HTTP File Server

თქვენი ამოცანაა დაწეროთ HTTP File სერვერი.

(კონფიგურაციის ფაილზე დეტალური ინფორმაცია იხილეთ ქვემოთ)

## მოთხოვნები

* **Basic HTTP File server:** 
  + Აპლიკაციამ კონფიგურაციის ფაილში სერვერების სიიდან უნდა აირჩიოს პირველი ელემენტი, ამ ელემენტის IP:PORT პარამეტრებზე გახსნას TCP სოკეტი და კლიენტების მოთხოვნებს მოემსახუროს documentroot დირექტორიიდან.
  + Სოკეტის დასასტარტად აიღეთ კონფიგურაციის ფაილში ჩამოთვლილი სერვერებიდან პირველის პარამეტრები
  + Უნდა მუშაობდეს HTTP მეთოდები: GET, HEAD
  + Უნდა მუშაობდეს HTTP Response Status Code 404 Not Found (თუ მოთხოვნილი ფაილი ან დირექტორია არ მოიძებნა)
  + ფაილის მისამართზე (path) გადასვლილას - სერვერმა უნდა დაიწყოს ფაილის მიწოდება კლიენტისთვის
  + დირექტორიის მისამართზე გადასვლისას - სერვერმა უნდა დააბრუნოს შესაბამის დირექტორიის კონტენტის სია (ფაილების და ფოლდერების სიის HTML-ი რომელშიც თითოეული არის ამ ფაილის ან ფოლდერის [hyperlink](https://techterms.com/definition/hyperlink#:~:text=A%20hyperlink%20is%20a%20word,way%20from%20page%20to%20page.) მისამართი).
* **Virtual Hosting:** კონფიგურაციის ფაილში ჩამოთვლილია სერვერების სია, Შეიძლება ერთიდაიმავე IP:PORT ის კომბინაციაზე რამდენიმე სერვერის ჩანაწერი არსებობდეს, სხვადასხვა vhost მნიშვნელობით. Აპლიკაციამ Თითოეული IP:PORT წყვილისთვის უნდა გაიხსნას TCP სოკეტი და მოთხოვნას მოემსახუროს იმ სერვერის კონფიგურაციის documentroot იდან რომელი კონფიგურაციის vhost საც ემთხვევა მოთხოვნის host ჰედერის მნიშვნელობა.
* **Parallel Connections:** სერვერს უნდა შეეძლოს პარალელურად არანაკლებ 1024 Connection-ის მომსახურება. (თითოეული ახალი connection-ის დასამუშავებლად უნდა შეიქმნას ახალი Thread-ი)
* **Persistent Connection:** Შესაძლებელი უნდა იყოს Persistent Connection-ის გაკეთება “Connection: keep-alive” ჰედერის საშუალებით, (ქონექშენის ხანგრძლივობა 5 წამი) და უნდა მუშაობდეს “Connection: close” ჰედერიც
* **Range Header:** თქვენ სერვერს [Range](http://stackoverflow.com/a/18745164/660408) ჰედერის აღქმა უნდა შეეძლოს. Range ჰედერი გამოიყენება chunked სტრიმინგისას, მაგ. ვიდეოს გადმოსაწერად ან live სტრიმინგისას. მითითებული ჰედერის არსებობა სერვერს მიანიშნებს, რომ კლიენტი ითხოვს ფაილის შესახებ ინფორმაციას(Range: bytes=0-) ან ფაილის მონაკვეთის წაკითხვას(Range: bytes=3744-). ამ ფუნქციონალს შევამოწმებთ ბრაუზერის (Chrome) დახმარებით. ვიდეოს დასაკვრელად შეიძლება დაგჭირდეთ Content-Type და Content-Length ჰედერების დამატება. [Დეტალურად](https://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html)

Კონფიგურაციის ფაილი

Სერვერს გაშვებისას უნდა გადაეცემოდეს კონფიგურაციის ფაილის ლოკალური მისამართი, რომელიც JSON ფორმატისაა

(JSON ფაილის წასაკითხად გამოიყენეთ რაიმე არსებული ბიბლიოთეკა, მაგალითად [JSON](https://docs.python.org/3/library/json.html) :)

Კონფიგურაციის ფაილის მაგალითი:

{

“log”: “/path/to/logs/”,

"server": [{

"vhost": "example1.com",

"ip": "127.0.0.1",

"port": 80,

"documentroot": "/path/to/example1/"

