

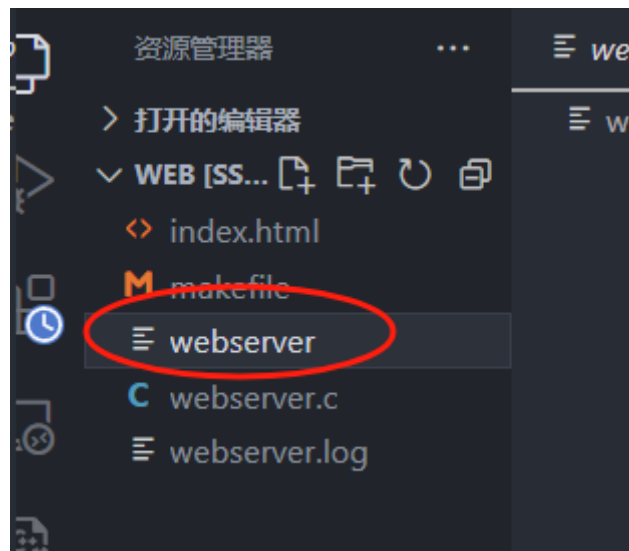
实验报告一

计科21-2 2021011587 吴维皓

题目一：

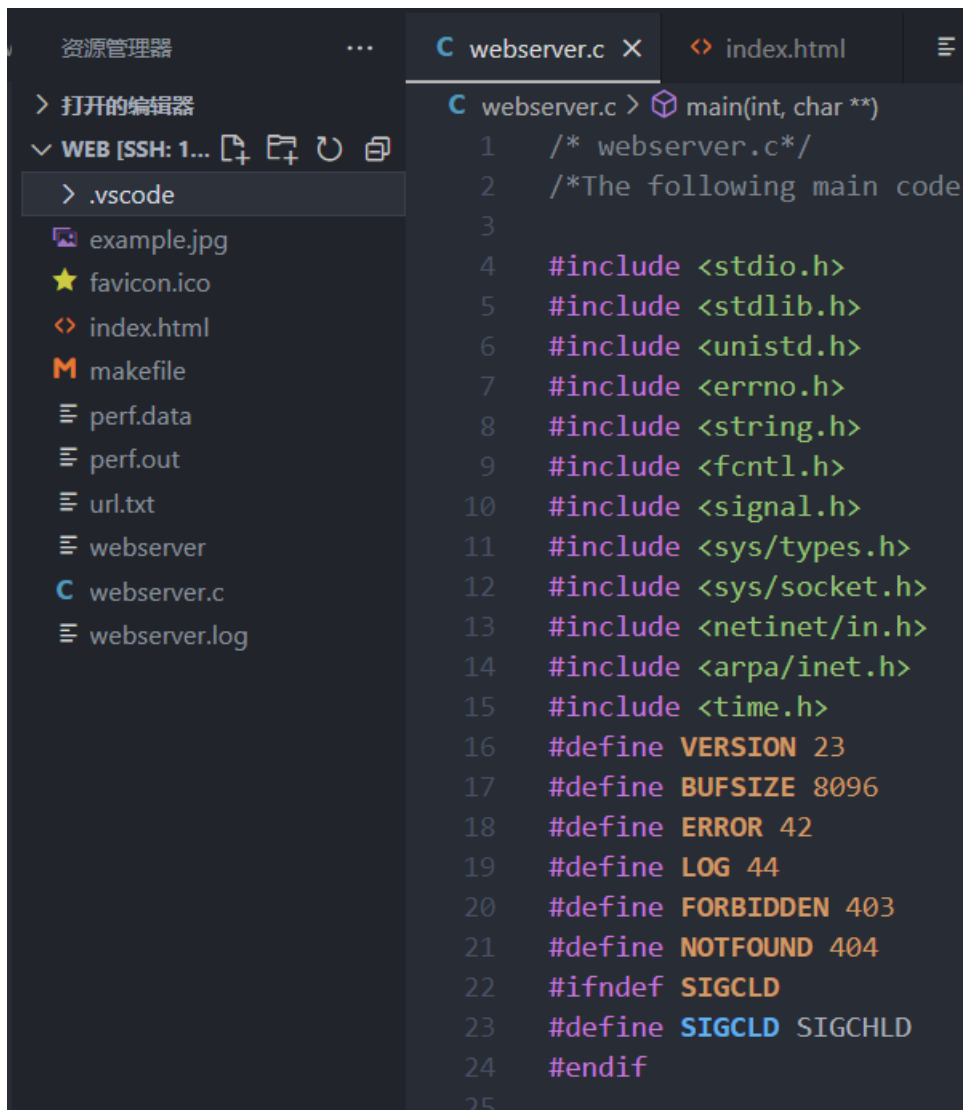
```
1 #makefile文件
2 CC = gcc
3 CFLAGS = -Wall -Wextra -O2
4
5 webserver: webserver.c
6     $(CC) $(CFLAGS) -o webserver webserver.c
7
8 clean:
9     rm -f webserver
10
```

```
[linux1@bogon web]$ make
gcc -Wall -Wextra -O2 -o webserver webserver.c
```



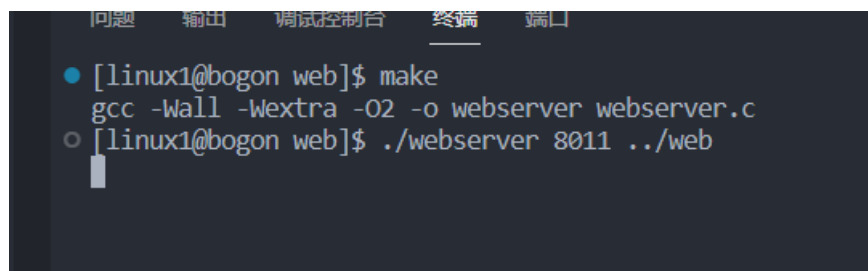
make后生成的可执行文件

题目二:



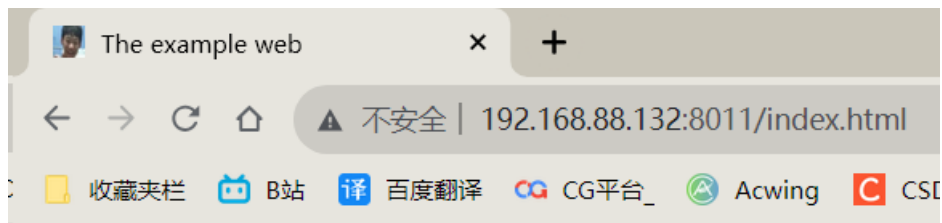
```
C webserver.c X index.html
C webserver.c > main(int, char **)
1  /* webserver.c */
2  /*The following main code
3
4  #include <stdio.h>
5  #include <stdlib.h>
6  #include <unistd.h>
7  #include <errno.h>
8  #include <string.h>
9  #include <fcntl.h>
10 #include <signal.h>
11 #include <sys/types.h>
12 #include <sys/socket.h>
13 #include <netinet/in.h>
14 #include <arpa/inet.h>
15 #include <time.h>
16 #define VERSION 23
17 #define BUFSIZE 8096
18 #define ERROR 42
19 #define LOG 44
20 #define FORBIDDEN 403
21 #define NOTFOUND 404
22 #ifndef SIGCLD
23 #define SIGCLD SIGCHLD
24 #endif
25
```

webserver.c 源代码



```
问题 输出 调试控制台 终端 端口
[linux1@bogon web]$ make
gcc -Wall -Wextra -O2 -o webserver webserver.c
[linux1@bogon web]$ ./webserver 8011 ../web
```

运行webserver 并指定端口为8011



webserver test page

Not pretty but it should prove that webserver works : -)



