Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №6-8 по курсу «Операционные системы»

Студент: А. О. Дубинин
Преподаватель: Е. С. Миронов
Группа: М8О-206Б

Дата:
Оценка:
Подпись:

Москва, 2018

Лабораторная работа №6-8

Цель работы: Целью является приобретение практических навыков в:

Управлении серверами сообщений (№6)

Применение отложенных вычислений (№7)

Интеграция программных систем друг с другом (№8)

Задача: Реализовать клиент-серверную систему по асинхронной обработке запросов. Необходимо составить программы сервера и клиента. При запуске сервер и клиент должны быть настраиваемы, то есть должна быть возможность поднятия на одной ЭВМ нескольких серверов по обработке данных и нескольких клиентов, которые к ним от носятся. Все общение между процессами сервера и клиентов должно осуществляться через сервер сообщений.

Серверное приложение – банк. Клиентское приложение – клиент банка. Клиент может отправить какую-то денежную сумму в банк на хранения. Клиент также может запросить из банка произвольную сумму. Клиенты могут посылать суммы на счета других клиентов. Запросить собственный счет. При снятии должна производиться проверка на то, что у клиента достаточно денег для снятия денежных средств. Идентификатор клиента задается во время запуска клиентского приложения, как и адрес банка. Считать, что идентификаторы при запуске клиентов будут уникальными.

Вариант 13 (2-1-1):

Сервер сообщений: ZeroMQ

Внутренне хранилище: вектор

Тип ключа: Целое число 32 бита

Дополнительные возможности сервера: Сохранение данных о счетах клиентов при завершении работы сервера и возобновлении

1 Описание

Для реализации связи клиент-сервер был выбран паттерн Request-Response. Клиент отправляет запрос на сервер и ждет ответа. После того, как ответ пришел, клиент может продолжать работу. Клиент подключается к серверу, аутентифицирует клиента в базе, затем работает с сервером. Сервер обрабатывает запросы клиента, на вход принимает специально структурированное сообщение, после обаботки запроса, клиенту отправляется текстовая строка с отчётом. В случае если у клиента нет возможности получить деньги, отчёт-ответ дополняется соответсвующей причиной.

Встроена возможность отправки денег на счёт другому клиенту этого банка.

Если все клиенты банка отключились от сервера, то банк отключается и записывает все о клиентах в файл.

Использованные функции ZMQ:

int zmq_connect(void *socket, const char *endpoint); – подключает socket к пути endpoint

int zmq_bind(void *socket, const char *endpoint); – присоединяет socket к пути endpoint

void *zmq_ctx_new (); создает новый контекст

void *zmq_socket(void *context, int type); – создает сокет типа type из контекста context.

int zmq_msg_send(zmq_msg t *msg, void *socket, int flags); – отправляет сообщение msg в socket с параметрами flags, возвращает количество отправленных байт, в случае ошибки возвращает -1.

int zmq_msg_init(zmq_msg t *msg) – инициализирует сообщение msg как пустой объект.

int zmq_msg recv(zmq_msg t *msg, void *socket, int flags); – получает сообщение из socket в msg с параметрами flags, возвращает количество полученных байт, в случае ошибки возвращает -1.

int zmq_msg close(zmq_msg t *msg) – очищает содержимое msg

int zmq_close(void *socket); – закрывает сокет socket.

int zmq_ctx destroy(void *context); – разрушает контекст context, блокирует доступ всем операциям кроме zmq_close.

