
Socket 编程实践指导书

《计算机网络》课程组

2023 年 5 月

一、概述

1. 项目简介

使用 BSD Socket API (Berkeley Sockets API) 实现具有并发性的 web server。依据 RFC 2616 文档^[1], 实现 HTTP/1.1 的 HEAD, GET, POST 等基本功能。项目完成后, 能够使用普通商用浏览器从所实现的 web server 下载静态网页。

2. 项目目标

- 1) 能够熟练阅读 Internet 标准 RFC 文件, 并实现相应协议;
- 2) 通过 HTTP 经典协议的复现, 掌握应用层协议的设计和实现方法;
- 3) 掌握应用层协议的并发设计和实现技术。

3. 平台、语言基本要求

- 1) 操作系统: Linux; 编程语言: C 语言。
- 2) 使用标准的 socket 库和提供的库函数, 禁止使用任何自定义的 socket 类或库。
- 3) 禁止使用 libpthread 线程库。
- 4) 在基础代码的框架下编程实现。基础代码可在智慧树平台上下载。

4. 提交方法

- 1) 在智慧树平台上提交实践报告(周进度报告和课程设计报告)和源码;
- 2) 在 Autolab 平台上提交源码进行自动测试, Autolab 平台使用方法详见“自动测试平台使用手册-学生端.pdf”。

不满足上述要求者, 一律 0 分。

二、 Web Server 功能的基本要求

根据 HTTP/1.1 标准 RFC 2616^[1], HTTP 支持多种方法 (method)。要求实现其中三种: GET, HEAD 和 POST。具体要求如下。

必做内容:

1. 实现 GET, HEAD 和 POST 三种基本方法。

方法	说明
GET	请求指定的页面信息, 并返回实体主体。
HEAD	类似于 GET 请求, 只不过返回的响应中没有具体的内容, 用于获取报文头部信息。
POST	向指定资源提交数据进行处理请求 (例如提交表单或者上传文件)。数据包含在消息 (message) 中。POST 请求可能会导致新的资源的建立和/或已有资源的修改。

2. 能够正确解析客户端发来的请求包, 并做出正确响应。

3. 支持 HTTP 的并行请求 (HTTP Pipelining)。

4. 支持多个客户端并发接入。

选做内容 (加分项):

1、根据 RFC 3875 实现 CGI (Common Gateway Interface)。

三、 每周工作进度具体要求及评分细则

总体要求：

1. 每周都要提交“周进度报告”，使用“计算机网络课程设计报告模版”，撰写文档中相应内容，总结本周的任务完成情况。
2. 每周都要提交源码，并进行本周任务的自动测试。
3. 除第一周外，此后每周的任务都需要在上周任务完成的基础上才能开展。因此，如果上周的任务没有完成，必须及时补上。补充的内容可以体现在“课程设计报告”中。
4. 本次实践结束后，需要提交“课程设计报告”（使用“计算机网络课程设计报告模版”）。课程设计报告是这次实践任务的总结。请在周进度报告的基础上，认真撰写，完整体现任务最终完成情况。
5. 本次实践的总分数由每周进度分数和课程设计报告分数构成。请同学们按进度要求认真完成。

注意事项：

1. 提交的源码必须保证和基础代码使用相同的 IP 地址和端口号。
2. 选做内容（CGI）在最后的课程设计报告中提交即可，不需要进行自动测试。

3.1 第一周：实现简单的 Echo Web Server

具体要求：

1. 掌握课本有关 HTTP 的内容；阅读 HTTP/1.1 的标准文档 RFC2616^[1]；
2. 搭建编程环境（参见“讲解 PPT-环境安装配置.pptx”）；
3. 熟悉 Socket 编程方法；
4. 掌握 lex 和 yacc^[7]正确解析消息（message）的方法；
5. 实现简单的 echo web server。