成功进入网页

```
1 //加载一次网址会产生3条log
2 :
3 INFO: request:GET /index.html HTTP/1.1**Host:
192.168.88.132:8011**Connection: keep-alive**Cache-Control: max-
age=0**Upgrade-Insecure-Requests: 1**User-Agent: Mozilla/5.0 (Windows NT
10.0; win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/117.0.0.0
Safari/537.36**Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7**Accept-
Encoding: gzip, deflate**Accept-Language: zh-CN,zh;q=0.9,en;q=0.8****:1
4 INFO: SEND:index.html:1
5 INFO: Header:HTTP/1.1 200 OK
6 Server: nweb/23.0
7 Content-Length: 391
8 Connection: close
9 Content-Type: text/html
10
11 :1
12 INFO: request:GET /example.jpg HTTP/1.1**Host:
192.168.88.132:8011**Connection: keep-alive**User-Agent: Mozilla/5.0
(Windows NT 10.0; win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/117.0.0.0 Safari/537.36**Accept:
image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8**Referer:
http://192.168.88.132:8011/index.html**Accept-Encoding: gzip,
deflate**Accept-Language: zh-CN,zh;q=0.9,en;q=0.8****:2
13 NOT FOUND: failed to open file:example.jpg
14 INFO: SEND:example.jpg:2
15 INFO: Header:HTTP/1.1 200 OK
16 Server: nweb/23.0
17 Content-Length: -1
18 Connection: close
19 Content-Type: image/jpg
20
21 :2
```

```

22  INFO: request:GET /favicon.ico HTTP/1.1**Host:
    192.168.88.132:8011**Connection: keep-alive**User-Agent: Mozilla/5.0
    (Windows NT 10.0; win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/117.0.0.0 Safari/537.36**Accept:
    image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8**Referer:
    http://192.168.88.132:8011/index.html**Accept-Encoding: gzip,
    deflate**Accept-Language: zh-CN,zh;q=0.9,en;q=0.8***:3
23  NOT FOUND: failed to open file:favicon.ico
24  INFO: SEND:favicon.ico:3
25  INFO: Header:HTTP/1.1 200 OK
26  Server: nweb/23.0
27  Content-Length: -1
28  Connection: close
29  Content-Type: image/ico

```

- 解释：从INFO字段可以看出，这3条log分别申请访问了3个资源，分别是：index.html、example.jpg、favicon.ico，也就是需要从服务器段加载相应的资源产生了这些记录。
- 为加快 html 网页显示的速度，采用的技术：
 - 缓存：通过设置缓存头，告诉浏览器在一段时间内重复访问同一资源时，直接使用缓存中的内容，而不需要再次向服务器请求
 - 优化数据库查询：如果网页需要从数据库中获取数据，可以通过优化查询语句、使用索引等方式来提高数据库查询速度
 - 压缩传输内容：使用 Gzip 或 Deflate 等压缩算法对传输的内容进行压缩，以减少传输数据量，提高加载速度
 - 使用内容分发网络(CDN)，减少服务器响应时间，提高加载速度
 - 减少 HTTP 请求次数：可以通过合并 CSS 和 JavaScript 文件、使用 Sprites 技术等方式来减少 HTTP 请求次数
 - 优化图片大小：对图片进行压缩和优化，以减少页面加载时间
 - 负载均衡：如果网站流量较大，可以使用负载均衡技术将流量分发到多个服务器上，以提高网站的性能和可用性

题目三：

```

1  //生成一个当前系统时间的变量
2  char time_now[255];
3  time_t rawtime;
4  struct tm * timeinfo;
5  time ( &rawtime );
6  timeinfo = localtime ( &rawtime );
7  strftime(time_now, sizeof(time_now), "%Y-%m-%d %H:%M:%S", timeinfo); //将时
    间变量格式化

```

```
48 char time_now[255];
49 time_t rawtime;
50 struct tm * timeinfo;
51 time ( &rawtime );
52 timeinfo = localtime ( &rawtime );
53 strftime(time_now, sizeof(time_now), "%Y-%m-%d %H:%M:%S", timeinfo);
54
55 /*根据消息类型, 将消息放入 logbuffer 缓存, 或直接将消息通过 socket 通道返回给客户端*/ switch (type) {
56 case ERROR:
57     (void)sprintf(logbuffer, "%s:ERROR: %s:%s Errno=%d exiting pid=%d", time_now, s1, s2, errno, getpid()); break;
58 case FORBIDDEN:
59     (void)write(socket_fd, "HTTP/1.1 403 Forbidden\nContent-Length: 185\nConnection: close\nContent-Type: text/html\n\n<html><head>\n<title>403 Forbidden</title>\n</head><body>\n<h1>Forbidden</h1>\n\nThe requested URL, file type or operation is not allowed on this simple static file webserver.\n</body></html>\n", 271);
60     (void)sprintf(logbuffer, "%s:FORBIDDEN: %s:%s", time_now, s1, s2); break;
61 case NOTFOUND:
62     (void)write(socket_fd, "HTTP/1.1 404 Not Found\nContent-Length: 136\nConnection: close\nContent-Type: text/html\n\n<html><head>\n<title>404 Not Found</title>\n</head><body>\n<h1>Not Found</h1>\n\nThe requested URL was not found on this server.\n</body></html>\n", 224);
63     (void)sprintf(logbuffer, "%s:NOT FOUND: %s:%s", time_now, s1, s2); break;
64 case LOG: (void)sprintf(logbuffer, "%s:INFO: %s:%s:%d", time_now, s1, s2, socket_fd); break;
65 }
66 /* 将 logbuffer 缓存中的消息存入 webserver.log 文件*/
67 if ((fd = open("webserver.log", O_CREAT | O_WRONLY | O_APPEND, 0644)) >= 0) {
68     (void)write(fd, logbuffer, strlen(logbuffer));
69     (void)write(fd, "\n", 1); (void)close(fd);
70 }

```

```
269
270 :2
271 2023-11-08 14:14:06:INFO: request:GET /index.html HTTP/1.1**Host: 192.168.88.132:8011**Connection: keep-alive**Upgrade-Insecure-Requests: 1**User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/117.0.0.0 Safari/537.36**Sec-Purpose: prefetch;prerender**Purpose: prefetch**Accept: text/html;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7**Accept-deflate**Accept-Language: zh-CN,zh;q=0.9,en;q=0.8****:1
272 2023-11-08 14:14:06:INFO: SEND:index.html:1
273 2023-11-08 14:14:06:INFO: Header:HTTP/1.1 200 OK
274 Server: nweb/23.0
275 Content-Length: 382
276 Connection: close
277 Content-Type: text/html
278
279 :1
280 2023-11-08 14:14:07:INFO: request:GET /example.jpg HTTP/1.1**Host: 192.168.88.132:8011**Connection: prerender**User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/117.0.0.0 Safari/537.36**Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8**Purpose: prefetch**Accept-Encoding: gzip, deflate**Accept-Language: zh-CN,zh;q=0.9,en;q=0.8****:2
281 2023-11-08 14:14:07:NOT FOUND: failed to open file:example.jpg