2 Исходный код

client.c

```
#include <zmq.h>
#include <stdio.h>
#include <assert.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <sys/mman.h>
#include <sys/file.h>
#include <unistd.h>
#include <sys/types.h>
#include "data.h"


void menu() {
  // printf("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n");
  // printf("1) Add cash to debit account \n");
  // printf("2) Withdraw cash \n");
  // printf("3) Account statistics \n");
  // printf("4) Send money \n");
  // printf("0) Exit \n");
  // printf("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n");
}

int main(int argc, char *argv[]) {

  Message message;
  int act = 0;
  char ans[256];

  void* context = zmq_ctx_new();
  void* server = zmq_socket(context, ZMQ_REQ);

  ans[0]='\0';
  strcat(ans,"tcp://localhost:");
  if(argc==2)
      strcat(ans,argv[1]);
    else
      strcat(ans,"4040");
  int rc = zmq_connect(server, ans);

  if (rc != 0) {
    perror("zmq_connect");
    zmq_close(server);
    zmq_ctx_destroy(context);
    exit(1);
  }
```

```
47
48    printf("Enter client id:\n");
49    scanf("%d", &message.clientId);
50    message.amount = 0;
51    message.receiverId = 0;
52    message.action = INIT;
53    zmq_msg_t clientReq;
54    zmq_msg_init_size(&clientReq, sizeof(Message));
55    memcpy(zmq_msg_data(&clientReq), &message, sizeof(Message));
56    zmq_msg_send(&clientReq, server, ZMQ_DONTWAIT);
57    zmq_msg_close(&clientReq);
58    zmq_msg_t reply;
59    zmq_msg_init(&reply);
60    zmq_msg_recv(&reply, server, 0);
61    strcpy(ans, (char*)zmq_msg_data(&reply));
62
63    if (strcmp(ans, "OK") == 0) {
64      printf("\nWelcome~\n");
65    } else if (strcmp(ans, "ERROR") == 0) {
66      printf("Sorry! Server returned error\n");
67      exit(1);
68    } else {
69      printf("Sorry! Bad server answer. Try again later\n");
70      exit(1);
71    }
72
73    zmq_msg_close(&reply);
74
75    for(;;) {
76      menu();
77      if (scanf("%d", &act) == EOF) {
78        act = 0;
79      }
80
81
82
83
84      if (act == 1) {
85        printf("Write sum you want to add: ");
86        scanf("%lld", &(message.amount));
87        message.action = ADD_DEBIT;
88        message.receiverId = 0;
89
90      } else if (act == 2) {
91        printf("Write sum you want to withdraw: ");
92        scanf("%lld", &(message.amount));
93        message.action = WTHDRAW_DEBIT;
94        message.receiverId = 0;
95
```

```
96      } else if (act == 3) {
97        message.amount = 0;
98        message.action = STAT;
99        message.receiverId = 0;
100
101     } else if (act == 4) {
102       printf("Write sum you want to send: ");
103       scanf("%lld", &(message.amount));
104       printf("Who you want to send to: ");
105       scanf("%d", &(message.receiverId));
106       message.action = SEND;
107
108     } else if (act == 0) {
109       message.amount = 0;
110       message.action = CLIENT_OFF;
111       message.receiverId = 0;
112       zmq_msg_init_size(&clientReq, sizeof(Message));
113       memcpy(zmq_msg_data(&clientReq), &message, sizeof(Message));
114
115       printf("Sending...\n");
116       zmq_msg_send(&clientReq, server, 0);
117       zmq_msg_close(&clientReq);
118
119       break;
120
121     } else {
122       printf("Try more...\n");
123       continue;
124     }
125
126     zmq_msg_init_size(&clientReq, sizeof(Message));
127     memcpy(zmq_msg_data(&clientReq), &message, sizeof(Message));
128
129     printf("Sending...\n");
130     zmq_msg_send(&clientReq, server, 0);
131     zmq_msg_close(&clientReq);
132     zmq_msg_init(&reply);
133     zmq_msg_recv(&reply, server, 0);
134
135     strcpy(ans, (char*)zmq_msg_data(&reply));
136     if (strcmp(ans, "ERROR") == 0) {
137         printf("Error occured\n");
138     } else {
139       printf("%s\n",ans);
140     }
141     zmq_msg_close(&reply);
142
143   }
144   zmq_close(server);
```

```
145   zmq_ctx_destroy(context);
146   return 0;
147 }
```

