

自行查阅相关资料,在 RHEL7.4 下初步掌握用 C/C++ 语言基于 TCP 非阻塞方式(也称为 TCP 异步方式)的 socket 编程的相关知识点并将答案写成文档:

注:本次所有作业,都是仅需要程序即可,不需要进程(写成进程也可以)

1、不允许分裂子进程

2、不允许使用线程

1、每个人的目录结构要求如下(假设学号为 1551234,各人按实修改):首先建立“学号-000108-server”(存放所有 server 端程序)和“学号-000108-client”子目录(存放所有 client 端程序),作业目录可位于任意子目录下,下面再建立若干空的子目录,示例如下:

```
1551234-000108-server      1551234-000108-client
|-- 01                      |-- 01
|-- ..                      |-- ..
|-- ..                      |-- ..
|-- 05                      |-- 05
```

2、(01 子目录)写一对 TCP Socket 的测试程序,分为 client 和 server,分别运行在不同虚拟机上

- 测试程序 tcp_server1-1(源程序名任意,允许多个,C/C++语言任选,make 后得到 tcp_server1 即可,下同),运行后绑定某个 TCP 端口号,并进入等待连接状态(下面称为 LISTEN 状态),要求端口号通过 main 函数带参数的方式传入(例:./tcp_server1 4000 表示绑定 TCP 4000 端口)
- 测试程序 tcp_client1-1,运行时带入服务端 IP 地址及端口号,即可向服务端发起连接,要求 IP 地址、端口号通过 main 函数带参数的方式传入(例:./tcp_client1 192.168.80.230 4000 则表示连接 192.168.80.230 的 TCP 4000 端口)
- Server 端用于 listen 的 socket,不设置为非阻塞方式,accept 成功后,将 accept 的 socket 设置为非阻塞方式
- Client 端建立的 socket,先不设置为非阻塞方式,待 connect 成功后,再设置为非阻塞方式
- 连接成功后,双方均进入 read(recv)状态,read(recv)函数后直接关闭 socket,程序退出
- read(recv)函数的表现会如何?程序会阻塞在 read(recv)还是立即结束? read(recv)函数返回什么?
- 测试程序 tcp_server1-2/tcp_client1-2,在 1-1 的基础上,用 select 使 read(recv)停下来而不立即返回
- 测试程序 tcp_server1-3,要求 socket 建立成功后,先设置非阻塞方式,再进行 bind、listen 和 accept,accept 的新 socket,也是立即设置为非阻塞方式,再进行后续操作
- 测试程序 tcp_client1-3,要求 socket 建立成功后,先设置非阻塞方式,再 connect
- 要求 tcp_client1-3 能连接 tcp_server1-3 成功,并在连接成功后,用 select 使 read(recv)停下来而不立即返回

注:后续所有小题,均要求 socket 建立后立即设置非阻塞方式,再进入后续操作

★ Server 端用于监听端口的 socket,顺序为 socket、设非阻塞、bind、listen、...

★ Server 端 accept 得到的新 socket,先设置为非阻塞,再进行后续的 read/write 等操作

★ Client 端的 socket,顺序为 socket、设非阻塞、connect、...

3、(02 子目录)写一对 TCP Socket 的测试程序,分为 client 和 server,分别运行在不同虚拟机上

- 测试程序 tcp_server2-1/tcp_client2-1,连接成功后,client 发数据(每次 10 字节,间隔 1 秒),server 用大小 100 的缓冲区收数据,死循环运行
- 此时在 client(server)端按 CTRL+C,server(client)端能否检测到连接中断?

- 如果新开一个会话窗口，用 `kill -9` 杀 `client (server)` 端程序，`server (client)` 端能否检测到连接中断？
 - 测试程序 `tcp_server2-2/tcp_client2-2`，连接成功后，`server` 发数据（每次 10 字节，间隔 1 秒），`client` 用大小 100 的缓冲区收数据，死循环运行（同时观察 2-1 的两种中断检测方式）
- 4、（03 子目录）写一对 TCP Socket 的测试程序，分为 `client` 和 `server`，分别运行在不同虚拟机上
- 测试程序 `tcp_server3-1/tcp_client3-1`，连接成功后，`server` 发数据（每次 10 字节，间隔 1 秒）并同时用大小 100 的缓冲区收数据，`client` 发数据（每次 15 字节，间隔 3 秒）并同时用大小 100 的缓冲区收数据，死循环运行
 - 注意：死循环中不是先发送若干字节、延时、再收若干字节，而是读写并发，以 `client` 端为例，相当于打印信息为收 3 次，发 1 次（也可能是其它收发值）
 - 测试程序 `tcp_server3-2/tcp_client3-2`，连接成功后，`server` 发数据（每次 10 字节，间隔 1 秒）并同时用大小为 88 的缓冲区收数据，能否在非阻塞模式下保证每次必须收到 88 字节才返回，即每次 `read` 或 `recv` 函数均返回读取了 88 字节？（注：不允许采用自己写循环保证读满 88 字节）`client` 发数据（每次 15 字节，间隔 3 秒）并同时用大小 100 的缓冲区收数据，死循环运行
- 5、（04 子目录）写一对 TCP Socket 的测试程序，分为 `client` 和 `server`，分别运行在不同虚拟机上
- 测试程序 `tcp_server4-1`，接受 `client` 的连接成功后，用 `getchar()` 进入暂停运行状态
 - 测试程序 `tcp_client4-1`，连接服务端成功后，用 `write` 向服务端不断写入，直到 `write` 失败为止，用 `netstat` 观察读写队列的值
 - 测试程序 `tcp_server4-2`，接受 `client` 的连接成功后，用每读 20 字节就延时 1 秒的方法循环读数据
 - 测试程序 `tcp_client4-2`，连接服务端成功后，用 `write` 向服务端不断大量写入，直到 `write` 失败为止，用 `netstat` 观察读写队列的值，`write` 失败后，如何重新恢复为继续写状态？
- 6、（05 子目录）写一对 TCP Socket 的测试程序，分为 `client` 和 `server`，分别运行在不同虚拟机上
- 测试程序 `tcp_server5`，接受连接成功后，`server` 发数据（每次 10 字节，间隔 1 秒）并同时用大小 100 的缓冲区收数据，死循环运行
 - 测试程序 `tcp_client5-1`，连接成功后，`client` 发数据（每次 15 字节，间隔 3 秒）并同时用大小 100 的缓冲区收数据，死循环运行
 - `Server` 端先接受一个 `Client` 端的连接，进入死循环读写状态
 - 要求此时 `Server` 能接受一个新的 `Client` 端的连接，也进入死循环读写状态（`Server` 端一个程序维护一个 `listen socket` 和多个 `accept` 的 `socket`，要保证 `accept` 的 `socket` 进入死循环读写的同时，仍然能接受新的 `Client` 端的连接）
 - 用两个会话窗口分别启动两个 `tcp_server5`（如：`./tcp_server5 4000 ./tcp_server5 5000`），
 - 测试程序 `tcp_client5-2`，运行时带两个端口号（例：`./tcp_client5-2 192.168.80.230 4000 5000`），表示在一个程序中建立两个 `socket`，分别连接两个不同端口号的 `server` 端，`client` 发数据（每次 15 字节，间隔 3 秒）并同时用大小 100 的缓冲区收数据，死循环运行，允许在不同会话窗口启动多个 `client` 端
 - `Server` 端和 `Client` 端均不允许采用分裂进程的方式，只能是一个程序