```

问题 输出 调试控制台 终端 端口

Linux1@hadoop-web1\$./webserver 8011 ./web

- 做法：将生成的时间变量加入log函数的每个sprintf中即可

题目四：

```

webserv.log
1 2023-11-08 15:47:11:INFO: request:GET /index.html HTTP/1.1**Host: 192.168.88.132:8011**Connection: keep-alive**Upgrade-Insecure-
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/117.0.0.0 Safari/537.36**Accept: text/ht
application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7**Accept-Encoding: gzip,
deflate**Accept-Language: zh-CN,zh;q=0.9,en;q=0.8****:1
2 2023-11-08 15:47:11:INFO: SEND:index.html:1
3 2023-11-08 15:47:11:INFO: Header:HTTP/1.1 200 OK
4 Server: nweb/23.0
5 Content-Length: 382
6 Connection: close
7 Content-Type: text/html
8
9 :1
10 2023-11-08 15:47:12:INFO: request:GET /example.jpg HTTP/1.1**Host: 192.168.88.132:8011**Connection: keep-alive**User-Agent: Mozi
0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/117.0.0.0 Safari/537.36**Accept: image/avif,image/webp,image/apng,i
q=0.8**Referer: http://192.168.88.132:8011/index.html**Accept-Encoding: gzip, deflate**Accept-Language: zh-CN,zh;q=0.9,en;q=0.8*
1 2023-11-08 15:47:12:NOT FOUND: failed to open file:example.jpg
2 2023-11-08 15:47:12:INFO: SEND:example.jpg:2
3 2023-11-08 15:47:12:INFO: Header:HTTP/1.1 200 OK
4 Server: nweb/23.0
5 Content-Length: -1
6 Connection: close
7 Content-Type: image/jpg
8
9 :2
10 2023-11-08 15:47:13:INFO: request:GET /favicon.ico HTTP/1.1**Host: 192.168.88.132:8011**Connection: keep-alive**User-Agent: Mozi
0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/117.0.0.0 Safari/537.36**Accept: image/avif,image/webp,image/apng,i
q=0.8**Referer: http://192.168.88.132:8011/index.html**Accept-Encoding: gzip, deflate**Accept-Language: zh-CN,zh;q=0.9,en;q=0.8*
1 2023-11-08 15:47:13:NOT FOUND: failed to open file:favicon.ico
2 2023-11-08 15:47:13:INFO: SEND:favicon.ico:3
3 2023-11-08 15:47:13:INFO: Header:HTTP/1.1 200 OK
4 Server: nweb/23.0

```

一个完整网页进程申请的资源

```

webserv.log
171 :13
172 2023-11-08 16:38:32:INFO: request:GET /index.html HTTP/1.1**Host: 192.168.88.132:8011**Connection: keep-alive**Cache
max-age=0**Upgrade-Insecure-Requests: 1**User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (K
Chrome/117.0.0.0 Safari/537.36**Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
application/signed-exchange;v=b3;q=0.7**Accept-Encoding: gzip, deflate**Accept-Language: zh-CN,zh;q=0.9,en;q=0.8****
173 2023-11-08 16:38:32:INFO: SEND:index.html:14
174 2023-11-08 16:38:32:INFO: Header:HTTP/1.1 200 OK
175 Server: nweb/23.0
176 Content-Length: 382
177 Connection: close
178 Content-Type: text/html
179
180 :14
181 2023-11-08 16:38:33:INFO: request:GET /index.html HTTP/1.1**Host: 192.168.88.132:8011**Connection: keep-alive**Cache
max-age=0**Upgrade-Insecure-Requests: 1**User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (K
Chrome/117.0.0.0 Safari/537.36**Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
application/signed-exchange;v=b3;q=0.7**Accept-Encoding: gzip, deflate**Accept-Language: zh-CN,zh;q=0.9,en;q=0.8****
182 2023-11-08 16:38:33:INFO: SEND:index.html:15
183 2023-11-08 16:38:33:INFO: Header:HTTP/1.1 200 OK
184 Server: nweb/23.0
185 Content-Length: 382
186 Connection: close
187 Content-Type: text/html
188
189 :15
190 2023-11-08 16:38:34:INFO: request:GET /index.html HTTP/1.1**Host: 192.168.88.132:8011**Connection: keep-alive**Cache
max-age=0**Upgrade-Insecure-Requests: 1**User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (K
Chrome/117.0.0.0 Safari/537.36**Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,

```

分析：结合log文件中的内容可以看出，多次快速刷新页面后，即使有些页面没有完全打开就被刷新掉，但是log文件中仍会有该次的访问数据，也就是说，只要有申请访问信号的到来，一个网页进程将会被创建。随之而来的是资源的加载，这个操作属于I/O操作，耗时相对较长，因为是多次快速刷新，意味着每个进程快速的被创建又被删除，又因为这是个单进程模型，即使每个进程只进行了一次I/O操作，所以排到最后一个进程（也就是最后刷新到的网页）加载资源仍会等待一段时间。

题目五：

1. 性能监测：

```

code 200 -- 10
• [linux1@bogon web]$ sudo /root/http_load-12mar2006/http_load -p 30 -s 17 url.txt
17 fetches, 30 max parallel, 6494 bytes, in 17.0016 seconds
382 mean bytes/connection
0.999909 fetches/sec, 381.965 bytes/sec
msecs/connect: 0.0817059 mean, 0.18 max, 0.013 min
msecs/first-response: 8005.67 mean, 16011 max, 0.349 min
HTTP response codes:
code 200 -- 17
• [linux1@bogon web]$ sudo /root/http_load-12mar2006/http_load -p 30 -s 25 url.txt
25 fetches, 30 max parallel, 9550 bytes, in 25.0025 seconds
382 mean bytes/connection
0.999899 fetches/sec, 381.962 bytes/sec
msecs/connect: 0.0818 mean, 0.202 max, 0.013 min
msecs/first-response: 12008.2 mean, 24017.3 max, 0.174 min
HTTP response codes:
code 200 -- 25
• [linux1@bogon web]$ sudo /root/http_load-12mar2006/http_load -p 300 -s 25 url.txt
25 fetches, 300 max parallel, 9550 bytes, in 25.0019 seconds
382 mean bytes/connection
0.999922 fetches/sec, 381.97 bytes/sec
msecs/connect: 0.10608 mean, 0.317 max, 0.015 min
msecs/first-response: 12007.5 mean, 24014.8 max, 0.317 min
HTTP response codes:
code 200 -- 25
• [linux1@bogon web]$ sudo /root/http_load-12mar2006/http_load -p 30 -s 60 url.txt
60 fetches, 30 max parallel, 22920 bytes, in 60.0014 seconds
382 mean bytes/connection
0.999977 fetches/sec, 381.991 bytes/sec
msecs/connect: 0.116233 mean, 0.873 max, 0.014 min
msecs/first-response: 22264.3 mean, 30021 max, 0.379 min
HTTP response codes:
code 200 -- 60

