Webserv

Bu, URL'lerin neden HTTP ile başladığını nihayet anladığınız zamandır

Özet:
Bu proje kendi HTTP sunucunuzu yazmakla ilgilidir.
Gerçek bir tarayıcı ile test edebileceksiniz.
HTTP, internet üzerinde en yaygın kullanılan protokollerden biridir.
Bir web sitesi üzerinde çalışmayacak olsanız bile, bunun inceliklerini anlamak faydalı olacaktır.

Sürüm: 21.4

İçindekiler

Bölüm I Giriş

Köprü Metni Aktarım Protokolü (HTTP) dağıtılmış, işbirliğine dayalı, hiper ortam bilgi sistemleri için bir uygulama protokolüdür.

HTTP, hiper metin belgelerinin kullanıcının kolayca erişebileceği diğer kaynaklara köprüler içerdiği World Wide Web için veri iletişiminin temelidir. Örneğin, bir fare düğmesine tıklayarak veya bir web tarayıcısında ekrana dokunarak.

HTTP, hiper metin işlevselliğini ve World Wide Web'in büyümesini desteklemek için geliştirilmiştir.

Bir web sunucusunun birincil işlevi web sayfalarını depolamak, işlemek ve istemcilere iletmektir. İstemci-sunucu iletişimi Köprü Metni Aktarım Protokolü (HTTP) aracılığıyla gerçekleşir.

Teslim edilen sayfalar çoğunlukla HTML belgeleridir ve metin içeriğine ek olarak resimler, stil sayfaları ve komut dosyaları içerebilir.

Yüksek trafikli bir web sitesi için birden fazla web sunucusu kullanılabilir.

Bir kullanıcı aracısı, genellikle bir web tarayıcısı veya web tarayıcısı, HTTP kullanarak belirli bir kaynağı yeniden arayarak iletişimi başlatır ve sunucu bu kaynağın içeriğiyle veya bunu yapamazsa bir hata mesajıyla yanıt verir. Kaynak genellikle sunucunun ikincil depolama alanındaki gerçek bir dosyadır, ancak bu her zaman geçerli değildir ve web sunucusunun nasıl uygulandığına bağlıdır.

Birincil işlevi içerik sunmak olsa da, HTTP istemcilerin veri göndermesini de sağlar. Bu özellik, dosyaların yüklenmesi de dahil olmak üzere web formlarının gönderilmesi için kullanılır.

Bölüm II Genel kurallar

- Programınız hiçbir koşulda çökmemeli (belleği bitse bile) veya beklenmedik bir şekilde sonlanmamalıdır.
  Bu durumda, projeniz işlevsiz olarak değerlendirilecek ve notunuz 0 olacaktır.
- Kaynak dosyalarınızı derleyen bir Makefile göndermelisiniz. Gereksiz yeniden bağlama yapmamalıdır.
- Makefile dosyanız en azından kuralları içermelidir:
  $(NAME), all, clean, fclean ve re.
- Kodunuzu c++ ve -Wall -Wextra -Werror bayrakları ile derleyin
- Kodunuz **C++ 98 standardına** uygun olmalıdır ve -std=c++98 bayrağı eklendiğinde derlenmeye devam etmelidir.
- Mümkün olduğunca çok C++ özelliğinden yararlandığınızdan emin olun (örneğin, .h> yerine <cstring> seçin). C fonksiyonlarını kullanmanıza izin verilir, ancak mümkünse her zaman C++ versiyonlarını tercih edin.
- Herhangi bir harici kütüphane ve Boost kütüphaneleri yasaktır.

Bölüm III

Zorunlu kısım

Program adı	webserv
Dosyaları teslim edin	Makefile, *.{h, hpp}, *.cpp, *.tpp, *.ipp, yapılandırma dosyaları
Makefile	NAME, all, clean, fclean, re
Argümanlar	[Bir yapılandırma dosyası]
Harici fonksiyonlar.	Tüm işlevler C++ 98'de uygulanmalıdır. execve, pipe, strerror, gai_strerror, errno, dup, dup2, fork, socketpair, htons, htonl, ntohs, ntohl, select, poll, epoll (epoll_create, epoll_ctl, epoll_wait), kqueue (kqueue, kevent), socket, accept, listen, send, recv, chdir, bind, connect, getaddrinfo, freeaddrinfo, setsockopt, getsockname, getprotobyname, fcntl, close, read, write, waitpid, kill, signal, access, stat, open, opendir, readdir ve closedir.
Libft yetkili	n/a
Açıklama	C++ 98'de bir HTTP sunucusu

C++ 98'de bir HTTP sunucusu yazmalısınız.

