

# Sprawozdanie nr 4 Programowanie Usług Sieciowych

**Mateusz Snoch**

## Zadanie

Na laboratorium należało napisać program (klient-serwer) z użyciem funkcji *select()*. Klient miał za zadanie odczytać liczby z pliku i przesłać je do serwera. Serwer mnoży liczby przez 1.5 i odsyła do klienta. Na koniec klient zapisuje otrzymane liczby do nowego pliku.

Format przykładowego pliku z liczbami:

1 2 4.5

1 3 5.5

1 5 6.5

Dwie pierwsze kolumny liczb są typu *int*, ostatnia *float*.

## Kod programu:

### Server.c

```
1. #include      "headers.h"
2. #include      <time.h>
3. int main(int argc, char **argv)
4. {
5.     int                    i,maxi, maxfd, listenfd, connfd, sockfd;
6.     int                    nready, client[FD_SETSIZE];
7.     ssize_t                n;
8.     fd_set                 rset, allset;
9.     char                   line[MAXLINE];
10.    char                   buff[MAXLINE];
11.    time_t                  ticks;
12.    socklen_t               clilen;
13.    struct sockaddr_in      cliaddr, servaddr;
14.    listenfd = socket(AF_INET, SOCK_STREAM, 0);
15.    bzero(&servaddr, sizeof(servaddr));
16.    servaddr.sin_family     = AF_INET;
17.    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
18.    servaddr.sin_port       = htons(4000);
19.    bind(listenfd, (SA *) &servaddr, sizeof(servaddr));
20.    listen(listenfd, LISTENQ);
21.    maxfd = listenfd;
22.    maxi = -1;
23.    for (i = 0; i < FD_SETSIZE; i++)
24.        client[i] = -1;
25.    FD_ZERO(&allset);
26.    FD_SET(listenfd, &allset);
27.    for ( ; ; ) {
28.        rset = allset;
```

```
29. nready = select(maxfd+1, &rset, NULL, NULL, NULL);
30. if (FD_ISSET(listenfd, &rset))
31. {
32. clilen = sizeof(cliaddr);
33. connfd = accept(listenfd, (SA *) &cliaddr, &clilen);
34. printf("connection from %s, port %d\n",
35. inet_ntop(AF_INET, &cliaddr.sin_addr, buff, sizeof(buff)),
36. ntohs(cliaddr.sin_port));
37. for (i = 0; i < FD_SETSIZE; i++)
38. if (client[i] < 0)
39. {
40. client[i] = connfd;
41. break;
42. }
43. if (i == FD_SETSIZE)
44. err_sys("Za duzo klientow");
45. FD_SET(connfd, &allset);
46. if (connfd > maxfd)
47. maxfd = connfd;
48. if (i > maxi)
49. maxi = i;
50. if (--nready <= 0)
51. continue;
52. }
53. for (i = 0; i <= maxi; i++)
54. {
55. if ( (sockfd = client[i]) < 0)
56. continue;
57. if (FD_ISSET(sockfd, &rset))
58. {
59. data _data;
60. int m = read(connfd, &_data, sizeof(data));
61. if(m==0)
62. {
63. close(sockfd);
64. FD_CLR(sockfd, &allset);
65. client[i] = -1;
66. }
67. else
68. {
69. printf("Nawiazano polaczenie\n");
70. int j;
71. for(j=0; j<9; j++){
72. _data.arr[j]*=1.5;
73. }
74. write(connfd, &_data, sizeof(data));
75. printf("Wyslano dane \n");
76. close(connfd);
77. }
78. if (--nready <= 0)
```

```
79. break;
80. }
81. }
82. }
83. }
```

Po zastosowaniu w programie funkcji select, jądro usypia proces i oczekuje na zdarzenie (np.: gotowy deskryptor połączenia) jądro wybudza proces w przypadku gdy nastąpi połączenie. Wywołanie funkcji select blokuje proces i oczekuje do momentu utworzenia jednego z deskryptorów gotowego do czytania, pisania, lub do pobrania wyjątku. Do programu serwera została dodana funkcja select().

## Client.c

```
1. #include      "headers.h"
2. int main(int argc, char **argv)
3. {
4.     int                                sockfd, n;
5.     struct sockaddr_in  servaddr;
6.     if (argc != 2)
7.         err_sys("Aby uruchomic : Adres IP");
8.     if ( ( sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
9.         err_sys("Blad utworzenia polaczenia");
10.    bzero(&servaddr, sizeof(servaddr));
11.    servaddr.sin_family = AF_INET;
12.    servaddr.sin_port  = htons(4000);
13.    if (inet_pton(AF_INET, argv[1], &servaddr.sin_addr) <= 0)
14.        err_sys("Blad konwersji do adresu IP dla %s", argv[1]);
15.    if (connect(sockfd, (SA *) &servaddr, sizeof(servaddr)) < 0)
16.        err_sys("Blad polaczenia");
17.    FILE *d=fopen("p1.txt","r");
18.    data _plik;
19.    int max = 0;
20.    while(!feof(d))
21.    {
22.        fscanf(d,"%f",&_plik.arr[max++]);
23.    }
24.    fclose(d);
25.    write(sockfd,&_plik,sizeof(data));
26.    data _plik2;
27.    n = read(sockfd, &_plik2, sizeof(data));
28.    if(n==sizeof(data))
29.    {
30.        d = fopen("p2.txt","w");
31.        printf("Odebrane dane\n");
32.        int i;
33.        for(i=0;i<9;i++)
34.        {
35.            fprintf(d,"%f\t",_plik2.arr[i]);
36.            if((i+1)%3==0) fprintf(d,"\n");
```

```
37. }
38. fclose(d);
39. }
40. exit(0);
41. }
```

Program klienta nie zmienił się w porównaniu do poprzednich zajęć. Niestety nie udało mi się dodać do programu klienta funkcji `select()`, w taki sposób, żeby poprawnie działał.

