МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

Курсовая работа №1
по курсу «Операционные системы»

Выполнил: Н. В. Храмов
Группа: М8О-208БВ-24
Преподаватель: Е. С. Миронов
Дата сдачи: 25.12.2025

Москва, 2025

Условие задачи

Цель работы: разработать клиент-серверную систему передачи мгновенных сообщений с использованием разделяемой памяти.

Задание:

Реализовать клиент-серверное приложение со следующим обязательным функционалом:

- Клиент может присоединиться к серверу, указав логин.
- Клиент может отправить личное сообщение другому клиенту по его логину.
- Клиент получает сообщения от других пользователей в реальном времени.
- На сервере должна храниться история всех переписок.
- Должен быть реализован поиск по истории переписки.
- Связь между сервером и клиентами должна быть реализована исключительно с помощью механизма отображения файлов в память (memory map, POSIX shared memory).

Вариант: 27

Метод решения

Для решения задачи выбрана архитектура на основе POSIX shared memory с использованием единого объекта разделяемой памяти, к которому подключаются один сервер и произвольное количество клиентов. Все процессы отображают в своё адресное пространство один и тот же файл объекта памяти с именем `/simple_chat_shm_v1`.

В разделяемой памяти размещена структура `SharedData`, содержащая:

- мьютекс и условную переменную с атрибутом `PTHREAD_PROCESS_SHARED`;
- массив до 16 зарегистрированных логинов;
- кольцевой буфер на 1024 сообщения;
- текущий индекс записи `msg_write_index` и флаг завершения работы сервера.

Отправка сообщения выполняется атомарно: клиент захватывает мьютекс, записывает сообщение в следующую ячейку буфера, увеличивает индекс и вызывает `pthread_cond_broadcast()`. Доставка в реальном времени реализована через отдельный поток в каждом клиенте: он ждёт на условной переменной, при пробуждении сканирует весь буфер, выводит новые сообщения для своего логина и помечает их как доставленные.

Хранение истории и поиск выполняются только на сервере. Поток-монитор сервера сохраняет непомеченные сообщения в файл `history.txt`. Поиск доступен через серверную команду `search`.

Архитектура работает исключительно в пределах одной машины, но обеспечивает минимальную задержку и простоту реализации.

Описание программы

Программа состоит из трёх файлов:

`common.h` содержит общие определения:

- константы `MAX_CLIENTS = 16`, `MAX_MESSAGES = 1024`, длины полей;
- структуру `Message` (отправитель, получатель, текст, timestamp, флаги `saved` и `delivered`);
- структуру `SharedData` — полное описание разделяемой памяти.

`server.cpp` реализует сервер:

- создаёт/открывает объект разделяемой памяти (`shm_open`, `ftruncate`, `mmap`);
- инициализирует мьютекс и условную переменную для межпроцессного использования;
- запускает поток, сохраняющий историю в `history.txt`;
- предоставляет консольные команды `list_clients`, `search <user1> <user2> [keyword]`, `exit`;
- обрабатывает SIGINT для корректного завершения.

`client.cpp` реализует клиент:

- подключается к существующей разделяемой памяти (`shm_open`, `mmap`);
- регистрирует введённый логин;
- создаёт отдельный поток для приёма сообщений;
- в основном потоке обрабатывает команды `send <login> <text>` и `exit`.

Основные использованные системные вызовы и функции:

- `shm_open()`, `ftruncate()`, `mmap()`, `munmap()`
- `pthread_mutexattr_setpshared()`, `pthread_condattr_setpshared()`
- `pthread_mutex_lock/unlock()`, `pthread_cond_wait()`, `pthread_cond_broadcast()`
- `sigaction()` для обработки SIGINT

Сетевая подсистема не используется — вся коммуникация осуществляется только через разделяемую память.

Результаты

Разработана клиент-серверная система мгновенных сообщений, полностью соответствующая требованиям задания. Связь между сервером и клиентами реализована исключительно через механизм разделяемой памяти POSIX (`shm_open` + `mmap`).

Ключевые особенности решения:

- Сервер создаёт (при первом запуске) или открывает объект разделяемой памяти `/simple_chat_shm_v1`, содержащий единую структуру `SharedData` с мьютексом и условной переменной (`PTHREAD_PROCESS_SHARED`), списком до 16 клиентов, кольцевым буфером на 1024 сообщения и служебными полями.
- Клиент подключается к существующей разделяемой памяти, вводит логин и регистрирует его в общем массиве клиентов.
- Отправка сообщения выполняется по логину получателя: клиент записывает сообщение в кольцевой буфер, обновляет `msg_write_index` и вызывает `pthread_cond_broadcast`.
- Приём сообщений в реальном времени обеспечивает отдельный поток в каждом клиенте: он ожидает на условной переменной, при пробуждении проверяет весь буфер, выводит новые сообщения для своего логина и помечает их как доставленные (`delivered = 1`).
- История переписки сохраняется на сервере в файле `history.txt`. Отдельный серверный поток при каждом срабатывании условной переменной записывает ещё не сохранённые сообщения в читаемом формате с меткой времени.
- Поиск по истории реализован через серверную команду `search <user1> <user2> [keyword]`. Команда выводит сообщения между указанными пользователями в обоих направлениях с возможностью фильтрации по ключевому слову.
- Корректное завершение работы по Ctrl+C и командам `quit`/`exit`: сервер выставляет флаг `server_stopping`, делает `pthread_cond_broadcast`, клиенты и серверные потоки завершаются чисто.

Все пункты задания выполнены полностью. Система работает под Linux, обеспечивает мгновенную доставку сообщений и постоянное хранение полной истории переписки.

Выводы

В ходе курсовой работы полностью решена поставленная задача. Реализована простая клиент-серверная система мгновенных сообщений с использованием POSIX shared memory (shm_open + mmap) и межпроцессной синхронизации средствами pthread. Созданы отдельные процессы — сервер и клиент, обменивающиеся сообщениями через общую область памяти. Организована регистрация клиентов по логину и хранение таблицы клиентов в общей памяти. Реализована отправка сообщений по логину и приём сообщений в реальном времени на стороне клиента. Сервер запускает фоновый поток (pthreads), который сохраняет новые сообщения в файл истории `history.txt`. В серверном CLI реализованы команды просмотра списка клиентов и поиска по истории переписок. Обеспечен корректный механизм завершения работы: обработка сигнала `SIGINT`, установка флага `server_stopping` в общей памяти и оповещение клиентов через `pthread_cond_broadcast`, вследствие чего клиенты корректно завершают работу. Проект собирается с помощью CMake и корректно работает под POSIX (проверено под Linux).