Çalıştırılabilir dosyanız aşağıdaki gibi çalıştırılmalıdır:

./webserv [yapılandırma dosyası]

Konu ve notlandırma kriterlerinde poll()'dan bahsedilse de select(), kqueue() veya epoll() gibi herhangi bir eşdeğer fonksiyonu kullanabilirsiniz.

Bu projeye başlamadan önce lütfen RFC'yi okuyun ve telnet ve NGINX ile testler yapın.
RFC'nin tamamını uygulamak zorunda olmasanız da, okumak gerekli özellikleri geliştirmenize yardımcı olacaktır.

III.1 Gereksinimler

- Programınız bir yapılandırma dosyasını argüman olarak almalı veya varsayılan bir yol kullanmalıdır.
- Başka bir web sunucusunu çalıştıramazsınız.
- Sunucunuz her zaman blokajsız kalmalı ve gerektiğinde istemci bağlantılarını düzgün bir ele almalıdır.
- Bloklama yapmamalı ve istemci ile sunucu arasındaki tüm G/Ç işlemleri için sadece **1** poll() (veya eşdeğeri) kullanmalıdır (dinleme dahil).
- poll() (veya eşdeğeri) hem okuma hem de yazmayı aynı anda izlemelidir.
- poll() (veya eşdeğeri) işlevinden geçmeden asla bir okuma veya yazma işlemi yapmamalısınız.
- Bir okuma veya yazma işlemi gerçekleştirdikten sonra errno değerinin kontrol kesinlikle yasaktır.
- Yapılandırma dosyanızı okumadan önce poll() veya eşdeğeri) kullanmanız gerekmez.

Bloklama yapmayan dosya tanımlayıcıları kullanmanız gerektiğinden, poll() (veya eşdeğeri) olmadan okuma/recv veya yazma/gönderme işlevlerini kullanmanız mümkündür ve sunucunuz bloklama yapmayacaktır.
Ancak daha fazla sistem kaynağı tüketecektir.
Bu nedenle, poll() (veya eşdeğeri) kullanmadan herhangi bir dosya tanımlayıcısında okuma/geri alma veya yazma/gönderme yapmaya çalışırsanız, notunuz 0 olacaktır.

- FD_SET, FD_CLR, FD_ISSET ve FD_ZERO gibi her makro ve tanımı kullanabilirsiniz (ne yaptıklarını ve nasıl çalıştıklarını anlamak çok faydalıdır).
- Sunucunuza gelen bir istek asla süresiz olarak askıda kalmamalıdır.
- Sunucunuz, seçtiğiniz standart **web tarayıcıları** ile uyumlu olmalıdır.
- NGINX'in HTTP 1.1 uyumlu olduğunu ve başlıkları ve cevap davranışlarını karşılaştırmak için kullanılabileceğini düşüneceğiz.
- HTTP yanıt durum kodlarınız doğru olmalıdır.
- Hiçbiri sağlanmamışsa sunucunuzda **varsayılan hata sayfaları** olmalıdır.
- CGI dışında başka bir şey için (PHP veya Python) fork kullanamazsınız.
- **Tamamen statik bir web sitesi sunabilmeniz** gerekir.
- Müşteriler **dosya yükleyebilmelidir**.
- En azından GET, POST ve DELETE yöntemlerine ihtiyacınız vardır.

- Sunucunuzun her zaman kullanılabilir kalmasını sağlamak için stres testi yapın.
- Sunucunuz birden fazla bağlantı noktasını dinleyebilmelidir (bkz. *Yapılandırma dosyası*).

III.2 Yalnızca MacOS için

macOS write() işlevini diğer Unix tabanlı işletim sistemlerinden farklı şekilde ele aldığından, fcntl() işlevini kullanmanıza izin verilir.
Diğer Unix işletim sistemlerindekine benzer bir davranış elde etmek için dosya tanımlayıcılarını engellemesiz modda kullanmanız gerekir.

Ancak, fcntl() işlevini yalnızca aşağıdaki bayraklarla kullanmanıza izin verilir:
F_SETFL, O_NONBLOCK ve FD_CLOEXEC.
Başka herhangi bir bayrak yasaktır.