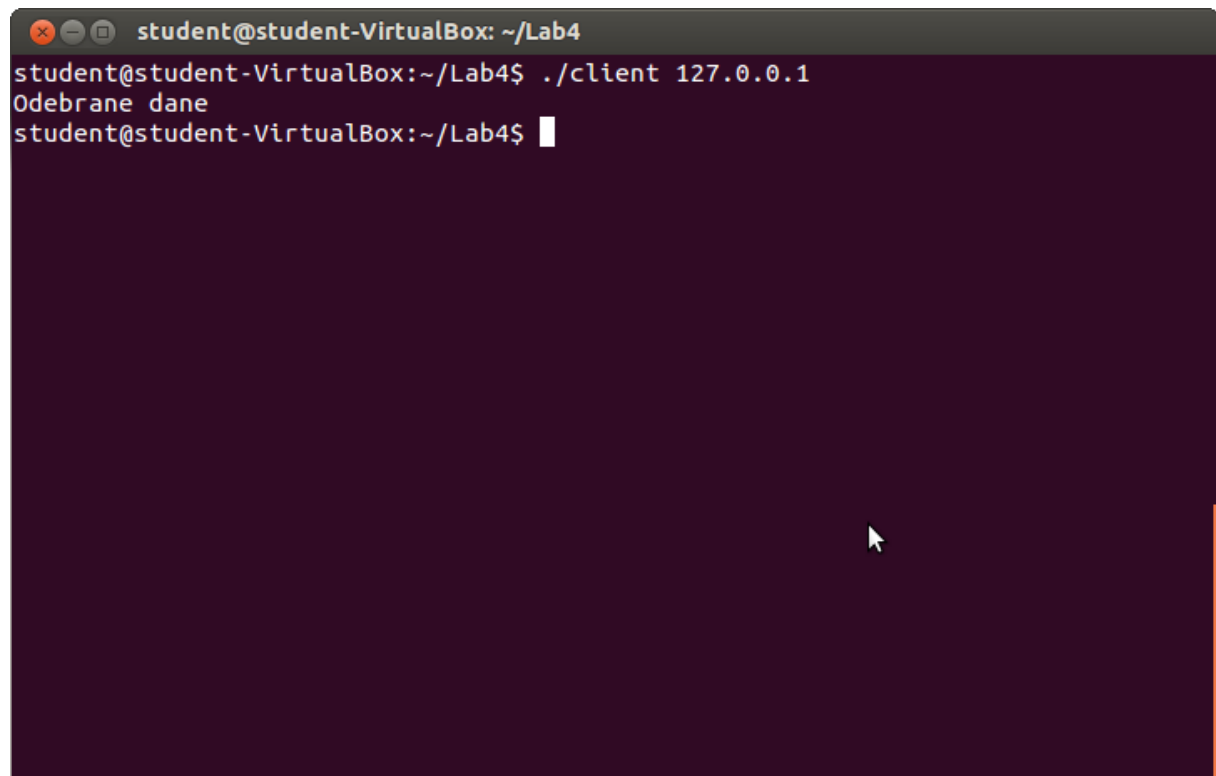
## Headers.h

```
1. #include      <netdb.h>
2. #include      <signal.h>
3. #include      <stdio.h>
4. #include      <stdlib.h>
5. #include      <string.h>
6. #include      <sys/stat.h>
7. #include      <sys/uio.h>
8. #include      <unistd.h>
9. #include      <sys/wait.h>
10. #include      <sys/un.h>
11. #include      <time.h>
12. #include      <netinet/in.h>
13. #include      <arpa/inet.h>
14. #include      <errno.h>
15. #include      <fcntl.h>
16. #define        LISTENQ          1024
17. #define        MAXLINE          4096
18. #define        MAXSOCKADDR      128
19. #define        BUFSIZE          8192
20. #define        SA      struct sockaddr
21. void sig_chld(int signo)
22. {
23.     pid_t pid;
24.     int stat;
25.     pid = wait (&stat);
26.     while((pid=waitpid(-1,&stat,WNOHANG))>0)
27.         printf("Potemek %d zakonczony\n", pid);
28.     return;
29. }
30. typedef struct
31. {
32.     float arr[9];
33. } data;
34. void err_sys(const char *, ...);
35. void err_sys(const char *msg, ...)
36. {
37.     printf("%s\n",msg);
38.     exit(1);
}
```

39. }

## Screeny z działania programu:

Client:



```
student@student-VirtualBox: ~/Lab4
student@student-VirtualBox:~/Lab4$ ./client 127.0.0.1
Odebrane dane
student@student-VirtualBox:~/Lab4$
```

The screenshot shows a terminal window with a dark background. The title bar at the top reads 'student@student-VirtualBox: ~/Lab4'. The terminal content shows a command prompt followed by the command './client 127.0.0.1'. The output of the command is 'Odebrane dane' on the next line. The prompt returns to 'student@student-VirtualBox:~/Lab4\$' with a cursor. A mouse cursor is visible on the right side of the terminal window.

Server:

```
student@student-VirtualBox: ~/Lab4
student@student-VirtualBox:~/Lab4$ ./server
connection from 127.0.0.1, port 56036
Nawiazano polaczenie
Wyslano dane
█
```

Na zajęciach udało mi się napisać server z funkcją `select()`, ale niestety nie udało mi się napisać klienta z tą funkcją.