Исходный код программы

Листинг 1: common.h — общие структуры данных и константы

```
#ifndef COMMON_H
#define COMMON_H

#include <pthread.h>
#include <cstdint>

const char *SHM_NAME = "/simple_chat_shm_v1";
const size_t MAX_CLIENTS = 16;
const size_t LOGIN_LEN = 32;
const size_t MAX_MESSAGES = 1024;
const size_t MESSAGE_TEXT_LEN = 256;

struct Message
{
    char sender[LOGIN_LEN];
    char recipient[LOGIN_LEN];
    char text[MESSAGE_TEXT_LEN];
    uint64_t timestamp;
    int saved;
    int delivered;
};

struct SharedData
{
    pthread_mutex_t mutex;
    pthread_cond_t cond;
    char clients[MAX_CLIENTS][LOGIN_LEN];
    Message messages[MAX_MESSAGES];
    int msg_write_index;
    int server_stopping;
};

#endif
```

Листинг 2: server.cpp — реализация сервера чата

```
#include "Server.h"
#include <sys/mman.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <cstring>
#include <iostream>
#include <fstream>
#include <ctime>
#include <pthread.h>
#include <sstream>
#include <signal.h>

size_t SHM_SIZE = sizeof(SharedData);

Server *Server::instance = nullptr;
```

```
17
18  void server_signal_handler(int sig)
19  {
20      if (Server::instance)
21      {
22          Server::instance->stop_requested = 1;
23      }
24  }
25
26  Server::Server() : data(nullptr), shm_fd(-1), stop_requested(0)
27  {
28      instance = this;
29
30      bool created = false;
31      int temp_fd = shm_open(SHM_NAME, O_RDWR, 0600);
32      if (temp_fd < 0)
33      {
34          created = true;
35      }
36      else
37      {
38          close(temp_fd);
39      }
40      init_shared_memory(created);
41  }
42
43  Server::~Server()
44  {
45      if (data)
46      {
47          munmap(data, SHM_SIZE);
48      }
49      if (shm_fd >= 0)
50      {
51          close(shm_fd);
52          shm_unlink(SHM_NAME);
53      }
54  }
55
56  void Server::init_shared_memory(bool create_new)
57  {
58      if (create_new)
59      {
60          shm_fd = shm_open(SHM_NAME, O_CREAT | O_EXCL | O_RDWR, 0600);
61          if (shm_fd < 0)
62          {
63              perror("shm_open create");
64              exit(1);
65          }
66          if (ftruncate(shm_fd, SHM_SIZE) == -1)
67          {
68              perror("ftruncate");
69              exit(1);
70          }
71      }
72      else
73      {
74          shm_fd = shm_open(SHM_NAME, O_RDWR, 0600);
```

```
75          if (shm_fd < 0)
76          {
77              perror("shm_open open");
78              exit(1);
79          }
80      }
81
82      void *ptr = mmap(nullptr, SHM_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, shm_fd, 0)
            ;
83      if (ptr == MAP_FAILED)
84      {
85          perror("mmap");
86          exit(1);
87      }
88      data = reinterpret_cast<SharedData *>(ptr);
89
90      if (create_new)
91      {
92          memset(data, 0, SHM_SIZE);
93          pthread_mutexattr_t mattr;
94          pthread_condattr_t cattr;
95          pthread_mutexattr_init(&mattr);
96          pthread_mutexattr_setpshared(&mattr, PTHREAD_PROCESS_SHARED);
97          pthread_mutex_init(&data->mutex, &mattr);
98          pthread_mutexattr_destroy(&mattr);
99
100         pthread_condattr_init(&cattr);
101         pthread_condattr_setpshared(&cattr, PTHREAD_PROCESS_SHARED);
102         pthread_cond_init(&data->cond, &cattr);
103         pthread_condattr_destroy(&cattr);
104
105         data->msg_write_index = 0;
106         data->server_stopping = 0;
107     }
108 }
109
110 void Server::append_history(const Message &m)
111 {
112     std::ofstream ofs("history.txt", std::ios::app);
113     if (!ofs)
114         return;
115
116     std::time_t t = (time_t)m.timestamp;
117     char buf[64];
118     std::strftime(buf, sizeof(buf), "%Y-%m-%d %H:%M:%S", std::localtime(&t));
119     ofs << "[" << buf << "] " << m.sender << " -> " << m.recipient << ": " << m.text
            << "\n";
120 }
121
122 void *Server::monitor_and_save(void *arg)
123 {
124     Server *self = static_cast<Server *>(arg);
125     while (!self->stop_requested)
126     {
127         pthread_mutex_lock(&self->data->mutex);
128         pthread_cond_wait(&self->data->cond, &self->data->mutex);
129
130         if (self->stop_requested || self->data->server_stopping)
```

```
131         {
132             pthread_mutex_unlock(&self->data->mutex);
133             break;
134         }
135 
136         for (int i = 0; i < (int)MAX_MESSAGES; ++i)
137         {
138             if (self->data->messages[i].timestamp != 0 && self->data->messages[i].saved
                    == 0)
139             {
140                 self->append_history(self->data->messages[i]);
141                 self->data->messages[i].saved = 1;
142             }
143         }
144         pthread_mutex_unlock(&self->data->mutex);
145     }
146     return nullptr;
147 }
148 
149 void Server::cli_loop()
150 {
151     std::string cmd;
152     while (!stop_requested)
153     {
154         std::cout << "server> ";
155         if (!std::getline(std::cin, cmd))
156         {
157             stop_requested = 1;
158             break;
159         }
160         if (stop_requested)
161             break;
162 
163         if (cmd == "quit" || cmd == "exit")
164         {
165             break;
166         }
167         if (cmd == "list_clients")
168         {
169             pthread_mutex_lock(&data->mutex);
170             std::cout << "Clients:\n";
171             for (size_t i = 0; i < MAX_CLIENTS; ++i)
172             {
173                 if (data->clients[i][0] != '\0')
174                 {
175                     std::cout << "- " << data->clients[i] << "\n";
176                 }
177             }
178             pthread_mutex_unlock(&data->mutex);
179         }
180         else if (cmd.rfind("search ", 0) == 0)
181         {
182             std::istringstream iss(cmd);
183             std::string tok, a, b, keyword;
184             iss >> tok >> a >> b;
185             std::getline(iss, keyword);
186             if (!keyword.empty() && keyword[0] == ' ')
187                 keyword.erase(0, 1);
```

```
188 
189             std::ifstream ifs("history.txt");
190             if (!ifs)
191             {
192                 std::cout << "No history file.\n";
193                 continue;
194             }
195             std::string line;
196             while (std::getline(ifs, line))
197             {
198                 bool users_ok = (line.find(" " + a + " -> " + b + ":") != std::string::
                        npos) ||
199                                 (line.find(" " + b + " -> " + a + ":") != std::string::
                                     npos);
200                 bool keyword_ok = keyword.empty() || (line.find(keyword) != std::string
                        ::npos);
201                 if (users_ok && keyword_ok)
202                 {
203                     std::cout << line << "\n";
204                 }
205             }
206         }
207         else if (cmd == "help")
208         {
209             std::cout << "Commands:\n list_clients\n search <user1> <user2> [keyword]\n
                     exit\n";
210         }
211         else
212         {
213             std::cout << "Unknown command. Type help.\n";
214         }
215     }
216 }
217 
218 void Server::run()
219 {
220     if (pthread_create(&monitor_thread, nullptr, &Server::monitor_and_save, this) !=
            0)
221     {
222         perror("pthread_create");
223         return;
224     }
225 
226     cli_loop();
227 
228     stop_requested = 1;
229     pthread_mutex_lock(&data->mutex);
230     data->server_stopping = 1;
231     pthread_cond_broadcast(&data->cond);
232     pthread_mutex_unlock(&data->mutex);
233 
234     pthread_join(monitor_thread, nullptr);
235     std::cout << "Server shutting down...\n";
236 }
237 
238 int main()
239 {
240     struct sigaction sa;
```

```
    sa.sa_handler = server_signal_handler;
    sigemptyset(&sa.sa_mask);
    sa.sa_flags = 0;
    sigaction(SIGINT, &sa, nullptr);

    std::cout << "Starting server...\n";
    Server s;
    s.run();
    return 0;
}
```