server.c

```
1  #include "vector.h"
2  #include <zmq.h>
3  #include <stdio.h>
4  #include <assert.h>
5  #include <unistd.h>
6  #include <string.h>
7  #include <stdlib.h>
8
9  int END = 0;
10
11 int main(int argc, char *argv[]) {
12   int countOfClients = 0;
13   Vector* vector = VectorCreate(10);
14   Message* message;
15   zmq_msg_t reply;
16   zmq_msg_t request;
17   char ans[256];
18   void* context = zmq_ctx_new();
19   void* responder = zmq_socket(context, ZMQ_REP);
20   ans[0]='\0';
21   strcat(ans,"tcp://*:");
22   char* port;
23   if(argc==2) {
24     port = argv[1];
25
26   }
27   else {
28     port = "4040";
29   }
30   strcat(ans,port);
31
32   FILE* fd;
33   fd = fopen(port, "rb");
34   if (fd != NULL)
35   {
36     while (1) {
37       int id;
38       long long money;
39       fread( &id, sizeof( int ), 1, fd );
40       if(id == END) {
41         break;
42       }
43       fread( &money, sizeof( long long ), 1, fd );
44       VectorAdd(vector, id);
```

```
45      CashToDebit(vector, money, id);
46    }
47  }
48
49
50  int rc = zmq_bind(responder, ans);
51
52  if (rc != 0) {
53  perror("zmq_bind");
54    zmq_close(responder);
55    zmq_ctx_destroy(context);
56  exit(1);
57  }
58
59
60  printf("Server initialized\n");
61  for(;;){
62    zmq_msg_init(&request);
63    zmq_msg_recv(&request, responder, 0);
64    message = (Message*) zmq_msg_data(&request);
65    printf("Recieved message from %d action: %d \n", message->clientId, message->action
          );
66
67    if (VectorSearch(vector, message->clientId)<0) {
68      //register new id
69      printf("Client %d added successfully\n",message->clientId);
70      VectorAdd(vector, message->clientId);
71
72    }
73
74    if (INIT == message->action) {
75      if(VectorSearch(vector, message->clientId)>=0) {
76            sprintf(ans, "OK");
77            countOfClients++;
78        }
79      else
80        sprintf(ans, "ERROR");
81
82    } else if (ADD_DEBIT == message->action) {
83      if(CashToDebit(vector, message->amount, message->clientId)==0)
84        sprintf(ans, "OK");
85      else
86        sprintf(ans, "ERROR");
87
88    } else if (WTHDRAW_DEBIT == message->action) {
89      int ret = Withdraw(vector, message->amount, message->clientId);
90      if(ret==0)
91        sprintf(ans, "OK");
92      else if(ret==1)
```

```
93          sprintf(ans, "OK: lack of funds withdrew from credit account");
94        else if(ret==2)
95          sprintf(ans, "Not enough money in the account");
96        else
97          sprintf(ans, "ERROR");
98
99      } else if (STAT == message->action) {
100       if(getStat(vector, message->clientId, ans)!=0)
101         sprintf(ans, "ERROR");
102
103     } else if (CLIENT_OFF == message->action) {
104       countOfClients--;
105       //printf("%d\n", countOfClients);
106       if(countOfClients<=0) {
107         printf("all clients are offline, server was down\n");
108         break;
109       }
110     } else if (SEND == message->action) {
111       if(VectorSearch(vector, message->receiverId)<0)
112         sprintf(ans, "No such receiver exist");
113       else {
114         int ret = Withdraw(vector, message->amount, message->clientId);
115         if(ret==0)
116           sprintf(ans, "OK");
117         else if(ret==1)
118           sprintf(ans, "OK: lack of funds withdrew from credit account");
119         else if(ret==2)
120           sprintf(ans, "Not enough money in the account");
121         else
122           sprintf(ans, "ERROR");
123
124         if(ret==1||ret==0)
125           if(CashToDebit(vector, message->amount, message->receiverId)!=0){
126             sprintf(ans, "ERROR");
127             //if error occured transaction failed
128             //return money back to sender
129             if(CashToDebit(vector, message->amount, message->clientId)!=0){
130               printf("FATAL ERROR: cannot revoke transaction\n ");
131               printf("\tcan not give back %lld moneys to %d\n ", message->amount,
                      message->clientId);
132               break;
133             }
134
135         }
136       }
137
138     } else {
139       sprintf(ans, "ERROR: Wrong request");
140     }
```

```
141
142       printf("Send answer to client: [%s]\n", ans);
143       zmq_msg_close(&request);
144       zmq_msg_init_size(&reply, strlen(ans)+1);
145       memcpy(zmq_msg_data(&reply), ans, strlen(ans)+1);
146       zmq_msg_send(&reply, responder, 0);
147       zmq_msg_close(&reply);
148     }
149
150     fd = fopen(port, "wb");
151     for ( int i = 0; i < vector->used; ++i ) {
152       fwrite( &vector->vector[i].id, sizeof( vector->vector[i].id ), 1, fd );
153       fwrite( &vector->vector[i].debit, sizeof( vector->vector[i].debit ), 1, fd );
154     }
155     fwrite( &END, sizeof( char ), 1, fd );
156
157     zmq_close(responder);
158     zmq_ctx_destroy(context);
159     VectorClean(vector);
160     return 0;
161 }
```

