

Sprawozdanie nr 3 Programowanie Usług Sieciowych

Mateusz Snoch

Zadanie

Zadaniem na tym laboratorium było przerobienie programu z poprzednich zajęć tak aby działał współbieżnie.

Kod programu:

Server.c

```
1. #include      "headers.h"
2. #include      <time.h>
3. int main(int argc, char **argv)
4. {
5.     int listenfd, connfd;
6.     pid_t childpid;
7.     struct sockaddr_in  servaddr;
8.     char                buff[MAXLINE];
9.     time_t              ticks;
10. listenfd = socket(AF_INET, SOCK_STREAM, 0);
11. bzero(&servaddr, sizeof(servaddr));
12. servaddr.sin_family    = AF_INET;
13. servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
14. servaddr.sin_port      = htons(4000);
15. bind(listenfd, (SA *) &servaddr, sizeof(servaddr));
16. listen(listenfd, LISTENQ);
17. signal(SIGCHLD,sig_chld);
18. for ( ; ; ) {
19.     connfd = accept(listenfd, (SA *) NULL, NULL);
20.     if ( (childpid = fork()) == 0)
21.     {
22.         close(listenfd);
23.         data _data;
24.         int n = read(connfd,&_data,sizeof(data));
25.         if(n==sizeof(data))
26.         {
27.             printf("Nawiazano polaczenie\n");
28.             int i;
29.             for(i=0;i<9;i++){
30.                 _data.arr[i]*=1.5;
31.             }
32.         }
33.     else printf("Nie udalo sie nawiazac polaczenia\n");
34.     write(connfd, &_data, sizeof(data));
```

```
35. printf("Wyslano dane \n");
36. exit(0);
37. }
38. close(connfd);
39. };
40. }
```

W porównaniu do poprzednich zajęć została użyta funkcja `fork()`. Dzięki użyciu tej funkcji serwer jest współbieżny, czyli może obsługiwać wielu klientów na raz. Gdy proces potomny zostaje zakończony proces macierzysty otrzymuje sygnał `SIGCHLD`. Po otrzymaniu sygnału, kończy proces potomny przy pomocy instrukcji `signal(SIGCHLD, sig_chld)`.

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("Bład 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("Bład konwersji do adresu IP dla %s", argv[1]);
15.    if (connect(sockfd, (SA *)&servaddr, sizeof(servaddr)) < 0)
16.        err_sys("Bład 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.

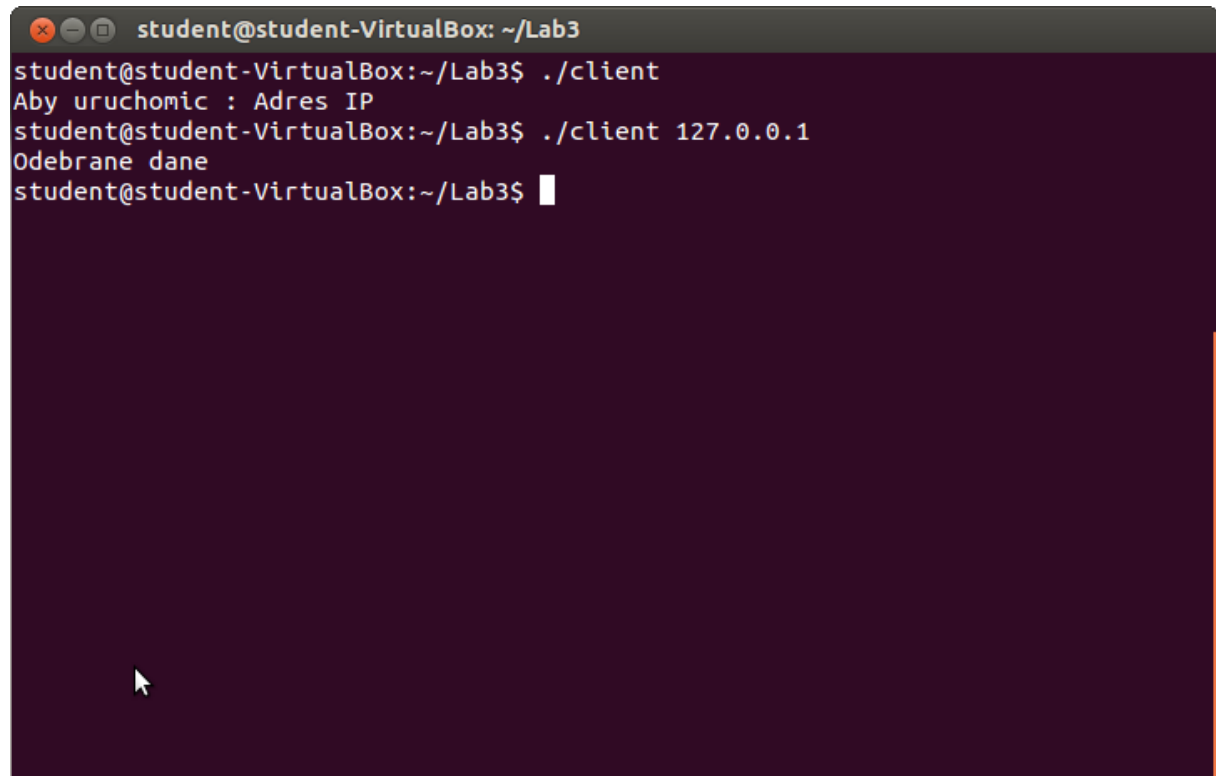
Headers.h

```
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:

A screenshot of a terminal window titled 'student@student-VirtualBox: ~/Lab3'. The terminal shows the following commands and output:

```
student@student-VirtualBox:~/Lab3$ ./client
Aby uruchomic : Adres IP
student@student-VirtualBox:~/Lab3$ ./client 127.0.0.1
Odebrane dane
student@student-VirtualBox:~/Lab3$
```

The terminal has a dark purple background and a white cursor is visible at the end of the last line.

Server:

```
student@student-VirtualBox: ~/Lab3
student@student-VirtualBox:~$ cd Lab 3
bash: cd: Lab: Nie ma takiego pliku ani katalogu
student@student-VirtualBox:~$ cd Lab3
student@student-VirtualBox:~/Lab3$ gcc server.c -o server
student@student-VirtualBox:~/Lab3$ ./server
Nawiazano polaczenie
Wyslano dane
█
```