实验七 进程间通信——Socket

**一、实验目的**

1、了解采用Socket通信的原理。

2、掌握Socket的创建及使用方法。

**二、实验原理**

1、通过Socket进行进程间通信的流程：服务器端先初始化Socket，然后与端口绑定(bind)，对端口进行监听(listen)，调用accept阻塞，等待客户端连接。在这时如果有客户端初始化一个Socket，然后连接服务器(connect)，若连接成功，则客户端与服务器端的连接就建立了。客户端发送数据请求，服务器端接收请求并处理请求，然后把回应数据发送给客户端，客户端读取数据，最后关闭连接，一次交互结束。

2、TCP服务器端依次调用socket()、bind()、listen()之后，就会监听指定的socket地址。TCP客户端依次调用socket()、connect()后将向TCP服务器发送一个连接请求。TCP服务器监听到这个请求后，调用accept()函数接收请求，连接建立成功。之后可以开始网络I/O操作，类同于管道的阻塞读写I/O操作。

**三、实验内容**

网络服务器通常用fork来同时服务多个客户端，父进程专门负责监听端口，每次accept一个新的客户端连接，就fork出一个子进程专门服务这个客户端。但子进程退出时会产生僵尸进程，最简单的办法就是父进程直接忽略SIGCHLD信号。

