Programowanie współbieżne

Laboratorium 8: Gniazdka

Kamil Zdeb, nr albumu: 235871

Piątek TN 9:15-12:15

05.06.2020

---

1 Wstęp

Programy miały być uruchamiane na komputerze z systemem Linux.

2 Realizacja zadań

Programy wykonałem na posiadanej na swoim komputerze dystrybucji Linux Manjaro przy użyciu edytora *Sublime Text*. W celu sprawniejszej pracy użyłem narzędzia *make* ułatwiającego kompilację plików.

2.1 Zadanie 1: Server i klient ftp

Zadaniem dzisiejszego laboratorium było stworzenie programów realizujących funkcję serwera oraz klienta ftp do komunikacji wykorzystującego gniazdka. Podczas laboratorium udało mi się zrealizować funkcjonalności odczytu i zapisu plików.

2.1.1 Kod programu

Program serwera tworzy gniazdko, a następnie oczekuje na przysłane mu przez klienta komunikaty. Po połączeniu się przez klienta serwer odpowiada na komunikaty przesyłane przez serwer.

```
// Proces odbierajacy komunikaty - wysyla udp_cli
// Wspolpracuje z udp_cli
// Kompilacja gcc udp_serw.c -o udp_serw -lrt
#include <arpa/inet.h>
#include <netinet/in.h>
#include <stdio.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <unistd.h>
#include <string.h>
#define PORT 9950

#define OPENR 1 // Otwarcie pliku do odczytu
#define OPENW 2 // Otwarcie pliku do zapisu
#define READ 3 // Odczyt fragmentu pliku
#define CLOSE 4 // Zamkniecie pliku
#define WRITE 5 // Zapis fragmentu pliku
#define OPENDIR 6 // Otworz zdalny katalog
#define READDIR 7 // Czytaj zdalny katalog
#define STOP 10 // Zatrzymanie serwera
#define SIZE 512
typedef struct {
 int typ; // typ zlecenia
 int ile; // liczba bajtow
 int fh; // uchwyt pliku
 char buf[SIZE]; // bufor
} mms_t;

```

```
30 void blad(char *s) {
31   perror(s);
32   _exit(1);
33 }
34 int main(void) {
35   struct sockaddr_in adr_moj, adr_cli;
36   int s, i, slen=sizeof(adr_cli),snd, rec, blen=sizeof(mms_t);
37   char buf[SIZE];
38   char buffer[512];
39   mms_t msg;
40   gethostname(buf,sizeof(buf));
41   printf("Host: %s\n",buf);
42   s=socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
43   if(s < 0) blad("socket");
44   printf("Gniazdko %d utworzone\n",s);
45  // Ustalenie adresu IP nadawcy
46   memset((char *) &adr_moj, 0, sizeof(adr_moj));
47   adr_moj.sin_family = AF_INET;
48   adr_moj.sin_port = htons(PORT);
49   adr_moj.sin_addr.s_addr = htonl(INADDR_ANY);
50   if (bind(s,(struct sockaddr *) &adr_moj, sizeof(adr_moj))==-1)
51   blad("bind");
52  // Odbior komunikatow ------------
53   do{
54     rec = recvfrom(s, &msg, blen, 0, &adr_cli, &slen);
55     if(rec < 0) blad("recvfrom()");
56     printf("Odebrano komunikat %d\n",msg.typ);
57     switch (msg.typ) {
58     case OPENR:
59       msg.fh = open(msg.buf,O_RDONLY);
60       break;
61
62     case READ:
63       msg.ile = read(msg.fh,msg.buf,SIZE);
64       break;
65
66     case OPENW:
67       strcpy(buffer, "ServerFiles/");
68       strcat(buffer, msg.buf);
69       msg.fh = open(buffer, O_CREAT | O_WRONLY | O_TRUNC, 0660);
70       if(msg.fh < 0){
71         blad("niepoprawny plik");
72       }
73       break;
74
75     case WRITE:
76       write(msg.fh, msg.buf, msg.ile);
77       break;
78     }
79     snd = sendto(s, &msg, blen, 0, &adr_cli, slen);
80     if(snd < 0) perror("sendto");
81   } while(msg.typ != STOP);
82   // Odpowiedz -----
83   close(s);
84   return 0;
85 }
```

