[toc]

课上测试

ch05

作业题目：网络编程

完成下面任务（29分）

1 在 Ubuntu 或 openEuler 中完成任务（推荐openEuler）

2 参考《head first C》实现knock knock服务器，提交代码knock.c，编译运行过程（13分）

3 使用多线程实现knock knock服务器,提交代码knockmt.c,编译运行过程,至少两个客户端测试，服务器运行结果中要打印线程id（13分）

4 提交git log结果（3分）

作业提交要求 (1')

0. 记录实践过程和 AI 问答过程，尽量不要截图，给出文本内容
1. (选做)推荐所有作业托管到 gitee或 github 上
2. (必做)提交作业 markdown文档，命名为“学号-姓名-作业题目.md”
3. (必做)提交作业 markdown文档转成的 PDF 文件，命名为“学号-姓名-作业题目.pdf”

- github链接

作业内容

参考《head first C》实现knock knock服务器，提交代码knock.c，编译运行过程

- 代码knock.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <sys/types.h>

#define MAX_PORTS 3           // 端口序列的长度
#define TIMEOUT 5             // 每个端口的超时时间（秒）
#define BUFFER_SIZE 1024      // 数据缓冲区大小

// 预定义端口序列
int knock_sequence[MAX_PORTS] = {9001, 9002, 9003};
```

```
// 检查客户端是否按顺序连接指定端口
void start_knock_server() {
    int server_sockets[MAX_PORTS], client_socket;
    struct sockaddr_in server_addr, client_addr;
    socklen_t client_len = sizeof(client_addr);
    char buffer[BUFFER_SIZE];
    int i, current_port;

    // 初始化所有服务器端口
    for (i = 0; i < MAX_PORTS; i++) {
        server_sockets[i] = socket(AF_INET, SOCK_STREAM, 0);
        if (server_sockets[i] == -1) {
            perror("Socket creation failed");
            exit(1);
        }

        server_addr.sin_family = AF_INET;
        server_addr.sin_addr.s_addr = INADDR_ANY;
        server_addr.sin_port = htons(knock_sequence[i]);

        if (bind(server_sockets[i], (struct sockaddr *)&server_addr,
sizeof(server_addr)) < 0) {
            perror("Bind failed");
            exit(1);
        }

        if (listen(server_sockets[i], 1) < 0) {
            perror("Listen failed");
            exit(1);
        }

        printf("Listening on port: %d (step %d)\n", knock_sequence[i], i +
1);
    }

    printf("Waiting for knock sequence...\n");

    // 依次等待客户端按顺序连接各个端口
    for (i = 0; i < MAX_PORTS; i++) {
        printf("Waiting for connection on port %d...\n", knock_sequence[i]);
        client_socket = accept(server_sockets[i], (struct sockaddr
*)&client_addr, &client_len);

        if (client_socket < 0) {
            perror("Accept failed");
            exit(1);
        }

        printf("Knock received on port %d from %s\n", knock_sequence[i],
inet_ntoa(client_addr.sin_addr));
        close(client_socket);
        close(server_sockets[i]);
    }
```

```
    printf("Knock sequence correct! Access granted.\n");
}

int main() {
    printf("Starting Knock Server...\n");
    start_knock_server();
    return 0;
}
```

- 编译运行过程
  - 服务器端

```
root@Youer:~/shiyan/test/bestidiocs2024/ch05/network/test# nano
knock_server.c
root@Youer:~/shiyan/test/bestidiocs2024/ch05/network/test# gcc -o
knock_server knock_server.c
root@Youer:~/shiyan/test/bestidiocs2024/ch05/network/test# ./knock_server
Starting Knock Server...
Listening on port: 9001 (step 1)
Listening on port: 9002 (step 2)
Listening on port: 9003 (step 3)
Waiting for knock sequence...
Waiting for connection on port 9001...
Knock received on port 9001 from 127.0.0.1
Waiting for connection on port 9002...
Knock received on port 9002 from 127.0.0.1
Waiting for connection on port 9003...
Knock received on port 9003 from 127.0.0.1
Knock sequence correct! Access granted.
```

  - 模拟的客户端

