**Содержание**

[**1. Цели работы** 2](#_Toc86758521)

[**2. Задачи** 2](#_Toc86758522)

[**3. Ход выполнения работы** 2](#_Toc86758523)

[**4. Вывод** 8](#_Toc86758524)

# **Лабораторная работа №10 – Создание соединения на сокетах**

# **1. Цели работы**

Освоение набора системных вызовов для создания сокетных соединений различных типов для обмена данными по сети.

**2. Задачи**

1. Войти в систему и скопировать с разделяемого ресурса в свой HOME-каталог набор исходных файлов для десятого занятия.

2. Скомпилировать и выполнить программу socketpair.cpp, которая демонстрирует создание простейшего вида сокета и обмен данными двух родственных процессов. Проанализировать вывод на консоль. Определить существует ли зависимость обмена от различных соотношений величин временных задержек.

3. Скомпилировать программы echo\_sever.cpp и echo\_client.cpp, задавая им при компиляции разные имена. Запустить программы сервера и клиента на разных терминалах. Ввести символьную информацию в окне клиента и проанализировать ввод.

4. Скомпилировать с разными именами программы sock\_c\_i\_srv.cpp и sock\_c\_iclt.cpp. Запустить программы сервера и клиента на разных терминалах. При запуске клиента указать в качестве параметра командной строки имя хоста localhost. Ввести символьную информацию в окне клиента и пояснить вывод.

5. Модифицировать программу echo\_server.cpp так, чтобы она при ответе на запросы клиента что-либо выводилось в окне сервера. Испытать работу эхо-сервера при одновременной работе с несколькими клиентами.

**3. Ход выполнения работы**

1. Войду в систему и скопирую в свой HOME-каталог с разделяемого ресурса набор исходных файлов для десятой лабораторной работы.

2. Скомпилирую и выполню программу socketpair.cpp, которая демонстрирует создание простейшего вида сокета и обмен между двумя родственными процессами. Результат выполнения приведен на рисунке 10.1. В результате работы программы в терминал была выведена информация в виде чисел от 0 до 9 включительно. Сама программа создает пару простейших сокетов. После создания потомка начинается ветвление и потомок, и родитель обрабатываются отдельно. Родитель отправляет потомку число, потомок, анализируя присланное сообщение, отвечает родителю, пока число не станет больше 9. В данном случае величина sleep() не оказывает влияние на функционал программы. При изменении данного параметра будет изменятся задержка отправки сообщений, так как ожидать будут сами процессы, а не скорость отправки сообщений не изменится.

|  |
| --- |
| Рис. 10.1 – Выполнение программы socketpair.cpp |

3. Скомпилирую и выполню программы echo\_server.cpp и echo\_client.cpp, задавая им при компиляции разные имена. Запущу программы сервера и клиента на разных терминалах, а также введу символьную информацию в окне клиента. Результаты приведены на рисунке 10.2 и 10.3. При выполнении программы создается специальный файл-сокет, который в качестве имени использует адрес имени файла. Результаты приведены на рисунке 10.4. По результатам с рисунка 10.3, что клиент получает от сервера сообщение, которое ввел сам же пользователь. В примере используются сокеты поточного типы. На рисунке 10.5 – приведена иллюстрация отключения клиента – видно, сервер остается в ожидании.

|  |
| --- |
| Рис. 10.2 – Компиляция файлов server и client |
| Рис. 10.3 – Результаты введение символьной информации |
| Рис. 10.4 - Файл-сокет |
| Рис. 10.5 – Результат отключения клиента |

4. Скомпилирую с разными именами программы sock\_c\_i\_srv.cpp и sock\_c\_iclt.cpp. Запущу программы сервера и клиента на разных терминалах. При запуске клиента был указан в качестве параметра командной строки имя хоста localhost. Введу символьную информацию в окне клиента. Результаты приведены на рисунке 10.6. Пользователю выводит его же сообщение, но в верхнем регистре. Важно уточнить, что в данном примере используются сетевые сокеты, так как требуется указывать и присваивать сетевой адрес, при этом файлы не создаются.

|  |
| --- |
| Рис. 10.6 – Результаты программы пункта 4. |

5. Модифицирую программы echo\_server.cpp так, чтобы она выводило при ответе информацию в терминал. Для этого достаточно добавить пару строк printf. Содержимое кода представлено в листинге 10.1.

|  |
| --- |
| Листинг 10.1 – Программа echo\_server\_m.cpp |
| /\*  \*\* echo\_server.cpp -- the echo server for echo\_cient.cpp; demonstrates UNIX sockets  \*/  #include <stdio.h>  #include <stdlib.h>  #include <unistd.h>  #include <errno.h>  #include <string.h>  #include <sys/types.h>  #include <sys/socket.h>  #include <sys/un.h>  #define SOCK\_PATH "echo\_socket"  int main(void)  {  int s, s2, t, len;  struct sockaddr\_un local, remote;  char str[100];  if ((s = socket(AF\_UNIX, SOCK\_STREAM, 0)) == -1) {  perror("socket");  exit(1);  }  local.sun\_family = AF\_UNIX;  // remote.sun\_family = AF\_UNIX; //  strcpy(local.sun\_path, SOCK\_PATH);  unlink(local.sun\_path);  len = strlen(local.sun\_path) + sizeof(local.sun\_family);  if (bind(s, (struct sockaddr \*)&local, len) == -1) {  perror("bind");  exit(1);  }  if (listen(s, 5) == -1) {  perror("listen");  exit(1);  }  for(;;) {  int done, n;  printf("Waiting for a connection...\n");  t = sizeof(remote);  if ((s2 = accept(s, (struct sockaddr \*)&remote, (socklen\_t \*)&t)) == -1) {  perror("accept");  exit(1);  }  printf("Connected.\n");  done = 0;  do {  n = recv(s2, str, 100, 0);  printf("CLIENT ----> %s", str);  if (n <= 0) {  if (n < 0) perror("recv");  done = 1;  }  if (!done)  printf("SERVER ----> %s", str);  if (send(s2, str, n, 0) < 0) {  perror("send");  done = 1;  }  } while (!done);  close(s2);  }  return 0;  } |

Далее протестирую скомпилирую модифицированную программу. Компиляция приведена на рисунке 10.7. Выполню программы и введу символьную информацию на стороне клиента. Результаты приведены на рисунке 10.8. Как видно, на стороне сервера отображаются сообщения о получении и отправки сообщений обратно.

|  |
| --- |
| Рис. 10.7 – Компиляция модифицированной программы |
| Рис. 10.8 – Результаты выполнения модифицированной программы |

Необходимо протестировать работу при запуске нескольких клиентов. Результаты приведены на рисунке 10.9. Данный эхо-сервер не предназначен для работы с несколькими клиентами, так как сообщение от второго клиента не доходят до сервера. Можно добавить создание дополнительных потоков для решения проблемы (fork).

|  |
| --- |
| Рис. 10.9 – Тестирование нескольких клиентов |

**4. Вывод**

В ходе выполнения программы было на практике изучено создание сокетов разных видов, а также их свойства. При этом были получены знания о работе сервера и его особенностях. Работа сервера была модифицирована для отображения дополнительной информации о сообщениях, которые он получает и отправляет.