

天津大学

《计算机网络》课程设计 周进度报告

题目：第四周 实现多个客户端的并发请求

学 号: 3020202184 3020244344

姓 名: 刘锦帆 李镇州

学 院: 智能与计算学部

专 业: 计算机科学与技术

年 级: 2020 级

任课教师: 石高涛

2022 年 4 月 11 日

目 录

第一章	协议设计	1
1.1	需求分析	1
1.2	并发客户端的设计方法和技术	1
第二章	协议实现	3
2.1	处理并发客户端的实现	3
第三章	实验结果及分析	4
3.1	手工测试	4
3.2	Autolab 测试	4
3.3	Apache 测试	5
第四章	进度总结及项目分工	6
4.1	本周进度情况	6
4.2	人员分工	6
附录		

一 协议设计

1.1 需求分析

网络编程的服务端程序在大多数的情况下，并不只是与一个客户端进行通讯。在嵌入式行业中，设备通常被要求至少同时需要与 5-10 个客户端同时通信，而对于嵌入式设备来说，其内部资源是非常有限的，我们本次使用 select 函数进行处理并发。本周要实现的是服务器能实现多个客户端的并发处理。即实现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。

1.2 并发客户端的设计方法和技术

我们是利用 select 函数去实现并发客户端。函数 `int select(int maxfd + 1,fd_set *readset,fd_set *writeset, fd_set *exceptset,const struct timeval * timeout);` 参数定义如表1-1 所示。

参数	描述
参数一	最大的文件描述符 + 1。是一个整数值，是指集合中所有文件描述符的范围，即所有文件描述符的最大值加 1。
参数二	用于检查可读性，是指向 fd_set 结构的指针，这个集合中应该包括文件描述符，我们是要监视这些文件描述符的读变化的，即我们关心是否可以从这些文件中读取数据了，如果这个集合中有一个文件可读，select 就会返回一个大于 0 的值，表示有文件可读，如果没有可读的文件，则根据 timeout 参数再判断是否超时，若超出 timeout 的时间，select 返回 0，若发生错误返回负值。可以传入 NULL 值，表示不关心任何文件的读变化。
参数三	用于检查可写性，具体的解释同参数二一致。
参数四	用于检查文件错误异常，具体的解释同参数二一致。
参数五	用于指定 select 函数运行到此处时等待的时间。

表 1-1 select 函数参数描述

参数 5 可以使 select 处于三种状态

1. 若将 NULL 以形参传入，即不传入时间结构，就是将 select 置于阻塞状态，一定等到监视文件描述符集合中某个文件描述符发生变化为止

2. 若将时间值设为 0 秒 0 毫秒，就变成一个纯粹的非阻塞函数，不管文件描述符是否有变化，都立刻返回继续执行，文件无变化返回 0，有变化返回一个正值；
 3. timeout 的值大于 0，这就是等待的超时时间，即 select 在 timeout 时间内阻塞，超时时间之内有事件到来就返回了，否则在超时后不管怎样一定返回，返回值同上述。

图 1-1 部分代码

部分代码如图1-1所示，我们通过select阻塞后再遍历fd的集合进而对每个真实存在的用户进行服务。具体流程将在下一章具体展示。

二 协议实现

本周修改的主要足 liso_server 的接收请求部分的代码。在 while(1) 的服务器处理循环中添加以 select 开头的预处理等操作，使服务器能够在等待一个客户端发送下一个请求时，同时处理来自其他客户端的请求，使服务器能够同时处理多个并发的客户端。

2.1 处理并发客户端的实现

大体逻辑如下图2-1所示。因空间限制，未完全采用规范画法。

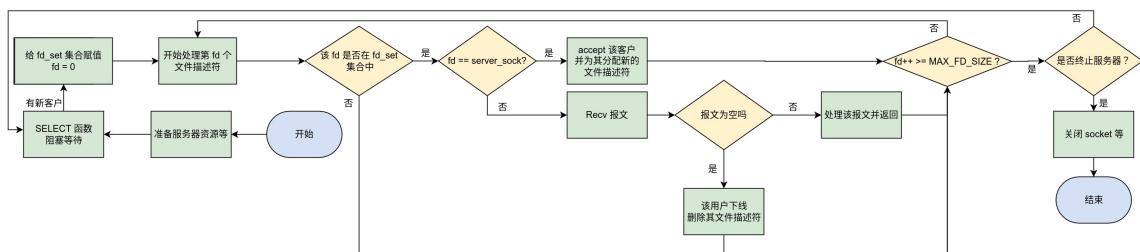


图 2-1 流程图

首先使用 `cnt=select(MAX_FD_SIZE+1, &tmp_fds, NULL, NULL, NULL)` 获得当前总共有多少个客户端有请求，同时将他们的文件描述符添加到类型为 `fd_set` 的集合中，将用户总数返回给 `cnt` 变量。

