Московский авиационный институт (национальный исследовательский университет)

Факультет прикладной математики и физики

Кафедра вычислительной математики и программирования

Курсовой проект по курсу «Операционные системы»

Студент: Т.А.Габдуллин
Преподаватель: Е. С. Миронов
Группа: 80-206Б

Дата:
Оценка:
Подпись:

Москва, 2017-2018

Описание

В качестве курсового проекта представлен сервер обмена сообщениями на основе очереди сообщения, который находится в стандарте POSIX5. Доступный функционал — авторизация на сервере, передача сообщения адресанту, проверка непрочитанных личных сообщений. При написании сообщения не авторизованному пользователю выдается варн.

2 Исходный код

dtaabase.h

```
#ifndef DATABASE
#define DATABASE
#include <stdio.h>
#include <sys/un.h>
#include <sys/mman.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include "config.h"
typedef struct field field;
struct field
{
   stored_message* msg;
   field* next;
   field* prev;
};
typedef struct
{
   field* begin;
   field* end;
   size_t size;
   bool flag;
}database;
void* create_shared_memory(size_t);
database* create_list();
bool add(database* list, stored_message* msg);
field* find(database* list, char* name);
bool purge(database* list, field* fld);
void print_list(database* list);
#endif
```

reciever.c

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
```

```
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include "config.h"
#include "database.h"
int main()
{
   int sock, listener;
   struct sockaddr_in addr;
   listener = socket(AF_INET, SOCK_STREAM, 0);
   if(listener < 0)
   {
      perror("socket");
      exit(1);
   }

   addr.sin_family = AF_INET;
   addr.sin_port = htons(3425);
   addr.sin_addr.s_addr = htonl(INADDR_ANY);

   if(bind(listener, (struct sockaddr *)&addr, sizeof(addr)) < 0)
   {
      perror("bind");
      exit(2);
   }
   stored_message* message = (stored_message*) malloc(sizeof(stored_message)); // buffer for
message
   listen(listener, MAX_QUEUE_LEN);
   database* list = create_list();
   while(1)
   {
      sock = accept(listener, NULL, NULL);
      if(sock < 0)
      {
         perror("accept");
         exit(3);
      }
      recv(sock, (stored_message*)message, sizeof(stored_message), 0);

      if(message->type == _msg)
      {
         if(add(list, message))
         {
            message->result = true;
         }
         else message->result = false;
      }
      else if(message->type == _request)
      {
         print_list(list);
         field* tmp = find(list, message->sender);
```

```
            if(tmp == NULL) message->result = false;
            else
            {
                strcpy(message->msg, tmp->msg->msg);
                strcpy(message->recipient, tmp->msg->sender);
                message->result = true;
                purge(list, tmp);
            }
        }
        send(sock, message, sizeof(stored_message), 0);
        close(sock);
    }
    close(listener);

    return 0;
}
```

sender.c

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <stdbool.h>
#include "config.h"
int main()
{
    int sock;
    struct sockaddr_in addr;
    stored_message* message = (stored_message*) malloc(sizeof(stored_message)); // buffer for
message
    printf("Enter your name: ");
    scanf("%s", message->sender);
    while(1)
    {
        char command;
        printf("What you want?\n");
        printf("m - compose and send message to someone\n");
        printf("r - refresh letter box. May be someone just texted to you?\n");
        printf("q - exit from client\n");
        scanf("%s", &command);
        if(command == 'm') // write message
        {
            message->type = _msg; //simple message
            printf("To :");
            scanf("%s", message->recipient);
            printf("Your message: ");
            scanf("%s", message->msg);
        }
        else if(command == 'r') //request for mail
```

```
{
   message->type = _request; // request
}
else if(command == 'q')
{
   free(message);
   exit(0);
}
else
{
   printf("Undefined command\n");
   continue;
}
A:;
sock = socket(AF_INET, SOCK_STREAM, 0);
if(sock < 0)
{
   perror("socket");
   exit(1);
}
addr.sin_family = AF_INET;
addr.sin_port = htons(3425); //порт...
addr.sin_addr.s_addr = htonl(INADDR_LOOPBACK);

if(connect(sock, (struct sockaddr *)&addr, sizeof(addr)) < 0)
{
   perror("connect");
   exit(2);
}

send(sock, message, sizeof(stored_message), 0);
recv(sock, message, sizeof(stored_message), 0);

if(message->type == _msg)
{
   if(message->result) printf("Sent!\n");
   else
   {
      printf("Try again later...\n");
      sleep(2);
      goto A;
   }
}
else if(message->type == _request)
{
   if(message->result)
   {
      printf("You have message from %s:", message->recipient);
      printf("%s\n", message->msg);
   }
   else printf("Letter box is empty\n");
}
```

```
        else
        {
            printf("Undefined command\n");
            exit(-1);
        }
        close(sock);
    }
    return 0;
}
```

Config.h

```
#ifndef CONFIG
#define CONFIG
#define MAX_QUEUE_LEN 256
#define MSG_SIZE 1028
#include <stdbool.h>
enum message_type
{
    _quit = 0,
    _msg,
    _request
};
typedef struct // struct of message
{
    char sender[14];
    char recipient[14];
    char msg[MSG_SIZE];
    bool result;
    int type;
} stored_message;
#endif
```

3 Консоль

server

ttimxag@KEKNOTE:~/Документы/kp$ ls
config.h config.h.gch database.c database.h database.h.gch reciever.c sender1.c sender.c test.c
timxag@KEKNOTE:~/Документы/kp$ gcc -o client sender.c database.c config.h
timxag@KEKNOTE:~/Документы/kp$ gcc -o server reciever.c database.c config.h
timxag@KEKNOTE:~/Документы/kp$ ./server
---

recipient is Timur
sender is NeTimur
message is test
---

client1

ttimxag@KEKNOTE:~/Документы/kp$ ./client
Enter your name: NeTimur
What you want?
m - compose and send message to someone
r - refresh letter box. May be someone just texted to you?
q - exit from client
m
To :Timur
Your message: test
Sent!
What you want?
m - compose and send message to someone
r - refresh letter box. May be someone just texted to you?
q - exit from client

client2

timxag@KEKNOTE:~/Документы/kp$ ./client
Enter your name: Timur
What you want?
m - compose and send message to someone
r - refresh letter box. May be someone just texted to you?
q - exit from client
r
You have message from NeTimur:test
What you want?
m - compose and send message to someone
r - refresh letter box. May be someone just texted to you?
q - exit from client

4 Вывод

Мной была написана программа, имитирующая мессенджер сообщений. Конечный продукт имеет довольно широкий функционал, позволяющий, к примеру, авторизация на сервере, передача сообщения адресанту, проверка непрочитанных личных сообщений.Это был очень занимательный опыт в моей жизни, я уверен, что в дальнейшем опыт написания программ с библиотеками обмена сообщений пригодится мне для будущих проектов.