Wojskowa Akademia Techniczna
Wydział Elektroniki
Instytut Telekomunikacji

Studia II°

Języki C/C++ w zastosowaniach sieciowych

Dodatek 4

dr inż. Jarosław Krygier

Warszawa 2020

Spis treści

1 Operacje na gnieździe typu RAW – różne protokoły (odbiornik)

```
/*
 ============================================================================
 Name        : Jcpp_odb.c
 Author      : J. Krygier
 Version     : 1.0
 Copyright   : JK
 Description : Odbiornik - operacje na gniezdzie RAW - rozne protokoly
 ============================================================================
 */
//UWAGA: program wymaga uprawnien administratora

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <errno.h>     //errno
#include <net/if.h>
#include <string.h>
#include <netinet/if_ether.h> // ETH_P_ALL

#define ROZMIAR_BUF             65536
#define IPv4 1
#define IPv6 0
#define ETH 0

int main(void) {
    puts("gniazdo RAW");

    struct sockaddr_in  adres_gniazda4;
    struct sockaddr_in6 adres_gniazda6;
    struct sockaddr     adres_gniazda_ogolny;

    int deskr_raw;
    unsigned char *buf = (unsigned char *) malloc(ROZMIAR_BUF);
    int rozmiar_danych;
    socklen_t rozmiar_gniazda;
    int i;
#if IPv4
    deskr_raw = socket(AF_INET, SOCK_RAW, IPPROTO_ICMP); //pakiety IPv4
#endif
#if IPv6
    deskr_raw = socket(AF_INET6, SOCK_RAW, IPPROTO_ICMPV6); //pakiety IPv6, bez naglowka
#endif
#if ETH
    deskr_raw = socket(AF_PACKET, SOCK_RAW, htons(ETH_P_ALL)); //ramki ETH
#endif
    if (deskr_raw < 0) {
        perror("Blad (socket())");
    } else {
        printf("Utworzono gniazdo, deskr:%d \n", deskr_raw);
    }

    struct ifreq ifr;
    memset(&ifr, 0, sizeof(ifr));
    snprintf(ifr.ifr_name, sizeof(ifr.ifr_name), "lo");
    setsockopt(deskr_raw, SOL_SOCKET, SO_BINDTODEVICE, (void *) &ifr,
            sizeof(ifr));

    for (;;) {
#if IPv4
        rozmiar_gniazda = sizeof (adres_gniazda4);
        rozmiar_danych = recvfrom(deskr_raw, buf, ROZMIAR_BUF, 0,(struct sockaddr *) &adres_gniazda4,
                &rozmiar_gniazda);
#endif
#if IPv6
```

```
        rozmiar_gniazda = sizeof (adres_gniazda6);
        rozmiar_danych = recvfrom(deskr_raw, buf, ROZMIAR_BUF, 0,(struct sockaddr *) &adres_gniazda6,
                                  &rozmiar_gniazda);
#endif
#if ETH
        rozmiar_gniazda = sizeof (adres_gniazda_ogolny);
        rozmiar_danych = recvfrom(deskr_raw, buf, ROZMIAR_BUF, 0, &adres_gniazda_ogolny,
                                  &rozmiar_gniazda);
#endif
        if (rozmiar_danych < 0) {
            printf("Recvfrom error , failed to get packets\n");
            return 1;
        }
        else {
            printf("Odebrano: %d bajtow\n",rozmiar_danych );
            for (i=0;i<rozmiar_danych;i++) {
                printf ("%.2x ", buf[i]);
            }
            printf ("\n");
        }
    }

    close(deskr_raw);
    puts("KONIEC");

    return EXIT_SUCCESS;
}
```

