BLGM 344 – Proje[1*]
WEB SUNUCUSU UYGULAMASI

Amaçlar

1. HTTP protokolünün öğrenilmesi
2. Verilen gereksinimleri sağlayan program yazılması
3. TCP sunucu programlamanın gerçek bir sistem oluşturarak kavranması

Proje

Kapsamı

Bu projede sizlerden basit bir web sunucusu yazmanız istenmektedir. Bu sunucu web tarayıcısı tarafından gönderilen "GET" isteklerine dosya sisteminde bulunan dosyaların içerikleriyle yanıt vermelidir. 4. deneyde yazmış olduğunuz sunucu, bu proje için temel oluşturacaktır. Bu proje üç adımdan oluşmaktadır.

1. Gelen isteğin okunarak yorumlanması
2. İstekte belirtilen kaynağın okunup yanıtın hazırlanarak gönderilmesi
3. Hata durumlarının kontrolü, erişim ve hata dosyaları ve sistemin testi

İstenilenler

Projenin ilk iki kısmından sonra ara raporlar, projenin sonunda ise yazılan programın kodları ve proje raporu teslim edilmelidir. Ara raporlar yaklaşık yarım sayfa, son rapor ise bir sayfa olmalıdır. Raporlara belirtilen kısımlarda uyguladığınız adımları, öğrendiklerinizi, yaşadığınız sorunları yazabilirsiniz. Son raporda kısaca programınızın çalışma mantığını anlatmanız da istenmektedir.

---

[*] BLGM 344 dersi için Bahar 2012/2013 döneminde Gürcü Öz ve Cem Kalyoncu tarafından hazırlanmıştır

[**] BLGM 344 dersi için Bahar 2015/2016 döneminde Can Genç tarafından modifiye edilmiştir.

Ön Bilgi

HTTP Protokolü

HTTP (Hypertext Transfer Protocol, hiper metin iletim protokolü) web kaynaklarının dağıtılmasını sağlayan bir iletişim kuralıdır. 1990 yılında kullanıma başlanmıştır. İstek cevap ilkesine göre çalışır. Şu anda HTTP 1.1 kuralları kullanılmaktadır.

HTTP üzerinden yapılan tüm istek ve veri transferleri okunabilir metin olarak yapılmaktadır. Yani, gelip giden veri rahatlıkla bir insan tarafından okunup anlaşılabilir yada oluşturulabilir. Bu, güvenlik açısından açık olarak görünse de, HTTP'nin yaygınlaşmasının en önemli sebeplerinden biridir. Çünkü yeni bir HTTP sunucu veya istemcisi programlamak oldukça kolaydır. Ayrıca **SSL** (Secure Sockets Layer, Güvenli Soket Katmanı) ile kolaylıkla güvenliği sağlanabilmektedir (Bkz https).

HTTP istek ve cevaplarında, öncelikle işlem veya cevap tanımı, sonra başlıklar ve en son varsa taşınacak veri bulunmaktadır. İşlem veya cevap tanımı ilk satırda bulunmaktadır. Bu satırın hemen ardından başlıklar her satırda bir başlık olacak şekilde gelmektedir. Veriler ise başlıkların ardından bir boş satır bırakıldıktan sonra gönderilmektedir. Başlığın adıyla değeri arasında iki nokta üst üste (:) bulunmaktadır. Alt satır için, standart olarak kullandığımız CRLF (\r\n) kullanılmaktadır. Diğer sayfada örnek bir istek-cevap verilmiştir. Burada gerçekleşen işlem, bir sitenin ana sayfasının (/) istenmesidir.

İstek:

```
GET / HTTP/1.1
Host: ornek.com.tr
```

Cevap:

```
HTTP/1.1 200 OK
Date: Sat, 13 Apr 2013 08:13:00 GMT
Content-Type: text/html
Content-Length: 76

<html><head><title>Merhaba</title></head><body>Merhaba
Dünya!</body></html>
```

İstekler

Bir isteğin yapılabilmesi için istek tanımının varsa başlıkların ve/veya taşınacak verinin verilmesi gerekmektedir. İstek tanımında önce isteğin tipi, sonra isteğin yapılacağı kaynak, daha sonra da kullanılan protokol adı bulunmalıdır. Yukarıdaki örnekte, **GET** istediğin tipi, **/** istenilen kaynak ve **HTTP/1.1** kullanılan protokoldür. Bu istek bir başlık içermektedir. **Host** başlığın adı, **ornek.com.tr** ise başlığın verisidir. Başlıkların ardında bir adet boş satır bulunması gerekmektedir.

HTTP'de bir çok istek cinsi bulunmaktadır. Ancak bu projede bunlardan yalnızca bir tanesini, **GET** isteğini kullanacağız. Ancak, standartlara uyan bir web sunucusunun GET ve HEAD isteklerine cevap vermesi beklenir. Bunun yanında web sunucularının tamamına yakını POST isteğini de gerçekleştirebilir.

