1. 使用 fork 函数,设计并实现 WebServer 以支持多进程并发处理众多客户端的请求。

- 问题：饿死，某些子进程一直运行不了，或一直运行父进程
- 用http_load 运行不了，只能运行前几个进程?
- 僵尸进程?
- 这两个问题，问了王霖同学，他在《深入了解计算机系统》一书中看了相应内容，完美解决问题，感谢就不多说了
- signal 与 close，详情见代码
- 还是得提高自学的能力，找相关资料
- 效果图：

```
❯ ./webserver 8000 ../1/HTML &
[1] 30238
❯ ./http_load -parallel 5 -fetches 50 -seconds 20 http
95 fetches, 5 max parallel, 28215 bytes, in 20.0015 seconds
297 mean bytes/connection
4.74964 fetches/sec, 1410.64 bytes/sec
msecs/connect: 0.266895 mean, 0.765 max, 0.037 min
msecs/first-response: 0.717537 mean, 1.141 max, 0.257 min
HTTP response codes:
  code 200 -- 95
```

2. 使用信号量、共享内存等系统接口函数,来统计每个子进程的消耗时间以及所有子进程消耗时间之和。

- clock 计时不好使了（并行 cpu 时钟问题），换 timeval
- 每个客户端并行？实现的是每个 accept 过程并行，一个客户端有三次 accept
- 基本问题：互斥，死锁与饿死
- 要点：关于共享变量的操作，一定要看加不加锁

1. 每个进程时间

- 计算开始时间，简单，入子进程时计即可
- 终止时间，如何确定哪句是一个进程的结束，结束后若去运行别的，造成错误。需要保证运行完后立刻计时?
- 牺牲掉最后一句并行性，在最后一句之前加锁，计时后释放；
- 但最后一句话并不重要，规定计时后才算子进程结束，解决问题。

2. 总时间

- 规定：所有子进程结束，一段时间里同时运行的子进程结束，为一段总时间，后面还可能产生子进程，计入下一个总时间。
- 用一共享内存 count，指示进程数
- 开始时间在子进程入口计，若 count 为 0，即为开始时间
- 结束时间，在子进程结束时加一判断 count 是否为 0 即可
- 不要写入父进程!

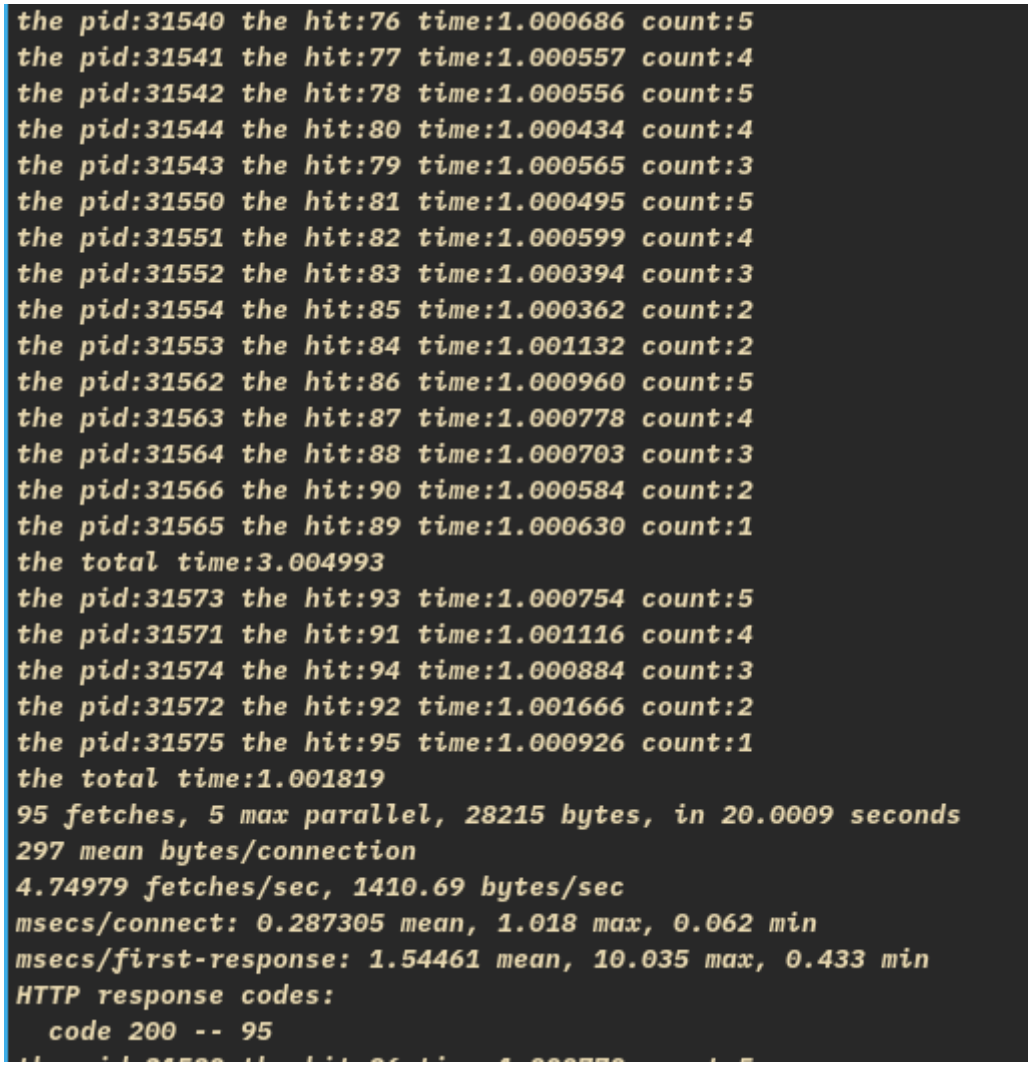
- 还有饿死的问题.....