Листинг 3: client.cpp — реализация клиента чата

```
#include "Client.h"
#include <sys/mman.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <cstring>
#include <iostream>
#include <pthread.h>
#include <ctime>
#include <signal.h>

size_t SHM_SIZE = sizeof(SharedData);

Client *Client::instance = nullptr;

void client_signal_handler(int sig)
{
    if (Client::instance)
    {
        Client::instance->stop_requested = 1;
    }
}

Client::Client() : data(nullptr), shm_fd(-1), stop_requested(0)
{
    instance = this;
}

Client::~Client()
{
    if (data)
    {
        munmap(data, SHM_SIZE);
    }
    if (shm_fd >= 0)
    {
        close(shm_fd);
    }
}

void Client::connect()
{
    shm_fd = shm_open(SHM_NAME, O_RDWR, 0600);
    if (shm_fd < 0)
```

```
45      {
46          perror("shm_open (client) - make sure server is running");
47          exit(1);
48      }
49
50      void *ptr = mmap(nullptr, SHM_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, shm_fd, 0)
            ;
51      if (ptr == MAP_FAILED)
52      {
53          perror("mmap (client)");
54          exit(1);
55      }
56      data = reinterpret_cast<SharedData *>(ptr);
57  }
58
59  bool Client::registerLogin(const std::string &l)
60  {
61      pthread_mutex_lock(&data->mutex);
62      if (data->server_stopping)
63      {
64          pthread_mutex_unlock(&data->mutex);
65          std::cerr << "Server is shutting down. Cannot register.\n";
66          return false;
67      }
68
69      for (size_t i = 0; i < MAX_CLIENTS; ++i)
70      {
71          if (std::string(data->clients[i]) == l)
72          {
73              pthread_mutex_unlock(&data->mutex);
74              login = l;
75              return true;
76          }
77      }
78
79      for (size_t i = 0; i < MAX_CLIENTS; ++i)
80      {
81          if (data->clients[i][0] == '\0')
82          {
83              strncpy(data->clients[i], l.c_str(), LOGIN_LEN - 1);
84              data->clients[i][LOGIN_LEN - 1] = '\0';
85              pthread_mutex_unlock(&data->mutex);
86              login = l;
87              return true;
88          }
89      }
90
91      pthread_mutex_unlock(&data->mutex);
92      return false;
93  }
94
95  void *Client::receiver_loop(void *arg)
96  {
97      Client *self = static_cast<Client *>(arg);
98      while (!self->stop_requested)
99      {
100         pthread_mutex_lock(&self->data->mutex);
101         pthread_cond_wait(&self->data->cond, &self->data->mutex);
```

```
102 
103         if (self->data->server_stopping)
104         {
105             pthread_mutex_unlock(&self->data->mutex);
106             std::cout << "\nServer is shutting down. Exiting client receiver.\n";
107             break;
108         }
109 
110         for (int i = 0; i < (int)MAX_MESSAGES; ++i)
111         {
112             if (self->data->messages[i].timestamp != 0 && self->data->messages[i].
                    delivered == 0)
113             {
114                 if (self->login == std::string(self->data->messages[i].recipient))
115                 {
116                     std::time_t t = (time_t)self->data->messages[i].timestamp;
117                     char buf[64];
118                     std::strftime(buf, sizeof(buf), "%Y-%m-%d %H:%M:%S", std::localtime
                            (&t));
119                     std::cout << "\n[" << buf << "] " << self->data->messages[i].sender
120                               << ": " << self->data->messages[i].text << "\n> ";
121                     self->data->messages[i].delivered = 1;
122                 }
123             }
124         }
125         pthread_mutex_unlock(&self->data->mutex);
126     }
127     return nullptr;
128 }
129 
130 void Client::send_message(const std::string &to, const std::string &text)
131 {
132     pthread_mutex_lock(&data->mutex);
133     if (data->server_stopping)
134     {
135         pthread_mutex_unlock(&data->mutex);
136         std::cout << "Server is shutting down; cannot send message.\n";
137         return;
138     }
139 
140     int idx = data->msg_write_index;
141     strncpy(data->messages[idx].sender, login.c_str(), LOGIN_LEN - 1);
142     data->messages[idx].sender[LOGIN_LEN - 1] = '\0';
143     strncpy(data->messages[idx].recipient, to.c_str(), LOGIN_LEN - 1);
144     data->messages[idx].recipient[LOGIN_LEN - 1] = '\0';
145     strncpy(data->messages[idx].text, text.c_str(), MESSAGE_TEXT_LEN - 1);
146     data->messages[idx].text[MESSAGE_TEXT_LEN - 1] = '\0';
147     data->messages[idx].timestamp = (uint64_t)time(nullptr);
148     data->messages[idx].saved = 0;
149     data->messages[idx].delivered = 0;
150 
151     data->msg_write_index = (idx + 1) % MAX_MESSAGES;
152     pthread_cond_broadcast(&data->cond);
153     pthread_mutex_unlock(&data->mutex);
154 }
155 
156 void Client::run()
157 {
```

```
158     if (pthread_create(&recv_thread, nullptr, &Client::receiver_loop, this) != 0)
159     {
160         perror("pthread_create");
161         return;
162     }
163 
164     std::string line;
165     std::cout << "> ";
166     while (!stop_requested && std::getline(std::cin, line))
167     {
168         if (data->server_stopping)
169         {
170             std::cout << "Server is shutting down. Exiting client.\n";
171             break;
172         }
173         if (stop_requested)
174             break;
175 
176         if (line == "quit" || line == "exit")
177             break;
178 
179         if (line.rfind("send ", 0) == 0)
180         {
181             size_t pos = line.find(' ', 5);
182             if (pos == std::string::npos)
183             {
184                 std::cout << "Usage: send <recipient> <message>\n";
185             }
186             else
187             {
188                 std::string recipient = line.substr(5, pos - 5);
189                 std::string text = line.substr(pos + 1);
190                 send_message(recipient, text);
191             }
192         }
193         else if (line == "help")
194         {
195             std::cout << "Commands:\n send <recipient> <message>\n exit\n";
196         }
197         else
198         {
199             std::cout << "Unknown command. Type help.\n";
200         }
201         std::cout << "> ";
202     }
203 
204     stop_requested = 1;
205     pthread_mutex_lock(&data->mutex);
206     pthread_cond_broadcast(&data->cond);
207     pthread_mutex_unlock(&data->mutex);
208 
209     pthread_join(recv_thread, nullptr);
210     std::cout << "Client exiting...\n";
211 }
212 
213 int main()
214 {
215     struct sigaction sa;
```

```
    sa.sa_handler = client_signal_handler;
    sigemptyset(&sa.sa_mask);
    sa.sa_flags = 0;
    sigaction(SIGINT, &sa, nullptr);

    std::cout << "Client starting. Connect to server's shared memory...\n";

    Client c;
    c.connect();

    std::string login;
    std::cout << "Enter login: ";
    std::getline(std::cin, login);
    if (login.empty())
    {
        std::cerr << "Login cannot be empty.\n";
        return 1;
    }

    if (!c.registerLogin(login))
    {
        std::cerr << "Cannot register login. Exiting.\n";
        return 1;
    }

    std::cout << "Registered as " << login << "\n";
    c.run();

    return 0;
}
```