data.h

```
1  #ifndef DATA_H
2  #define DATA_H
3
4  typedef enum {
5      INIT, ADD_DEBIT,
6      WTHDRAW_DEBIT,
7      STAT, SEND, CLIENT_OFF
8  } RequestToken;
9
10 typedef struct {
11   RequestToken action;
12   int clientId;
13   int receiverId;
14   long long amount;
15 } Message ;
16
17 #endif
```

Makefile

```
1  all: client server
2  client:
3    gcc -std=c99 -Wall -D_POSIX_SOURCE=600 -D_XOPEN_SOURCE=600 client.c -o client -L/usr
         /local/lib -lzmq
4  server:
5    gcc -std=c99 -Wall -D_POSIX_SOURCE=600 server.c vector.c -o server -L/usr/local/lib
         -lzmq
```

```
6 clear:
7   rm server client
```

3 Тесты

Сервер

```
//параллельная работа разных серверов
art@mars:~/study/semester_3/OS/lab_6/code$ ./server 4041
Server initialized
Recieved message from 1 action: 0
Client 1 added successfully
Send answer to client: [OK]
Recieved message from 2 action: 0
Client 2 added successfully
Send answer to client: [OK]
Recieved message from 1 action: 1
Send answer to client: [OK]
Recieved message from 2 action: 3
Send answer to client: [Id: 2
Debit balace: 0
]
Recieved message from 1 action: 5
Send answer to client: [Id: 2
Debit balace: 0
]
Recieved message from 2 action: 5
all clients are offline,server was down
art@mars:~/study/semester_3/OS/lab_6/code$

////////////////////////////////
art@mars:~/study/semester_3/OS/lab_6/code$ ./server 4042
Server initialized
Recieved message from 1 action: 0
Client 1 added successfully
Send answer to client: [OK]
Recieved message from 1 action: 2
Send answer to client: [Not enough money in the account]
Recieved message from 1 action: 5
all clients are offline,server was down
art@mars:~/study/semester_3/OS/lab_6/code$
```

Клиент

```
//параллельная работа разных серверов
art@mars:~/workdir/OS/lab_6/code$ ./client 4041
Enter client id:
1

Welcome~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1) Add cash to debit account
2) Withdraw cash
3) Account statistics
4) Send money
0) Exit
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1
Write sum you want to add: 2
Sending...
OK
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1) Add cash to debit account
2) Withdraw cash
3) Account statistics
4) Send money
0) Exit
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
0
Sending...

////////////////////////////////////
art@mars:~/study/semester_3/OS/lab_6/code$ ./client 4041
Enter client id:
2

Welcome~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1) Add cash to debit account
2) Withdraw cash
3) Account statistics
4) Send money
0) Exit
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
3
```

```
Sending...
Id: 2
Debit balace: 0

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1) Add cash to debit account
2) Withdraw cash
3) Account statistics
4) Send money
0) Exit
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
0
Sending...
///////////////////////////////////
art@mars:~/study/semester_3/OS/lab_6/code$ ./client 4042
Enter client id:
1

Welcome~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1) Add cash to debit account
2) Withdraw cash
3) Account statistics
4) Send money
0) Exit
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
2
Write sum you want to withdraw: 78
Sending...
Not enough money in the account
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1) Add cash to debit account
2) Withdraw cash
3) Account statistics
4) Send money
0) Exit
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
0
Sending...
```

