfejsu, to funkcja if_indextoname() zwraca nazwę interfejsu w buforze określonym przez parametr ifname. W przeciwnym wypadku funkcja zwraca NULL i odpowiednio ustawia wartość zmiennej errno.

Funkcja if_nameindex() zwraca tablicę struktur opisujących wszystkie interfejsy w systemie (nawet interfejsy administracyjnie wyłączone). Elementy tablicy zawierają nazwę oraz indeks interfejsu.

```
#include <net/if.h>

struct if_nameindex *if_nameindex(void);
```

Struktura if_nameindex przedstawiona jest poniżej:

```
#include <net/if.h>

struct if_nameindex {
 unsigned int if_index; /* Indeks interfejsu, np.: 1 */
 char *if_name; /* Nazwa interfejsu, np.: "eth0" */
};
```

Ostatni element zwróconej tablicy zawiera strukturę, której pole if_index ustawione jest na 0, a pole if_name ustawione na NULL. W przypadku błędu funkcja zwraca NULL i odpowiednio ustawia wartość zmiennej errno. Pamięć zajmowana przez zwróconą tablicę jest alokowana dynamicznie przez funkcję if_nameindex() i w celu jej zwolnienia należy wywołać funkcję if_freenameindex().

Funkcja if_freenameindex() zwalnia pamięć zaalokowaną przez funkcję if_nameindex():

```
#include <net/if.h>

void if_freenameindex(struct if_nameindex *ptr);
```

Parametr ptr jest wskaźnikiem zwróconym przez funkcję if_nameindex().

1.3.4. Adresy podstawowe, drugorzędne i aliasy

W pewnych przypadkach zachodzi konieczność skonfigurowania kilku adresów IP na jednym interfejsie. Przykładowe zastosowania takiego rozwiązania to:

- Udostępnienie wielu usług sieciowych na jednym interfejsie. Każdej usłudze można przypisać niezależny adres IP.
- Połączenie kilku sieci VLAN za pomocą jednego interfejsu (technika routeron-the-stick). Stacja sieciowa z systemem Linux (jądro >= 2.4.14) pełni wówczas rolę routera i zajmuje się przekazywaniem pakietów (forwardingiem) między sieciami VLAN.

Kiedy na jednym interfejsie skonfigurowanych jest wiele adresów IP, jądro systemu nie musi traktować ich jako równoważnych. Na podstawie prefiksu (RFC 4632: „CIDR Address Strategy”) wprowadzane jest rozróżnienie między adresami podstawowymi (ang. primary), a adresami drugorzędnymi (ang. secondary).

Adres jest uznawany za drugorzędny, jeżeli jego prefiks jest taki sam jak prefiks innego adresu (skonfigurowanego wcześniej) na danym interfejsie. Każdy interfejs może mieć dowolną liczbę adresów podstawowych i drugorzędnych. Dla danego prefiksu, tylko jeden adres może być traktowany jako podstawowy, a każdy kolejny adres zostanie zakwalifikowany jako drugorzędny.

Przykład konfiguracji adresów IP dla interfejsu eth0 za pomocą programu ip (pakiet iproute2):

```
root@host:~# ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> (…) /* pominięto */
 inet 192.168.1.3/24 brd 192.168.1.255 scope global eth0

root@host:~# ip addr add 192.168.1.4/24 dev eth0
root@host:~# ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> (…) /* pominięto */
 inet 192.168.1.3/24 brd 192.168.1.255 scope global eth0
 inet 192.168.1.4/24 scope global secondary eth0
```

```
root@host:~# ip addr add 192.168.1.5/26 dev eth0
root@host:~# ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> (...) /* pominięto */
 inet 192.168.1.3/24 brd 192.168.1.255 scope global eth0
 inet 192.168.1.5/26 scope global eth0
 inet 192.168.1.4/24 scope global secondary eth0
```

Adresy 192.168.1.3 oraz 192.168.1.4 posiadają ten sam prefiks (192.168.1.0/24). Ponieważ adres 192.168.1.4 został skonfigurowany później od adresu 192.168.1.3, jest on adresem drugorzędnym. Adres 192.168.1.5 posiada maskę o długości 26 (prefiks 192.168.1.0/26). Jest on adresem podstawowym dla tego prefiksu - nie istnieje żaden inny adres o tym samym prefiksie.

