§0.Linux 知识补充 - Socket 编程 - 基础知识综合

基于之前的 socket 编程知识，完成下面的题目：

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-000111"，作业目录可位于任意子目录下，下面不在包含子目录，示例如下：

1551234-000111

【注】：部分同学认为应该分 client/server 两个子目录，其实是不正确的，因为 client 和 server 有很多的公共函数，所以放在一个目录中更好，调试时一键拷贝可执行文件即可

2、程序的基本要求为：一对 socket 测试程序，分为 client 和 server，分别运行在不同虚拟机上，client 向 server 端发起若干连接，每个连接按要求传送数据，双方各自将传送/收到的数据写成文件，比较双方的文件内容，完全相同者为测试通过

- client 端向 server 发起连接，连接的协议既可以是 tcp，也可以是 udp
- 多个连接的处理，既可以是 fork 子进程方式（每个子进程处理一个连接），也可以由一个主进程处理全部连接
- 每个连接的方式，既可以是阻塞，也可以是非阻塞（一个主进程处理多个连接则不能是阻塞方式）
- 如果要求为非阻塞连接，则必须在 socket 建立后立即设置，即 socket 在进行 listen、accept 和 connect 时必须已经是非阻塞方式
- client 和 server 端的连接处理方式（fork/nofork）、阻塞/非阻塞方式均独立设置，不受对端影响（例：client 端 fork+阻塞方式连接，sevver 以非 fork+非阻塞方式处理连接）
- 每一对连接传输的顺序要求如下：
  - S -> C：字符串"StuNo"（不准发尾零,大小写按要求）**（注：UDP 忽略此步骤）**
  - C -> S：Client 进程的所有者学号（4 字节的 int 型，网络序）**（注：UDP 主动发此包）**
  - S -> C：字符串"pid"（不准发尾零，小写）
  - C -> S：**若 Client 端为 fork 方式，则为 client 进程的 pid（4 字节 int 型，网络序）**
    若 Client 端为 nofork 方式，则为 client 进程的（pid<<16+socket_id）（4 字节 int 型，网络序）
  - S -> C：字符串"TIME"（带尾零，大写）
  - C -> S：Client 进程的当前系统时间（"yyyy-mm-dd hh:mm:ss"形式，定长 19 字节，不准发尾零）
  - S -> C：字符串"str*****"**（带尾零，str 小写）**，*****为 5 位随机数字，00001-99999 之间
  - C -> S：长度为*****的随机字符串，每个字符的 ASCII 值范围 0~255 之间
  - S -> C：字符串"end"**（不带尾零，小写）**
  - C -> S：收到 end 后，Client 端主动关闭连接，将发送的四项信息（学号、Client 子进程的 pid 号**（十进制形式）**、Client 端发送的时间戳、长度 1-99999 间的随机字符串）写入"学号.进程号.pid.txt"文件中（例：1551234.3764.pid.txt），文件内容为四行，分别对应四项信息，完成后 client 子进程退出（文件换行符为 Linux 格式）
  - S -> C：Server 端侦测到 client 端已关闭后，关闭 socket，将收到的四项内容写入文件中，文件命名及格式同 client 端
  - 本次测试完成后，将 client 和 server 两端的文件取在一起，每对同名文件按字节比较完全相同则测试通过
  - 任何一步收到的数据错误则中断连接（例：server 发送了"time"，则 client 收到后中断连接）
  - client 必须保证收到"end"的数量与要求的数量一致，即 client 遇到非法终止的连接后要重新发起连接
  - 测试完成后，双方生成文件的数量不匹配、或匹配但多于或少于要求的数量，均视为未通过测试

3、server/client 端程序要求可由命令行带入以下参数（假设可执行文件名为 server/client）：

./server -ip x.x.x.x -port xx -p tcp/udp -block/-nonblock -fork/-nofork

./client -ip x.x.x.x -port xx -num 1-1000 -p tcp/udp -block/-nonblock -fork/-nofork

参数解释如下：

-ip x.x.x.x ： 对 server 而言，表示要绑定的本机 IP 地址，缺省为 0.0.0.0
对 client 而言，表示要链接的服务端 IP 地址，无缺省值

-port xx ： 对 server 而言，表示要 bind 的 TCP/UDP 端口号，无缺省值
对 client 而言，表示要连接的 TCP/UDP 端口号，无缺省值

-num 1-1000 ： 产生的连接数，缺省 500

-p 协议 ： 只有 tcp/udp 两个选项，缺省 tcp

-block/-nonblock ： 选择阻塞/非阻塞方式，缺省 nonblock

-fork/-nofork ： 选择分裂进程/单个进程方式，缺省-nofork

运行示例：

./server -port 4000：
tcp 协议，绑定本机所有 IP 地址
监听 4000 端口
listen 及 accept 的 socket 均为 nonblock
不分裂进程（所有 accept 的 socket 均在一个进程中处理）

./server -port 4000 -ip 192.168.80.230 -fork：
tcp 协议，绑定本机的 192.168.80.230 网卡地址（本机其它网卡不接受连接）
监听 4000 端口
listen 及 accept 的 socket 均为 nonblock
分裂进程方式，每次 accept 一个 socket 即分裂一个进程去单独处理

./client -ip 192.168.80.230 -port 4000 -p udp -fork -block -num 1000：
udp 协议，连接服务器 IP 地址为 192.168.80.230
连接端口为 4000
listen 及 accept 的 socket 均为 block
分裂进程方式，数量为 500 个

【注:】

1、各参数无顺序要求（例： -fork -p tcp ⇔ -p tcp -fork）

2、-nofork 与-block 同时出现时，-block 无效

3、 Server 端为 UDP 协议时，-fork 无效

【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-000111"子目录，下面不再需要子目录

2、提交作业时，每位同学上交一个 linux-socket-test.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-000111 目录中

示例如下：

```
total-000111
|-- 1551234-linux-socket-test.tar.bz2      （第 1 位同学的作业压缩包）
...
`-- 1554321-linux-socket-test.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-000111
|-- 1551234-000111                    （第 1 位同学的 client 作业目录）
...
`-- 1554321-000111                    （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
|-- 1551234-000111                    （第 1 位同学的 client 作业目录）
...
|-- 1554321-000111                    （最后 1 位同学的 client 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、12 月 3 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
4、**本次作业鼓励大家相互测试（即甲的 client 去连接乙的 server）**
5、**本次作业会安排现场测试验收，具体时间另行通知（12.3 后）**
6、**最终的测试方法会是和老师提供的参考版本间互测（即学生的 client 连接老师的 server，老师的 client 连接学生的 server），参考测试版本 11.30 日放出（参考版本会有故意发错数据/故意无理由关闭连接的情况出现）**