Московский авиационный институт
(Национальный исследовательский университет)

Факультет: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»

Отчет по курсовой работе
по курсу "Операционные системы"

Студент: Дюсекеев А.Е.

Группы: М8О-204Б-17

Преподаватель: Миронов Е.С.

Оценка

Москва 2019

Цель:

- Приобретение практических навыков в использовании знаний, полученных в течении курса
- Проведение исследования в выбранной предметной области

Задание:

Необходимо спроектировать и реализовать программный прототип в соответствии с выбранным вариантом. Произвести анализ и сделать вывод на основании данных, полученных при работе программного прототипа.

Вариант задания:

Создание клиента для передачи мгновенных личных сообщений
На основе любой из выбранных технологий:

- Pipes
- Sockets
- Сервера очередей
- И другие

Создать собственный клиент быстрых сообщений (возможно, и сервер – зависит от выбранной архитектуры), который бы работали в рамках сети. Также есть возможность создания собственного клиента быстрых сообщений для уже существующих систем обмена сообщений (например, telegram).

Для написания программы выбрано : ZeroMQ .

Введение:

Я начал пользоваться мессенджерами только в 2012 году (VK), тогда я еще не понимал и не хотел понять , как работает данный сайт , в том числе , как осуществляется отправка сообщений . Казалось , что это простой процесс , но курс операционных систем показал , как все сложно может быть устроено .

Отправка сообщений может происходить по-разному :

- peer-to-peer .В этом случае сервер нужен только для того, что бы люди находили друг друга. Ну или можно исхитриться и сервер тогда вообще не нужен. В этом случае клиенты соединяются друг к другу и шлют сообщения напрямую.
- обычная модель обмена сообщений через общий сервер. Самый простой вариант. Все соединяются к одному серверу, который выступает как маршрутизатор сообщений. У каждого сообщения есть идентификатор откуда и куда оно его нужно доставить . Далее происходит довольно понятный процесс пересылки сообщений .

Далее в разборе решения данной задачи я покажу , как я реализовал свой мессенджер с помощью ZeroMQ .

Решение:

В программе клиента создается пустой контекст , затем сокет , который соединяется к удаленному адресу , который по умолчанию имеет tcp порт : 4029 . Еще до того , как пользователь начинает вводить сообщение , процесс клиента посылает свой идентификатор процесса серверу , чтобы он внес его в базу данных . Это понадобится далее .

Далее происходит чтение запроса от пользователя , которое отправляется на сервер с помощью zmq_msg_send . Далее сервер принимает сообщение и посылает сигналы прерывания с помощью kill(pid,SIGINT) всем процессам за исключением процесса , который отправил данное сообщение . Это сообщение сохраняется в переменную структуры people . Затем все процессы обращаются к серверу после получения сигнала с помощью функции signal , чтобы прочитать последнее сообщение lastmsg и напечатать его в стандартный поток вывода в процессах клиентов .

В результате , когда кто-то отправляет сообщение в сервер , сервер посылает сигналы всем клиентам , и у каждого печатается соответствующие сообщение .

Клиент может выйти из чата с помощью печати и отправки серверу сообщения 'exit' после чего программа клиента завершается , а в списке клиентов происходит удаление по ключу - идентификатору клиента .

Сервер прекращает свою работу , если в голове списка , который хранит информацию о клиентах , будет равен NULL .

В результате получился довольно неплохой мессенджер , который использует библиотеку ZeroMQ и один из его паттернов - REQUEST - REPLY .

Сценарий выполнения работы:

Между двумя людьми :

```
alisher@Duglas-VirtualBox ~/Desktop/os/kursach $ ./server
waiting for message
number of people ++
waiting for message
id : 111 ; name : Nick ; message : hi
waiting for message
number of people ++
waiting for message
waiting for message
id : 110 ; name : John ; message : hi,Nick
waiting for message
waiting for message
id : 111 ; name : Nick ; message : HI!)
waiting for message
waiting for message
id : 111 ; name : Nick ; message : how are you?
```

```
waiting for message
waiting for message
id : 110 ; name : John ; message : fine and you?
waiting for message
waiting for message
id : 111 ; name : Nick ; message : me too
waiting for message
waiting for message
id : 110 ; name : John ; message : by
waiting for message
waiting for message
id : 111 ; name : Nick ; message : by
waiting for message
waiting for message
id : 111 ; name : Nick ; message : exit
waiting for message
waiting for message
id : 110 ; name : John ; message : exit
```

```
alisher@Duglas-VirtualBox ~/Desktop/os/kursach $ ./client 111 Nick
hello!,this is an instant messenger,
you can exit it by sending 'exit'
hi
John joined the conversation
John | hi,Nick
HI!)
how are you?
John | fine and you?
me too
John | by
by
exit
```

```
alisher@Duglas-VirtualBox ~/Desktop/os/kursach $ ./client 110 John
hello!,this is an instant messenger,
you can exit it by sending 'exit'
hi,Nick
Nick | HI!)
Nick | how are you?
fine and you?
Nick | me too
by
Nick | by
Nick exit from chat
exit
```