Nie należy mylić prefiksu sieci z długością prefiksu (ang. prefix-length). Przykładowo, 192.168.1.1/24 i 192.168.100.1/24 posiadają te same długości prefiksów (24), ale inne prefiksy: 192.168.1.0/24 oraz 192.168.100.0/24.

Adresy podstawowe posiadają następujące właściwości:

- W przypadku usunięcia adresu podstawowego, wszystkie adresy drugorzędne dla danego prefiksu i interfejsu są usuwane. Jest to domyślne zachowanie, ale istnieje możliwość jego skonfigurowania za pomocą systemu plików proc. Wykonanie polecenia:

  ```
  echo "1" > /proc/sys/net/ipv4/conf/eth0/promote_secondaries
  ```

  spowoduje, że w przypadku usunięcia adresu podstawowego interfejsu eth0, adres drugorzędny zostanie promowany na jego miejsce.
- Kiedy stacja sieciowa określa adres źródłowy dla lokalnie generowanych datagramów IP, bierze pod uwagę tylko adresy podstawowe.

Program ifconfig (najczęściej stosowany przez administratorów) nie rozróżnia adresów podstawowych i drugorzędnych. Adresy drugorzędne skonfigurowane za pomocą programu ip (pakiet iproute2) bez opcji label nie są wyświetlane przez ifconfig.

Przed wprowadzeniem iproute2, system Linux używał tzw. aliasów interfejsu. Aliasy są nadal dostępne w nowszych wersjach jądra, celem zachowania wstecznej kompatybilności. W przypadku ifconfig jedynym sposobem skonfigurowania wielu adresów dla jednego interfejsu jest zdefiniowanie wirtualnych interfejsów (aliasów) – eth0:0, eth0:1, itp. Kiedy konfiguruje się alias interfejsu, status podstawowy/drugorzędny jest przypisywany do adresu na zasadach opisanych w tym podrozdziale (pomimo, że dla ifconfig pojęcia te są obce). Program ip z pakietu iproute2 nie traktuje aliasów jako niezależnych interfejsów. Dla iproute2 alias jest tylko etykietą przypisaną do adresu.

```
root@host:~# ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> (...) /* pominięto */
 inet 192.168.1.3/24 brd 192.168.1.255 scope global eth0

root@host:~# ifconfig eth0:1 192.168.1.4 netmask 255.255.255.0
```

```
root@host:~# ifconfig eth0:5 192.168.2.1 netmask 255.255.255.0
root@host:~# ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> (...) /* pominięto */
 inet 192.168.1.3/24 scope global eth0
 inet 192.168.2.1/24 brd 192.168.2.255 scope global eth0:5
 inet 192.168.1.4/24 brd 192.168.1.255 scope global secondary eth0:1
```

Na listingu powyżej wyróżnione zostały etykiety interfejsu eth0 (aliasy dla ifconfig). Proszę zwrócić uwagę na fakt, że skonfigurowany za pomocą ifconfig adres 192.168.1.4 otrzymał status adresu drugorzędnego.

1.4. Cel laboratorium

Celem laboratorium jest zapoznanie się z interfejsem ioctl oraz systemem plików proc. Za pomocą wymienionych mechanizmów można tworzyć aplikacje pozwalające na zarządzanie lokalną konfiguracją sieciową. Podczas realizacji tego laboratorium zapoznasz się z:

- pobieraniem informacji konfiguracyjnych i parametrów adapterów sieciowych,
- zarządzaniem interfejsami,
- zarządzaniem adresami IP,
- zarządzaniem tablicą ARP oraz tablicą routingu.

2. Przebieg laboratorium

Druga część instrukcji zawiera zadania do praktycznej realizacji, które demonstrują zastosowanie technik z omawianego zagadnienia.

2.1. Zadanie 1. Parametry interfejsów sieciowych

Zadanie polega na analizie kodu przykładowego programu. Program wypisuje na standardowe wyjście parametry konfiguracyjne wszystkich interfejsów sieciowych (administracyjnie włączonych). Proszę zwrócić szczególną uwagę na sposób w jaki alokowany jest obszar pamięci dla zwracanych informacji. Funkcje malloc() oraz ioctl() wywoływane są w pętli do czasu, aż rozmiar zaalokowanego dynamicznie obszaru pamięci okaże się wystarczający dla zwracanych przez funkcję danych. Proszę pamiętać, że informacje zwracane przez funkcję ioctl() mogą ulec zmianie między kolejnymi wywołaniami.

