|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **ПРОГРАМНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.03 ПРОГРАММНАЯ ИНЖЕНЕРИЯ**

**Отчет**

|  |  |
| --- | --- |
| **По лабораторной работе №** | 6 |

**Дисциплина:** Операционные системы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ7-62Б |  |  | Н.А. Гарасев |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | Н.Ю. Рязанова |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2020

**Задание**

**Часть 1.**

Написать приложение по модели клиент-сервер, демонстрирующее взаимодействие параллельных процессов на отдельном компьютере с использованием сокетов в файловом пространстве имен: семейство - AF\_UNIX, тип - SOCK\_DGRAM. При демонстрации работы программного комплекса необходимо запустить несколько клиентов (не меньше 5) и продемонстрировать, что сервер обрабатывает обращения каждого запущенного клиента.

Код программы.

Код сервера.

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <unistd.h>  #include <sys/types.h>  #include <signal.h>  #include <sys/socket.h>  #define MSG\_LEN 256  #define SOCKET\_NAME "socket.soc"  int sock;  void sigint\_handler(int signum)  {  close(sock);  unlink(SOCKET\_NAME);  printf("Socket was closed!\n");  }  int main(void)  {  struct sockaddr addr;  sock = socket(AF\_UNIX, SOCK\_DGRAM, 0);  if (sock < 0)  {  perror("Can't open socket!");  exit(1);  }  addr.sa\_family = AF\_UNIX;  strcpy(addr.sa\_data, SOCKET\_NAME);  if (bind(sock, &addr, sizeof(addr)) < 0)  {  printf("Can't bind name to socket!\n");  close(sock);  unlink(SOCKET\_NAME);  perror("Error in bind() ");  exit(-1);  }  printf("\nServer is waiting\n");  signal(SIGINT, sigint\_handler);  char msg[MSG\_LEN];  while(1)  {  int recievedSize = recv(sock, msg, sizeof(msg), 0);  if (recievedSize < 0)  {  close(sock);  unlink(SOCKET\_NAME);  perror("Error in recv()");  exit(1);  }  msg[recievedSize] = 0;  printf("Client send: %s\n", msg);  }    printf("Closing socket\n");  close(sock);  unlink(SOCKET\_NAME);  return 0;  } |

Код клиента.

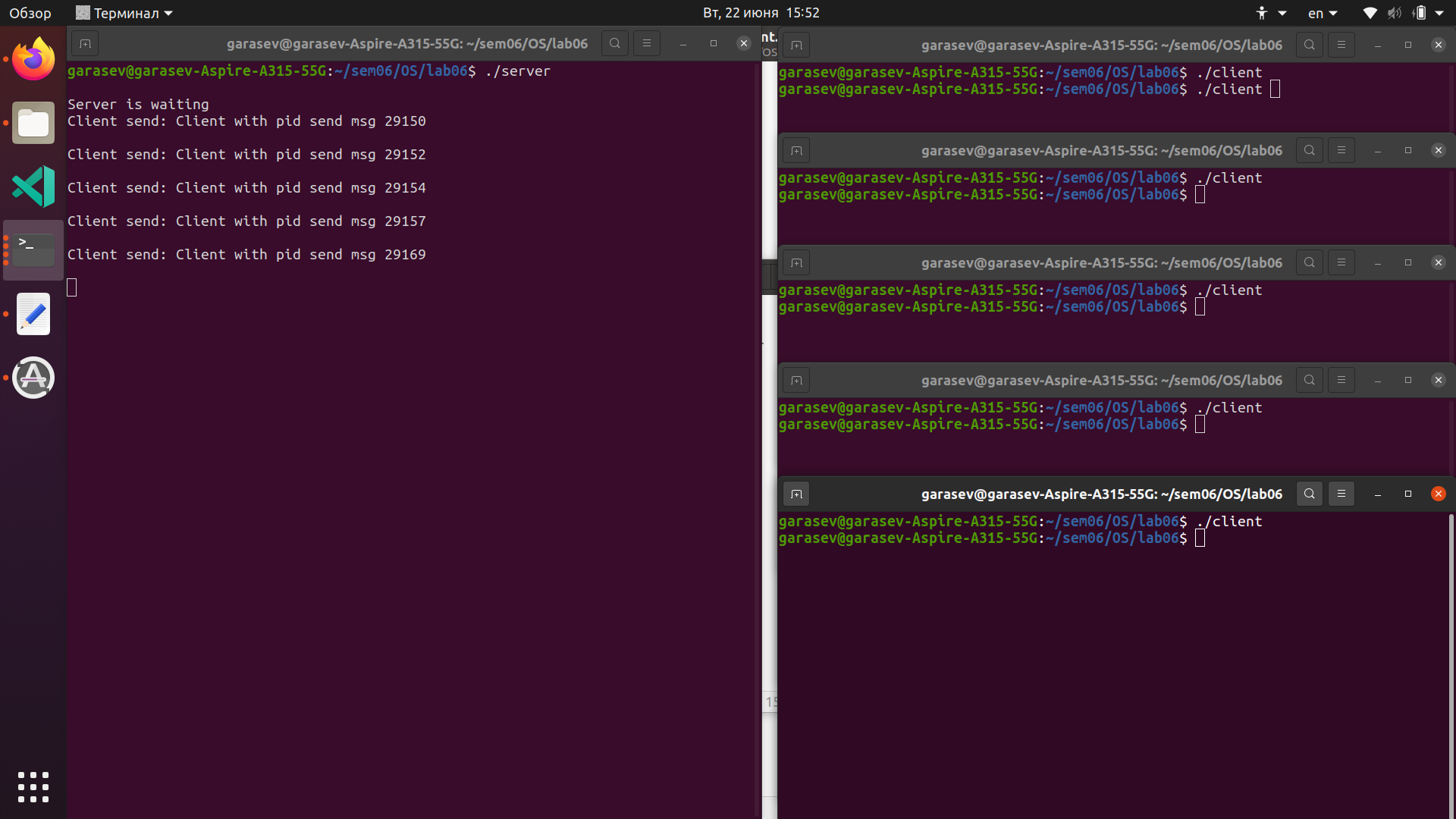
|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <unistd.h>  #include <sys/types.h>  #include <sys/socket.h>  #include "info.h"  int main(void)  {  int sockfd = socket(PF\_LOCAL, SOCK\_DGRAM, 0);  if (sockfd < 0)  {  perror("Can't open socket!");  exit(1);  }  struct sockaddr server\_addr;  server\_addr.sa\_family = AF\_UNIX;  strcpy(server\_addr.sa\_data, SOCKET\_NAME);  char msg[MSG\_LEN];  sprintf(msg, "Client with pid send msg %d\n", getpid());  sendto(sockfd, msg, strlen(msg), 0, &server\_addr, sizeof(server\_addr));  close(sockfd);  return 0;  } |

