**《计算机网络》课程设计**

**周进度报告**

**第四周：实现多个客户端的并发处理**

**学 号 3020244294**

**姓 名 石子跃**

**学 院 智能与计算**

**专 业 网络安全**

**年 级 2020**

**任课教师 仇超**

**2022年 4月 8日**

# 任务需求分析

具体要求： 当服务器在等待一个客户端发送下一个请求时，能够同时处理来自其它客户 端的请求，使服务器能够同时处理多个并发的客户端。

注意： 1) 将服务器能够支持的最大连接数设置为 1024，这是操作系统可用文件描 述符数量的最大值。

2) 客户端可能会“暂停”（即请求发送了一半突然暂停）或出错，但这些问 题不应对其他并发用户产生不良影响。也就是说，如果一个客户端只发 送了请求的一半就停止了，那么服务端应继续为另一个客户端提供服务。

3) 只能通过 select()方法实现并发，禁止使用多线程。使用除 select()外的方 法实现，一经发现，则本周实验报告和自动测试分数均为零分。

# 、二、具体实现

# 

int select(int maxfdp, fd\_set \*readset, fd\_set \*writeset, fd\_set \*exceptset,struct timeval \*timeout);

int FD\_ZERO(int fd, fd\_set \*fdset); //一个 fd\_set类型变量的所有位都设为 0

int FD\_CLR(int fd, fd\_set \*fdset); //清除某个位时可以使用

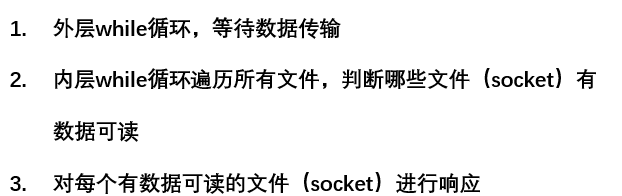
int FD\_SET(int fd, fd\_set \*fd\_set); //设置变量的某个位置位

int FD\_ISSET(int fd, fd\_set \*fdset); //测试某个位是否被置位

select模型的关键是使用一种有序的方式，对多个套接字进行统一管理与调度 。当声明了一个文件描述符集后，必须用FD\_ZERO将所有位置零。

用户首先将需要进行IO操作的socket添加到select中，然后阻塞等待select系统调用返回。当数据到达时，socket被激活，select函数返回。用户线程正式发起read请求，读取数据并继续执行。

从流程上来看，使用select函数进行IO请求和同步阻塞模型没有太大的区别，甚至还多了添加监视socket，以及调用select函数的额外操作，效率更差。但是，使用select以后最大的优势是用户可以在一个线程内同时处理多个socket的IO请求。用户可以注册多个socket，然后不断地调用select读取被激活的socket，即可达到在同一个线程内同时处理多个IO请求的目的。而在同步阻塞模型中，必须通过多线程的方式才能达到这个目的。