4 Диагностика strace

```
//Client:
...
read(3,"# Internet (IP) protocols## Up"...,4096) = 2932
read(3,"",4096)                         = 0
close(3)                                  = 0
eventfd2(0,EFD_CLOEXEC)                  = 3
fcntl(3,F_GETFL)                          = 0x2 (flags O_RDWR)
fcntl(3,F_SETFL,O_RDWR|O_NONBLOCK)     = 0
fcntl(3,F_GETFL)                          = 0x802 (flags O_RDWR|O_NONBLOCK)
fcntl(3,F_SETFL,O_RDWR|O_NONBLOCK)     = 0
getrandom("61630438435d35849e51281b3e",16,0) = 16
getrandom("6398044d506b7841f8160872b",16,0) = 16
eventfd2(0,EFD_CLOEXEC)                  = 4
fcntl(4,F_GETFL)                          = 0x2 (flags O_RDWR)
fcntl(4,F_SETFL,O_RDWR|O_NONBLOCK)     = 0
fcntl(4,F_GETFL)                          = 0x802 (flags O_RDWR|O_NONBLOCK)
fcntl(4,F_SETFL,O_RDWR|O_NONBLOCK)     = 0
epoll_create1(EPOLL_CLOEXEC)              = 5
epoll_ctl(5,EPOLL_CTL_ADD,4,0,u32=2250577952,u64=94641854945312) = 0
epoll_ctl(5,EPOLL_CTL_MOD,4,EPOLLIN,u32=2250577952,u64=94641854945312) = 0
mmap(NULL,8392704,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,-1,0) = 0x7f0fe303300
mprotect(0x7f0fe3034000,8388608,PROT_READ|PROT_WRITE) = 0
clone(child_stack=0x7f0fe3832b70,flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CL
= 14882
futex(0x7f0fe438207c,FUTEX_WAKE_PRIVATE,2147483647) = 0
futex(0x7f0fe4382088,FUTEX_WAKE_PRIVATE,2147483647) = 0
openat(AT_FDCWD,"/proc/self/task/14882/comm",O_RDWR) = 6
write(6,"ZMQbg/0",7)                    = 7
close(6)                                  = 0
eventfd2(0,EFD_CLOEXEC)                  = 6
fcntl(6,F_GETFL)                          = 0x2 (flags O_RDWR)
fcntl(6,F_SETFL,O_RDWR|O_NONBLOCK)     = 0
fcntl(6,F_GETFL)                          = 0x802 (flags O_RDWR|O_NONBLOCK)
fcntl(6,F_SETFL,O_RDWR|O_NONBLOCK)     = 0
epoll_create1(EPOLL_CLOEXEC)              = 7
epoll_ctl(7,EPOLL_CTL_ADD,6,0,u32=2250593392,u64=94641854960752) = 0
epoll_ctl(7,EPOLL_CTL_MOD,6,EPOLLIN,u32=2250593392,u64=94641854960752) = 0
mmap(NULL,8392704,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,-1,0) = 0x7f0fe283200
mprotect(0x7f0fe2833000,8388608,PROT_READ|PROT_WRITE) = 0
```

```
clone(child_stack=0x7f0fe3031b70,flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CL
= 14883
openat(AT_FDCWD,"/proc/self/task/14883/comm",O_RDWR) = 8
write(8,"ZMQbg/1",7)                    = 7
close(8)                                 = 0
eventfd2(0,EFD_CLOEXEC)                  = 8
fcntl(8,F_GETFL)                         = 0x2 (flags O_RDWR)
fcntl(8,F_SETFL,O_RDWR|O_NONBLOCK)     = 0
fcntl(8,F_GETFL)                         = 0x802 (flags O_RDWR|O_NONBLOCK)
fcntl(8,F_SETFL,O_RDWR|O_NONBLOCK)     = 0
poll([fd=8,events=POLLIN],1,0)      = 0 (Timeout)
brk(0x561386281000)                      = 0x561386281000
write(6,"",8)          = 8
write(8,"",8)          = 8