Server 收到 client 的带多请求行的消息后，能够正确解析出来，并且返回响应消息（response message）。分以下 3 种情况处理：

- 1) Echo: 如果收到客户端发来的是 GET, HEAD 和 POST 方法，则 echo 回去，即重新封装（encapsulation）消息并返回给客户端。
 - 2) 没实现: 如果收到客户端发来的是除 GET, HEAD 和 POST 以外的其它方法，服务器并没有实现，则需要返回响应消息“HTTP/1.1 501 Not Implemented\r\n\r\n”。
 - 3) 格式错误: 如果收到的客户端消息的格式错误，应能够识别出来，并返回错误代码为 400 的 HTTP 响应消息“HTTP/1.1 400 Bad request\r\n\r\n”。
6. 使用浏览器测试，能够完成评分细则表中列出的功能。给出测试样例，并将测试结果展示在实验结果部分。

评分细则：

评分分为周进度报告和自动测试两项。两项独立评分，满分均为 100 分。最后按照权重计算本次实践的总分。

具体评分标准如下表所示。

表 3.1 第一周任务及评分标准

	任务点	完成内容及评分标准	分值比例
实验报告	1	协议设计。 详细阐述请求消息的解析方法。	30%
	2	协议实现。 详细阐述请求消息的解析实现。	30%
	3	实验结果及分析。 1) 能够给出所实现任务点的测试样例，并将测试结果截图展示； 2) 能够对结果进行合理的分析说明。	30%
	4	结论。 按照周进度任务要求详细填写完成进度表格。	10%
自动测试	1	能够正确解析客户端消息（带多请求行），识别出 GET, HEAD, POST，并返回（echo）给客户端。	50%
	2	能够正确解析客户端消息，识别出不是 GET, HEAD, POST 的其它方法，并返回代码为 501 的响应消息。	20%
	3	能够正确解析客户端消息，识别出 5 种以上格式错误，并返回代码为 400 的响应消息。	30%

3.2 第二周：实现 HEAD、GET 和 POST 方法

具体要求：

1. 完善服务器的功能，使其能够正确响应 HTTP1.1 请求消息，并按照 RFC 2616 中的定义实现 HEAD、GET 和 POST 的持久连接（persistent connection）^[5]。
 - 1) 如果收到客户端发来的 GET, HEAD 和 POST 方法，服务器按照 RFC2616 的规定进行处理并发回响应消息。
 - 2) 支持 4 种 HTTP 1.1 出错代码：400，404，501，505。能够准确判别客户端消息，并发回响应消息。
 - 3) 妥善管理接收缓冲区，避免由于客户端请求消息太长导致的缓冲区溢出问题。
2. 服务器能够处理读写磁盘文件时遇到的错误（例如，权限、文件不存在、IO 错误等）。
3. 创建简化的日志记录模块，记录格式化日志。

使用日志记录程序运行的过程，方便调试和追踪。可以参考 Apache 的日志文件^[4]。

 - 1) 按照 Apache 日志文件中“Error Log”的格式记录服务器的出错情况；
 - 2) 按照 Apache 日志文件中“Access Log”的“Common Log Format”记录服务器处理的请求；
 - 3) 可以创建其它方便调试的日志（只是为了方便调试，不做格式要求，不作为得分点）。
4. 不需要实现分块（Chunking）。
5. 不需要实现 Conditional GETs。
6. 响应的要求
 - 1) 对 GET/HEAD 请求发回的响应必须包含"HTTP/1.1 200 OK\r\n"。
 - 2) GET 方法获取的网页使用基础代码提供的默认网页，网页位于“/webServerStartCodes-new/static_site/index.html”。
 - 3) 400、404、501、505 的响应为

```
RESPONSE_400 = "HTTP/1.1 400 Bad request\r\n\r\n"
RESPONSE_404 = "HTTP/1.1 404 Not Found\r\n\r\n"
RESPONSE_501 = "HTTP/1.1 501 Not Implemented\r\n\r\n"
RESPONSE_505 = "HTTP/1.1 505 HTTP Version not supported\r\n\r\n"
```