然后循环遍历 `fd_set` 集合中 0 ~ `MAX_FD_SIZE` 个位置。通过函数 `FD_ISSET()` 判断当前位置是否有客户。如果有客户占用，则进行处理，否则跳过。

在处理时，先判断其文件描述符是否和服务器的 `sock` 相同，如果相同则表明是一个新的连接，需要进行 `accept` 操作，并为其分配一个新的文件描述符。如果和服务器的 `sock` 不相同，则按照一般流程处理。如果使用 `select` 选择了这个客户，且后续 `recv` 函数接收到的信息长度为 0，我们就可以认为这个客户已经离去，可以关掉它的 `socket` 了。

和前几次的一个小区别在于服务器需要为每个客户分配一个私有的空间来存储它的地址、端口以及缓存。对此，我们的实现方式，是使用地址类型和动态伸缩的数组。（也可以使用结构体，不过原理是一样的）。

三 实验结果及分析

3.1 手工测试

我们将 liso_client 设置为忙等待，及其不会自动关闭 socket 并退出，而是占用着服务器的资源等待着。如图3-1，我们同时开启了 4 个陷入忙等待的客户端对服务器发出请求，服务器仍然能正常处理发出 pipeline 请求的客户端。

图 3-1 手工测试

3.2 Autolab 测试

如图3-2 所示，在 3 月 29 日下午完成了基础功能，后续修改了部分细节（同时也在进行 CGI 的编程）。

Ver	File	Submission Date	lab4 (100.0)	Late Days Used	Total Score
4	1051666563@qq.com_4_Liao.tar	2022-04-01 22:02:27 +0000	100.0	Submitted 0 days late	100.0
3	1051666563@qq.com_3_Liao.tar	2022-04-01 19:06:15 +0000	100.0	Submitted 0 days late	100.0
2	1051666563@qq.com_2_Liao.tar	2022-04-01 19:26:36 +0000	100.0	Submitted 0 days late	100.0
1	1051666563@qq.com_1_Liao.tar	2022-03-29 17:02:46 +0000	100.0	Submitted 0 days late	100.0

图 3-2 Autolab Test Result

3.3 Apache 测试

我们分别进行了并发压力测试以及多请求压力测试。测试截图统一放在附录中。第三周的 liso_server 由于没有进行并发优化。在测试时由于会一直等待客户端提出退出申请，在测试时会出现 Apache Bench 忙等待。故而我们只对第四周的 liso_server 进行了测试。

并发压力测试 控制总请求数不变的情况下，我们分别进行了并发数量为 10、100 以及 1000 的测试，结果如图所示。如表3-1 所示，总请求数量不变情况下，随着并发等级的提高，单个请求的平均反应时间并没有太大的变化。所以可以认为我们的方案是能够解决高并发问题的。

	并发等级	总请求数量	TPR(ms)
1	10	1000	0.238
2	100	1000	0.247
3	1000	1000	0.222

表 3-1 并发压力测试结果

多请求压力测试 通过控制并发等级为 10 不变，改变总请求数量，我们得到了多请求压力测试的结果，如表3-2所示。可以轻易看出，在并发等级一定的情况下，总请求数量在 100,000 量级及以下时，单个请求的平均反应时间都能在正常的范围内。但当请求量达到 1,000,000 数量级时，单个请求的平均响应时间则会增加很多，推测是因为请求数量太多，导致频繁接受、删除用户，进而导致反应时间增加。

	并发等级	总请求数量	TPR(ms)
1	10	100	0.332
2	10	1000	0.238
3	10	10000	0.225
4	10	100000	0.264
5	10	1000000	0.463

表 3-2 多请求压力测试

四 进度总结及项目分工

4.1 本周进度情况

通过 select 函数实现了对并发请求。具体进度如表 4-1 所示。

	本周任务要求	完成	备注
1	完成并发客户端的实现	✓	无
2	进行 Apache bench 压力测试	✓	无
3	并发效果以及性能测试	✓	无

表 4-1 本周进度完成表

4.2 人员分工

人员分工如表4-2所示。

人员	项目分工
刘锦帆	完成大部分代码工作，以及协议实现部分
李镇州	完成 client 端的处理以及协议设计、实验结果及分析部分的写作

表 4-2 人员分工表

附 录



图 4-1 Apache Bench 测试