```

- 平均每秒响应量为：0.9999
- 客户端与服务器建立连接平均时间：0.09324 ms
- http请求发出到服务器接收第一个响应平均时间：13571.4 ms


```

0 0 7944 114000 0 525048 0 0 0 0 755 930 1 1 99 0 0
● [linux1@bogon web]$ vmstat 1 30
procs -----memory----- ---swap-- ---io--- -system-- -----cpu-----
r b swpd free buff cache si so bi bo in cs us sy id wa st
1 0 11016 145788 0 478416 0 0 26 2 63 90 0 0 100 0 0
1 0 11016 143368 0 478416 0 0 0 0 815 1086 1 1 98 0 0
0 0 11016 145244 0 478416 0 0 0 100 796 919 1 1 98 0 0
0 0 11016 145292 0 478416 0 0 0 20 723 1012 1 1 99 0 0
0 0 11016 144368 0 478416 0 0 0 0 751 911 1 1 98 0 0
0 0 11016 143840 0 478416 0 0 0 0 806 952 0 1 99 0 0
2 0 11016 142148 0 478416 0 0 0 0 1172 1618 1 1 98 0 0
0 0 11016 144492 0 478420 0 0 0 0 614 686 1 1 99 0 0
0 0 11016 144568 0 478420 0 0 0 0 509 623 0 1 99 0 0
0 0 11016 143704 0 478420 0 0 0 8 377 529 1 0 99 0 0
1 0 11016 143088 0 478420 0 0 0 0 540 653 1 1 99 0 0
1 0 11016 142276 0 478420 0 0 0 0 681 790 0 1 99 0 0
0 0 11016 142844 0 478420 0 0 0 0 621 653 1 2 97 0 0
0 0 11016 143496 0 478420 0 0 0 0 465 563 0 1 99 0 0
0 0 11016 144628 0 478420 0 0 0 0 437 573 1 1 99 0 0
0 0 11016 144136 0 478420 0 0 0 4 411 557 1 0 99 0 0
1 0 11016 141032 0 478420 0 0 0 0 794 716 2 1 97 0 0
0 0 11016 144168 0 478420 0 0 0 0 674 731 1 1 99 0 0
0 0 11016 144304 0 478420 0 0 0 0 652 866 1 1 99 0 0
0 0 11016 142972 0 478420 0 0 0 0 459 635 1 1 99 0 0
0 0 11016 142972 0 478420 0 0 0 0 573 745 1 0 99 0 0
1 0 11016 140292 0 478424 0 0 0 4 711 892 1 2 98 0 0
0 0 11016 142028 0 478424 0 0 0 0 538 639 1 1 99 0 0
0 0 11016 143392 0 478424 0 0 0 0 514 625 1 1 99 0 0
0 0 11016 143020 0 478424 0 0 0 0 523 753 0 1 99 0 0
0 0 11016 142896 0 478424 0 0 0 0 640 921 1 1 99 0 0
1 0 11016 141020 0 478424 0 0 0 8 667 762 1 1 98 0 0
0 0 11016 142352 0 478424 0 0 0 0 556 652 1 1 99 0 0
0 0 11016 144736 0 478424 0 0 0 0 716 879 1 1 99 0 0
0 0 11016 143448 0 478424 0 0 0 0 562 579 1 1 98 0 0
○ [linux1@bogon web]$

```

- 在 free 还有剩余的情况下 swpd 大于0，且保持在一个稳定的数，说明这可能是虚拟内存被用于内存映射（将一些必须加载的大型文件的部分映射到虚拟内存中）
- 从 in 可以看出，CPU每秒平均中断次数在500次，最高可到1172次
- 从 cs 可以看出，CPU上下文切换次数平均在800次，最高可达1618次
- 从 us sy id可以看出，用户进程和系统进程占用CPU时间的百分比都很低，最多维持在1%，而大部分时间CPU都处于空闲状态，可以知道，网页程序的运行不会占用太多CPU时间，不是一个CPU密集型程序

```

● [linux1@bogon web]$ iostat -d -x /dev/sda /home/linux1/web/webserver
Linux 3.10.0-1160.el7.x86_64 (bogon) 11/08/2023 _x86_64_ (4 CPU)

Device:            rrqm/s   wrqm/s     r/s     w/s   rkB/s   wkB/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %util
sda                  0.01     0.26     1.86     0.56  103.54    8.73   92.96     0.00    0.70   0.48   1.43   0.19   0.05

● [linux1@bogon web]$ iostat -d -x /dev/sda /home/linux1/web/webserver
Linux 3.10.0-1160.el7.x86_64 (bogon) 11/08/2023 _x86_64_ (4 CPU)

Device:            rrqm/s   wrqm/s     r/s     w/s   rkB/s   wkB/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %util
sda                  0.01     0.26     1.86     0.56  103.53    8.73   92.96     0.00    0.70   0.48   1.43   0.19   0.05

● [linux1@bogon web]$ iostat -d -x /dev/sda /home/linux1/web/webserver
Linux 3.10.0-1160.el7.x86_64 (bogon) 11/08/2023 _x86_64_ (4 CPU)

Device:            rrqm/s   wrqm/s     r/s     w/s   rkB/s   wkB/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %util
sda                  0.01     0.31     1.84     0.57  102.39    9.09   92.70     0.00    0.69   0.48   1.39   0.19   0.05

○ [linux1@bogon web]$

```

- rkB/s 和 wkB/s 数值稳定，对应了每次网页打开时加载的资源比较固定（照片和图标），相比与上面的CPU使用情况，webserver程序算是I/O密集型程序

Total DISK READ :		0.00 B/s	Total DISK WRITE :		3.92 K/s		
Actual DISK READ:		0.00 B/s	Actual DISK WRITE:		0.00 B/s		
TID	PRIO	USER	DISK READ	DISK WRITE	SWAPIN	IO>	COMMAND
21549	be/4	linux1	0.00 B/s	3.92 K/s	0.00 %	0.00 %	./webserver 8011 ../web
1	be/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	systemd --switched-root --system
2	be/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[kthreadd]
4	be/0	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[kworker/0:0H]
6	be/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[ksoftirqd/0]
7	rt/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[migration/0]
8	be/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[rcu_bh]
9	be/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[rcu_sched]
10	be/0	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[lru-add-drain]
11	rt/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[watchdog/0]
12	rt/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[watchdog/1]
13	rt/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[migration/1]
14	be/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[ksoftirqd/1]
6829	be/4	linux1	0.00 B/s	0.00 B/s	0.00 %	0.00 %	gvfsd-network --spawner :1.4 /org
16	be/0	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[kworker/1:0H]
17	rt/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[watchdog/2]
18	rt/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[migration/2]
19	be/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[ksoftirqd/2]
21	be/0	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[kworker/2:0H]
22	rt/4	root	0.00 B/s	0.00 B/s	0.00 %	0.00 %	[watchdog/3]

(Not all processes could be identified, non-owned process info will not be shown, you would have to be root to see it all.)

Active Internet connections (servers and established)

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State	PID/Program name	Timer
tcp	0	0	192.168.122.1:53	0.0.0.0:*	LISTEN	-	off (0.00/0/0)
tcp	0	0	0.0.0.0:22	0.0.0.0:*	LISTEN	-	off (0.00/0/0)
tcp	0	0	127.0.0.1:631	0.0.0.0:*	LISTEN	-	off (0.00/0/0)
tcp	0	0	127.0.0.1:39864	0.0.0.0:*	LISTEN	3314/code-1a5daa3a0	off (0.00/0/0)
tcp	0	0	127.0.0.1:25	0.0.0.0:*	LISTEN	-	off (0.00/0/0)
tcp	0	0	127.0.0.1:43744	0.0.0.0:*	LISTEN	2921/node	off (0.00/0/0)
tcp	20	0	0.0.0.0:8011	0.0.0.0:*	LISTEN	9374/./webserver	off (0.00/0/0)
tcp	0	0	0.0.0.0:111	0.0.0.0:*	LISTEN	-	off (0.00/0/0)
tcp	83	0	192.168.88.132:8011	192.168.88.132:60974	ESTABLISHED	-	off (0.00/0/0)
tcp	0	0	192.168.88.132:60994	192.168.88.132:8011	ESTABLISHED	-	off (0.00/0/0)
tcp	0	0	192.168.88.132:60992	192.168.88.132:8011	ESTABLISHED	-	off (0.00/0/0)
tcp	83	0	192.168.88.132:8011	192.168.88.132:60994	ESTABLISHED	-	off (0.00/0/0)
tcp	83	0	192.168.88.132:8011	192.168.88.132:60966	ESTABLISHED	-	off (0.00/0/0)
tcp	83	0	192.168.88.132:8011	192.168.88.132:60982	ESTABLISHED	-	off (0.00/0/0)
tcp	0	0	192.168.88.132:60968	192.168.88.132:8011	ESTABLISHED	-	off (0.00/0/0)
tcp	0	0	192.168.88.132:60996	192.168.88.132:8011	ESTABLISHED	-	off (0.00/0/0)
tcp	83	0	192.168.88.132:8011	192.168.88.132:60968	ESTABLISHED	-	off (0.00/0/0)
tcp	83	0	192.168.88.132:8011	192.168.88.132:60964	ESTABLISHED	-	off (0.00/0/0)
tcp	0	0	192.168.88.132:60984	192.168.88.132:8011	ESTABLISHED	-	off (0.00/0/0)
tcp	83	0	192.168.88.132:8011	192.168.88.132:60962	ESTABLISHED	-	off (0.00/0/0)
tcp	0	0	192.168.88.132:60986	192.168.88.132:8011	ESTABLISHED	-	off (0.00/0/0)
tcp	83	0	192.168.88.132:8011	192.168.88.132:60984	ESTABLISHED	-	off (0.00/0/0)
tcp	0	0	127.0.0.1:43744	127.0.0.1:57846	ESTABLISHED	3021/node	off (0.00/0/0)
tcp	83	0	192.168.88.132:8011	192.168.88.132:60966	ESTABLISHED	-	off (0.00/0/0)
tcp	0	0	192.168.88.132:60952	192.168.88.132:8011	TIME WAIT	-	timewait (59.66/0/0)
tcp	0	0	192.168.88.132:60988	192.168.88.132:8011	ESTABLISHED	-	off (0.00/0/0)
tcp	0	0	192.168.88.132:60978	192.168.88.132:8011	ESTABLISHED	-	off (0.00/0/0)
tcp	0	0	192.168.88.132:60954	192.168.88.132:8011	FIN WAIT2	-	timewait (58.76/0/0)
tcp	1	0	192.168.88.132:8011	192.168.88.132:60954	CLOSE WAIT	9374/./webserver	off (0.00/0/0)
tcp	0	0	192.168.88.132:60948	192.168.88.132:8011	TIME WAIT	-	timewait (57.66/0/0)