注意：

- 1) 收到 POST 后，echo 返回即可。
- 2) 如果需要用到哈希表，请自行编写，禁止使用现有的哈希表库函数。
- 3) 服务器会提供静态文件供客户端请求使用。但是也可以使用 <http://svn.apache.org/repos/asf/httpd/httpd/trunk/docs/conf/mime.types>。不需要支持所有的媒体类型，只支持最常见的 MIME 类型即可，例如，text/html, text/css, image/png, image/jpeg, image/gif 等。
- 4) Stat()是系统调用函数，用于检查文件状态，供大家参考使用。
- 5) 如果请求包的头部大于 8192 字节，可以不解析，发回错误代码为 400 的响应消息。注意，要和内容长度大于 8192 字节的情况相区别。

评分细则：

评分包括周进度报告和自动测试两项，两项给分独立，具体评分标准如下表所示。

表 3.2 第二周任务及评分标准

	任务点	完成内容	分值比例
周进度报告	1	协议设计（第二周）。 1）详细阐述 HTTP1.1 三种基本方法的设计规则（persistent connection）和协议头部格式； 2）说明接收缓冲区的设计； 3）说明日志记录模块的设计。 4）其它设计细节。	30%
	2	协议实现（第二周）。 1）详细阐述 HTTP1.1 三种基本方法（persistent connection）的实现； 2）说明接收缓冲区的实现； 3）说明日志记录模块的实现； 4）说明读写磁盘文件时遇到的错误的处理； 5）其它实现细节。	30 %
	3	实验结果及分析（第二周） 1）能够给出所实现任务点的测试样例，并将测试结果截图展示； 2）能够对结果进行合理的分析说明； 3）实验结果要求包含正确的日志格式记录运行过程。	30%
	4	结论。按照周进度任务要求详细填写完成进度表格	10%
自动测试	1	能够正确解析客户端消息： 识别出 GET,HEAD，并返回相应响应消息给客户端； 识别出 POST，echo 返回。	60%
	2	能够正确解析客户端消息，并正确返回 4 种 HTTP 1.1 出错代码：400，404，501，505。	40%

3.3 第三周：实现 HTTP 的并发请求

具体要求：

1. 服务器能连续响应客户端使用同一个 TCP 连接同时发送的多个请求 GET/HEAD/POST，即支持 HTTP pipelining^[6]。
2. 服务器按照 RFC2616 规定的顺序处理 HTTP 的并发请求。
3. 对于 HTTP 的并发请求，如果服务器认为其中一个请求是错误的并拒绝该请求，那么服务器需要能够正确识别并解析出并发到达的下一条请求。

评分细则：

评分包括周进度报告和自动测试两项，两项给分独立，具体评分标准如下表所示。

表 3.3 第三周任务及评分标准

	任务点	完成内容	分值比例
周进度报告	1	协议设计（第三周）。 详细阐述 HTTP pipelining 的设计。	30%
	2	协议实现（第三周）。 详细阐述 HTTP pipelining 的实现。	30%
	3	实验结果及分析（第三周）。 1) 能够给出所实现任务点的测试样例，并将测试结果截图展示； 2) 能够对结果进行合理的分析说明。	30%
	4	结论。 按照周进度任务要求详细填写完成进度表格。	10%
自动测试	1	正确解析的并发请求数：少于 4 条	0%
	2	正确解析的并发请求数：4 条	20%
	3	正确解析的并发请求数：8 条	40%
	4	正确解析的并发请求数：12 条	60%
	5	正确解析的并发请求数：16 条	80%
	6	正确解析的并发请求数：20 条及以上	100%

3.4 第四周：实现多个客户端的并发处理

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。

评分细则：

评分包括周进度报告和自动测试两项，两项给分独立，具体评分标准如下表所示。

	任务点	完成内容	分值比例
实验报告	1	协议设计（第四周） 详细阐述并发客户端的设计方法与技术。	20%
	2	协议实现（第四周） 详细阐述并发客户端的实现。	30%
	3	实验结果及分析（第四周） 1. 实验结果及分析包括并发效果以及性能测试； 2. 性能测试部分包括但不限于以下内容： 使用 Apache bench，设置不同数量级的并发数量，对第三周实现的服务器（不支持并发客户端）和第四周实现的服务器（支持并发客户端），进行对比测试，分析测试结果中的各项性能参数，并讨论本周工作对服务器性能提升所起到的作用。	40%
	4	结论。 按照周进度任务要求详细填写完成进度表格	10%
自动测试	1	用 <code>select</code> 方法实现多个客户端的并发请求	100%