},

"vhost": "example3.com",

"ip": "127.0.0.1",

"port": 80,

"documentroot": "/path/to/example3/"

},{

"vhost": "example2.com",

"ip": "0.0.0.0",

"port": 8080,

"documentroot": "/path/to/example2/"

}]

}

Კონფიგურაციის ფაილის აღწერა:

* **log** - დირექტორია თუ სად უნდა ინახებოდეს სერვერის ლოგები, ლოგირების ფორმატი დეტალურად იხილეთ ქვემოთ
* **server** - Virtual Host ების სია (List) რომელშიდაც თითოეული ელემენტი შედგება შემდეგი პარამეტრებისგან:
  + **vhost** - virtual host - ის სახელი
  + **ip/port** - ინტერფეისის მისამართი და პორტი რომელზეც ვაკვირდებით შემოსულ მოთხოვნებს
  + **documentroot -** host ის შესაბამისი დირექტორია საიდანაც შემოსულ რექვესთებზე ვაწვდით ფაილებს ან დირექტორიის კონტენტის სიას.

## Ლოგის სტრუქტურა

* ყოველი ვირტუალური ჰოსტისთვის(vhost) log დირექტორიაში შექმენით თითო ფაილი შემდეგი სახელით: [vhost\_name].log. მაგ. /path/to/logs/example1.com.log
* თითოული ვირტუალური ჰოსტის ლოგ ფაილში, თითოეული მოთხოვნის შესაბამისად უნდა ჩაიწეროს:
  + რა დროს მოხდა მომართვა
  + რა IP მისამართით
  + რომელ დომეინი იყო მითითებული მოთხოვნისას
  + მოთხოვნილი ფაილის ან დირექტორიის მისამართი (path)
  + რა სტატუს კოდი დავუბრუნეთ
  + რამდნი ბაიტი ინფორმაცია გაეგზავნა კლიენტს
  + კლიენტის user-agent ჰედერში მოწოდებული ინფორმაცია.
* Მაგალითად:

**[Sun Oct 28 12:30:37 2012] 192.168.100.231** [**www.example.com**](http://www.google.com/) **/index.html 200 124 "Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/31.0.1650.63 Safari/537.36"**

## DNS ჩანაწერები

Იმისათვის რომ ტესტებმა იმუშავოს საჭიროა კონფიგურაციის ფაილში არსებული სერვერების სიის შესაბამისი DNS ჩანაწერების გაკეთება, (server\_ip, server\_vhost ის წყვილების სახით უნდა ჩამოწეროთ “ლოკალური DNS ჩანაწერებში” ) ამისთვის იხილეთ ინსტრქუცია შემდეგ ლინკზე: <https://www.tecmint.com/set-permanent-dns-nameservers-in-ubuntu-debian/>.

## შეფასების კრიტერიუმი:

1. Basic HTTP File server: 30%
2. Virtual Hosting support: 20%
3. Parallel Connections support: 20%
4. Persistent Connection support: 15%
5. Range Header support: 15%
6. Logging: Bonus 5%

## Დამხმარე თულები: <3

Პარალელური შეერთებები:

Პარალელური შეერთებების სიმულირებისთვის გამოგადგებათ პროგრამა **ab (apache benchmark)** ან **siege**

sudo apt-get install apache2-utils

sudo apt-get install siege

გაშვების მაგალითი siege -v -r 1 -c 1024 example1.com/dir1/mgs1.png

ასევე გამოგადგებათ [postman](https://www.getpostman.com/) HTTP კლიენტი. უფრო დეტალური ანალიზისთვის - [telnet](http://www.anta.net/misc/telnet-troubleshooting/http.shtml). Chrome-ის გარდა Firefox-ზეც შეამოწმეთ თქვენი აპლიკაცია. ეს უკანასკნელი უფრო მკაცრად იცავს HTTP-ის სტანდარტებს, რაც მარტივად გამოავლენს თქვენი სერვერის შეცდომებს.

ასევე შეგიძლიათ გაუშვათ [httpbin](https://httpbin.org/) სერვერი, რომელიც პასუხად აბრუნებს თქვენგან მიღებული მოთხოვნის აღწერას.

მარტივი HTTP სერვერები <https://gist.github.com/willurd/5720255>

Range ჰედერის შესამოწმებლად შეგიძიათ ვიდეოები ნახოთ [ამ](http://www.sample-videos.com/) მისამართზე. Chrome-ის devtools-ით ან wireshark-ით დააკვირდით mp4 ფაილების გადმოსაწერად გაგზავნილ მოთხოვნებს