【注：】1、每个示例程序都写好 makefile 文件，一次 make 形成多个可执行文件

2、本次作业需要打开多个 SecureCRT 的会话窗口观察信息，建议在屏幕上平铺，以便同时观察各个窗口的输出信息

【本次作业的统一批改方法说明：】

- 1、每个人的目录结构要求如下（假设学号为 1551234，各人**按实修改**）：首先建立“学号-000108-server”子目录“学号-000108-client”子目录，下面再建立 01-05 的子目录，示例如下：

1551234-000108-server	1551234-000108-client
-- 01	-- 01
-- ..	-- ..
-- ..	-- ..
-- 05	-- 05
~-- makefile	~-- makefile

（每位同学的总 makefile 文件，make 后能生成所有子目录下的可执行文件）

- 2、提交作业时，每位同学上交一个 linux-tcp-socket-async.tar.bz2 文件，解压后能得到上述的完整目录结构，截止时间到后，会从每人的交作业目录中复制出来，全部放在 total-000108 目录中

示例如下：

total-000108	
-- 1551234-linux-socket-async.tar.bz2	（第 1 位同学的作业压缩包）
...	
~-- 1554321-linux-socket-async.tar.bz2	（最后 1 位同学的作业压缩包）

依次解压后，能得到如下目录结构：

total-000108	
-- 1551234-000108-client	（第 1 位同学的 client 作业目录）
-- 1551234-000108-server	（第 1 位同学的 server 作业目录）
...	
-- 1554321-000108-client	（最后 1 位同学的 client 作业目录）
-- 1554321-000108-server	（最后 1 位同学的 server 作业目录）

- 3、进入 total-000108 目录，进行一次 make，就能生成所有可执行文件，示例如下：

total-000108	
-- 1551234-000108-client	（第 1 位同学的 client 作业目录）
-- 1551234-000108-server	（第 1 位同学的 server 作业目录）
...	
-- 1554321-000108-client	（最后 1 位同学的 client 作业目录）
-- 1554321-000108-server	（最后 1 位同学的 server 作业目录）
~-- makefile	（老师事先建好的 makefile 文件，准备编译所有同学的本次作业，具体的实现方式是进入到每个学号对应的目录后调用该目录下的总 makefile）

- 4、无法顺利编译则不能得分，对应学号及子目录名错则不能得分

- 5、作业提交时清除所有的中间文件及生成的可执行文件、源程序备份文件等

【作业要求：】

- 1、**11 月 5 日前**网上提交
- 2、每题所占平时成绩的具体分值见网页
- 3、超过截止时间提交作业则不得分

下面还有...

本次作业的提示：

- 1、本次作业的重点在于掌握 select 函数的使用
- 2、设置非阻塞方式后，accept、read、write 均即时返回，因此只有在需要 accept、read、write 的时候才能去做相应操作（如何判断需要？）
- 3、accept 和 read，因为是接受对端信息，置 select 的 readfds 是容易理解的，网络上的很多资料说 write 不需要置 writefds，其实是不对的（例：连接成功后，一端 getchar 暂停，另一端持续 write 直到缓冲区满，此时 write 会失败，但如果置 writefds 则 select 不会返回可写）
- 4、如果在死循环中，先 select 再 sleep 来达到读写并发并且若干秒后发数据，是不正确的，因为 select 的延时不确定，取决于对端，而且 select 和 sleep 是顺序，不是并发，即 sleep 过程中若有对端数据到来则无法即时处理。正确的做法是 select+设置定时器，当定时器到后会发送信号，导致 select 返回<0，处理定时器信号即可（操作系统的中断概念）
- 5、select 返回<0/=0，不代表一定是错，要分情况处理