III.3 Yapılandırma dosyası

NGINX yapılandırma dosyasının 'sunucu' bölümünden ilham alabilirsiniz.

Yapılandırma dosyasında şunları yapabilmeniz gerekir:

- Her bir 'sunucunun' bağlantı noktasını ve ana bilgisayarını seçin.
- Sunucu_adlarını ayarlayın ya da .
- Bir ana bilgisayar:port için ilk sunucu bu ana bilgisayar:port için varsayılan olacaktır (yani başka bir sunucuya ait olmayan tüm isteklere yanıt verecektir).
- Varsayılan hata sayfalarını ayarlayın.
- İstemci istek gövdeleri için izin verilen maksimum boyutu ayarlayın.
- Aşağıdaki kurallardan/yapılandırmalardan bir veya birden fazlasıyla rotalar oluşturun (rotalar regexp kullanmayacaktır):
  - Rota için kabul edilen HTTP yöntemlerinin bir listesini tanımlayın.
  - Bir HTTP yönlendirmesi tanımlayın.
  - İstenen dosyanın bulunması gereken bir dizin veya dosya tanımlayın (örneğin, url /kapouet /tmp/www'ye köklenmişse, url /kapouet/pouic/toto/pouet /tmp/www/pouic/toto/pouet).
  - Dizin listelemeyi etkinleştirin veya devre dışı bırakın.

- İstek bir dizin için olduğunda sunulacak varsayılan bir dosya ayarlayın.
- Belirli dosya uzantısına göre CGI çalıştırın (örneğin .php).
- POST ve GET yöntemleri ile çalışmasını sağlayın.
- Rotanın yüklenen dosyaları kabul etmesine izin verin ve nereye kaydedileceklerini yapılandırın.
  - CGI'ın ne olduğunu merak ediyor musunuz?
  - CGI'yi doğrudan çağırmayacağınız için PATH_INFO olarak tam yolu kullanın.
  - Yalnızca, yığınlanmış istekler için sunucunuzun bunları yığınlamaması gerektiğini, CGI'nin gövdenin sonu olarak EOF bekleyeceğini .
  - Aynı durum CGI çıktısı için de geçerlidir. CGI'dan content_length döndürülmezse, EOF döndürülen verinin sonunu işaretleyecektir.
  - Programınız CGI'yi ilk argüman olarak istenen dosya ile çağırmalıdır.
  - CGI, göreli yol dosya erişimi için doğru dizinde çalıştırılmalıdır.
  - Sunucunuz en az bir CGI (php-CGI, Python, .) desteklemelidir.

Değerlendirme sırasında her özelliğin çalıştığını test etmek ve göstermek için yapılandırma dosyaları ve varsayılan dosyalar sağlamalısınız.

Belirli bir davranışla ilgili bir sorunuz varsa, programınızın davranışını NGINX'inkiyle karşılaştırmalısınız.
Örneğin, server_name'in nasıl çalıştığını kontrol edin.
Küçük bir test aracı sağladık. Tarayıcınız ve testlerinizle her şey yolunda gidiyorsa bunu kullanmak zorunlu değildir, ancak hataları bulmanıza ve düzeltmenize yardımcı olabilir.

Esneklik çok önemlidir. Sunucunuz her zaman çalışır durumda kalmalıdır.

Sadece tek bir programla test yapmayın. Testlerinizi Python veya Golang gibi daha uygun bir dilde, hatta tercih ederseniz C veya C++'da yazın.

Bölüm IV Bonus kısmı

İşte uygulayabileceğiniz bazı ek özellikler:

- Çerezleri ve oturum yönetimini destekleyin (basit örnekler verin).
- Birden fazla CGI işleyin.

Bonus bölümü yalnızca zorunlu bölümün sorunsuz bir şekilde tamamlanması halinde değerlendirilecektir. Tüm zorunlu gereklilikleri karşılayamazsanız, bonus bölümünüz değerlendirmeye alınmayacaktır.

Bölüm V

Sunum ve akran değerlendirmesi

ödevinizi her zamanki gibi Git deponuza gönderin. Savunma sırasında yalnızca içeriği değerlendirilecektir. Doğru olduklarından emin olmak için dosyalarınızın adlarını iki kez kontrol ettiğinizden emin olun.

16D85ACC441674FBA2DF65190663F42A3832CEA21E024516795E1223BBA77916734D1
26120A16827E1B16612137E59ECD492E46EAB67D109B142D49054A7C281404901890F
619D682524F5