Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

# Курсовой проект по курсу

# «Операционные системы»

Группа: М8О-209БВ-24

Студент: Артонкин В.Н.

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 20.12.25

Москва, 2025

# Постановка задачи

**Вариант 27.**

Создать собственный клиент быстрых сообщений (возможно, и сервер – зависит от выбранной архитектуры: pipes/sockets/брокеры сообщений), который бы работал в рамках сети. Необходимо предусмотреть возможность хранения истории переписок (на сервере) и поиска по ним. Связь между сервером и клиентом должна быть реализована при помощи memory map. Необходимо предусмотреть возможность хранения истории переписок (на сервере) и поиска по ним. Связь между сервером и клиентом должна быть реализована при помощи memory map

# Общий метод и алгоритм решения

**Использованные системные вызовы:**

- shm_open(const char *name, int oflag, mode_t mode) - создаёт или открывает объект POSIX разделяемой памяти, доступный нескольким процессам.
- ftruncate(int fd, off_t length) - изменяет размер объекта разделяемой памяти до заданного значения.
- mmap(void *addr, size_t length, int prot, int flags, int fd, off_t offset) - отображает объект разделяемой памяти в виртуальное адресное пространство процесса.
- munmap(void *addr, size_t length) - удаляет отображение разделяемой памяти из адресного пространства процесса.
- sem_init(sem_t *sem, int pshared, unsigned int value) - инициализирует POSIX-семафор, размещённый в разделяемой памяти.
- sem_wait(sem_t *sem) - уменьшает значение семафора и при необходимости блокирует процесс до освобождения ресурса.
- sem_post(sem_t *sem) - увеличивает значение семафора и разблокирует ожидающие процессы.
- sleep(unsigned int seconds) - приостанавливает выполнение процесса на заданное количество секунд.
- usleep(useconds_t usec) - приостанавливает выполнение потока на заданное количество микросекунд.
- pthread_create(pthread_t *thread, const pthread_attr_t *attr, void *(*start_routine)(void *), void *arg) - создаёт новый поток выполнения внутри процесса клиента.
- exit(int status) - завершает выполнение процесса с заданным кодом возврата.

**Алгоритм работы**

Общая архитектура

Программа состоит из двух независимых процессов:

- серверного приложения;
- клиентского приложения.

Сначала запускается сервер, после чего может быть запущено произвольное количество клиентов. Все клиенты взаимодействуют с сервером через разделяемую память, отображённую в адресное пространство процессов (memory mapping).

Сервер выполняет инфраструктурную роль:

- создаёт и инициализирует разделяемую область памяти;
- хранит список активных клиентов;
- хранит историю всех сообщений;
- обеспечивает синхронизированный доступ к данным.

Клиентские приложения:

- регистрируются на сервере по логину;
- записывают сообщения в общую память;
- читают сообщения из общей памяти в реальном времени;
- выполняют поиск и просмотр истории сообщений.

Алгоритм работы сервера

1. Сервер создаёт объект POSIX shared memory с помощь shm_open.
2. Сервер задаёт размер разделяемой памяти, равный размеру общей структуры данных.
3. Сервер отображает разделяемую память в своё адресное пространство с помощью mmap с флагом MAP_SHARED.
4. Сервер инициализирует структуру общей памяти:
    - список клиентов;
    - массив сообщений;
    - счётчик сообщений.
5. Сервер инициализирует POSIX-семафор, размещённый в разделяемой памяти, для синхронизации доступа.
6. После инициализации сервер переходит в бесконечный цикл ожидания.

7. Сервер не обрабатывает пользовательский ввод и не выполняет активную обработку сообщений, а обеспечивает корректное существование и доступность общей памяти.
8. Сервер продолжает работу до принудительного завершения.