3.5 选作内容：CGI

具体要求：

1. 阅读 RFC3875^[2]和 RFC2396^[3]，实现 CGI。
2. 当服务器接收到一个使用 CGI URI 的请求时，能够开启一个新的进程，该进程启动 CGI 程序并将客户端请求提供给 CGI 程序，由 CGI 程序负责处理请求并将响应的内容返回给服务器，最后由服务器负责将响应内容返回给客户端。
3. CGI 环境变量需要包括以下条目：
 - 1) CONTENT_LENGTH—可直接从请求中获取
 - 2) CONTENT_TYPE—指示所传递来的信息的 MIME 类型，可直接从请求中获取
 - 3) GATEWAY_INTERFACE—“CGI/1.1”
 - 4) PATH_INFO—紧接在 CGI 程序名之后的其他路径信息，是 URI 中的<path>部分
 - 5) QUERY_STRING—如果服务器与 CGI 程序信息的传递方式是 GET，这个环境变量的值为传递的信息，即从 URI 中解析出来的“？”后的内容
 - 6) REMOTE_ADDR—发送请求的客户端的 IP 地址
 - 7) REQUEST_METHOD—提供脚本被调用的方法，可直接从请求中获取
 - 8) REQUEST_URI—可直接从请求中获取
 - 9) SCRIPT_NAME—CGI 脚本的名称
 - 10) SERVER_PORT—服务器的端口
 - 11) SERVER_PROTOCOL—“HTTP/1.1”
 - 12) SERVER_SOFTWARE—“Liso/1.0”
 - 13) HTTP_ACCEPT—可直接从请求中获取
 - 14) HTTP_REFERER—可直接从请求中获取
 - 15) HTTP_ACCEPT_ENCODING—可直接从请求中获取
 - 16) HTTP_ACCEPT_LANGUAGE—可直接从请求中获取
 - 17) HTTP_ACCEPT_CHARSET—可直接从请求中获取
 - 18) HTTP_HOST—发送 CGI 请求的客户端的主机名，可直接从请求中获取

-
- 19) HTTP_COOKIE—客户端内的 COOKIE 内容，可直接从请求中获取
 - 20) HTTP_USER_AGENT—提供包含了版本数或其他专有数据的客户浏览器信息，可直接从请求中获取
 - 21) HTTP_CONNECTION—可直接从请求中获取
- 4. 无论 CGI 进程以何种形式终止，服务器能够向客户端返回 500 出错代码。
 - 5. **完成任务点 2 的所有内容，才能计算 CGI 任务的分数**，只设置了 CGI 环境变量或搭建了 HTML 网页均不计算分数。

评分细则：

不参与自动测试，实现过程、实验结果及分析（包括所有任务点）详细写入最终的课程设计报告，以附加分的形式参与最终课程设计报告的打分。

任务点	完成内容	分值比例
1	CGI 环境变量齐全	10%
2	完成一个具有表单的 HTML 页面（能够实现账号和密码的输入、提交）。服务器能够根据用户在浏览器提交的信息，动态生成网页（网页显示的内容包括用户动态提交的账号和密码）	80%
3	CGI 进程出错终止时，服务器向客户端返回 500 出错代码	10%

四、 实践报告要求

4.1 实践报告内容要求

实践报告包括两种：周进度报告和课程设计报告。周进度报告是课程设计报告的一部分。

实践报告使用“计算机网络课程设计报告模版”撰写文档，命名规则：“计算机网络课程设计报告_学号_姓名”。文件类型：docx 或者转换为 pdf。

每周的实践完成后，撰写当周的周进度报告，填写“计算机网络课程设计报告”文档中相应章节。把文档另存为“计算机网络课程设计报告_第 x 周_学号_姓名”，其中“x”为周序号， $x=[1, 4]$ 。提交到智慧树。