- 资源释放的问题，程序是不会终止的，只有强制终止，资源释放?
- 统计如下：

```
) make
gcc -std=gnu99 -Wall -g -o webservert1 timemultiwebserver1.c -lrt -lpthread
) ./webservert1 8000 ../1/HTML
the pid:218608 the hit:1 time:1.001540 count:3
the pid:218609 the hit:2 time:1.001005 count:2
the pid:218618 the hit:3 time:1.000486 count:1
the total time:1.737417
the pid:218623 the hit:4 time:1.000652 count:5
the pid:218624 the hit:5 time:1.000820 count:4
the pid:218632 the hit:6 time:1.000458 count:4
the pid:218633 the hit:7 time:1.000590 count:4
the pid:218634 the hit:8 time:1.000937 count:4
the pid:218638 the hit:9 time:1.000462 count:3
the pid:218642 the hit:10 time:1.000708 count:3
the pid:218644 the hit:11 time:1.001391 count:2
the pid:218647 the hit:12 time:1.000475 count:1
the total time:3.258792
the pid:218698 the hit:13 time:1.000652 count:6
the pid:218699 the hit:14 time:1.001089 count:7
the pid:218704 the hit:15 time:1.000271 count:8
the pid:218705 the hit:16 time:1.000344 count:7
the pid:218706 the hit:17 time:1.000600 count:6
the pid:218709 the hit:18 time:1.000708 count:6
the pid:218712 the hit:19 time:1.000342 count:6
the pid:218713 the hit:20 time:1.000404 count:6
the pid:218717 the hit:21 time:1.000265 count:7
the pid:218719 the hit:22 time:1.000521 count:7
the pid:218720 the hit:23 time:1.000269 count:8
the pid:218721 the hit:24 time:1.000406 count:7
the pid:218722 the hit:25 time:1.000344 count:6
the pid:218723 the hit:26 time:1.000427 count:6
the pid:218725 the hit:27 time:1.000355 count:5
the pid:218726 the hit:28 time:1.000879 count:4
the pid:218732 the hit:29 time:1.000272 count:5
the pid:218734 the hit:30 time:1.000483 count:4
the pid:218735 the hit:31 time:1.000382 count:4
the pid:218736 the hit:32 time:1.000764 count:4
the pid:218737 the hit:33 time:1.001176 count:4
the pid:218742 the hit:34 time:1.000409 count:3
the pid:218743 the hit:35 time:1.000592 count:2
the pid:218744 the hit:36 time:1.000605 count:1
the total time:5.530132
^C
```

3. 使用 http_load 来测试当前设计的多进程 WebServer 服务性能,根据测试结果来分析其比单进程 Web 服务性能提高的原因。同时结合题目 2, 来分析当前多进程 WebServer 的性能瓶颈在何处?是否还能够继续提高此 WebServer 服务的性能?

- http_load:



- 并行处理 accept 请求，每发一次请求都是一个子进程，加快了速度
- 性能可进一步并行化，每次写入？