Алгоритм работы клиента

1. Клиент запускается и запрашивает у пользователя логин.
2. Клиент открывает ранее созданную сервером разделяемую память с помощью shm_open.
3. Клиент отображает разделяемую память в своё адресное пространство с помощью mmap.
4. Клиент захватывает семафор и регистрируется в массиве клиентов:
    - сохраняет свой логин;
    - помечает себя как активного;
    - сохраняет индекс последнего прочитанного сообщения.
5. Клиент запускает отдельный поток, который:
    - периодически проверяет наличие новых сообщений;
    - отображает входящие сообщения в реальном времени.
6. Основной поток клиента отображает консольное меню, позволяющее:
    - отправить сообщение другому клиенту по логину;
    - выполнить поиск сообщений по ключевому слову;
    - просмотреть все входящие сообщения за всё время;
    - завершить работу клиента.
7. При отправке сообщения клиент:
    - захватывает семафор;
    - добавляет новое сообщение в массив сообщений в разделяемой памяти;
    - увеличивает счётчик сообщений;
    - освобождает семафор.
8. При поиске или просмотре истории клиент:
    - захватывает семафор;
    - выполняет линейный просмотр массива сообщений;
    - отображает подходящие сообщения;

- освобождает семафор.

9. При завершении работы клиент помечает себя как неактивного и корректно завершает процесс.

Особенности взаимодействия процессов

- Все процессы работают с одной и той же областью памяти, отображённой с флагом MAP_SHARED.
- Передача сообщений между клиентами осуществляется путём записи и чтения структур данных в разделяемой памяти.
- Для предотвращения гонок данных используется POSIX-семафор, общий для всех процессов.

# Код программы

## common.h

```
#ifndef COMMON_H

#define COMMON_H

#include <semaphore.h>

#define SHM_NAME "/chat_shm"

#define MAX_CLIENTS 10

#define MAX_MESSAGES 500

#define LOGIN_LEN 32

#define TEXT_LEN 256

typedef struct {

char from[LOGIN_LEN];

char to[LOGIN_LEN];

char text[TEXT_LEN];

} Message;

typedef struct {

char login[LOGIN_LEN];
```

```
int active;
int last_read;
} Client;
typedef struct {
Client clients[MAX_CLIENTS];
Message messages[MAX_MESSAGES];
int message_count;
sem_t mutex;
} SharedData;
#endif
```

**server.c**

```
#include <stdio.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <unistd.h>
#include <string.h>
#include <semaphore.h>
#include "common.h"
int main() {
int fd = shm_open(SHM_NAME, O_CREAT | O_RDWR, 0666);
if (fd < 0) {
perror("shm_open");
return 1;
}
ftruncate(fd, sizeof(SharedData));
SharedData *data = mmap(NULL, sizeof(SharedData),
PROT_READ | PROT_WRITE,
MAP_SHARED, fd, 0);
```

```
if (data == MAP_FAILED) {
perror("mmap");
return 1;
}
memset(data, 0, sizeof(SharedData));
sem_init(&data->mutex, 1, 1);
printf("=== Chat server started ===\n");
while (1) {
sleep(1);
}
}
```