fstat(1,st_mode=S_IFCHR|0620,st_rdev=makedev(136,0),...) = 0
write(1,"Enter client id:",17Enter client id:
)      = 17
fstat(0,st_mode=S_IFCHR|0620,st_rdev=makedev(136,0),...) = 0
read(0,1
"1",1024)                     = 2
poll([fd=8,events=POLLIN],1,0)      = 1 ([fd=8,revents=POLLIN])
read(8,"",8)           = 8
poll([fd=8,events=POLLIN],1,0)      = 0 (Timeout)
write(6,"",8)          = 8
poll([fd=8,events=POLLIN],1,-1)     = 1 ([fd=8,revents=POLLIN])
read(8,"",8)           = 8
poll([fd=8,events=POLLIN],1,0)      = 0 (Timeout)
poll([fd=8,events=POLLIN],1,-1)     = 1 ([fd=8,revents=POLLIN])
read(8,"",8)           = 8
poll([fd=8,events=POLLIN],1,0)      = 0 (Timeout)
write(6,"",8)          = 8
write(1,"",1
)                        = 1
write(1,"Welcome~",9Welcome~
)               = 9
...
//server
...
read(3,"# Internet (IP) protocols## Up"...,4096) = 2932
read(3,"",4096)                          = 0
close(3)                                 = 0
```

```
eventfd2(0,EFD_CLOEXEC)                 = 3
fcntl(3,F_GETFL)                        = 0x2 (flags O_RDWR)
fcntl(3,F_SETFL,O_RDWR|O_NONBLOCK)    = 0
fcntl(3,F_GETFL)                        = 0x802 (flags O_RDWR|O_NONBLOCK)
fcntl(3,F_SETFL,O_RDWR|O_NONBLOCK)    = 0
getrandom("196c5a26351326b98929a21b72",16,0) = 16
getrandom("4b4e7b6672e5b0b13487f",16,0) = 16
eventfd2(0,EFD_CLOEXEC)                 = 4
fcntl(4,F_GETFL)                        = 0x2 (flags O_RDWR)
fcntl(4,F_SETFL,O_RDWR|O_NONBLOCK)    = 0
fcntl(4,F_GETFL)                        = 0x802 (flags O_RDWR|O_NONBLOCK)
fcntl(4,F_SETFL,O_RDWR|O_NONBLOCK)    = 0
epoll_create1(EPOLL_CLOEXEC)            = 5
epoll_ctl(5,EPOLL_CTL_ADD,4,0,u32=1554872560,u64=93928194668784) = 0
epoll_ctl(5,EPOLL_CTL_MOD,4,EPOLLIN,u32=1554872560,u64=93928194668784) = 0
mmap(NULL,8392704,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,-1,0) = 0x7f83bde41000
mprotect(0x7f83bde42000,8388608,PROT_READ|PROT_WRITE) = 0
clone(child_stack=0x7f83be640b70,flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLO
= 14899
futex(0x7f83bf19007c,FUTEX_WAKE_PRIVATE,2147483647) = 0
futex(0x7f83bf190088,FUTEX_WAKE_PRIVATE,2147483647) = 0
openat(AT_FDCWD,"/proc/self/task/14899/comm",O_RDWR) = 6
write(6,"ZMQbg/0",7)                   = 7
close(6)                                = 0
eventfd2(0,EFD_CLOEXEC)                 = 6
fcntl(6,F_GETFL)                        = 0x2 (flags O_RDWR)
fcntl(6,F_SETFL,O_RDWR|O_NONBLOCK)    = 0
fcntl(6,F_GETFL)                        = 0x802 (flags O_RDWR|O_NONBLOCK)
fcntl(6,F_SETFL,O_RDWR|O_NONBLOCK)    = 0
epoll_create1(EPOLL_CLOEXEC)            = 7
epoll_ctl(7,EPOLL_CTL_ADD,6,0,u32=1554888000,u64=93928194684224) = 0
epoll_ctl(7,EPOLL_CTL_MOD,6,EPOLLIN,u32=1554888000,u64=93928194684224) = 0
mmap(NULL,8392704,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,-1,0) = 0x7f83bd640000
mprotect(0x7f83bd641000,8388608,PROT_READ|PROT_WRITE) = 0