Системные вызовы

```
01:49:35.735282 execve("./server",["./server"],0x7ffcb6521d58 /* 37 vars */)
= 0 <0.000354>
01:49:35.735888 brk(NULL)                       = 0x62b3d261c000 <0.000069>
01:49:35.736211 mmap(NULL,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0)
= 0x753306011000 <0.000089>
01:49:35.736558 access("/etc/ld.so.preload",R_OK) = -1 ENOENT (No such file
or directory) <0.000081>
01:49:35.736840 openat(AT_FDCWD,"/etc/ld.so.cache",O_RDONLY|O_CLOEXEC) = 3
<0.000157>
01:49:35.737203 fstat(3,{st_mode=S_IFREG|0644,st_size=31547,...}) = 0 <0.000072>
01:49:35.737504 mmap(NULL,31547,PROT_READ,MAP_PRIVATE,3,0) = 0x753306009000
<0.000136>
01:49:35.737778 close(3)                        = 0 <0.000082>
01:49:35.738003 openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libstdc++.so.6",O_RDONLY|O_CLO
= 3 <0.000080>
01:49:35.738315 read(3,"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0
\205'\0\0\0\0\0\0\0\0@\08\0\f\0@\0#\0\"\0\1\0\0\0\4\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\2\0\0\0\0\0\0 \2\0\0\0\0\0\0\10\0\0\0\0\0\0\0\4\0\0\0\4\0\0\0\340\2\0\0\0\0\0\340\
\0\0\0\0\0\0\0 \0\0\0\0\0\0\0\10\0\0\0\0\0\0\0\4\0\0\0\4\0\0\0\0\3\0\0\0\0\0\0\3\0\
\0\0\0\0\0\0\0\10\0\0\0\0\0\0\0"...,832) = 832 <0.000108>
```

```
01:49:35.738573 fstat(3,{st_mode=S_IFREG|0644,st_size=2592224,...}) = 0 <0.000037>
01:49:35.738730 mmap(NULL,2609472,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0)
= 0x753305c00000 <0.000085>
01:49:35.738932 mmap(0x753305c9d000,1343488,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED
= 0x753305c9d000 <0.000099>
01:49:35.739159 mmap(0x753305de5000,552960,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWR
= 0x753305de5000 <0.000080>
01:49:35.739355 mmap(0x753305e6c000,57344,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|
= 0x753305e6c000 <0.000107>
01:49:35.739593 mmap(0x753305e7a000,12608,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|
= 0x753305e7a000 <0.000073>
01:49:35.739928 close(3)                = 0 <0.000057>
01:49:35.740088 openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libgcc_s.so.1",O_RDONLY|O_CLOE
= 3 <0.000052>
01:49:35.740224 read(3,"\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0 \0\0\0\0\0\0\0\10\0\0\0\0\0\0\0\4\0\0\0\4\0\0\0\0\3\0\0\0\0\0\0\3\0\0
= 832 <0.000066>
01:49:35.740379 fstat(3,{st_mode=S_IFREG|0644,st_size=183024,...}) = 0 <0.000064>
01:49:35.740536 mmap(NULL,185256,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0) =
0x753305fdb000 <0.000055>
01:49:35.740720 mmap(0x753305fdf000,147456,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|
= 0x753305fdf000 <0.000051>
01:49:35.740951 mmap(0x753306003000,16384,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRI
= 0x753306003000 <0.000038>
01:49:35.741102 mmap(0x753306007000,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|M
= 0x753306007000 <0.000061>
01:49:35.741236 close(3)                = 0 <0.000052>
01:49:35.741345 openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libc.so.6",O_RDONLY|O_CLOEXEC)
= 3 <0.000031>
01:49:35.741446 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\0\0\0\0\0\0@\08\0\16\0@\0@\0?\0\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\0\0\0@) \0\0\0\0\0@\2\0\0\0\0\0\0@\2\0\0\0\0\0\0\10\0\0\0\0\0\0\0\4
= 832 <0.000050>
01:49:35.741605 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\0\0\0\0@) \0\0\0\0\0@) \0\0\0\0\0@\2\0\0\0\0\0\0@\2\0\0\0\0\0\0\10\0\0\0\0\0\0\0\4
= 784 <0.000050>
01:49:35.741715 fstat(3,{st_mode=S_IFREG|0755,st_size=2125328,...}) = 0 <0.000048>
01:49:35.741813 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\0\0\0\0@) \0\0\0\0\0@) \0\0\0\0\0@\2\0\0\0\0\0\0@\2\0\0\0\0\0\0\10\0\0\0\0\0\0\0\4
= 784 <0.000046>
01:49:35.741944 mmap(NULL,2170256,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0)
= 0x753305800000 <0.000050>
01:49:35.742097 mmap(0x753305828000,1605632,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED
= 0x753305828000 <0.000097>
01:49:35.742286 mmap(0x7533059b0000,323584,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWR
= 0x7533059b0000 <0.000050>
01:49:35.742381 mmap(0x7533059ff000,24576,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|
= 0x7533059ff000 <0.000056>
```

```
01:49:35.742489 mmap(0x753305a05000,52624,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|
= 0x753305a05000 <0.000059>
01:49:35.742603 close(3)                = 0 <0.000053>
01:49:35.742712 openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libm.so.6",O_RDONLY|O_CLOEXEC)
= 3 <0.000029>
01:49:35.742798 read(3,"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0
= 832 <0.000064>
01:49:35.742918 fstat(3,{st_mode=S_IFREG|0644,st_size=952616,...}) = 0 <0.000063>