- 通过观察8011端口的 Recv-Q 和 Send-Q，可以注意到这两个数据量并不大，其中一个甚至为零，表明接收消息的程序以及网络本身都在正常运行，没有出现明显的阻塞或延迟情况

结论：

1. 网页平均每秒响应一次请求，且并行性不高
2. 磁盘的读写请求的服务时间不长且利用率不高
3. 网页对服务器的读写请求量小

2. 源码分析:

```
/*根据消息类型, 将消息放入 logbuffer 缓存, 或直接将消息通过 socket 通道返回给客户端*/ switch (type)
{
case ERROR:
    (void)sprintf(logbuffer, "%s:ERROR: %s:%s Errno=%d exiting pid=%d", time_now, s1, s2, errno, getpid());
    break;
case FORBIDDEN:
    (void)write(socket_fd, "HTTP/1.1 403 Forbidden\nContent-Length: 185\nConnection: close\nContent-Type: text/html\n\n<html><head>\n<title>403 Forbidden</title>\n</head><body>\n<h1>Forbidden</h1>\n\nThe requested URL, file type or operation is not allowed on this simple static file webserver.\n</body></html>\n", 271);
    (void)sprintf(logbuffer, "%s:FORBIDDEN: %s:%s", time_now, s1, s2);
    break;
case NOTFOUND:
    (void)write(socket_fd, "HTTP/1.1 404 Not Found\nContent-Length: 136\nConnection: close\nContent-Type: text/html\n\n<html><head>\n<title>404 Not Found</title>\n</head><body>\n<h1>Not Found</h1>\n\nThe requested URL was not found on this server.\n</body></html>\n", 224);
    (void)sprintf(logbuffer, "%s:NOT FOUND: %s:%s", time_now, s1, s2);
    break;
case LOG:
    (void)sprintf(logbuffer, "%s:INFO: %s:%s:%d", time_now, s1, s2, socket_fd);
    break;
}

/* 将 logbuffer 缓存中的消息存入 webserver.log 文件*/
if ((fd = open("webserver.log", O_CREAT | O_WRONLY | O_APPEND, 0644)) >= 0)
{
    (void)write(fd, logbuffer, strlen(logbuffer));
    (void)write(fd, "\n", 1);
    (void)close(fd);
}
```

```
145     logger(LOG, "Header", buffer, hit);
146     (void)write(fd, buffer, strlen(buffer));
147
148     /* 不停地从文件里读取文件内容, 并通过 socket 通道向客户端返回文件内容*/
149     while ((ret = read(file_fd, buffer, BUFSIZE)) > 0)
150     {
151         (void)write(fd, buffer, ret);
152     }
153     sleep(1); /* sleep 的作用是防止消息未发出, 已经将此 socket 通道关闭*/
154     close(fd);
155 }
```

分析: 在webserver中, 唯一涉及到的write函数存在于logger和web函数中。结合之前的性能监测, 这些I/O操作对系统性能的损害相对较小。由于请求的网页不需要加载大量资源, 因此对内存和磁盘数据量的读写并不频繁。然而, 由于这些I/O操作的存在, 请求服务的时间会有一定的损耗。

```
197     /* 建立服务端侦听 socket*/
198     if ((listenfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
199     {
200         logger(ERROR, "system call", "socket", 0);
201         port = atoi(argv[1]);
202         if (port < 0 || port > 60000)
203             logger(ERROR, "Invalid port number (try 1->60000)", argv[1], 0);
204         serv_addr.sin_family = AF_INET;
205         serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
206         serv_addr.sin_port = htons(port);
207         if (bind(listenfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
208             logger(ERROR, "system call", "bind", 0);
209         if (listen(listenfd, 64) < 0)
210             logger(ERROR, "system call", "listen", 0);
211         for (hit = 1;; hit++)
212         {
213             length = sizeof(cli_addr);
214             if ((socketfd = accept(listenfd, (struct sockaddr *)&cli_addr, &length)) < 0)
215                 logger(ERROR, "system call", "accept", 0);
216             web(socketfd, hit); /* never returns */
217         }
218     }
```

服务端侦听部分, 对系统的损耗如下:

- 内存泄漏, 该程序没有释放已分配的内存, 这可能导致内存泄漏
- 无限循环, 没有强制退出则一直占用系统资源
- 单线程处理, 只有一个线程处理所有客户端连接, 可能导致性能问题

题目六:

1. 添加相关计时函数监测程序各个部分执行时间

```
1 //计时函数设计
2 struct timeval time_s, time_e; //设置全局变量, 为timeval结构体的对象
3
4 gettimeofday(&time_s, NULL); //获取当前时间
5 .
6 {相关程序部分}
7 .
8 gettimeofday(&time_e, NULL); //获取当前时间
9
10 printf(" - {相关程序部分}耗时: %ld us\n", time_e.tv_usec - time_s.tv_usec); //
    打印输出时间差
```

```
1 //webserver.c
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <unistd.h>
5 #include <errno.h>
6 #include <string.h>
7 #include <fcntl.h>
8 #include <signal.h>
9 #include <sys/types.h>
10 #include <sys/socket.h>
11 #include <netinet/in.h>
12 #include <arpa/inet.h>
13 #include <sys/time.h>
14 #include <time.h>
15 #define VERSION 23
16 #define BUFSIZE 8096
17 #define ERROR 42
18 #define LOG 44
19 #define FORBIDDEN 403
20 #define NOTFOUND 404
21 #ifndef SIGCLD
22 #define SIGCLD SIGCHLD
23 #endif
24
25 // 时间监测
26 struct timeval time_s, time_e;
27
28 struct
29 {
30     char *ext;
31     char *filetype;
32 } extensions[] = {
33     {"gif", "image/gif"},
34     {"jpg", "image/jpg"},
35     {"jpeg", "image/jpeg"},
```

```

36     {"png", "image/png"},
37     {"ico", "image/ico"},
38     {"zip", "image/zip"},
39     {"gz", "image/gz"},
40     {"tar", "image/tar"},
41     {"htm", "text/html"},
42     {"html", "text/html"},
43     {0, 0}};
44
45 /* 日志函数，将运行过程中的提示信息记录到 webserver.log 文件中*/
46 void logger(int type, char *s1, char *s2, int socket_fd)
47 {
48     int fd;
49     char logbuffer[BUFSIZE * 2];
50
51     char time_now[30];
52     time_t rawtime;
53     struct tm *timeinfo;
54     time(&rawtime); // 获取当前系统时间
55     timeinfo = localtime(&rawtime); // 转化成当地时间
56     strftime(time_now, sizeof(time_now), "%Y-%m-%d %H:%M:%S", timeinfo); //
    对时间进行格式化
57
58     /*根据消息类型，将消息放入 logbuffer 缓存，或直接将消息通过 socket 通道返回给客户
    端*/
59     gettimeofday(&time_s, NULL);
60     switch (type)
61     {
62     case ERROR:
63         (void)sprintf(logbuffer, "%s:ERROR: %s:%s Errno=%d exiting pid=%d",
    time_now, s1, s2, errno, getpid());
64         break;
65     case FORBIDDEN:
66         (void)write(socket_fd, "HTTP/1.1 403 Forbidden\nContent-Length:
    185\nConnection: close\nContent-Type: text/html\n\n<html><head>\n<title>403
    Forbidden</title>\n</head><body>\n<h1>Forbidden</h1>\n The requested URL,
    file type or operation is not allowed on this simple static file
    webserver.\n</body></html>\n", 271);
67         (void)sprintf(logbuffer, "%s:FORBIDDEN: %s:%s", time_now, s1, s2);
68         break;
69     case NOTFOUND:
70         (void)write(socket_fd, "HTTP/1.1 404 Not Found\nContent-Length:
    136\nConnection: close\nContent-Type: text/html\n\n<html>
    <head>\n<title>404 Not Found</title>\n</head><body>\n<h1>Not
    Found</h1>\nThe requested URL was not found on this server.\n</body>
    </html>\n", 224);
71         (void)sprintf(logbuffer, "%s:NOT FOUND: %s:%s", time_now, s1, s2);
72         break;
73     case LOG:
74         (void)sprintf(logbuffer, "%s:INFO: %s:%s:%d", time_now, s1, s2,
    socket_fd);
75         break;
76     }
77
78     /* 将 logbuffer 缓存中的消息存入 webserver.log 文件*/

```

```

79     if ((fd = open("webserver.log", O_CREAT | O_WRONLY | O_APPEND, 0644))
    >= 0)
80     {
81         (void)write(fd, logbuffer, strlen(logbuffer));
82         (void)write(fd, "\n", 1);
83         (void)close(fd);
84     }
85     gettimeofday(&time_e, NULL);
86     printf(" - 将 logbuffer 缓存中的消息存入 webserver.log 文件耗时: %ld
    us\n", time_e.tv_usec - time_s.tv_usec);
87 }
88
89 /* 此函数完成了 webServer 主要功能，它首先解析客户端发送的消息，然后从中获取客户端请求
    的文件名，然后根据文件名从本地将此文件读入缓存，并生成相应的 HTTP 响应消息；最后通过服务
    器与客户端的 socket 通道向客户端返回 HTTP 响应消息*/
90 void web(int fd, int hit)
91 {
92     printf("接收客户端请求\n");
93     gettimeofday(&time_s, NULL);
94     int j, file_fd, buflen;
95     long i, ret, len;
96     char *fstr;
97     static char buffer[BUFSIZE + 1]; /* 设置静态缓冲区 */
98     ret = read(fd, buffer, BUFSIZE); /* 从连接通道中读取客户端的请求消息 */
99     if (ret == 0 || ret == -1)
100     { // 如果读取客户端消息失败，则向客户端发送 HTTP 失败响应信息
101         logger(FORBIDDEN, "failed to read browser request", "", fd);
102     }
103     if (ret > 0 && ret < BUFSIZE) /* 设置有效字符串，即将字符串尾部表示为 0 */
104         buffer[ret] = 0;
105     else
106         buffer[0] = 0;
107     for (i = 0; i < ret; i++) /* 移除消息字符串中的“CF”和“LF”字符*/
108         if (buffer[i] == '\r' || buffer[i] == '\n')
109             buffer[i] = '*';
110     logger(LOG, "request", buffer, hit);
111     gettimeofday(&time_e, NULL);
112     printf(" - 从连接通道中读取客户端的请求消息耗时: %ld us\n", time_e.tv_usec -
    time_s.tv_usec);
113
114     /*判断客户端 HTTP 请求消息是否为 GET 类型，如果不是则给出相应的响应消息*/
115     gettimeofday(&time_s, NULL);
116     if (strncmp(buffer, "GET ", 4) && strncmp(buffer, "get ", 4))
117     {
118         logger(FORBIDDEN, "Only simple GET operation supported", buffer,
    fd);
119     }
120     for (i = 4; i < BUFSIZE; i++)
121     { /* null terminate after the second space to ignore extra stuff */
122         if (buffer[i] == ' ')
123         { /* string is "GET URL " +lots of other stuff */
124             buffer[i] = 0;
125             break;
126         }
127     }
128 }

```

```

127     gettimeofday(&time_e, NULL);
128     printf(" - 判断客户端 HTTP 请求消息类型耗时: %ld us\n", time_e.tv_usec -
time_s.tv_usec);
129
130     /* 在消息中检测路径, 不允许路径中出现 "." */
131     gettimeofday(&time_s, NULL);
132     for (j = 0; j < i - 1; j++)
133         if (buffer[j] == '.' && buffer[j + 1] == '.')
134             {
135                 logger(FORBIDDEN, "Parent directory (..) path names not
supported", buffer, fd);
136             }
137     if (!strncmp(&buffer[0], "GET /\0", 6) || !strncmp(&buffer[0], "get
/\0", 6))
138         (void)strcpy(buffer, "GET /index.html"); // 如果请求消息中没有包含有效
的文件名, 则使用默认的文件名 index.html
139     gettimeofday(&time_e, NULL);
140     printf(" - 检测消息路径耗时: %ld us\n", time_e.tv_usec - time_s.tv_usec);
141
142     /* 根据预定义在 extensions 中的文件类型, 检查请求的文件类型是否本服务器支持 */
143     gettimeofday(&time_s, NULL);
144     buflen = strlen(buffer);
145     fstr = (char *)0;
146     for (i = 0; extensions[i].ext != 0; i++)
147     {
148         len = strlen(extensions[i].ext);
149         if (!strncmp(&buffer[buflen - len], extensions[i].ext, len))
150             {
151                 fstr = extensions[i].filetype;
152                 break;
153             }
154     }
155     if (fstr == 0)
156         logger(FORBIDDEN, "file extension type not supported", buffer, fd);
157
158     if ((file_fd = open(&buffer[5], O_RDONLY)) == -1)
159     { /* 打开指定的文件名*/
160         logger(NOTFOUND, "failed to open file", &buffer[5], fd);
161     }
162     logger(LOG, "SEND", &buffer[5], hit);
163     len = (long)lseek(file_fd, (off_t)0, SEEK_END); /* 通过 lseek 获取文件长
度*/
164     (void)lseek(file_fd, (off_t)0, SEEK_SET); /* 将文件指针移到文件首位置
*/
165     /* Header + a blank line */
166     (void)sprintf(buffer, "HTTP/1.1 200 OK\nServer: nweb/%d.0\nContent-
Length: %ld\nConnection: close\nContent-Type: %s\n\n", VERSION, len, fstr);
167     logger(LOG, "Header", buffer, hit);
168     (void)write(fd, buffer, strlen(buffer));
169     gettimeofday(&time_e, NULL);
170     printf(" - 检查请求的文件类型是否本服务器支持耗时: %ld us\n", time_e.tv_usec
- time_s.tv_usec);
171
172     /* 不停地从文件里读取文件内容, 并通过 socket 通道向客户端返回文件内容*/
173     gettimeofday(&time_s, NULL);

