# Operating System Lab\_03

22CST 蒋云翔 2022102330

## Task1:

### Idea:

1. Read the lab3 file carefully and understand the core of the connection between the server and client. Under such a situation, the task will be easy to deal with.(***Tips: ①Socket creating ②Bind ③Listen ④Accept the client***)

日程表

描述已自动生成

1. According to the execution result, I design an interesting test case, which is a cute talking with teacher Bobo!!!!😊

文本

中度可信度描述已自动生成

1. I think two of the most important things which is may be ignored is that: ①IP ②Port, these two must be the same.

文本

低可信度描述已自动生成

### Code structure

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <sys/socket.h>  #include <netinet/in.h>  #include <unistd.h>  // Author:Yusi 2022102330  // Time:2024/4/19  #define MAX 80  #define PORT 8080  #define SA struct sockaddr  void func(int sockfd)  {  char buff[MAX];  int n;  for (;;) {  bzero(buff, MAX);  //read the data from client  read(sockfd, buff, sizeof(buff));  printf("From client: %s", buff);    // exit the loop if receiver "exit" message  if (strncmp(buff, "exit", 4) == 0) {  printf("Server Exit...\n");  write(sockfd, buff, sizeof(buff));  break;  }    else if (strncmp(buff, "hi", 2) == 0) {  strcpy(buff, "Hello\n");  }  else if (strncmp(buff, "Who are u, sir???",12) == 0) {  strcpy(buff, "I'm your OS Teacher He!!!!\n");  }  else if (strncmp(buff, "Omg!!I did not recognize u teacher Bobo!!",40) == 0) {  strcpy(buff, "Keep my handsome face in your mind, boy!\n");  }  printf("\tTo client: %s", buff);  // Send message to client  write(sockfd, buff, sizeof(buff));  bzero(buff, sizeof(buff));  }  }  int main()  {  int sockfd, connfd, len;  struct sockaddr\_in servaddr, cli;  // Create socket  sockfd = socket(AF\_INET, SOCK\_STREAM, 0);  if (sockfd == -1) {  printf("socket creation failed...\n");  exit(0);  }  else  printf("Socket successfully created..\n");  bzero(&servaddr, sizeof(servaddr));  // assign IP and Port  servaddr.sin\_family = AF\_INET;  servaddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");  servaddr.sin\_port = htons(PORT);  // Bind socket  if ((bind(sockfd, (SA\*)&servaddr, sizeof(servaddr))) != 0) {  printf("socket bind failed...\n");  exit(0);  }  else  printf("Socket successfully binded..\n");  // Listen  if ((listen(sockfd, 5)) != 0) {  printf("Listen failed...\n");  exit(0);  }  else  printf("Server listening..\n");  len = sizeof(cli);  // Accept client  connfd = accept(sockfd, (SA\*)&cli, &len);  if (connfd < 0) {  printf("server acccept failed...\n");  exit(0);  }  else  printf("server acccept the client...\n");  func(connfd);  close(sockfd);  } |

### Execution results

|  |
| --- |
|  |

## Task2:

### Idea:

1. Very similar to Task1, the only difference is to listen to more client. Teacher Bobo gave us a hint to use fork() function to realize the problem, so do I.

文本

描述已自动生成

2. ***Question:*** All my clients can exit perfectly, but my server seems can not exit correctly seeing the execution results below. I’m wondering what’s wrong with my code????

### Code Structure

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <sys/socket.h>  #include <netinet/in.h>  #include <unistd.h>  // Author: Yusi 2022102330  // Date: 2024/4/19  #define MAX 80  #define PORT 8080  #define SA struct sockaddr  void func(int sockfd) {  char buff[MAX];  int n;  for (;;) {  bzero(buff, sizeof(buff));  read(sockfd, buff, sizeof(buff));  printf("From client: %s", buff);  if (strncmp(buff, "exit", 4) == 0) {  printf("Server Exit...\n");  write(sockfd, buff, sizeof(buff));  break;  }  write(sockfd, buff, sizeof(buff));  }  }  int main() {  int sockfd, connfd, len;  struct sockaddr\_in servaddr, cli;  // socket create and verification  sockfd = socket(AF\_INET, SOCK\_STREAM, 0);  if (sockfd == -1) {  printf("socket creation failed...\n");  exit(0);  }  else  printf("Socket successfully created..\n");  bzero(&servaddr, sizeof(servaddr));  // assign IP, PORT  servaddr.sin\_family = AF\_INET;  servaddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");  servaddr.sin\_port = htons(PORT);  // Binding newly created socket to given IP and verification  if ((bind(sockfd, (SA\*)&servaddr, sizeof(servaddr))) != 0) {  printf("socket bind failed...\n");  exit(0);  }  else  printf("Socket successfully binded..\n");  // Now server is ready to listen and verification  if ((listen(sockfd, 5)) != 0) {  printf("Listen failed...\n");  exit(0);  }  else  printf("Server listening..\n");  len = sizeof(cli);  for (;;) {  // Accept the data packet from client and verification  connfd = accept(sockfd, (SA\*)&cli, &len);  if (connfd < 0) {  printf("server accept failed...\n");  exit(0);  }  else  printf("server accept the client...\n");  // Forking a child process to handle client  if (fork() == 0) {  close(sockfd); // Child process closes listening socket  func(connfd); // Child process handles the client  close(connfd); // Child process closes client socket  exit(0); // Child process exits after handling client  }  else {  close(connfd); // Parent process closes client socket  }  }  // Close the listening socket  close(sockfd);  return 0;  } |

### Execution results

|  |
| --- |
|  |