```
root@Youer:~/shiyan/test/bestidiocs2024/ch05/network/C# nc 127.0.0.1 9001
^X^C
root@Youer:~/shiyan/test/bestidiocs2024/ch05/network/C# nc 127.0.0.1 9002
^C
root@Youer:~/shiyan/test/bestidiocs2024/ch05/network/C# nc 127.0.0.1 9003
^C
```

- 代码实现符合需求

使用多线程实现knock knock服务器,提交代码knockmt.c,编译运行过程,至少两个客户端测试，服务器运行结果中要打印线程id

- 代码knockmt.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <errno.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <pthread.h>
#include <stdint.h> // For intptr_t

#define PORT1 9001
#define PORT2 9002
#define PORT3 9003
#define MAX_CLIENTS 5

void *handle_client(void *arg) {
    int client_socket = *(int *)arg;
    char buffer[1024] = {0};
    pthread_t tid = pthread_self();

    printf("Thread ID: %lu - Client connected.\n", tid);
    read(client_socket, buffer, sizeof(buffer));
    printf("Thread ID: %lu - Message received: %s\n", tid, buffer);

    close(client_socket);
    printf("Thread ID: %lu - Client disconnected.\n", tid);
    pthread_exit(NULL);
}

int start_server(int port) {
    int server_fd, new_socket;
    struct sockaddr_in address;
    int addrlen = sizeof(address);
    int opt = 1;

    // 创建 socket 文件描述符
    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
        perror("Socket failed");
        exit(EXIT_FAILURE);
    }

    // 设置 SO_REUSEADDR 选项，避免 "Address already in use" 错误
    if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR, &opt, sizeof(opt)))
{
        perror("setsockopt failed");
        exit(EXIT_FAILURE);
    }

    // 绑定端口和 IP 地址
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(port);
```

```
    if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0) {
        perror("Bind failed");
        close(server_fd);
        exit(EXIT_FAILURE);
    }

    if (listen(server_fd, MAX_CLIENTS) < 0) {
        perror("Listen failed");
        close(server_fd);
        exit(EXIT_FAILURE);
    }

    printf("Server listening on port %d...\n", port);

    // 接收客户端连接
    while (1) {
        if ((new_socket = accept(server_fd, (struct sockaddr *)&address,
(socklen_t *)&addrlen)) < 0) {
            perror("Accept failed");
            continue;
        }

        pthread_t thread_id;
        if (pthread_create(&thread_id, NULL, handle_client, (void
*)&new_socket) != 0) {
            perror("Thread creation failed");
            close(new_socket);
        }
        pthread_detach(thread_id);
    }

    close(server_fd);
    return 0;
}

int main() {
    pthread_t t1, t2, t3;

    // 创建三个线程分别监听三个端口
    if (pthread_create(&t1, NULL, (void *)start_server, (void *)
(intptr_t)PORT1)) {
        perror("Thread 1 failed");
        exit(EXIT_FAILURE);
    }
    if (pthread_create(&t2, NULL, (void *)start_server, (void *)
(intptr_t)PORT2)) {
        perror("Thread 2 failed");
        exit(EXIT_FAILURE);
    }
    if (pthread_create(&t3, NULL, (void *)start_server, (void *)
(intptr_t)PORT3)) {
        perror("Thread 3 failed");
        exit(EXIT_FAILURE);
```

```
    }

    // 等待线程结束
    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
    pthread_join(t3, NULL);

    return 0;
}
```

- 编译运行过程,至少两个客户端测试
  - 服务器端

```
root@Youer:~/shiyan/test/bestidiocs2024/ch05/network/test# nano
knockmt.c
root@Youer:~/shiyan/test/bestidiocs2024/ch05/network/test# gcc -o
knockmt knockmt.c -pthread
root@Youer:~/shiyan/test/bestidiocs2024/ch05/network/test# ./knockmt
Server listening on port 9001...
Server listening on port 9002...
Server listening on port 9003...
Thread ID: 140081700533824 - Client connected.
Thread ID: 140081692141120 - Client connected.
Thread ID: 140081700533824 - Message received: ����
Thread ID: 140081700533824 - Client disconnected.
Thread ID: 140081700533824 - Client connected.
Thread ID: 140081700533824 - Message received: ����
Thread ID: 140081700533824 - Client disconnected.
Thread ID: 140081692141120 - Message received: ����
Thread ID: 140081692141120 - Client disconnected.
Thread ID: 140081692141120 - Client connected.
```

  - 通过多个窗口模拟多个客户端
  - 模拟的客户端01

```
root@Youer:~/shiyan/test/bestidiocs2024/ch05/network/C# telnet
127.0.0.1 9001
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
^CConnection closed by foreign host.
root@Youer:~/shiyan/test/bestidiocs2024/ch05/network/C# telnet
127.0.0.1 9003
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
^CConnection closed by foreign host.
```

    - 模拟的客户端02

```
root@Youer:~# telnet 127.0.0.1 9002
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
^CConnection closed by foreign host.
root@Youer:~# telnet 127.0.0.1 9001
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
```

- 代码实现符合需求

提交git log结果

- git log 结果

```
root@Youer:~/shiyan/test/bestidiocs2024/ch05/network/test# git log
commit 1b9a84fe3bdd93e36c6cd2750d0459f81f9f1e22 (HEAD -> master)
Author: 徐鹿鸣 <xlm20040219@qq.com>
Date:   Tue Dec 17 10:52:49 2024 +0800

    finish knockmt.c

commit 7e68502d90abed3e85aede55fdf2207671b9e06b
Author: 徐鹿鸣 <xlm20040219@qq.com>
Date:   Tue Dec 17 10:39:25 2024 +0800

    finish knock.c
```