Между тремя людьми :

```
alisher@Duglas-VirtualBox ~/Desktop/os/kursach $ ./server
waiting for message
number of people ++
waiting for message
number of people ++
waiting for message
waiting for message
```

```
number of people ++
waiting for message
waiting for message
waiting for message
id : 100 ; name : Anna ; message : hi guys!
waiting for message
waiting for message
waiting for message
id : 111 ; name : Nick ; message : hello
waiting for message
waiting for message
waiting for message
id : 110 ; name : John ; message : hi
waiting for message
waiting for message
waiting for message
id : 110 ; name : John ; message : exit
waiting for message
waiting for message
waiting for message
id : 111 ; name : Nick ; message : exit
waiting for message
waiting for message
id : 100 ; name : Anna ; message : exit
```

```
alisher@Duglas-VirtualBox ~/Desktop/os/kursach $ ./client 111 Nick
hello!,this is an instant messenger,
you can exit it by sending 'exit'
Anna | hi guys!
hello
John | hi
John exit from chat
exit
alisher@Duglas-VirtualBox ~/Desktop/os/kursach $
```

```
alisher@Duglas-VirtualBox ~/Desktop/os/kursach $ ./client 110 John
hello!,this is an instant messenger,
you can exit it by sending 'exit'
Nick joined the conversation
Anna | hi guys!
Nick | hello
hi
exit
alisher@Duglas-VirtualBox ~/Desktop/os/kursach $
```

```
alisher@Duglas-VirtualBox ~/Desktop/os/kursach $ ./client 100 Anna
hello!,this is an instant messenger,
you can exit it by sending 'exit'
John joined the conversation
Nick joined the conversation
hi guys!
Nick | hello
John | hi
John exit from chat
Nick exit from chat
exit
```

Листинг программы :

CLIENT.C

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>
#include <zmq.h>
#include <signal.h>

#include "message.h"

int pid;
char text[100];
void* req;

void usage(){
      printf("hello!,this is an instant messenger,\n");
      printf("you can exit it by sending \'exit\' \n");
}
void send_recv(Message *mes)
{
    zmq_msg_t request;

    zmq_msg_init_size(&request, sizeof(Message));
    memcpy(zmq_msg_data(&request), mes, sizeof(Message));
    zmq_msg_send(&request, mes->requester, 0);
    zmq_msg_close(&request);

      zmq_msg_init(&request);
    zmq_msg_recv(&request, mes->requester, 0);
    mes = (Message*)zmq_msg_data(&request);
    zmq_msg_close(&request);
      if(mes->action==1){
            strcpy(text,mes->text);
      }
}

void printmsg(){
      Message mes;
      mes.action=1;
      mes.requester=req;
      send_recv(&mes);
      if(strcmp(text,"exit"))
            printf("%s\n",text);
}

int main (int argc, char *argv[])
{
      if(argc!=3){
            printf("enter ./client id name\n");
            return 0;
      }

    Message mes;
      void* context = zmq_ctx_new();
```

```
      if(context == NULL)
            return 0;
      void* request = zmq_socket(context, ZMQ_REQ);
      if(request == NULL)
            return 0;
      usage();
    zmq_connect(request,"tcp://localhost:4029");//create socket here
      mes.pid=getpid();
      mes.requester=request;
      req=request;
      mes.action=-1;
    mes.id=atoi(argv[1]);
    strcpy(mes.name,argv[2]);
      send_recv(&mes);
      signal(SIGINT,printmsg);
    do{
        gets(mes.text);
        mes.id=atoi(argv[1]);
            mes.action=0;
        mes.requester=request;
        send_recv(&mes);
    }
    while(strcmp(mes.text,"exit"));
    mes.id=atoi(argv[1]);
      zmq_close(request);
      zmq_ctx_destroy(context);

      return 0;
}
```

SERVER.C

```
#include <string.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <signal.h>

#include <time.h>
#include <sys/ioctl.h>
#include <stropts.h>

#include "zmq.h"
#include "message.h"
#include "people.h"

volatile sig_atomic_t flag = 0;

void quit_server_func()
{
      printf("\nserver quit\n");
      flag = 3;
}

int main (int argc, char *argv[])
{
    People *list=NULL;
      zmq_msg_t request;
    zmq_msg_t reply;
      void* context = zmq_ctx_new();
      if(context == NULL)
```