2 Operacje na gnieździe typu RAW – manipulacja nagłowkami (nadajnik)

```
/*
 ============================================================================
 Name        : Jcpp_nad.c
 Author      : J. Krygier
 Version     : v1.0
 Copyright   : JK
 Description : Nadajnik – operacje na dniezdzie typu RAW – manipulacja nagłowkami
 ============================================================================
 */
//UWAGA: program wymaga uprawnien administratora

#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h> //socket(), sendto()
#include <arpa/inet.h> //IPPROTO_RAW
#include <unistd.h>    //sleep
#include <netinet/ip.h> //nagl. ip
#include <string.h>
#include <stdlib.h>
#include <errno.h>

unsigned short csum();

int main(void) {
    puts("Prosty nadajnik RAW");
    int gniazdo_raw;
    unsigned char datagram[1500];
    struct sockaddr_in adres_gniazda_celu;
    int rozmiar_danych;
    int one = 1;
    const int *val = &one;
    unsigned char dane[1480];

    memset(datagram, 0, 1500);

    struct iphdr *iph = (struct iphdr *) datagram;

    gniazdo_raw = socket(AF_INET, SOCK_RAW, IPPROTO_ICMP);
    //gniazdo_raw = socket(PF_INET, SOCK_RAW, IPPROTO_RAW);
    if (gniazdo_raw == -1) {
        //nie moge utworzyc gniaza
        perror("Blad:");
        exit(1);
    }
    adres_gniazda_celu.sin_family = AF_INET;
    //adres_gniazda_celu.sin_port = htons(8080); //nie ma znaczenia jesli ustawiamy w naglowku IP oddzielnie
    adres_gniazda_celu.sin_addr.s_addr = inet_addr("127.10.10.10"); //adres docelowy

#if 1
    if (setsockopt(gniazdo_raw, IPPROTO_IP, IP_HDRINCL, val, sizeof(one)) < 0) {
        perror("Blad ustawienia opcji IP_HDRINCL");
        exit(0);
    }
#endif

    //uzupelnienie naglowka ipv4
    iph->ihl = 5;
    iph->version = 4;
    iph->tos = 0xe0;
    iph->tot_len = sizeof(struct iphdr) + sizeof(dane);
    iph->id = htonl(54321); //jakis identyfikator
    iph->frag_off = 0;
    iph->ttl = 255;
    iph->protocol = IPPROTO_ICMP;
    iph->check = 0; //tymczasowo
```

```
    iph->saddr = inet_addr("127.0.0.1");
    iph->daddr = adres_gniazda_celu.sin_addr.s_addr;

    //suma kontrolna wlasna funkcja
    //iph->check = csum((unsigned short *) datagram, iph->tot_len); // powinien system tez obliczyc

    //ustawimy dane
    /*memset(dane, 1, 1480);
    memset(dane, 0xff, 2); // ustawimy 2 bajty
    memset(dane+2, 0xff, 2); // ustawimy kolejne
    dane[4]=0;
    dane[5]=10;*/

    memcpy(datagram + sizeof(struct iphdr), dane, sizeof(dane));

    memset(dane, 0xff, 2);

    int i;
    for (i=0;i<40;i++) {
        printf ("%.2x ", datagram[i]);
    }
    printf("\n");

    rozmiar_danych
            = sendto(gniazdo_raw, datagram, sizeof(datagram), 0,
                            (struct sockaddr *) &adres_gniazda_celu,
                            sizeof(adres_gniazda_celu));
    if (rozmiar_danych < 0) {
        perror("Blad transmisji");
    } else {
        printf("Wyslano : %d [B]\n", rozmiar_danych);
    }
    //sleep(1);

    return EXIT_SUCCESS;
}

unsigned short csum(unsigned short *ptr, int nbytes) {
    register long sum;
    unsigned short oddbyte;
    register short answer;

    sum = 0;
    while (nbytes > 1) {
        sum += *ptr++;
        nbytes -= 2;
    }
    if (nbytes == 1) {
        oddbyte = 0;
        *((u_char*) &oddbyte) = *(u_char*) ptr;
        sum += oddbyte;
    }

    sum = (sum >> 16) + (sum & 0xffff);
    sum = sum + (sum >> 16);
    answer = (short) ~sum;

    return (answer);
}
```