W celu uruchomienia programu należy wykonać następujące czynności:

1. Przejść do katalogu ze źródłami (rozpakować je w razie konieczności).
2. Skompilować program źródłowy do postaci binarnej:

   ```
   $ gcc -o iolist iolist.c
   ```

3. Uruchomić program i przeanalizować jego działanie:

   ```
   $ ./iolist
   ```

4. Proszę porównać informacje zwrócone przez program z informacjami jakie można uzyskać za pomocą polecenia ifconfig lub ip addr.

2.2. Zadanie 2. System plików proc

Zadanie polega na analizie kodu przykładowego programu. Program otwiera plik /proc/net/if_inet6 i odczytuje z niego dane tekstowe. Odpowiednio sformatowane dane są przesyłane na standardowe wyjście. Po zamknięciu pliku, program wypisuje listę interfejsów uzyskaną za pomocą funkcji if_nameindex().

W celu uruchomienia programu należy wykonać następujące czynności:

1. Przejść do katalogu ze źródłami.
2. Skompilować program źródłowy do postaci binarnej:

   ```
   $ gcc -o proclist proclist.c
   ```

3. Uruchomić program i przeanalizować jego działanie:

   ```
   $ ./proclist
   ```

4. O czym informuje zakres adresu IPv6 (ang. address scope)?.
5. Czy lista interfejsów uzyskana za pomocą if_nameindex() jest zawsze zgodna z listą interfejsów w pliku /proc/net/if_inet6?

2.3. Zadanie 3. Modyfikacja tablicy ARP

Zadanie polega na analizie kodu programu, który dodaje wpis do tablicy ARP. Program przyjmuje adres IP oraz adres MAC jako argumenty wywołania. Proszę zwrócić uwagę na fakt, że pole określające nazwę interfejsu (arp_dec) w strukturze arpreq jest wyzerowane. Nazwa interfejsu dla wpisu zostanie określona przez jądro systemu na podstawie tablicy routingu.

Dokonywany wpis ma charakter permanentny, o czym decyduje flaga ATF_PERM.

W celu uruchomienia programu należy wykonać następujące czynności:

1. Przejść do katalogu ze źródłami.
2. Skompilować program źródłowy do postaci binarnej:

   $ gcc –o ioarp ioarp.c

3. Uruchomić sniffer (np. tcpdump) z odpowiednimi opcjami – tak aby przeanalizować adresy MAC dla wysyłanych komunikatów ICMP Echo.
4. Sprawdzić możliwość połączenia z dowolnym hostem w sieci lokalnej – za pomocą polecenia ping.
5. Na podstawie przechwyconych pakietów proszę określić jaki adres MAC posiada wybrana stacja sieciowa.
6. Uruchomić program podając adres IPv4 należący do stacji sieciowej wybranej w punkcie 4 oraz dowolny adres MAC (różny od adresu wybranej stacji):

   $ ./ioarp <adres IPv4> <adres MAC>

7. Proszę ponownie wywołać program ping z adresem IP hosta, dla którego zmodyfikowano wpis w tablicy ARP i za pomocą sniffera zaobserwować adresy MAC wysyłanych komunikatów.
8. Czy otrzymywane są odpowiedzi na komunikaty ICMP Echo?
9. Po zakończeniu zadania proszę usunąć wpis z tablicy ARP za pomocą polecenia:

   $ arp -d <adres IPv4>

2.4. Zadanie 4. Konfiguracja adresu MAC i MTU

Celem zadania jest zaimplementowanie programu umożliwiającego modyfikację adresu MAC oraz MTU interfejsu sieciowego za pomocą funkcji ioctl().

Program powinien:

- przyjmować jako argumenty wywołania nazwę interfejsu, adres MAC oraz liczbę określającą MTU,
- wypisywać na standardowe wyjście aktualny adres MAC i MTU,
- modyfikować konfigurację interfejsu,
- wypisywać adres MAC i MTU po modyfikacji.

Adres MAC ma zostać zapisany do pola ifr_hwaddr struktury ifreq w sposób podobny do zastosowanego w programie z zadania 3 (można posłużyć się funkcją sscanf()).