```
            return 0;
    Message *mes;
      void* respond = zmq_socket(context, ZMQ_REP);
      if(respond == NULL)
            return 0;

      if(zmq_bind(respond, "tcp://*:4029")==-1)
                return 0;
      signal(SIGINT,quit_server_func);
      while(flag!=3){
        printf("waiting for message\n");
            zmq_msg_init(&request);
            zmq_msg_recv(&request, respond, 0);
            mes=(Message*)zmq_msg_data(&request);
            zmq_msg_close(&request);
        if(mes->action==-1){
            insert_client(&list,mes->id,mes->pid,mes->name);
            zmq_msg_init_size(&reply,sizeof(Message));
            memcpy(zmq_msg_data(&reply),mes,sizeof(Message));
                zmq_msg_send(&reply, respond, 0);
                zmq_msg_close(&reply);
                sprintf(list->lastmsg,"%s joined the conversation",mes-
>name);
            printf("number of people ++\n");
                People *tmp = list;
                while(tmp!=NULL){
                    if(tmp->pid!=mes->pid){
                        kill(tmp->pid,SIGINT);}
                    tmp=tmp->next;
                    if(tmp==NULL)break;
                }
        }
            else if(mes->action==1){
                  zmq_msg_init_size(&reply,sizeof(Message));
                  strcpy(mes->text,list->lastmsg);
            memcpy(zmq_msg_data(&reply),mes,sizeof(Message));
                zmq_msg_send(&reply, respond, 0);
                zmq_msg_close(&reply);
            }
        else{
                if(!find(&list,mes->id))
                    insert_client(&list,mes->id,mes->pid,mes->name);
                  if(!strcmp(mes->text,"exit"))
                  sprintf(list->lastmsg,"%s %s from chat ",mes->name,mes-
>text);
                  else
                  sprintf(list->lastmsg,"%s | %s ",mes->name,mes->text);
            printf("id : %d ; name : %s ; message : %s \n",mes->id,mes-
>name,mes->text);
                People *tmp = list;
                while(tmp!=NULL){
                    if(tmp->pid!=mes->pid){
                        kill(tmp->pid,SIGINT);}
                    tmp=tmp->next;
                    if(tmp==NULL)break;
                }
                zmq_msg_init_size(&reply,sizeof(Message));
                memcpy(zmq_msg_data(&reply),mes,sizeof(Message));
                  zmq_msg_send(&reply, respond, 0);
                  zmq_msg_close(&reply);
                if(!strcmp(mes->text,"exit")){
```

```
                    delete_client(&list,mes->id);
                    if(list==NULL)
                        break;
                }
            }
        }
        zmq_close(respond);
        zmq_ctx_destroy(context);

        return 0;
}
```

PEOPLE.C

```
#include "people.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>


People* find(People **head,int id){
    People *prev,*tmp=*head;
    if(*head==NULL){
        return NULL;
    }
    while(tmp!=NULL){
        prev=tmp;
        if(tmp->id==id){
            return tmp;
        }
        if(tmp!=NULL)
        tmp=tmp->next;
    }
    return NULL;
}
void insert_client(People **head,int id,pid_t pid,char *name){
    People *tmp=*head;
    People *prev;
    if(*head==NULL){
        (*head)=(People*)malloc(sizeof(People));
        (*head)->id=id;
        (*head)->pid=pid;
        strcpy((*head)->name,name);
        (*head)->next=NULL;
        return;
    }
    while(tmp!=NULL){
        if(tmp->id==id){
            return;
        }
        prev=tmp;
        tmp=tmp->next;
    }
    if(tmp==NULL){
        prev->next=(People*)malloc(sizeof(People));
        prev->next->id=id;
        prev->next->pid=pid;
        strcpy(prev->next->name,name);
        prev->next->next=NULL;
    }

    return;
```

```
}


void delete_client(People **head,int id){
      People *prev=*head,*tmp=*head,*tmp2=*head;
      while(tmp!=NULL){
            if(tmp->id==id)
                  break;
            prev=tmp;
            tmp=tmp->next;
      }
      if(prev==tmp){
        (*head)=(*head)->next;
        free(tmp);
        if(*head==NULL)
            return;
      }
      else{
            prev->next=tmp->next;
            free(tmp);
      }
    strcpy((*head)->lastmsg,tmp2->lastmsg);
}
```

Вывод:

При реализации программы мессенджера не были использованы многие системные вызовы, поскольку не нашел им применение . Но знания и опыт пригодились . Например необходимо было понять , как будет происходить взаимодействие между клиентами за счет сервера : по сути это обычные процессы , в которых создаются сокеты и происходит взаимодействие между программами за счет функций библиотеки zmq.h . Но без второй лабораторной работы было бы очень трудно понять , как процессы вообще могут общаться . Так же и другие задачи помогли накопить опыт в подобном роде программировании .

Мой мессенджер довольно прост , возможно можно было по-другому его реализовать , но мне кажется , что данный метод - самый простой . Его можно также переделать под программу рассылки новостей от сервера , который будет посылать каждому процессу сигналы , чтобы они приняли сообщение с сервера . Это , например , как ежемесячная рассылка новостей от какого-то издательства . Да и вообще с помощью ZeroMQ можно реализовать множество идей .

В целом курс операционных систем был очень полезным . Поскольку , все что я делал было увлекательным и отложилось в памяти .