01:49:35.743026 mmap(NULL,950296,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0) =
0x753305ef2000 <0.000055>
01:49:35.743125 mmap(0x753305f02000,520192,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|
= 0x753305f02000 <0.000069>
01:49:35.743236 mmap(0x753305f81000,360448,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWR
= 0x753305f81000 <0.000069>
01:49:35.743348 mmap(0x753305fd9000,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|M
= 0x753305fd9000 <0.000050>
01:49:35.743450 close(3)                = 0 <0.000126>
01:49:35.743668 mmap(NULL,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0)
= 0x753305ef0000 <0.000087>
01:49:35.743866 mmap(NULL,12288,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0)
= 0x753305eed000 <0.000102>
01:49:35.744120 arch_prctl(ARCH_SET_FS,0x753305eed740) = 0 <0.000097>
01:49:35.744338 set_tid_address(0x753305eeda10) = 12454 <0.000101>
01:49:35.744588 set_robust_list(0x753305eeda20,24) = 0 <0.000144>
01:49:35.744823 rseq(0x753305eee060,0x20,0,0x53053053) = 0 <0.000064>
01:49:35.745075 mprotect(0x7533059ff000,16384,PROT_READ) = 0 <0.000053>
01:49:35.745230 mprotect(0x753305fd9000,4096,PROT_READ) = 0 <0.000063>
01:49:35.745387 mprotect(0x753306007000,4096,PROT_READ) = 0 <0.000052>
01:49:35.746183 mprotect(0x753305e6c000,45056,PROT_READ) = 0 <0.000096>
01:49:35.746442 mprotect(0x62b3ac309000,4096,PROT_READ) = 0 <0.000031>
01:49:35.746541 mprotect(0x753306049000,8192,PROT_READ) = 0 <0.000035>
01:49:35.746674 prlimit64(0,RLIMIT_STACK,NULL,{rlim_cur=8192*1024,rlim_max=RLIM64_INF
= 0 <0.000091>
01:49:35.746971 munmap(0x753306009000,31547) = 0 <0.000092>
01:49:35.747253 futex(0x753305e7a7bc,FUTEX_WAKE_PRIVATE,2147483647) = 0 <0.000322>
01:49:35.747852 getrandom("\x4e\x80\x67\x46\x82\x32\x35\x91",8,GRND_NONBLOCK)
= 8 <0.000088>
01:49:35.748042 brk(NULL)               = 0x62b3d261c000 <0.000088>
01:49:35.748224 brk(0x62b3d263d000)     = 0x62b3d263d000 <0.000033>
01:49:35.748330 rt_sigaction(SIGINT,{sa_handler=0x62b3ac3068a9,sa_mask=[],sa_flags=SA
= 0 <0.000066>
01:49:35.748478 fstat(1,{st_mode=S_IFCHR|0620,st_rdev=makedev(0x88,0x4),...})
= 0 <0.000028>
01:49:35.748598 write(1,"Starting server...\n",19) = 19 <0.000049>
01:49:35.748721 openat(AT_FDCWD,"/dev/shm/simple_chat_shm_v1",O_RDWR|O_NOFOLLOW|O_CLO
= -1 ENOENT (No such file or directory) <0.000035>
01:49:35.748821 openat(AT_FDCWD,"/dev/shm/simple_chat_shm_v1",O_RDWR|O_CREAT|O_EXCL|O
= 3 <0.000042>
```

```
01:49:35.748917 ftruncate(3,344672)     = 0 <0.000028>
01:49:35.748978 mmap(NULL,344672,PROT_READ|PROT_WRITE,MAP_SHARED,3,0) = 0x753305e9800
<0.000031>
01:49:35.749256 rt_sigaction(SIGRT_1,{sa_handler=0x753305899530,sa_mask=[],sa_flags=S
= 0 <0.000066>
01:49:35.749432 rt_sigprocmask(SIG_UNBLOCK,[RTMIN RT_1],NULL,8) = 0 <0.000073>
01:49:35.749623 mmap(NULL,8392704,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,-1,0)
= 0x753304fff000 <0.000064>
01:49:35.749774 mprotect(0x753305000000,8388608,PROT_READ|PROT_WRITE) = 0 <0.000034>
01:49:35.749855 rt_sigprocmask(SIG_BLOCK,~[],[],8) = 0 <0.000025>
01:49:35.749920 clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREA
=>{parent_tid=[12455]},88) = 12455 <0.000263>
01:49:35.750256 rt_sigprocmask(SIG_SETMASK,[],NULL,8) = 0 <0.000053>
01:49:35.750447 write(1,"server>",8) = 8 <0.000153>
01:49:35.750716 fstat(0,{st_mode=S_IFCHR|0620,st_rdev=makedev(0x88,0x4),...})
= 0 <0.000216>
01:49:35.751066 read(0,0x62b3d262e800,1024) = ? ERESTARTSYS (To be restarted
if SA_RESTART is set) <0.000100>
01:49:35.751298 ---SIGTTIN {si_signo=SIGTTIN,si_code=SI_KERNEL} ---
01:49:35.751449 ---stopped by SIGTTIN ---
01:49:35.750225 rseq(0x7533057fffe0,0x20,0,0x53053053) = 0 <0.000023>
01:49:35.750329 set_robust_list(0x7533057ff9a0,24) = 0 <0.000160>
01:49:35.750545 rt_sigprocmask(SIG_SETMASK,[],NULL,8) = 0 <0.000059>
01:49:35.750850 futex(0x753305e98050,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,0,NULL,FU
= ? ERESTARTSYS (To be restarted if SA_RESTART is set) <0.000528>
01:49:35.751541 ---stopped by SIGTTIN ---
01:49:40.736844 execve("./client",["./client"],0x7fffb3ec03e8 /* 37 vars */)
= 0 <0.000237>
01:49:40.737318 brk(NULL)               = 0x6101caf9c000 <0.000053>
01:49:40.737488 mmap(NULL,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0)
= 0x7e28b7ace000 <0.000064>
01:49:40.737616 access("/etc/ld.so.preload",R_OK) = -1 ENOENT (No such file
or directory) <0.000051>
01:49:40.737734 openat(AT_FDCWD,"/etc/ld.so.cache",O_RDONLY|O_CLOEXEC) = 3
<0.000067>
01:49:40.737893 fstat(3,{st_mode=S_IFREG|0644,st_size=31547,...}) = 0 <0.000041>
01:49:40.738004 mmap(NULL,31547,PROT_READ,MAP_PRIVATE,3,0) = 0x7e28b7ac6000
<0.000068>
01:49:40.738146 close(3)                = 0 <0.000095>
01:49:40.738315 openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libstdc++.so.6",O_RDONLY|O_CLO