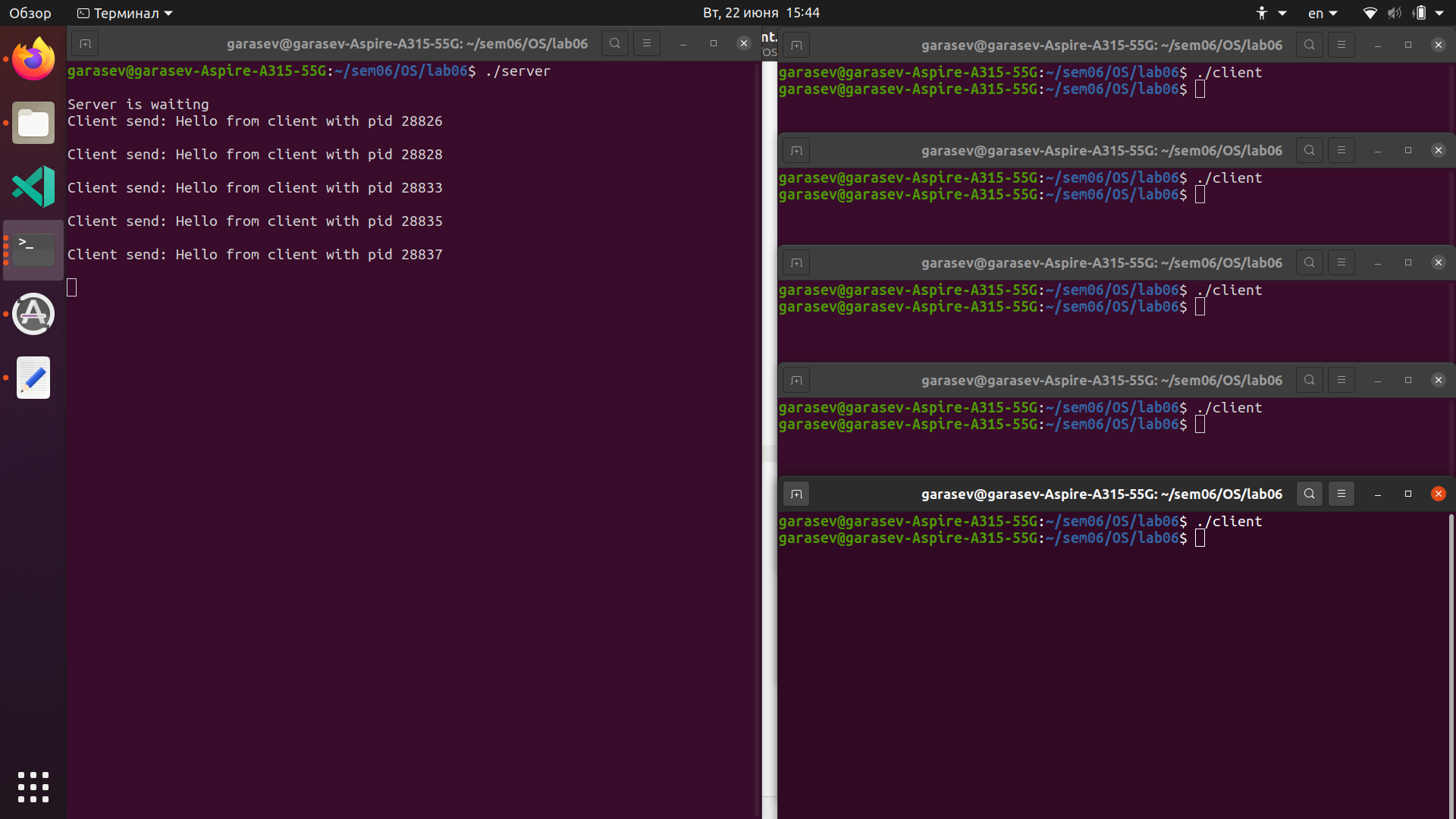
**Описание работы программы.**

В процессе-сервере с помощью с помощью вызова socket() создается сокет семейства AF\_UNIX с типом SOCK\_DGRAM. С помощью системного вызова bind() происходит привязка сокета и локального адреса. Сервер блокируется на функции recv() и ждет сообщения от процессов-клиентов.

В процессе-клиенте создается сокет семейства AF\_UNIX с типом SOCK\_DGRAM с помощью системного вызова socket(). С помощью функции sendto() отправляется сообщение к процессу-серверу.

**Пример работы программы.**





**Часть 2.**