FD\_SET(sock, &readfds); //将服务器端socket加入到集合中

for (fd = 0; fd < FD\_SETSIZE; fd++)/\*扫描所有的文件描述符\*/

if (FD\_ISSET(fd, &testfds)) /\*找到相关文件描述符\*/

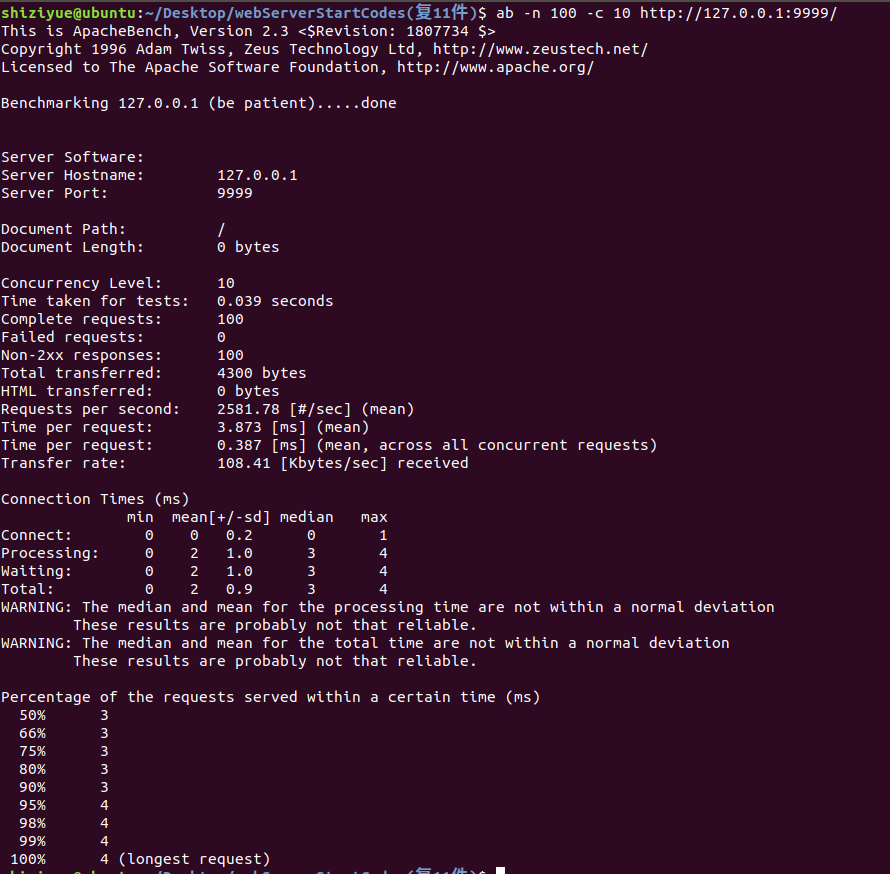
if (fd == sock)/\*找到服务器程序的文件描述符\*/

else/\*是客户端socket则对其进行响应

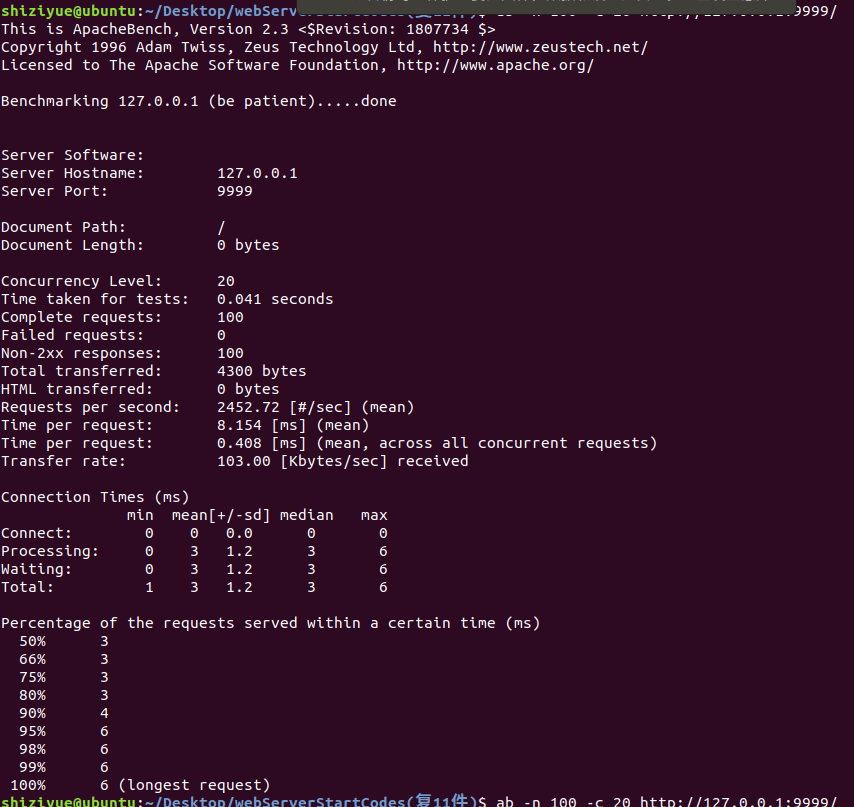
# 三、实验结果及分析

实现了select的多线程之后，发现相比于实验3中的代码速度改进许多，

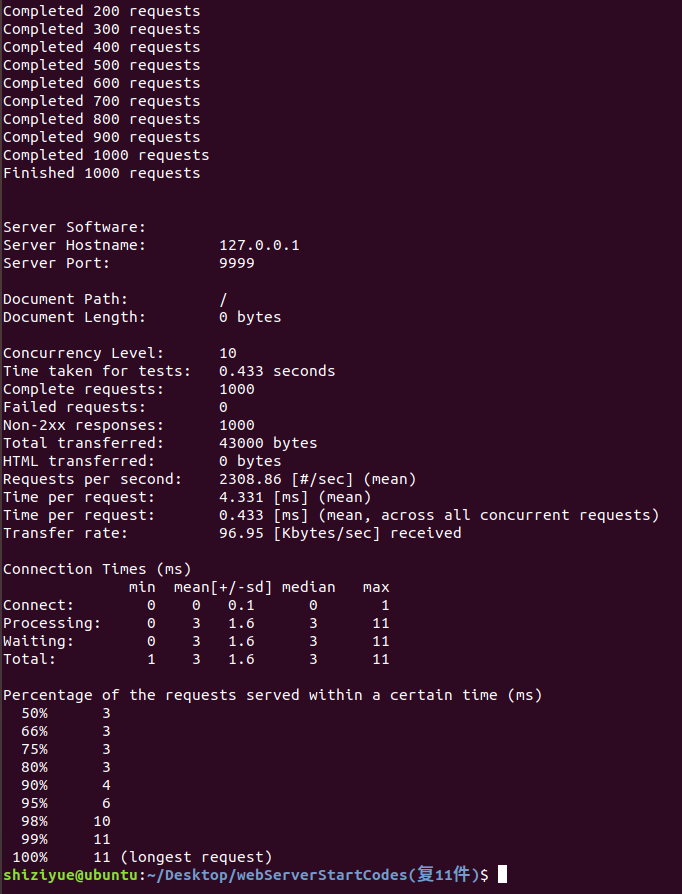
下面是共计发送总计100个，每次十个并发的结果

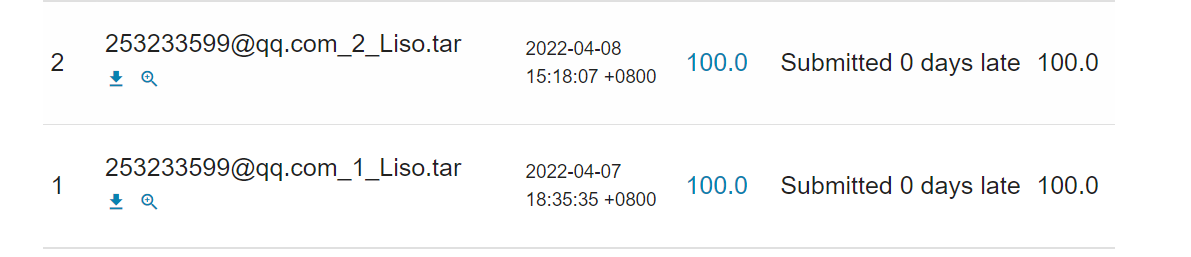


下面是共计发送总计100个，每次20个并发的结果



下面是共计发送总计1000个，每次10个并发的结果





发现总数一样时，改变并发数，响应时间随着并发数增加而增加，当并发数一样时，响应时间随总数增加而增加。

作用：

使用select以后最大的优势是用户可以在一个线程内同时处理多个socket的IO请求。在网络编程中，当涉及到多客户访问服务器的情况，我们首先想到的办法就是fork出多个进程来处理每个客户连接。现在，我们同样可以使用select来处理多客户问题，而不用fork。

但是，对socket进行扫描时是线性扫描，即采用轮询的方法，效率较低：当套接字比较多的时候，每次select()都要通过遍历FD\_SETSIZE个Socket来完成调度,不管哪个Socket是活跃的,都遍历一遍。这会浪费很多CPU时间。还需要维护一个用来存放大量fd的数据结构，这样会使得用户空间和内核空间在传递该结构时复制开销大。

# 四、进度总结

**本周任务完成表**

在“完成”“没完成”列对应打“√”

|  |  |  |  |
| --- | --- | --- | --- |
| 本周任务  要求 | 完成 | 没完成 | 备注 |
| 1、实现多个客户端的并发处理 | √ |  |  |
| 2、讨论第四周工作对服务器性能提升所起到的作用 | √ |  |  |

**上周任务改进表**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 上周任务 | 改进内容 | 备注 |
| 1 |  |  |  |
| 2 |  |  |  |