= 3 <0.000051>
01:49:40.738418 read(3,"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0
\205'\0\0\0\0\0\0\0\0@\08\0\f\0@\0#\0\"\0\1\0\0\0\4\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\2\0\0\0\0\0\0 \2\0\0\0\0\0\0\10\0\0\0\0\0\0\0\4\0\0\0\4\0\0\0\340\2\0\0\0\0\0\0\340\
\0\0\0\0\0\0\0 \0\0\0\0\0\0\0\10\0\0\0\0\0\0\0\4\0\0\0\4\0\0\0\0\3\0\0\0\0\0\0\3\0\
\0\0\0\0\0\0\10\0\0\0\0\0\0\0"...,832) = 832 <0.000049>
01:49:40.738526 fstat(3,{st_mode=S_IFREG|0644,st_size=2592224,...}) = 0 <0.000068>
01:49:40.738664 mmap(NULL,2609472,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0)
```

```
= 0x7e28b7800000 <0.000056>
01:49:40.738780 mmap(0x7e28b789d000,1343488,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED
= 0x7e28b789d000 <0.000067>
01:49:40.738917 mmap(0x7e28b79e5000,552960,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWR
= 0x7e28b79e5000 <0.000051>
01:49:40.739029 mmap(0x7e28b7a6c000,57344,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|
= 0x7e28b7a6c000 <0.000076>
01:49:40.739179 mmap(0x7e28b7a7a000,12608,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|
= 0x7e28b7a7a000 <0.000055>
01:49:40.739310 close(3)                = 0 <0.000053>
01:49:40.739428 openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libgcc_s.so.1",O_RDONLY|O_CLOE
= 3 <0.000063>
01:49:40.739554 read(3,"\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0 \0\0\0\0\0\0\0\10\0\0\0\0\0\0\0\4\0\0\0\4\0\0\0\0\3\0\0\0\0\0\0\3\0\0
= 832 <0.000063>
01:49:40.739707 fstat(3,{st_mode=S_IFREG|0644,st_size=183024,...}) = 0 <0.000052>
01:49:40.739849 mmap(NULL,185256,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0) =
0x7e28b7a98000 <0.000073>
01:49:40.740028 mmap(0x7e28b7a9c000,147456,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|
= 0x7e28b7a9c000 <0.000083>
01:49:40.740172 mmap(0x7e28b7ac0000,16384,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRI
= 0x7e28b7ac0000 <0.000053>
01:49:40.740281 mmap(0x7e28b7ac4000,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|M
= 0x7e28b7ac4000 <0.000055>
01:49:40.740403 close(3)                = 0 <0.000041>
01:49:40.740504 openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libc.so.6",O_RDONLY|O_CLOEXEC)
= 3 <0.000057>
01:49:40.740632 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\0\0\0\0\0@\08\0\16\0@\0@\0?\0\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\0\0\0@) \0\0\0\0\0@\2\0\0\0\0\0\0@\2\0\0\0\0\0\0\10\0\0\0\0\0\0\0\4
= 832 <0.000055>
01:49:40.740767 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\0\0\0@) \0\0\0\0\0@) \0\0\0\0\0@\2\0\0\0\0\0\0@\2\0\0\0\0\0\0\10\0\0\0\0\0\0\0\4
= 784 <0.000067>
01:49:40.740952 fstat(3,{st_mode=S_IFREG|0755,st_size=2125328,...}) = 0 <0.000063>
01:49:40.741102 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\0\0\0@) \0\0\0\0\0@) \0\0\0\0\0@\2\0\0\0\0\0\0@\2\0\0\0\0\0\0\10\0\0\0\0\0\0\0\4
= 784 <0.000114>
01:49:40.741329 mmap(NULL,2170256,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0)
= 0x7e28b7400000 <0.000113>
01:49:40.741590 mmap(0x7e28b7428000,1605632,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED
= 0x7e28b7428000 <0.000083>
01:49:40.741784 mmap(0x7e28b75b0000,323584,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWR
= 0x7e28b75b0000 <0.000078>
01:49:40.741972 mmap(0x7e28b75ff000,24576,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|
= 0x7e28b75ff000 <0.000134>
01:49:40.742291 mmap(0x7e28b7605000,52624,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|
= 0x7e28b7605000 <0.000059>
```

```
01:49:40.742495 close(3)                = 0 <0.000072>
01:49:40.742705 openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libm.so.6",O_RDONLY|O_CLOEXEC)
= 3 <0.000061>
01:49:40.742867 read(3,"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0
= 832 <0.000052>
01:49:40.743072 fstat(3,{st_mode=S_IFREG|0644,st_size=952616,...}) = 0 <0.000068>
01:49:40.743235 mmap(NULL,950296,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0) =
0x7e28b7717000 <0.000071>
01:49:40.743410 mmap(0x7e28b7727000,520192,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|
= 0x7e28b7727000 <0.000078>
01:49:40.743591 mmap(0x7e28b77a6000,360448,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWR
= 0x7e28b77a6000 <0.000103>
01:49:40.743790 mmap(0x7e28b77fe000,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|M
= 0x7e28b77fe000 <0.000071>
01:49:40.743956 close(3)                = 0 <0.000038>
01:49:40.744068 mmap(NULL,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0)
= 0x7e28b7a96000 <0.000073>
01:49:40.744246 mmap(NULL,12288,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0)
= 0x7e28b7a93000 <0.000052>
01:49:40.744367 arch_prctl(ARCH_SET_FS,0x7e28b7a93740) = 0 <0.000040>