实践结束后，在周进度报告的基础上完善内容形成最终的课程设计报告：“计算机网络课程设计报告_学号_姓名”。提交到智慧树。

注意：源代码单独提交，不要出现在实践报告的任何部分。

4.2 实践报告格式要求

- 1) 严格按照模版的格式撰写。
- 2) 按内容分章节撰写，排版美观，结构清晰。

五、 实践方式

1. 分组完成，每组 1~2 人，自由组队。
2. 每组独立完成。以组为单位评分，组内成员得分相同。如果 2 人一组，则必须在周进度报告和课程设计报告中明确 2 人的具体分工。
3. 每周都要提交本周的工作内容，确保按计划推进实践工作。文档和源码提交到智慧树；在自动测试平台 Autolab 进行功能测试。
4. 鼓励同学们就编程实现问题积极开展讨论和交流，但是需要独立编程实现。**禁止任何形式的抄袭，一经发现，抄袭与被抄袭者均以零分处理。**源码和报告文档都会使用查重软件查重。

六、评分标准

本实践项目满分 100 分。为鼓励同学们完成选做内容，额外增加 20 分。分数分布如下表所示。

表 1 分数比值分配表

序号	内容	占比	备注
1	按进度完成阶段性任务	90%	
2	实践报告	10%	
	合计	100%	
	完成选做内容	最高 +20 分	选做内容得分直接加到总分上，如果超过 100 分，则计为 100。

最后实践得分按照比例折合到平时成绩里，不超过平时成绩的总分。

具体要求如下：

1、按进度完成阶段性任务：80%

设每次阶段性任务总分为 S ，包括进度报告得分 R 和功能自动测试得分 T ，按下式计算：

$$S = \alpha \times 0.2 \times R + \beta \times 0.8 \times T.$$

其中 α, β 为延迟提交因子，如果按时提交为 1，否则为 0.8。

每次进度报告，功能测试得分满分均按 100 计，最后再折合到总分里。

各阶段任务的权重如下：

阶段	内容	权重	备注
1	实现简单的 Echo	0.3	
2	实现基本方法	0.3	
3	实现 HTTP Pipelining	0.2	
4	实现多个客户端的并发处理	0.2	
合计		1	

2、实践报告文档：10%

- 1) 按照要求，撰写实践报告内容中的每个部分。(6%)
- 2) 内容详实，重点突出。(2%)
- 3) 语言规范流畅，条理清晰，格式满足要求。(2%)

关于诚信问题：

1、源代码和报告提交后，将由学校查重系统或相关查重软件查重，
如有雷同，本次实践总分记为 0 分。

2、本次实践的源码 **3 年内禁止公开**：禁止在学生中间流传，禁止放到网上公开。请同学们严格遵守此项要求。

参考文献

- [1] RFC 2616, Hypertext Transfer Protocol -- HTTP/1.1, <http://www.ietf.org/rfc/rfc2616.txt>
- [2] RFC 3875, Common Gateway Interface, <http://www.ietf.org/rfc/rfc3875.txt>
- [3] RFC 2396, Uniform Resource Identifiers (URI): Generic Syntax,
<http://www.ietf.org/rfc/rfc2396.txt>
- [4] Apache Log Files, <https://httpd.apache.org/docs/2.4/logs.html>
- [5] Persistent Connection: https://en.m.wikipedia.org/wiki/HTTP_persistent_connection
- [6] HTTP pipelining: https://en.m.wikipedia.org/wiki/HTTP_pipelining
- [7] A Guide to Lex & Yacc:
https://arcb.csc.ncsu.edu/~mueller/codeopt/codeopt00/y_man.pdf

推荐资源

- [1] GNU make, <https://www.gnu.org/software/make/manual/make.html>
- [2] UNIX Socket FAQ, <http://developerweb.net/?s=f47b63594e6b831233c4b8ebaf10a614&f=70>
- [3] siege - An HTTP/HTTPS stress tester, <https://linux.die.net/man/1/siege>
- [4] ab - Apache HTTP server benchmarking tool, <https://linux.die.net/man/1/ab>
- [5] The GNU C Library, <https://www.gnu.org/software/libc/manual/>
- [6] Valgrind, <https://www.valgrind.org>

[7] GDB: The GNU Project Debugger, <https://www.gnu.org/software/gdb/>

[8] The Apache HTTP Server Project, <http://httpd.apache.org>