1 пункт

На консоль мы получили числа от 0 до 9.   
В ходе программы была создана пара сокетов. После форка в операторе case был выбор родителя и потомка. Закрываются ненужные сокеты. Родитель отправляет потомку нечётные числа и читает чётные от ребёнка и наоборот.  
Зависимость от величины переданной в sleep() проявляется лишь в задержках перед выводом следующей цифры. Т.к. программа не идёт дальше пока не будет что прочитать из другого сокета.

Изображение выглядит как текст

Автоматически созданное описание

2 пункт

Данный сокет явялется ПОТОЧНЫМ – обеспечивает двухсторонний, последовательный, надежный поток данных без определённых границ. Ввод/ввывод буферезуются. Если ввести довольно длинную строку, то она разделится на несколько echo.

Изображение выглядит как текст

Автоматически созданное описание

Файл нужен для создания сокета.

Изображение выглядит как текст

Автоматически созданное описание

3 пункт

Изображение выглядит как текст

Автоматически созданное описание

Данный сокет так же **поточный**. Он возвращает введённый текст в верхнем регистре.

4 пункт

Изображение выглядит как текст

Автоматически созданное описание

|  |
| --- |
| /\*  \*\* 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("client <- %s",str);  if (send(s2, str, n, 0) < 0) {  perror("send");  done = 1;  }  } while (!done);  close(s2);  }  return 0;  } |

Всё работает

Попробуем подключить несколько клиентов.

Изображение выглядит как текст, снимок экрана, экран

Автоматически созданное описание

Можно сделать вывод, что данный эхо сервер не предназначен для подключения нескольких клиентов. Чтобы он мог работать в многопотоке необходимо добавить или fork или exec.

5 пункт допольнительный

Для каждого клиента будем делать свой дочерний процесс сервера. Вот что получилось:

Код в папке)

Изображение выглядит как текст

Автоматически созданное описание