01:49:40.744462 set_tid_address(0x7e28b7a93a10) = 12561 <0.000048>
01:49:40.744575 set_robust_list(0x7e28b7a93a20,24) = 0 <0.000063>
01:49:40.744727 rseq(0x7e28b7a94060,0x20,0,0x53053053) = 0 <0.000045>
01:49:40.744873 mprotect(0x7e28b75ff000,16384,PROT_READ) = 0 <0.000067>
01:49:40.745023 mprotect(0x7e28b77fe000,4096,PROT_READ) = 0 <0.000071>
01:49:40.745161 mprotect(0x7e28b7ac4000,4096,PROT_READ) = 0 <0.000052>
01:49:40.745467 mprotect(0x7e28b7a6c000,45056,PROT_READ) = 0 <0.000056>
01:49:40.745604 mprotect(0x610198be7000,4096,PROT_READ) = 0 <0.000050>
01:49:40.745716 mprotect(0x7e28b7b06000,8192,PROT_READ) = 0 <0.000052>
01:49:40.745831 prlimit64(0,RLIMIT_STACK,NULL,{rlim_cur=8192*1024,rlim_max=RLIM64_INF
= 0 <0.000045>
01:49:40.745936 munmap(0x7e28b7ac6000,31547) = 0 <0.000057>
01:49:40.746122 futex(0x7e28b7a7a7bc,FUTEX_WAKE_PRIVATE,2147483647) = 0 <0.000039>
01:49:40.746297 getrandom("\x34\x6c\xf9\x86\x71\x4a\xf5\x59",8,GRND_NONBLOCK)
= 8 <0.000049>
01:49:40.746414 brk(NULL)               = 0x6101caf9c000 <0.000047>
01:49:40.746530 brk(0x6101cafbd000)     = 0x6101cafbd000 <0.000052>
01:49:40.746651 rt_sigaction(SIGINT,{sa_handler=0x610198be45a9,sa_mask=[],sa_flags=SA
= 0 <0.000050>
01:49:40.746780 fstat(1,{st_mode=S_IFCHR|0620,st_rdev=makedev(0x88,0x4),...})
= 0 <0.000049>
01:49:40.746888 write(1,"Client starting. Connect to server's shared memory...\n",54)
= 54 <0.000065>
01:49:40.747077 openat(AT_FDCWD,"/dev/shm/simple_chat_shm_v1",O_RDWR|O_NOFOLLOW|O_CLO
= 3 <0.000109>
01:49:40.747264 mmap(NULL,344672,PROT_READ|PROT_WRITE,MAP_SHARED,3,0) = 0x7e28b76c200
<0.000041>
01:49:40.747369 write(1,"Enter login: ",13) = 13 <0.000057>
```

```
01:49:40.747485 fstat(0,{st_mode=S_IFCHR|0620,st_rdev=makedev(0x88,0x4),...})
= 0 <0.000077>
01:49:40.747622 read(0,0x6101cafae6c0,1024) = ? ERESTARTSYS (To be restarted
if SA_RESTART is set) <0.000062>
01:49:40.747742 ---SIGTTIN {si_signo=SIGTTIN,si_code=SI_KERNEL} ---
01:49:40.747794 ---stopped by SIGTTIN ---
01:49:43.742590 execve("./client",["./client"],0x7fff73114bb8 /* 37 vars */)
= 0 <0.000443>
01:49:43.743326 brk(NULL)               = 0x5d2cb3ff5000 <0.000097>
01:49:43.743646 mmap(NULL,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0)
= 0x742301655000 <0.000108>
01:49:43.743910 access("/etc/ld.so.preload",R_OK) = -1 ENOENT (No such file
or directory) <0.000103>
01:49:43.744211 openat(AT_FDCWD,"/etc/ld.so.cache",O_RDONLY|O_CLOEXEC) = 3
<0.000072>
01:49:43.744477 fstat(3,{st_mode=S_IFREG|0644,st_size=31547,...}) = 0 <0.000097>
01:49:43.744734 mmap(NULL,31547,PROT_READ,MAP_PRIVATE,3,0) = 0x74230164d000
<0.000075>
01:49:43.744959 close(3)                = 0 <0.000063>
01:49:43.745139 openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libstdc++.so.6",O_RDONLY|O_CLO
= 3 <0.000080>
01:49:43.745360 read(3,"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0
\205'\0\0\0\0\0\0\0\0@\08\0\f\0@\0#\0\"\0\1\0\0\0\4\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\2\0\0\0\0\0\0 \2\0\0\0\0\0\0\10\0\0\0\0\0\0\0\4\0\0\0\4\0\0\0\340\2\0\0\0\0\0\340\
\0\0\0\0\0\0\0 \0\0\0\0\0\0\0\10\0\0\0\0\0\0\0\4\0\0\0\4\0\0\0\0\3\0\0\0\0\0\0\3\0\
\0\0\0\0\0\0\10\0\0\0\0\0\0\0"...,832) = 832 <0.000070>
01:49:43.745566 fstat(3,{st_mode=S_IFREG|0644,st_size=2592224,...}) = 0 <0.000064>
01:49:43.745720 mmap(NULL,2609472,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0)
= 0x742301200000 <0.000070>
01:49:43.745897 mmap(0x74230129d000,1343488,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED
= 0x74230129d000 <0.000076>
01:49:43.746079 mmap(0x7423013e5000,552960,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWR
= 0x7423013e5000 <0.000073>
01:49:43.746244 mmap(0x74230146c000,57344,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|
= 0x74230146c000 <0.000069>
01:49:43.746411 mmap(0x74230147a000,12608,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|
= 0x74230147a000 <0.000075>
01:49:43.746636 close(3)                = 0 <0.000067>
01:49:43.746828 openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libgcc_s.so.1",O_RDONLY|O_CLOE
= 3 <0.000072>
01:49:43.747018 read(3,"\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0 \0\0\0\0\0\0\0\10\0\0\0\0\0\0\0\4\0\0\0\4\0\0\0\0\3\0\0\0\0\0\0\3\0\
= 832 <0.000068>
01:49:43.747212 fstat(3,{st_mode=S_IFREG|0644,st_size=183024,...}) = 0 <0.000064>
01:49:43.747401 mmap(NULL,185256,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0) =
0x74230161f000 <0.000076>
01:49:43.747597 mmap(0x742301623000,147456,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|
= 0x742301623000 <0.000074>
```

```
01:49:43.747778 mmap(0x742301647000,16384,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRI
= 0x742301647000 <0.000070>