## client.c

```
#include <stdio.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <unistd.h>
#include <string.h>
#include <pthread.h>
#include <semaphore.h>
#include <stdlib.h>
#include "common.h"
SharedData *data;
char my_login[LOGIN_LEN];
int my_index;
void clear_screen() {
printf("\033[2J\033[H");
}
void header() {
```

```
printf("====================================\n");
printf(" IPC CHAT | user: %s\n", my_login);
printf("====================================\n");
}
/* ===== Receiver thread ===== */
void *receiver(void *arg) {
while (1) {
sem_wait(&data->mutex);
while (data->clients[my_index].last_read < data->message_count) {
int i = data->clients[my_index].last_read;
Message *msg = &data->messages[i];
if (strcmp(msg->to, my_login) == 0) {
printf("\n[%s]: %s\n", msg->from, msg->text);
}
data->clients[my_index].last_read++;
}
sem_post(&data->mutex);
usleep(200000);
}
return NULL;
}
void show_all_incoming() {
clear_screen();
header();
printf("All incoming messages:\n\n");
sem_wait(&data->mutex);
int found = 0;
for (int i = 0; i < data->message_count; i++) {
```

```
Message *msg = &data->messages[i];
if (strcmp(msg->to, my_login) == 0) {
printf("[%s]: %s\n", msg->from, msg->text);
found = 1;
}
}
sem_post(&data->mutex);
if (!found) {
printf("No messages.\n");
}
printf("\nPress Enter...");
getchar();
}
void search_messages() {
char key[TEXT_LEN];
printf("Keyword: ");
fgets(key, TEXT_LEN, stdin);
key[strcspn(key, "\n")] = 0;
clear_screen();
header();
printf("Search results:\n\n");
sem_wait(&data->mutex);
int found = 0;
for (int i = 0; i < data->message_count; i++) {
Message *msg = &data->messages[i];
if ((strcmp(msg->to, my_login) == 0 ||
strcmp(msg->from, my_login) == 0) &&
strstr(msg->text, key)) {
```

```
printf("[%s -> %s]: %s\n",
msg->from, msg->to, msg->text);
found = 1;
}
}
sem_post(&data->mutex);
if (!found) printf("Nothing found.\n");
printf("\nPress Enter...");
getchar();
}
int main() {
clear_screen();
printf("Login: ");
scanf("%31s", my_login);
getchar();
int fd = shm_open(SHM_NAME, O_RDWR, 0666);
if (fd < 0) {
perror("shm_open");
return 1;
}
data = mmap(NULL, sizeof(SharedData),
PROT_READ | PROT_WRITE,
MAP_SHARED, fd, 0);
sem_wait(&data->mutex);
my_index = -1;
for (int i = 0; i < MAX_CLIENTS; i++) {
if (!data->clients[i].active) {
strcpy(data->clients[i].login, my_login);
```

```
data->clients[i].active = 1;
data->clients[i].last_read = data->message_count;
my_index = i;
break;
}
}
sem_post(&data->mutex);
if (my_index == -1) {
printf("Server full\n");
return 1;
}
pthread_t t;
pthread_create(&t, NULL, receiver, NULL);
while (1) {
clear_screen();
header();
printf("1) Send message\n");
printf("2) Search messages\n");
printf("3) Show all incoming\n");
printf("4) Exit\n");
printf("Choose: ");
int c;
scanf("%d", &c);
getchar();
if (c == 1) {
char to[LOGIN_LEN], text[TEXT_LEN];
printf("To: ");
scanf("%31s", to);
```

```
getchar();
printf("Message: ");
fgets(text, TEXT_LEN, stdin);
text[strcspn(text, "\n")] = 0;
sem_wait(&data->mutex);
if (data->message_count < MAX_MESSAGES) {
Message *m = &data->messages[data->message_count++];
strcpy(m->from, my_login);
strcpy(m->to, to);
strcpy(m->text, text);
}
sem_post(&data->mutex);
} else if (c == 2) {
search_messages();
} else if (c == 3) {
show_all_incoming();
} else if (c == 4) {
sem_wait(&data->mutex);
data->clients[my_index].active = 0;
sem_post(&data->mutex);
exit(0);
}
}
}
```

## Протокол работы программы

```
strace -o server_trace.txt -f ./server
=== Chat server started ===
```

$ strace -e 'trace=!clock_nanosleep' -o client1_trace.txt ./client

Login: lenka

==========

IPC CHAT | user: lenka

==========

1) Send message

2) Search messages

3) Show all incoming

4) Exit

Choose:

[tolik]: privet krasotka

3

==========

IPC CHAT | user: lenka

==========

All incoming messages:

[tolik]: privet krasotka

Press Enter...

==========

IPC CHAT | user: lenka

==========

1) Send message

2) Search messages

3) Show all incoming

4) Exit

Choose: 1

To: tolik

Message: не понимаю по-английски

==========

IPC CHAT | user: lenka

==========

1) Send message

2) Search messages

3) Show all incoming

4) Exit

Choose: 2

Keyword: не

==========

IPC CHAT | user: lenka

==========

Search results:

[lenka -> tolik]: не понимаю по-английски

Press Enter...^C

execve("./client", ["./client"], 0x7ffe7fbaeb70 /* 34 vars */) = 0

brk(NULL) = 0x56436d31f000

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f3b02ba7000

access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

fstat(3, {st_mode=S_IFREG|0644, st_size=11939, ...}) = 0

mmap(NULL, 11939, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f3b02ba4000

close(3) = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\243\2\0\0\0\0\0"..., 832) = 832