clone(child_stack=0x7f83bde3fb70,flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLO
= 14900
openat(AT_FDCWD,"/proc/self/task/14900/comm",O_RDWR) = 8
write(8,"ZMQbg/1",7)                   = 7
close(8)                                = 0
eventfd2(0,EFD_CLOEXEC)                 = 8
fcntl(8,F_GETFL)                        = 0x2 (flags O_RDWR)
```

```
fcntl(8,F_SETFL,O_RDWR|O_NONBLOCK)    = 0
fcntl(8,F_GETFL)                       = 0x802 (flags O_RDWR|O_NONBLOCK)
fcntl(8,F_SETFL,O_RDWR|O_NONBLOCK)    = 0
openat(AT_FDCWD,"4040",O_RDONLY)      = 9
fstat(9,st_mode=S_IFREG|0644,st_size=25,...) = 0
read(9,"F",4096) = 25
read(9,"",4096)                        = 0
poll([fd=8,events=POLLIN],1,0)     = 0 (Timeout)
socket(AF_INET,SOCK_STREAM|SOCK_CLOEXEC,IPPROTO_TCP) = 10
setsockopt(10,SOL_SOCKET,SO_REUSEADDR,[1],4) = 0
bind(10,sa_family=AF_INET,sin_port=htons(4040),sin_addr=inet_addr("0.0.0.0"),16)
= 0
listen(10,100)                          = 0
getsockname(10,sa_family=AF_INET,sin_port=htons(4040),sin_addr=inet_addr("0.0.0.0"),[1
= 0
write(6,"",8)         = 8
write(8,"",8)         = 8
fstat(1,st_mode=S_IFCHR|0620,st_rdev=makedev(136,1),...) = 0
write(1,"Server initialized",19Server initialized
)    = 19
poll([fd=8,events=POLLIN],1,-1)    = 1 ([fd=8,revents=POLLIN])
read(8,"",8)          = 8
poll([fd=8,events=POLLIN],1,0)     = 0 (Timeout)
poll([fd=8,events=POLLIN],1,-1)    = 1 ([fd=8,revents=POLLIN])
read(8,"",8)          = 8
poll([fd=8,events=POLLIN],1,0)     = 0 (Timeout)
poll([fd=8,events=POLLIN],1,-1)    = 1 ([fd=8,revents=POLLIN])
read(8,"",8)          = 8
write(6,"",8)         = 8
poll([fd=8,events=POLLIN],1,0)     = 0 (Timeout)
poll([fd=8,events=POLLIN],1,-1)    = 1 ([fd=8,revents=POLLIN])
read(8,"",8)          = 8
poll([fd=8,events=POLLIN],1,0)     = 0 (Timeout)
write(6,"",8)         = 8
write(1,"Recieved message from 1 action: "...,35Recieved message from 1 action:
0
) = 35
write(1,"Send answer to client: [OK]",28Send answer to client: [OK]
) = 28
...
```

5 Выводы

Выполнив лабораторную работу, я приобрёл практические навыки в управлении серверами сообщений, приминил отложенные вычисления, интегрировал программные системы друг с другом . Самой главной проблемой получившейся системы является база клиентов: она содержится целиком в оперативной памяти. Также реалзация вектора, который из себя представляет база, основан на принципе: если закончилась память – аллоцируем в два раза больше, что при большом потоке клиентов, может создать излишнюю нагрузку. Реализованная концепция Request-Response очень проста в понимании и удобная для создания подобных программ. Использование очередей собощений для меня было замечательным опытом, применение готового и удобного интерфейса, очень простой в прочтении и легкий в написании код, который в дальнейшем легко поддерживать и улучшать.