Program klienta tworzy gniazdko i łączy się z serwerem na podstawie adresu

ip podanego jako argument uruchamianego programu. Następnie prosi o wybranie odpowiedniej opcji. Po wybraniu opcji użytkownik jest proszony o podanie nazwy pliku, który ma być odczytany lub zapisany.

```c
// Proces wysyla a potem odbiera komunikaty udp
// Wspolpracuje z udp_serw
// Kompilacja gcc udp_cli.c -o udp_cli -lrt
#include <netinet/in.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>
#define PORT 9950
#define SRV_IP "127.0.0.1"

#define OPENR 1 // Otwarcie pliku do odczytu
#define OPENW 2 // Otwarcie pliku do zapisu
#define READ 3 // Odczyt fragmentu pliku
#define CLOSE 4 // Zamkniecie pliku
#define WRITE 5 // Zapis fragmentu pliku
#define OPENDIR 6 // Otworz zdalny katalog
#define READDIR 7 // Czytaj zdalny katalog
#define STOP 10 // Zatrzymanie serwera
#define SIZE 512
typedef struct {
 int typ; // typ zlecenia
 int ile; // liczba bajtow
 int fh; // uchwyt pliku
 char buf[SIZE]; // bufor
} mms_t;

void blad(char *s) {
  perror(s);
  _exit(1);
 }
int main(int argc, char * argv[]) {
  struct sockaddr_in adr_moj, adr_serw, adr_x;
  int s, i, slen=sizeof(adr_serw), snd, blen=sizeof(mms_t),rec;
  int choice = 0;
  char buf[SIZE];
  int fh;
  mms_t msg;
  s=socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
  if(s < 0) blad("socket");
  printf("Gniazdko %d utworzone\n",s);
  memset((char *) &adr_serw, 0, sizeof(adr_serw));
  adr_serw.sin_family = AF_INET;
  adr_serw.sin_port = htons(PORT);
  if (inet_aton(argv[1], &adr_serw.sin_addr)==0) {
    fprintf(stderr, "inet_aton() failed\n");
    _exit(1);
  }
  printf("Wybierz opcje: \n\t1.Odczyt pliku \n\t2.Zapis pliku\n");
  scanf("%d", &choice);
  getchar();
```

```
  switch(choice){
    case 1:
      do { //Otwarcie pliku
        printf("Podaj nazwe pliku: ");
        gets(msg.buf);
        msg.typ = OPENR;
        snd = sendto(s, &msg, blen, 0, &adr_serw, slen);
        if(snd < 0) perror("sendto()");
        rec = recvfrom(s, &msg, blen, 0, &adr_moj, &slen);
        if(rec < 0) perror("recvfrom()");
      } while(msg.fh < 0);

      do { // Odczyt --------------------
        msg.typ = READ;
        msg.ile = SIZE;
        snd = sendto(s, &msg, blen, 0, &adr_serw, slen);
        if(snd < 0) perror("sendto()");
        rec = recvfrom(s, &msg, blen, 0, &adr_moj, &slen);
        if(rec < 0) perror("recvfrom()");
        printf("Odebrano: %d bajtow\n",msg.ile);
        msg.buf[msg.ile] = '\0';
        if(msg.ile > 0) printf("%s\n",msg.buf);
      } while(msg.ile == SIZE);
      break;
    case 2:
      do { //Otwarcie pliku
        printf("Podaj nazwe pliku: ");
        gets(msg.buf);
        msg.typ = OPENW;
        snd = sendto(s, &msg, blen, 0, &adr_serw, slen);
        if(snd < 0) perror("sendto()");
        rec = recvfrom(s, &msg, blen, 0, &adr_moj, &slen);
        if(rec < 0) perror("recvfrom()");
      } while(msg.fh < 0);

      fh = open(msg.buf,O_RDONLY);

      do { // Zapis --------------------
        printf("%s\n", msg.buf);
        msg.ile = read(fh, msg.buf, SIZE);
        msg.typ = WRITE;
        //if(msg.ile > 0) printf("%s\n\n",msg.buf);
        snd = sendto(s, &msg, blen, 0, &adr_serw, slen);
        if(snd < 0) perror("sendto()");
        rec = recvfrom(s, &msg, blen, 0, &adr_moj, &slen);
        if(rec < 0) perror("recvfrom()");
      } while(msg.ile == SIZE);
      break;
  }
  close(s);
  return 0;
}
```