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

**mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f3b02992000**

**mmap(0x7f3b029ba000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7f3b029ba000**

**mmap(0x7f3b02b42000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000) = 0x7f3b02b42000**

**mmap(0x7f3b02b91000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x7f3b02b91000**

**mmap(0x7f3b02b97000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f3b02b97000**

close(3) = 0

**mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f3b0298f000**

arch_prctl(ARCH_SET_FS, 0x7f3b0298f740) = 0

set_tid_address(0x7f3b0298fa10) = 32875

set_robust_list(0x7f3b0298fa20, 24) = 0

rseq(0x7f3b02990060, 0x20, 0, 0x53053053) = 0

mprotect(0x7f3b02b91000, 16384, PROT_READ) = 0

mprotect(0x564367d40000, 4096, PROT_READ) = 0

mprotect(0x7f3b02bdf000, 8192, PROT_READ) = 0

prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

munmap(0x7f3b02ba4000, 11939) = 0

fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...}) = 0

getrandom("\xbe\x4e\x87\x2f\x4e\x98\x07\x51", 8, GRND_NONBLOCK) = 8

brk(NULL) = 0x56436d31f000

brk(0x56436d340000) = 0x56436d340000

fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...}) = 0

write(1, "\33[2J\33[HLogin: ", 14) = 14

read(0, "lenka\n", 1024) = 6

openat(AT_FDCWD, "/dev/shm/chat_shm", O_RDWR|O_NOFOLLOW|O_CLOEXEC) = 3

mmap(NULL, 160440, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f3b02967000

rt_sigaction(SIGRT_1, {sa_handler=0x7f3b02a2b530, sa_mask=[], sa_flags=SA_RESTORER|SA_ONSTACK|SA_RESTART|SA_SIGINFO, sa_restorer=0x7f3b029d7330}, NULL, 8) = 0

rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f3b02166000

mprotect(0x7f3b02167000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

**clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_C**

**LEARTID, child_tid=0x7f3b02966990, parent_tid=0x7f3b02966990, exit_signal=0, stack=0x7f3b02166000, stack_size=0x7fff80, tls=0x7f3b029666c0}, 88) = -1 ENOSYS (Function not implemented)**

**clone(child_stack=0x7f3b02965f70, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[33073], tls=0x7f3b029666c0, child_tidptr=0x7f3b02966990) = 33073**

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

write(1, "\33[2J\33[H==========="..., 45) = 45

write(1, " IPC CHAT | user: lenka\n", 24) = 24

write(1, "==========="..., 38) = 38

write(1, "1) Send message\n", 16)  = 16

write(1, "2) Search messages\n", 19)  = 19

write(1, "3) Show all incoming\n", 21)  = 21

write(1, "4) Exit\n", 8)  = 8

write(1, "Choose: ", 8)  = 8

read(0, "3\n", 1024)  = 2

write(1, "\33[2J\33[H==========="..., 45) = 45

write(1, " IPC CHAT | user: lenka\n", 24) = 24

write(1, "==========="..., 38) = 38

write(1, "All incoming messages:\n", 23) = 23

write(1, "\n", 1)  = 1

write(1, "[tolik]: privet krasotka\n", 25) = 25

write(1, "\n", 1)  = 1

write(1, "Press Enter...", 14)  = 14

read(0, "\n", 1024)  = 1

write(1, "\33[2J\33[H==========="..., 45) = 45

write(1, " IPC CHAT | user: lenka\n", 24) = 24

write(1, "==========="..., 38) = 38

write(1, "1) Send message\n", 16)  = 16

write(1, "2) Search messages\n", 19)  = 19

write(1, "3) Show all incoming\n", 21)  = 21

write(1, "4) Exit\n", 8)  = 8

write(1, "Choose: ", 8)  = 8

read(0, "1\n", 1024)  = 2

write(1, "To: ", 4)                    = 4

read(0, "tolik\n", 1024)               = 6

write(1, "Message: ", 9)               = 9

read(0, "\320\275\320\265 \320\277\320\276\320\275\320\270\320\274\320\260\321\216 \320\277\320\276-\320\260\320\275\320\263\320"..., 1024) = 44

