§0.Linux 知识补充 - Socket 编程 - TCP 阻塞方式 - 基本知识

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

0、补充知识

- 将 RHEL7 虚拟机克隆一个/多个，新的虚拟机如何设置网卡并使生效？
- RHEL7 的虚拟机中，如何在一个网卡上设置多地址（多地址要求属于不同网段，且两台虚拟机间相应的同网段地址能 ping 通）【注意：不是在 VMware 中增加一个网卡，而是仅有一张网卡的情况下如何设置多地址】

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-000107"子目录（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：

```
1551234-000107
  |-- 01
  |-- ..
  |-- ..
  |-- 06
```

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（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）
- 如果服务端绑定的端口号已被使用（比如两次运行 ./tcp_server1 4000 或 ./tcp_server1 80），则无法进入 LISTEN 状态，会在哪个函数上出错？
- 测试程序 tcp_client1，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）
- 如果 client 端连接时的 IP 地址不正确（例如不存在的 IP 地址），会在哪一步出错？如果连接的端口号不正确，会在哪一步出错？
- 连接成功后，双方给出相应的提示信息，双方均进入 read(recv)状态，此时 read/recv 函数会阻塞
- 连接成功后，用 CTRL+C 中断 client（server）端，Server（client）端能否能侦测到连接已中断？
- 连接成功后，用 kill -9 杀死 client（server）端，Server（client）端能否能侦测到连接已中断？（另外启动一个 SecureCRT 的会话来做 kill）
- 在双方连接成功后，再新的会话中再启动一个 tcp_client1 连接 server，会出现什么情况？
- tcp_sevrer1 运行终止后，立即再次启动，绑定相同端口号，能否成功？（REUSEADDR 选项的作用，加或不加的区别是什么？）

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

- 前提：第 1 小题中，client 每次连接 server 时，client 的端口号是随机分配的
- 测试程序 tcp_server2，与 tcp_server1 功能相同，但接受连接时打印 client 端的 IP 地址和端口号
- 测试程序 tcp_client2，要求连接 server 端的时候使用固定端口号，通过 main 函数带参数的方式传入（例：./tcp_client2 12345 192.168.80.230 4000 则表示 client 的 12345 端口连接 server 的 4000 端口）

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
- 两台用于测试的 RHEL7 虚拟机均设置多个地址（例如：192.168.80.230/ 172.18.12.230）
- 测试程序 tcp_server3，能读到本机所有网卡的所有 IP 地址，然后只绑定其中的某个 IP 地址的某个端口，要求 IP 地址和端口号通过 main 函数带参数的方式传入（例：./tcp_sevrer3 172.18.12.230 4000 表示只绑定 172.18.12.230 的 4000 端口）
- 测试程序 tcp_client3，运行时带入服务端 IP 地址及其中任意一个端口号，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_client3 172.18.12.230 4000 则表示连接 172.18.12.230 的 TCP 4000 端口）
- 如果 tcp_client3 连接未绑定的 IP 地址（例如：192.168.80.230），会怎样？

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 read 函数一次读 20 字节（此时应进入阻塞状态，即 read 函数执行后，不读满 20 字节一直不返回，如何做到？注：不允许采用自己写循环保证读满 20 字节）
- 测试程序 tcp_client4-1，连接服务端成功后，用 write 函数向服务端写入 20 字节，要求每次写两字节，然后延时 1 秒，再写 2 字节…，观察 server 端的 read 函数何时返回并执行后续语句，打印 read 函数读到的内容，是否与 client 发送的内容相同？
- 测试程序 tcp_server4-2/tcp_client_4-2，将 read/write 换成 recv/send 函数，用法是否相同？结果是否相同？
- 给出 read/recv 函数的使用区别，给出 write/send 函数的使用区别