```

```

174     while ((ret = read(file_fd, buffer, BUFSIZE)) > 0)
175     {
176         (void)write(fd, buffer, ret);
177     }
178     gettimeofday(&time_e, NULL);
179     printf(" - 从文件里读取文件内容, 并通过 socket 通道向客户端返回耗时: %ld us\n",
time_e.tv_usec - time_s.tv_usec);
180
181     sleep(1); /* sleep 的作用是防止消息未发出, 已经将此 socket 通道关闭*/
182     close(fd);
183 }
184
185 int main(int argc, char **argv)
186 {
187
188     int i, port, listenfd, socketfd, hit;
189     socklen_t length;
190     static struct sockaddr_in cli_addr; /* static = initialised to zeros
*/
191     static struct sockaddr_in serv_addr; /* static = initialised to zeros
*/
192
193     /*解析命令参数*/
194     if (argc < 3 || argc > 3 || !strcmp(argv[1], "-?"))
195     {
196         (void)printf("hint: nweb Port-Number Top-Directory\t\tversion
%d\n\n"
197                     "\tnweb is a small and very safe mini web server\n"
198                     "\tnweb only servers out file/web pages with
extensions named below\n"
199                     "\t and only from the named directory or its sub-
directories.\n"
200                     "\tThere is no fancy features = safe and secure.\n\n"
201                     "\tExample:webserver 8181 /home/nwebdir &\n\n"
202                     "\tOnly supports:",
VERSION);
203         for (i = 0; extensions[i].ext != 0; i++)
204             (void)printf(" %s", extensions[i].ext);
205
206         (void)printf("\n\tNot Supported: URLs including \"..\", Java,
Javascript, CGI\n"
207                     "\tNot Supported: directories / /etc /bin /lib /tmp
/usr /dev /sbin \n"
208                     "\tNo warranty given or implied\n\tNigel Griffiths
nag@uk.ibm.com\n");
209         exit(0);
210     }
211
212     if (!strncmp(argv[2], "/", 2) || !strncmp(argv[2], "/etc", 5) ||
213         !strncmp(argv[2], "/bin", 5) || !strncmp(argv[2], "/lib", 5) ||
214         !strncmp(argv[2], "/tmp", 5) || !strncmp(argv[2], "/usr", 5) ||
215         !strncmp(argv[2], "/dev", 5) || !strncmp(argv[2], "/sbin", 6))
216     {
217         (void)printf("ERROR: Bad top directory %s, see nweb -?\n",
argv[2]);
218         exit(3);

```



```

219     }
220     if (chdir(argv[2]) == -1)
221     {
222         (void)printf("ERROR: Can't Change to directory %s\n", argv[2]);
223         exit(4);
224     }
225
226     /* 建立服务端侦听 socket*/
227     gettimeofday(&time_s, NULL);
228     if ((listenfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
229         logger(ERROR, "system call", "socket", 0);
230     port = atoi(argv[1]);
231     if (port < 0 || port > 60000)
232         logger(ERROR, "Invalid port number (try 1->60000)", argv[1], 0);
233     serv_addr.sin_family = AF_INET;
234     serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
235     serv_addr.sin_port = htons(port);
236     if (bind(listenfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) <
237         0)
238         logger(ERROR, "system call", "bind", 0);
239     if (listen(listenfd, 64) < 0) // 开始侦听socket连接, 最大连接数为64
240         logger(ERROR, "system call", "listen", 0);
241     gettimeofday(&time_e, NULL);
242     printf(" - 建立服务端侦听耗时: %ld us\n", time_e.tv_usec -
243         time_s.tv_usec);
244
245     for (hit = 1;; hit++)
246     {
247         gettimeofday(&time_s, NULL);
248         length = sizeof(cli_addr);
249
250         if ((socketfd = accept(listenfd, (struct sockaddr *)&cli_addr,
251             &length)) < 0)
252             logger(ERROR, "system call", "accept", 0);
253         web(socketfd, hit); /* never returns */
254         gettimeofday(&time_e, NULL);
255         printf("完成第 %d 次响应请求耗时: %ld us\n", hit, time_e.tv_usec -
256             time_s.tv_usec);
257         printf("\n");
258     }
259 }

```



```




○ [linux1@bogon web]$ ./webserver 8011 ../web
- 建立服务端侦听耗时: 91 us
接收客户端请求
- - 将 logbuffer 缓存中的消息存入 webserver.log 文件耗时: 94 us
- 从连接通道中读取客户端的请求消息耗时: 117 us
- 判断客户端 HTTP 请求消息类型耗时: 0 us
- 检测消息路径耗时: 0 us
- - 将 logbuffer 缓存中的消息存入 webserver.log 文件耗时: 26 us
- - 将 logbuffer 缓存中的消息存入 webserver.log 文件耗时: 5 us
- 检查请求的文件类型是否本服务器支持耗时: 59 us
- 从文件里读取文件内容, 并通过 socket 通道向客户端返回耗时: 4 us
完成第 1 次响应请求耗时: 398 us

接收客户端请求
- - 将 logbuffer 缓存中的消息存入 webserver.log 文件耗时: 66 us
- 从连接通道中读取客户端的请求消息耗时: 68 us
- 判断客户端 HTTP 请求消息类型耗时: 0 us
- 检测消息路径耗时: 1 us
- - 将 logbuffer 缓存中的消息存入 webserver.log 文件耗时: 40 us
- - 将 logbuffer 缓存中的消息存入 webserver.log 文件耗时: 3 us
- - 将 logbuffer 缓存中的消息存入 webserver.log 文件耗时: 3 us
- 检查请求的文件类型是否本服务器支持耗时: 5 us
- 从文件里读取文件内容, 并通过 socket 通道向客户端返回耗时: 0 us
完成第 2 次响应请求耗时: 322 us

接收客户端请求
- - 将 logbuffer 缓存中的消息存入 webserver.log 文件耗时: 104 us
- 从连接通道中读取客户端的请求消息耗时: 118 us
- 判断客户端 HTTP 请求消息类型耗时: 0 us
- 检测消息路径耗时: 0 us
- - 将 logbuffer 缓存中的消息存入 webserver.log 文件耗时: 3 us
- - 将 logbuffer 缓存中的消息存入 webserver.log 文件耗时: 4 us
- 检查请求的文件类型是否本服务器支持耗时: 36 us
- 从文件里读取文件内容, 并通过 socket 通道向客户端返回耗时: 5 us
完成第 3 次响应请求耗时: 663 us

```

分析：这是一次网页访问程序响应的请求次数。我用  表示一级功能调用， 表示二级功能调用，从输出的信息也能大致的反映程序的执行过程和函数调用关系。不过我们的重点是分析时间损耗。

- 建立服务端侦听
 - 涉及大量网络通信操作，网络延迟影响耗时
 - 系统资源紧张，会影响创建socket、绑定地址和端口等操作
- 将日志缓存中的信息写入.log文件
 - 涉及到大量I/O操作和文件的打开/关闭操作
- 接收客户端请求
 -  `accept` 在接收客户端连接请求时受网络延迟影响
- 从连接通道读取客户端请求信息
 -  `read` 函数在读取客户端的请求消息时受网络延迟影响
- 从文件里读取内容，并通过 socket 通道向客户端返回
 -  `read` 函数涉及网络通信，受网络延迟影响
 - 涉及到文件读取和.log文件写入等I/O操作

