SPRAWOZDANIE LABORATORIUM 2.3 – PROGRAMOWANIE SIECIOWE

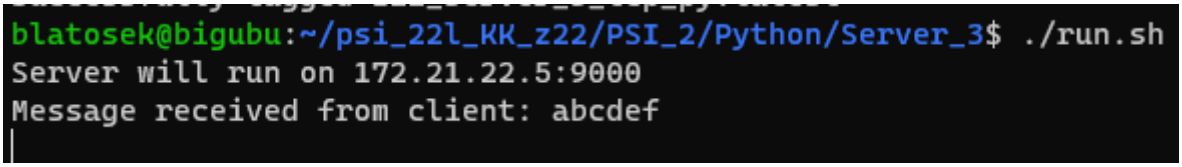
Z22 - Bartosz Latosek, Adam Sudoł, Karol Rogoziński, Bartłomiej Dudek

2.3

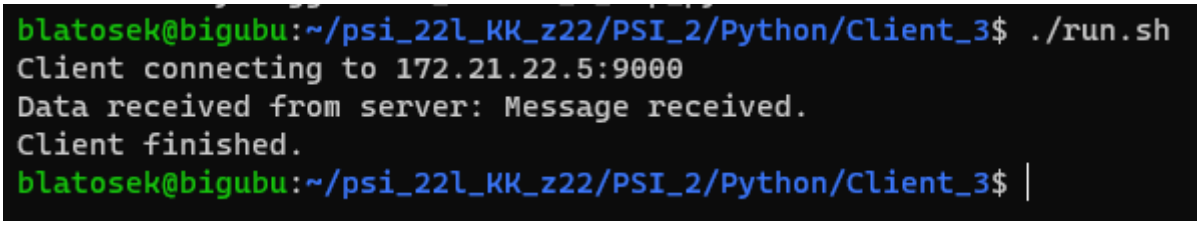
Python - posługując się funkcją settimeout() zmodyfikować program z Z 2.1 tak, aby obsługiwał timeout-y dla funkcji connect() i accept().

Działanie:

Server python:



Klient python:



Kod źródłowy:

Server Python:

```python
import socket as s
from socket import *

PORT = 9000
BUFFER = 1024
HOST = s.gethostbyname(s.gethostname())
TIMEOUT_TIME = 1

with s.socket(s.AF_INET, s.SOCK_STREAM) as server:
  server.bind((HOST, PORT))
  server.listen( 5 )
  print(f"Server will run on {HOST}:{PORT}")
  while True:
    client, address = server.accept()
    with client:
      try:
        client.settimeout(TIMEOUT_TIME)
        data = client.recv(BUFFER)
      except timeout:
        print("Couldn't recveive data (TIMEOUT)")
      if not data:
        break
      print(f"Message received from client: {data.decode('utf-8')}")
      client.sendall("Message received.\0".encode('utf-8'))
    client.close()
  server.close()
```

Klient Python:

```python
import socket as s
import random
from socket import *

HOST = '172.21.22.5'
PORT = 9000
BUFFER = 1024
TIMEOUT_TIME = 0.0001

messages = ["Hello world!", "Bazinga", "Random Data", "Top secret", "abcdef"]

print(f"Client connecting to {HOST}:{PORT}")

with s.socket(s.AF_INET, s.SOCK_STREAM) as client:
  try:
    client.settimeout(TIMEOUT_TIME)
    client.connect((HOST, PORT))
  except timeout:
    print("Failed to connect (timeout)")
    exit(-1)
  data = random.choice(messages)
  try:
    client.sendall(data.encode('utf-8'))
  except timeout:
    print("Failed to send data (timeout)")
  data = client.recv(BUFFER)
  print(f"Data received from server: {data.decode('utf-8')}")
  client.close()

print("Client finished.")
```

C – zrealizować timeout dla accept() korzystając z funkcji select(); zrealizować connect() w wersji nieblokującej.

Działanie:

Klient C:

```
blatosek@bigubu:~/psi_22l_KK_z22/PSI_2/C/Client_3$ ./run.sh
Setting default port to 9000
blatosek@bigubu:~/psi_22l_KK_z22/PSI_2/C/Client_3$ ./run.sh
^[[ASetting default port to 9000
blatosek@bigubu:~/psi_22l_KK_z22/PSI_2/C/Client_3$ ./run.sh
Setting default port to 9000
blatosek@bigubu:~/psi_22l_KK_z22/PSI_2/C/Client_3$
```

Serwer C:

```
blatosek@bigubu:~/psi_22l_KK_z22/PSI_2/C/Server_3$ ./run.sh
Setting default port to 9000
C Server listening on 172.21.22.5:9000
Message From TCP client: Message sent from c client
Message From TCP client: Message sent from c client
Message From TCP client: Message sent from c client
```

Kod źródłowy:

Server C:

```c
  if (bind(sockfd, (struct sockaddr *)&serveraddr, sizeof serveraddr) == -1){
    perror("BINDING ERROR");
    exit(1);
  }
  if (listen(sockfd, 5) != 0){
    perror("ERROR LISTENING");
    exit(1);
  }

  printf("C Server listening on 172.21.22.5:%d\n", portno);

  int i;
  while (1)
  {
    FD_ZERO(&ready);
    FD_SET(sockfd, &ready);
    for (i = 0; i < MAX_FDS; ++i) {
      if (socktab[i] > 0)
        FD_SET(socktab[i], &ready);
    }
    timeout.tv_sec = TIMEOUT_TIME;
    timeout.tv_usec = 0;

    if ((nactive = select(nfds, &ready, NULL, NULL, &timeout)) < -1) {
      perror("ERROR with select");
      continue;
    }

    if (FD_ISSET(sockfd, &ready)) {
      len = sizeof(clientaddr);
      msgsock = accept(sockfd, (struct sockaddr *)&clientaddr, &len);
      if ((childpid = fork()) == 0) {
        close(sockfd);
        memset(&buffer, 0, sizeof(buffer));
        printf("Message From TCP client: ");
        read(msgsock, buffer, sizeof(buffer));
        printf("%s\n", buffer);
        exit(0);
      }
      close(msgsock);
    }
    if (nactive == 0)
        printf("Timeout");
  }

  if (close(sockfd) < 0) {
      perror("ERROR closing socket");
```

Klient c:

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>

#define DATA "Message sent from c client"
#define HOST "172.21.22.5"
#define PORT 9000

int main(int argc, char *argv[]){
  int sockfd, portno;
  char buffer[1024];
  struct sockaddr_in serveraddr;

  if (argc < 3) {
      fprintf(stderr,"Setting default port to 9000\n");
      portno = PORT;
  }

  sockfd = socket(AF_INET, SOCK_STREAM, 0);
  if (sockfd < 0) {
        perror("ERROR while opening socket");
        exit(1);
  }
  memset(&serveraddr, 0, sizeof(serveraddr));

  serveraddr.sin_family = AF_INET;
  serveraddr.sin_addr.s_addr = inet_addr(HOST);
  serveraddr.sin_port = htons(portno);

  if (connect(sockfd, (struct sockaddr *)&serveraddr, sizeof(serveraddr)) < 0){
    perror("ERROR connecting to socket");
    exit(1);
  }

  strncpy(buffer, DATA, sizeof(DATA));

  if (send(sockfd, buffer, strlen(buffer), 0) < 0){
    perror("ERROR sending data");
    exit(1);
  }

  shutdown(sockfd,SHUT_WR);
  close(sockfd);

  return 0;
```