1、编写服务端service.c源程序，实现如下功能：

1）创建套接字，绑定一个端口，监听套接字，忽略子进程的退出消息。

2）监听端口等待客户端连接。循环检查否有客户连接到服务器，如果有，则调用fork创建一个子进程来处理请求。

3）读取客户端发送的数据并向客户端回送数据，回送数据内容为：客户端发送的数据内容以及系统时间。接着关闭套接字并退出。

2、编写客户端client.c源程序，实现如下功能：

1）创建Socket并连接服务器，如果连接成功，则向服务器发送数据。

2）接收并输出显示服务器回传的数据，关闭套接字并退出。服务器端则继续监听端口等待其他客户端下一次连接请求。

3、编译并执行服务器端和客户端代码。

**四、实验代码**

**Service.c**

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<sys/types.h>**

**#include<sys/socket.h>**

**#include<netinet/in.h>**

**#include<arpa/inet.h>**

**#include<unistd.h>**

**#include<signal.h>**

**#include<string.h>**

**#include<time.h>**

**#include<assert.h>**

**#define SERVER\_PORT 6666**

**int sockfd;**

**void request\_handler(int fd);**

**void sigint\_handler(int num);**

**char \*get\_time();**

**int main(int argc, char \*argv[]){**

**// ignore sigchld**

**signal(SIGCHLD, SIG\_IGN);**

**signal(SIGINT, sigint\_handler);**

**sockfd = socket(AF\_INET, SOCK\_STREAM, 0);**

**assert(sockfd != -1);**

**struct sockaddr\_in serv\_addr;**

**memset(&serv\_addr, 0, sizeof(serv\_addr));**

**serv\_addr.sin\_family = AF\_INET;**

**serv\_addr.sin\_addr.s\_addr = inet\_addr("0.0.0.0");**

**serv\_addr.sin\_port = htons(SERVER\_PORT);**

**assert(bind(sockfd, (struct sockaddr\*)&serv\_addr, sizeof(serv\_addr)) != -1);**

**assert(listen(sockfd, 20) != -1);**

**printf("run tcp server in %s:%d ......\n", "127.0.0.1", SERVER\_PORT);**

**struct sockaddr\_in clnt\_addr;**

**socklen\_t clnt\_addr\_size = sizeof(clnt\_addr);**

**while(1){**

**int cfd = accept(sockfd, (struct sockaddr\*)&clnt\_addr, &clnt\_addr\_size);**

**printf("[%s] connect from %s:%d\n", get\_time(), inet\_ntoa(clnt\_addr.sin\_addr), clnt\_addr.sin\_port);**

**pid\_t pid = fork();**

**if(pid == 0){**

**signal(SIGCHLD, SIG\_DFL);**

**signal(SIGINT, SIG\_DFL);**

**request\_handler(cfd);**

**exit(0);**

**}**

**}**

**close(sockfd);**

**return 0;**

**}**

**void request\_handler(int fd){**

**char buffer[64];**

**int n = -1;**

**char \*time = get\_time();**

**write(fd, time, strlen(time));**

**write(fd, "\n", 1);**

**while((n = read(fd, buffer, 64)) > 0){**

**write(fd, buffer, n);**

**}**

**close(fd);**

**}**

**void sigint\_handler(int num){**

**close(sockfd);**

**exit(0);**

**}**

**char \*get\_time(){**

**time\_t t = time(NULL);**

**char \*time = ctime(&t);**

**char \*res = time;**

**while(\*time != '\n'){**

**time++;**

**}**

**\*time = '\0';**

**return res;**

**}**

**Client.c**

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<sys/types.h>**

**#include<sys/socket.h>**

**#include<netinet/in.h>**

**#include<arpa/inet.h>**

**#include<unistd.h>**

**#include<signal.h>**

**#include<string.h>**

**#include<assert.h>**

**#define SERVER\_ADDR "39.107.83.159"**

**#define SERVER\_PORT 6666**

**int main(int *argc*, char \**argv*[]){**

**int sockfd = socket(AF\_INET, SOCK\_STREAM, 0);**

**assert(sockfd != -1);**

**struct sockaddr\_in serv\_addr;**

**memset(&serv\_addr, 0, sizeof(serv\_addr));**

**serv\_addr.sin\_family = AF\_INET;**

**serv\_addr.sin\_addr.s\_addr = inet\_addr(SERVER\_ADDR);**

**serv\_addr.sin\_port = htons(SERVER\_PORT);**

**socklen\_t addrlen = sizeof(serv\_addr);**

**assert(connect(sockfd, (struct sockaddr \*)&serv\_addr, addrlen) != -1);**

**char \*data = "there is client!\n";**

**write(sockfd, data, strlen(data));**

**char buffer[256];**

**int n = -1;**

***// read server's time info***

**if((n = read(sockfd, buffer, 64)) > 0){**

**write(1, buffer, n);**

**}**

***// read server's data***

**if((n = read(sockfd, buffer, 64)) > 0){**

**write(1, buffer, n);**

**}**

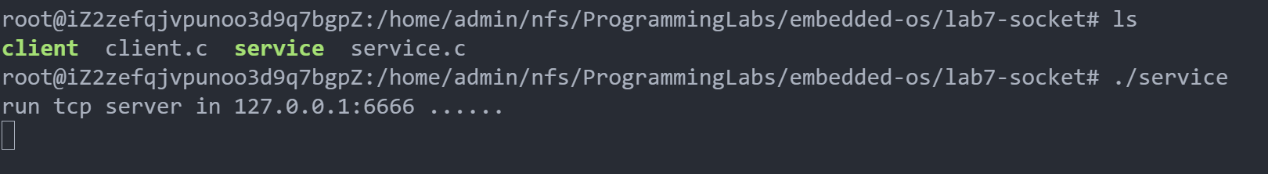
**close(sockfd);**

**return 0;**

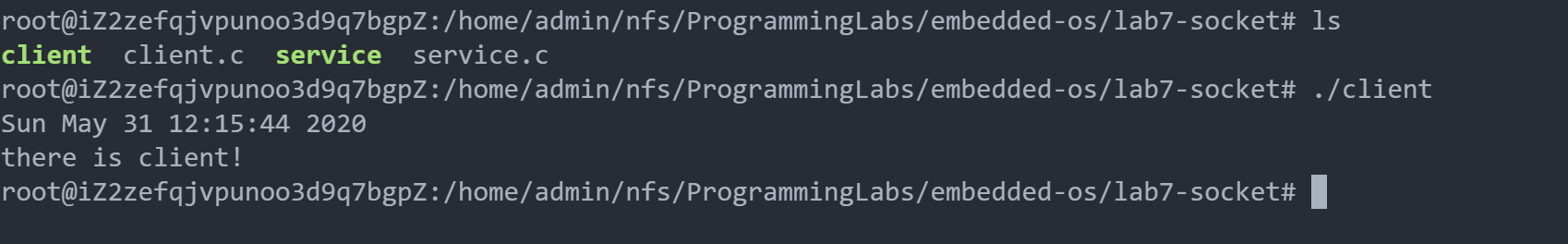
**}**

**截图**

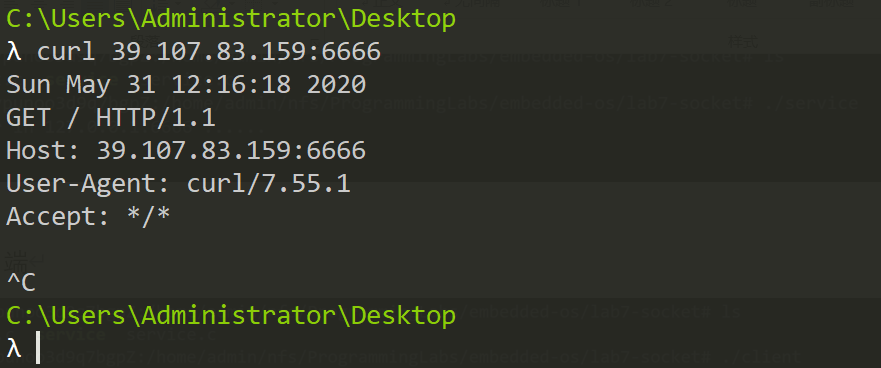
1. 运行服务端



1. 运行客户端



1. Curl 测试



4 结果