W celu uruchomienia programu należy wykonać następujące czynności:

1. Skompilować program źródłowy do postaci binarnej:

   ```
   $ gcc -o iomac iomac.c
   ```

2. Uruchomić program podając nazwę interfejsu, adres MAC oraz MTU:

   ```
   $ ./iomac <nazwa interfejsu> <adres MAC> <MTU>
   ```

3. Zweryfikować poprawność działania programu za pomocą polecenia ifconfig lub ip link.
4. Po zakończeniu zadania proszę przywrócić początkowe parametry interfejsu.

2.5. Zadanie 5. Zarządzanie adresami IPv4

Celem zadania jest zaimplementowanie programu umożliwiającego konfigurację adresów IPv4 za pomocą funkcji ioctl(). Program powinien przyjmować jako argumenty wywołania:

- nazwę interfejsu (lub aliasu), dla którego przeprowadzana jest konfiguracja,
- opcję określającą typ operacji – „add” dla dodania nowego adresu lub „down” dla usunięcia adresu. Usunięcie adresu IPv4 za pomocą funkcji ioctl() jest możliwe tylko przez wyzerowanie flagi IFF_UP interfejsu.
- dla opcji „add” wymagane jest dodatkowo określenie adresu IP oraz maski sieciowej w notacji kropkowo-dziesiętnej.

W przypadku dodawania nowego adresu dla aliasu interfejsu (np.: eth0:7), należy sprawdzić, czy interfejs „bazowy” (np.: eth0) jest administracyjnie włączony. Jeżeli nie jest, program powinien wypisywać odpowiedni komunikat i zakończyć działanie.

W celu uruchomienia programu należy wykonać następujące czynności:

1. Skompilować program źródłowy do postaci binarnej:

   ```
   $ gcc -o ioaddr ioaddr.c
   ```

2. Uruchomić program podając alias interfejsu (interfejs „bazowy” dla konfigurowanego aliasu musi istnieć w systemie), adres IPv4 oraz maskę sieciową:

   ```
   $ ./ioaddr <nazwa aliasu> add <adres IPv4> <maska>
   ```

   Proszę tak określić adres IP oraz maskę sieciową, aby prefiks konfigurowanego adresu różnił się od prefiksów adresów należących do interfejsu „bazowego” oraz wszystkich jego aliasów.
3. Zweryfikować poprawność działania programu za pomocą polecenia ifconfig lub ip addr.
4. Dodać kolejny adres o tym samym prefiksie sieciowym, co poprzedni:

   ```
   $ ./ioaddr <nazwa aliasu> add <adres IPv4> <maska>
   ```

   Wskazówka:
   Pomiędzy aliasami interfejsów, a adresami IP istnieje związek jeden-do-jeden. Każdy dodawany adres powinien mieć własny alias.
5. Za pomocą polecenia ip addr sprawdzić czy adres skonfigurowany w poprzednim punkcie otrzymał status adresu drugorzędnego.
6. Usunąć adres skonfigurowany w punkcie 2:

   ```
   $ ./ioaddr <nazwa aliasu> down
   ```

7. Proszę ponownie zweryfikować konfigurację za pomocą polecenia ip addr.
8. Czy usunięcie adresu podstawowego spowodowało usunięcie adresu drugorzędnego?

3. Opracowanie i sprawozdanie

Realizacja laboratorium pt. „Zarządzanie lokalną konfiguracją sieciową” polega na wykonaniu wszystkich zadań programistycznych podanych w drugiej części tej instrukcji. Wynikiem wykonania powinno być sprawozdanie w formie wydruku papierowego dostarczonego na kolejne zajęcia licząc od daty laboratorium, kiedy zadania zostały zadane.

Sprawozdanie powinno zawierać:

- opis metodyki realizacji zadań (system operacyjny, język programowania, biblioteki, itp.),

- algorytmy wykorzystane w zadaniach (zwłaszcza, jeśli zastosowane zostały rozwiązania nietypowe),
- opisy napisanych programów wraz z opcjami,
- trudniejsze kawałki kodu, opisane tak, jak w niniejszej instrukcji,
- uwagi oceniające ćwiczenie: trudne/łatwe, nie/realizowalne, nie/wymagające wcześniejszej znajomości zagadnień (wymienić jakich),
- wskazówki dotyczące ewentualnej poprawy instrukcji celem lepszego zrozumienia sensu oraz treści zadań.