write(1, "\33[2J\33[H=========================="..., 45) = 45

write(1, " IPC CHAT | user: lenka\n", 24) = 24

write(1, "================================"..., 38) = 38

write(1, "1) Send message\n", 16)       = 16

write(1, "2) Search messages\n", 19)    = 19

write(1, "3) Show all incoming\n", 21)  = 21

write(1, "4) Exit\n", 8)               = 8

write(1, "Choose: ", 8)                = 8

read(0, "2\n", 1024)                   = 2

write(1, "Keyword: ", 9)               = 9

read(0, "\320\275\320\265\n", 1024)    = 5

write(1, "\33[2J\33[H=========================="..., 45) = 45

write(1, " IPC CHAT | user: lenka\n", 24) = 24

write(1, "================================"..., 38) = 38

write(1, "Search results:\n", 16)       = 16

write(1, "\n", 1)                      = 1

write(1, "[lenka -> tolik]: \320\275\320\265 \320\277\320\276\320\275\320\270\320"..., 62) = 62

write(1, "\n", 1)                      = 1

write(1, "Press Enter...", 14)         = 14

read(0, 0x56436d31f6b0, 1024)          = ? ERESTARTSYS (To be restarted if SA_RESTART is set)

--- SIGINT {si_signo=SIGINT, si_code=SI_KERNEL} ---

+++ killed by SIGINT +++

$ strace -e 'trace=!clock_nanosleep' -o client2_trac

e.txt ./client

Login: tolik

```
======================================
 IPC CHAT | user: tolik
======================================
1) Send message
2) Search messages
3) Show all incoming
4) Exit
Choose: 1
To: lenka
Message: privet krasotka
======================================
 IPC CHAT | user: tolik
======================================
1) Send message
2) Search messages
3) Show all incoming
4) Exit
Choose:
[lenka]: не понимаю по-английски
2
Keyword: не
======================================
 IPC CHAT | user: tolik
======================================
Search results:

[lenka -> tolik]: не понимаю по-английски

Press Enter...
======================================
 IPC CHAT | user: tolik
======================================
1) Send message
2) Search messages
```

3) Show all incoming

4) Exit

Choose: ^C

execve("./client", ["./client"], 0x7ffeddc47a60 /* 34 vars */) = 0

brk(NULL) = 0x558c799c6000

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f302048b000

access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

fstat(3, {st_mode=S_IFREG|0644, st_size=11939, ...}) = 0

mmap(NULL, 11939, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f3020488000

close(3) = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\243\2\0\0\0\0\0"..., 832) = 832

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

**mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f3020276000**

**mmap(0x7f302029e000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7f302029e000**

**mmap(0x7f3020426000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000) = 0x7f3020426000**

**mmap(0x7f3020475000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x7f3020475000**

**mmap(0x7f302047b000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f302047b000**

close(3) = 0

**mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f3020273000**

arch_prctl(ARCH_SET_FS, 0x7f3020273740) = 0

set_tid_address(0x7f3020273a10) = 32984

set_robust_list(0x7f3020273a20, 24) = 0

rseq(0x7f3020274060, 0x20, 0, 0x53053053) = 0

mprotect(0x7f3020475000, 16384, PROT_READ) = 0

mprotect(0x558c76af4000, 4096, PROT_READ) = 0

mprotect(0x7f30204c3000, 8192, PROT_READ) = 0

prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

munmap(0x7f3020488000, 11939) = 0

fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x2), ...}) = 0

getrandom("\xf1\x78\x6f\x44\x98\xb5\x78\x0f", 8, GRND_NONBLOCK) = 8

brk(NULL) = 0x558c799c6000

brk(0x558c799e7000) = 0x558c799e7000

fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x2), ...}) = 0

write(1, "\33[2J\33[HLogin: ", 14) = 14

read(0, "tolik\n", 1024) = 6

openat(AT_FDCWD, "/dev/shm/chat_shm", O_RDWR|O_NOFOLLOW|O_CLOEXEC) = 3

mmap(NULL, 160440, PROT_READ|PROT_WRITE, MAP_SHARED, 3, 0) = 0x7f302024b000

rt_sigaction(SIGRT_1, {sa_handler=0x7f302030f530, sa_mask=[], sa_flags=SA_RESTORER|SA_ONSTACK|SA_RESTART|SA_SIGINFO, sa_restorer=0x7f30202bb330}, NULL, 8) = 0

rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f301fa4a000

mprotect(0x7f301fa4b000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

**clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f302024a990, parent_tid=0x7f302024a990, exit_signal=0, stack=0x7f301fa4a000, stack_size=0x7fff80, tls=0x7f302024a6c0}, 88) = -1 ENOSYS (Function not implemented)**

**clone(child_stack=0x7f3020249f70, flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, parent_tid=[33130], tls=0x7f302024a6c0, child_tidptr=0x7f302024a990) = 33130**

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

write(1, "\33[2J\33[H==========="..., 45) = 45

write(1, " IPC CHAT | user: tolik\n", 24) = 24

write(1, "================================"..., 38) = 38

write(1, "1) Send message\n", 16) = 16

write(1, "2) Search messages\n", 19) = 19

```
write(1, "3) Show all incoming\n", 21)  = 21
write(1, "4) Exit\n", 8)               = 8
write(1, "Choose: ", 8)                = 8
read(0, "1\n", 1024)                   = 2
write(1, "To: ", 4)                    = 4
read(0, "lenka\n", 1024)                = 6
write(1, "Message: ", 9)                = 9
read(0, "privet krasotka\n", 1024)      = 16
write(1, "\33[2J\33[H=========================="..., 45) = 45
write(1, " IPC CHAT | user: tolik\n", 24) = 24
write(1, "================================="..., 38) = 38
write(1, "1) Send message\n", 16)       = 16
write(1, "2) Search messages\n", 19)    = 19
write(1, "3) Show all incoming\n", 21)  = 21
write(1, "4) Exit\n", 8)               = 8
write(1, "Choose: ", 8)                = 8
read(0, "2\n", 1024)                   = 2
write(1, "Keyword: ", 9)                = 9
read(0, "\320\275\320\265\n", 1024)     = 5
write(1, "\33[2J\33[H=========================="..., 45) = 45
write(1, " IPC CHAT | user: tolik\n", 24) = 24
write(1, "================================="..., 38) = 38
write(1, "Search results:\n", 16)       = 16
write(1, "\n", 1)                      = 1
write(1, "[lenka -> tolik]: \320\275\320\265 \320\277\320\276\320\275\320\270\320"..., 62) = 62
write(1, "\n", 1)                      = 1
write(1, "Press Enter...", 14)         = 14
read(0, "\n", 1024)                    = 1
write(1, "\33[2J\33[H=========================="..., 45) = 45
write(1, " IPC CHAT | user: tolik\n", 24) = 24
write(1, "================================="..., 38) = 38
write(1, "1) Send message\n", 16)       = 16
write(1, "2) Search messages\n", 19)    = 19
write(1, "3) Show all incoming\n", 21)  = 21
```

write(1, "4) Exit\n", 8)  = 8

write(1, "Choose: ", 8)  = 8

read(0, 0x558c799c66b0, 1024)  = ? ERESTARTSYS (To be restarted if SA_RESTART is set)

--- SIGINT {si_signo=SIGINT, si_code=SI_KERNEL} ---

+++ killed by SIGINT +++

## Вывод

В результате выполнения лабораторной работы была реализована клиент-серверная консольная система мгновенного обмена сообщениями с поддержкой нескольких пользователей. Для межпроцессного взаимодействия была использована POSIX shared memory с отображением в адресное пространство процессов через mmap и синхронизация доступа с помощью POSIX-семафоров. Реализация полностью соответствует требованиям задания и демонстрирует практическое применение системных вызовов операционной системы для организации межпроцессного взаимодействия и совместного доступа к памяти.