2. 使用perf工具监测webserver程序

```
• [linux1@localhost web]$ sudo perf stat ./webserver 8011 ../web
./webserver: Broken pipe

Performance counter stats for './webserver 8011 ../web':

          9.49 msec task-clock                #    0.000 CPUs utilized
             20      context-switches        #    0.002 M/sec
              4      cpu-migrations           #    0.422 K/sec
            173      page-faults              #    0.018 M/sec
<not supported>      cycles
<not supported>      instructions
<not supported>      branches
<not supported>      branch-misses

      21.361158639 seconds time elapsed

      0.000000000 seconds user
      0.010260000 seconds sys
```

- 在CPU上运行时间为9.49 ms，CPU利用率接近0
- 缺页失效次数173次，这与程序中大量的函数调用和循环结构有关

问题	输出	调试控制台	终端	端口
Samples: 35 of event 'cpu-clock', Event count (approx.): 8750000				
Children	Self	Command	Shared Object	Symbol
+ 77.14%	0.00%	webserver	[kernel.kallsyms]	[k] tcp_transmit_skb
+ 77.14%	0.00%	webserver	[kernel.kallsyms]	[k] ip_queue_xmit
+ 77.14%	0.00%	webserver	[kernel.kallsyms]	[k] ip_local_out_sk
+ 77.14%	0.00%	webserver	[kernel.kallsyms]	[k] ip_output
+ 77.14%	0.00%	webserver	[kernel.kallsyms]	[k] ip_finish_output
+ 77.14%	0.00%	webserver	[kernel.kallsyms]	[k] dev_queue_xmit
+ 77.14%	0.00%	webserver	[kernel.kallsyms]	[k] __dev_queue_xmit
+ 77.14%	0.00%	webserver	[kernel.kallsyms]	[k] sch_direct_xmit
+ 77.14%	0.00%	webserver	[kernel.kallsyms]	[k] dev_hard_start_xmit
+ 74.29%	74.29%	webserver	[kernel.kallsyms]	[k] e1000_xmit_frame
+ 74.29%	0.00%	webserver	[kernel.kallsyms]	[k] __tcp_push_pending_frames
+ 74.29%	0.00%	webserver	[kernel.kallsyms]	[k] tcp_write_xmit
+ 60.00%	0.00%	webserver	[kernel.kallsyms]	[k] system_call_fastpath
+ 51.43%	0.00%	webserver	libc-2.17.so	[.] __GI__libc_write
+ 51.43%	0.00%	webserver	[kernel.kallsyms]	[k] sys_write
+ 51.43%	0.00%	webserver	[kernel.kallsyms]	[k] vfs_write
+ 40.00%	0.00%	webserver	libc-2.17.so	[.] __GI__libc_close
+ 40.00%	0.00%	webserver	[kernel.kallsyms]	[k] __int_signal
+ 40.00%	0.00%	webserver	[kernel.kallsyms]	[k] do_notify_resume
+ 40.00%	0.00%	webserver	[kernel.kallsyms]	[k] task_work_run
+ 40.00%	0.00%	webserver	[kernel.kallsyms]	[k] __fput
+ 40.00%	0.00%	webserver	[kernel.kallsyms]	[k] __fput
+ 40.00%	0.00%	webserver	[kernel.kallsyms]	[k] sock_close
+ 40.00%	0.00%	webserver	[kernel.kallsyms]	[k] sock_release
+ 40.00%	0.00%	webserver	[kernel.kallsyms]	[k] inet_release
+ 40.00%	0.00%	webserver	[kernel.kallsyms]	[k] tcp_close
+ 40.00%	0.00%	webserver	[kernel.kallsyms]	[k] tcp_send_fin
+ 37.14%	0.00%	webserver	[kernel.kallsyms]	[k] do_sync_write
+ 34.29%	0.00%	webserver	[kernel.kallsyms]	[k] sock_aio_write
Tip: Compare performance results with: perf diff [old file] [new file]				

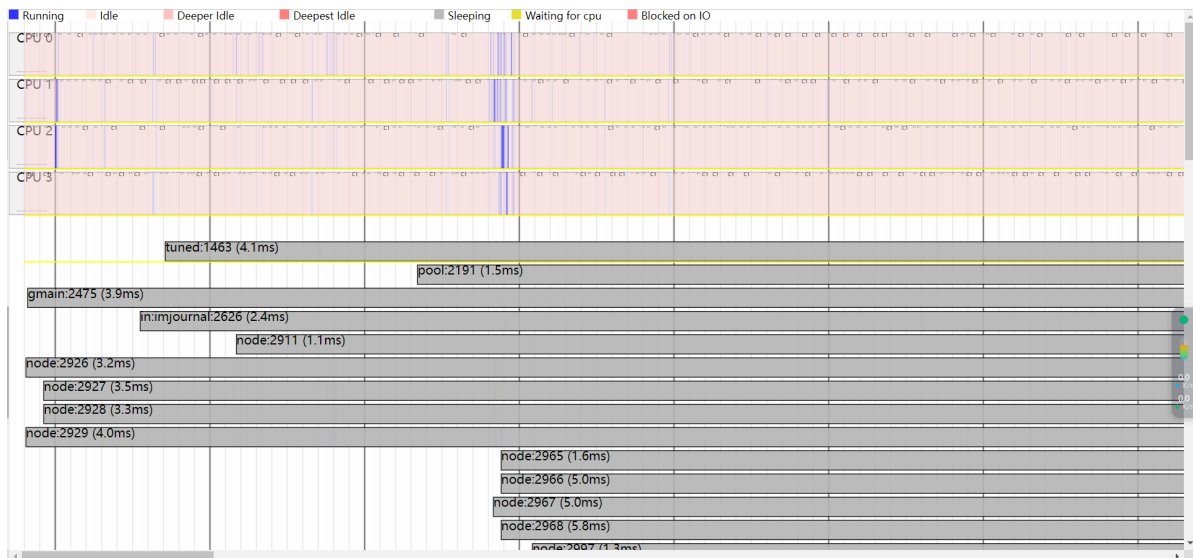
- 可以看出，消耗CPU时间较多的函数都是些网络栈中的核心函数，说明程序中网络数据处理方面需要优化

```
• [linux1@bogon web]$ sudo perf timechart record ./single-process-server 8011 ../web
[ perf record: Woken up 24 times to write data ]
[ perf record: Captured and wrote 7.044 MB perf.data (60902 samples) ]
```

```

• [linux1@bogon web]$ sudo /root/http_load-12mar2006/http_load -p 20 -s 30 url.txt
30 fetches, 20 max parallel, 8190 bytes, in 30.0016 seconds
273 mean bytes/connection
0.999946 fetches/sec, 272.985 bytes/sec
msecs/connect: 0.1035 mean, 0.376 max, 0.013 min
msecs/first-response: 13007 mean, 20010.9 max, 0.207 min
HTTP response codes:
code 200 -- 30

```



- 可以看出该程序CPU利用率低，大部分时间CPU都处于空闲状态

耗时函数：

- accept(), 接收用户请求，受网络延迟影响
- logger(), 涉及大量I/O操作
- web(), 涉及大量网络通信操作，受网络延迟影响
- read(), 读取客户端的请求消息受网络延迟影响
- write(), I/O操作

题目七：

性能低下原因：

- 单进程处理客户端请求
- 缺少对程序发生错误时的处理
- I/O操作频繁
- 对网络延迟依赖较高

解决方法：

- 采用多进程异步处理编程，每个进程单独处理各自的请求
- 多进程编程下，用子进程处理用户请求，并等待子进程处理的返回结果，以便在父进程中处理错误
- 采用缓存机制，将要读/写的内容先存入缓存，一次性完成I/O操作
- 采用预加载机制，在接收网页请求时提前从服务器获取资源，需要时直接读取并渲染，避免等待下载时间
- 采用多级存储层次结构，将频繁使用的资源存在相对CPU较近的地方，减少存取延迟
- 采用更高效的协议（如HTTP/2）来减少网络延迟的影响