01:49:43.747934 mmap(0x74230164b000,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|M
= 0x74230164b000 <0.000135>
01:49:43.748219 close(3)                = 0 <0.000076>
01:49:43.748440 openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libc.so.6",O_RDONLY|O_CLOEXEC)
= 3 <0.000084>
01:49:43.748646 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\0\0\0\0\0@\08\0\16\0@\0@\0?\0\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\0\0\0@) \0\0\0\0\0@\2\0\0\0\0\0\0@\2\0\0\0\0\0\0\10\0\0\0\0\0\0\0\4
= 832 <0.000070>
01:49:43.748846 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\0\0\0@) \0\0\0\0\0@) \0\0\0\0\0@\2\0\0\0\0\0\0@\2\0\0\0\0\0\0\10\0\0\0\0\0\0\0\4
= 784 <0.000069>
01:49:43.749036 fstat(3,{st_mode=S_IFREG|0755,st_size=2125328,...}) = 0 <0.000059>
01:49:43.749196 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\0\0\0@) \0\0\0\0\0@) \0\0\0\0\0@\2\0\0\0\0\0\0@\2\0\0\0\0\0\0\10\0\0\0\0\0\0\0\4
= 784 <0.000066>
01:49:43.749376 mmap(NULL,2170256,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0)
= 0x742300e00000 <0.000079>
01:49:43.749591 mmap(0x742300e28000,1605632,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED
= 0x742300e28000 <0.000085>
01:49:43.749805 mmap(0x742300fb0000,323584,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWR
= 0x742300fb0000 <0.000079>
01:49:43.749992 mmap(0x742300fff000,24576,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|
= 0x742300fff000 <0.000120>
01:49:43.750256 mmap(0x742301005000,52624,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|
= 0x742301005000 <0.000115>
01:49:43.750558 close(3)                = 0 <0.000064>
01:49:43.750768 openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libm.so.6",O_RDONLY|O_CLOEXEC)
= 3 <0.000157>
01:49:43.751086 read(3,"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0
= 832 <0.000239>
01:49:43.751773 fstat(3,{st_mode=S_IFREG|0644,st_size=952616,...}) = 0 <0.000105>
01:49:43.752114 mmap(NULL,950296,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0) =
0x742301536000 <0.000082>
01:49:43.752311 mmap(0x742301546000,520192,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|
= 0x742301546000 <0.000085>
01:49:43.752485 mmap(0x7423015c5000,360448,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWR
= 0x7423015c5000 <0.000070>
01:49:43.752644 mmap(0x74230161d000,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|M
= 0x74230161d000 <0.000070>
01:49:43.752845 close(3)                = 0 <0.000068>
01:49:43.753030 mmap(NULL,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0)
= 0x742301534000 <0.000067>
01:49:43.753260 mmap(NULL,12288,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0)
= 0x742301531000 <0.000144>
01:49:43.753557 arch_prctl(ARCH_SET_FS,0x742301531740) = 0 <0.000067>
```

```
01:49:43.753773 set_tid_address(0x742301531a10) = 12597 <0.000080>
01:49:43.754130 set_robust_list(0x742301531a20,24) = 0 <0.000064>
01:49:43.754340 rseq(0x742301532060,0x20,0,0x53053053) = 0 <0.000080>
01:49:43.754631 mprotect(0x742300fff000,16384,PROT_READ) = 0 <0.000081>
01:49:43.754864 mprotect(0x74230161d000,4096,PROT_READ) = 0 <0.000084>
01:49:43.755149 mprotect(0x74230164b000,4096,PROT_READ) = 0 <0.000092>
01:49:43.755850 mprotect(0x74230146c000,45056,PROT_READ) = 0 <0.000091>
01:49:43.756138 mprotect(0x5d2ca755c000,4096,PROT_READ) = 0 <0.000086>
01:49:43.756380 mprotect(0x74230168d000,8192,PROT_READ) = 0 <0.000093>
01:49:43.756673 prlimit64(0,RLIMIT_STACK,NULL,{rlim_cur=8192*1024,rlim_max=RLIM64_INF
= 0 <0.000069>
01:49:43.756917 munmap(0x74230164d000,31547) = 0 <0.000111>
01:49:43.757287 futex(0x74230147a7bc,FUTEX_WAKE_PRIVATE,2147483647) = 0 <0.000066>
01:49:43.757577 getrandom("\xe0\x4b\xf7\x9e\x43\xea\x6f\x09",8,GRND_NONBLOCK)
= 8 <0.000074>
01:49:43.757793 brk(NULL)               = 0x5d2cb3ff5000 <0.000064>
01:49:43.757958 brk(0x5d2cb4016000)     = 0x5d2cb4016000 <0.000115>
01:49:43.758227 rt_sigaction(SIGINT,{sa_handler=0x5d2ca75595a9,sa_mask=[],sa_flags=SA
= 0 <0.000062>
01:49:43.758455 fstat(1,{st_mode=S_IFCHR|0620,st_rdev=makedev(0x88,0x4),...})
= 0 <0.000063>
01:49:43.758632 write(1,"Client starting. Connect to server's shared memory...\n",54)
= 54 <0.000091>
01:49:43.758860 openat(AT_FDCWD,"/dev/shm/simple_chat_shm_v1",O_RDWR|O_NOFOLLOW|O_CLO
= 3 <0.000075>
01:49:43.759054 mmap(NULL,344672,PROT_READ|PROT_WRITE,MAP_SHARED,3,0) = 0x7423014dc00
<0.000071>
01:49:43.759246 write(1,"Enter login: ",13) = 13 <0.000085>
01:49:43.759531 fstat(0,{st_mode=S_IFCHR|0620,st_rdev=makedev(0x88,0x4),...})
= 0 <0.000061>
01:49:43.759706 read(0,0x5d2cb40076c0,1024) = ? ERESTARTSYS (To be restarted
if SA_RESTART is set) <0.000091>
01:49:43.759909 ---SIGTTIN {si_signo=SIGTTIN,si_code=SI_KERNEL} ---
01:49:43.759983 ---stopped by SIGTTIN ---
```