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_server5-1，接受 client 的连接成功后，用一句 getchar()进入等待输入状态
- 测试程序 tcp_client5-1，连接服务端成功后，用 write 函数不断向服务端写入数据（加计数器统计写入了多少字节），大约写入多少字节后会使 write 函数不再返回（阻塞状态）
- server 端在 getchar()后用 read 进行读（假设每次读 n 个字节），读入多少字节后，client 端的 write 函数可以返回？这说明了什么问题？
- 在整个过程中用新会话打开终端后，用 netstat 命令观察 tcp 连接的各种信息（netstat 可以带哪些参数？显示的内容代表什么？）
- 测试程序 tcp_server5-2/tcp_client_5-2，双方角色互换，即 server 写至阻塞为止，然后 client 开始读，直到 server 端解除阻塞，观察整个过程
- 测试程序 tcp_server5-3/tcp_client_5-3，功能同 5-1，在其中通过设置函数改变 TCP 收发缓冲区大小，通过 netstat 观察整个过程

7、（06 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_server6-1，接受 client 的连接成功后，进入死循环，死循环中先 read 再 write 反复进行；测试程序 tcp_client6-1，连接服务端成功后，也进入死循环，死循环中同样先 read 再 write 反复进行，此时双方能否正常收发数据？
  - 用 main 函数带参数方式带入每次读写的字节数
    例：./tcp_server 4000 1000 900 表示绑定端口 4000，每次读 1000 字节，写 900 字节
    ./tcp_client 192.168.80.230 4000 800 700 表示连接 192.168.80.230 的 4000 端口，每次读 800，写 700
  - 假设双方每次 read/write 都是 1000 字节
- 测试程序 tcp_server6-2，接受 client 的连接成功后，进入死循环，死循环中先 write 再 read 反复进行；测试程序 tcp_client6-2，连接服务端成功后，也进入死循环，死循环中也是先 write 再 read 反复进行，此时双方能否正常收发数据？
  - 用 main 函数带参数方式带入每次读写的字节数，格式规则同上
  - 假设双方每次 read/write 都是 1000 字节
  - 假设 server 端每次 read 1000 字节/write 500 字节，client 端每次 read 500 字节/write

1000 字节
    - 假设 server 端每次 read/write 是 1000 字节，client 端每次 read/write 是 700 字节
- 测试程序 tcp_server6-3，接受 client 的连接成功后，进入死循环，死循环中先 write 再 read 反复进行；测试程序 tcp_client6-3，连接服务端成功后，也进入死循环，死循环中先 read 再 write 反复进行，此时双方能否正常收发数据？
    - 用 main 函数带参数方式带入每次读写的字节数，格式规则同上
    - 假设双方每次 read/write 都是 1000 字节
    - 假设 server 端每次 read 1000 字节/write 500 字节，client 端每次 read 500 字节/write 1000 字节
    - 假设 server 端每次 read/write 是 1000 字节，client 端每次 read/write 是 700 字节
- 测试程序 tcp_server6-4，接受 client 的连接成功后，进入死循环，死循环中先 read 再 write 反复进行；测试程序 tcp_client6-4，连接服务端成功后，也进入死循环，死循环中先 write 再 read 反复进行，此时双方能否正常收发数据？
    - 用 main 函数带参数方式带入每次读写的字节数，格式规则同上
    - 假设双方每次 read/write 都是 1000 字节
    - 假设 server 端每次 read 1000 字节/write 500 字节，client 端每次 read 500 字节/write 1000 字节
    - 假设 server 端每次 read/write 是 1000 字节，client 端每次 read/write 是 700 字节

【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-000107"子目录（可位于任意子目录下），下面再建立 01-05 的子目录，示例如下：

```
1551234-000107
|-- 01
|-- 02
|-- ...
`-- 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-sync.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-000107 目录中

示例如下：

```
total-000107
|-- 1551234-linux-socket-sync.tar.bz2      （第 1 位同学的作业压缩包）
...
`-- 1554321-linux-socket-sync.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-000107
|-- 1551234-000107                    （第 1 位同学的作业目录）
...
`-- 1554321-000107                    （最后 1 位同学的作业目录）
```

3、 进入 total-000107 目录，进行一次 make，就能生成所有可执行文件，示例如下：

```
total-000107
|-- 1551234-000107                    （第 1 位同学的作业目录）
...
```

```
|-- 1554321-000107                    （最后 1 位同学的作业目录）
`-- 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、10 月 29 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