- **GET**: bir kaynağın gönderilmesini istemektedir. En sık kullanılan istek cinsidir.
- **POST**: bir kaynağa veri gönderilmesini istemektedir.
- **HEAD**: yalnızca belirtilen kaynağa ait başlıkların gönderilmesini ister
- **OPTIONS**: verilen kaynağa hangi isteklerin yapılabileceğini listeler
- **PUT**: gönderilen verinin belirtilmiş olan kaynağa yazılmasını ister
- **DELETE**: belirtilmiş olan kaynağın silinmesini ister.

Yapılan bir istekte bulunması zorunlu olan tek bir başlık, Host, mevcuttur. **Host** başlığı istenilen sitenin adresini barındırır. Böylece aynı sunucu üzerinde birden fazla web sitesi barındırılabilir. Bunun dışında bir çok standart ve standart olmayan başlıklar mevcuttur. Ancak bu proje çerçevesinde yazılacak olan web sunucularında bu başlıklar ve görevleriyle ilgilenmeyeceğiz.

Cevaplar

Bir web sunucusu, yapılan bir isteğe cevap vermelidir. Bu cevap isteği gerçekleştirebilir, isteği başka bir kaynağa yönlendirebilir veya isteğin hatalı olduğunu bildirebilir. Verilen cevabın ilk satırı cevap tipini belirtmektedir. Yukarıdaki örnekte **HTTP/1.1** protokolü, **200** cevap kodunu, **OK** verilen cevabı belirtmektedir. Bu cevabın üç adet başlığı bulunmaktadır. Bu üç başlık da başarılı bir GET isteğine yanıt gönderirken gereklidir. **Date** başlığı şu anki sistem tarih ve saatini, **Content-Type** gönderilen içeriğin tipini, **Content-Length** ise gönderilen içeriğin uzunluğunu bildirmektedir. Başlıklardan sonra bir boş satırın ardından gönderilecek içerik gelmektedir. İçeriklerin verilerini sistemde bulunan dosyalardan okuyacağız. Projenin ikinci kısmında, kullanmanız için örnek dosyalar sağlanacaktır. İçerik tipi olarak bu projede .html dosyaları için **text/html**, .txt dosyaları

için **text/plain** ve diğerleri için **application/x-unknown** kullanacağız.

Veri Birleştirme

TCP üzerinden gönderilen verilerin tek parça halinde elimize ulaşacağı kesin değildir. Bu sebepten dolayı, parça parça gelen verilerin birleştirilmesi gerekmektedir. Ancak bu parçaları birleştirilebilmesi için, parça sonunu nasıl tespit edilebileceğinin bilinmesi gerekmektedir. Yapılacak olan projede yalnızca GET istediği kullanılacağından ve GET isteği veri taşımadığından dolayı, başlıklardan hemen sonra verinin bittiği bilinmektedir. Başlıkların bitişi ise ek bir boş satırla belirtilmelidir. Yani, verinin en sonunda iki kere alt satıra (\r\n\r\n) geçiş mevcuttur. Verinin sonunun tespitinin yapılabilindiğine göre, gelen verinin, parça parça bir döngü içerisinde hedef diziye aktarılması sağlanabilir. Dizinin en sonuna '\0' eklenmesi, bu dizinin dizge olarak kullanılabilmesi için gereklidir.

Not: Benzer şekilde iki paket bir biriyle birleşerek de ulaşabilir, ancak bu proje kapsamında böyle bir sorunla karşılaşmayacağız.

Metin İşleme

Projenin birinci kısmı kapsamında gönderilen isteği işlemeniz gerekmektedir. Aşağıda, verilen bir yazıyı boşluklardan bölerek bir diziye atan program parçası mevcuttur. Bu program çalışırken, verilen yazıdaki tüm karakterleri tek tek işlemektedir. Bu işlemi yaparken bulduğu karakterleri tek tek parçalara eklemektedir. Ayırma karakteri olan boşluğu bulduğunda ise sıradaki parçaya geçmektedir.

```
char yazi[1024]; //en fazla yazı uzunluğu 1023 karakter
char parcalar[100][256]; //en fazla 100 kelime,
                         //her kelime 256 karakter

... //burada yazının elde edildiğini varsayıyoruz

int i; //yazıdaki karakter indisi
int parcasayisi=0; //işlenilen parçanın numarası ve parça sayısı
int parcauzunlugu=0; //işlenilen parçanın karakter indisi

//işimizi kolaylaştırmak için tüm parçaları 0'lıyoruz
memset(parcalar, 0, 100*256);
```

```
//yazı metninde bulunan tüm karakterler için
for(i=0;i<strlen(yazi);i++) {
     if(yazi[i]==' ') { //eğer boşluksa
          parcasayisi++; //diger parcaya gec
          parcauzunlugu=0; //ve diger parçanın başından başla
     }
     else { //degilse
          //bulunan karakteri parcaya at
          parcalar[parcasayisi][parcauzunlugu]=yazi[i];
          //şu anki parçanın sıradaki karakterine geç
          parcauzunlugu++;
     }
}
```

Siz de gelen isteği satırlara ayırırken yukarıdaki programa benzer şekilde ayırabilirsiniz.

Dosya ve dizin tespiti

Bu proje içerisinde gönderilen bir adın dosya veya dizin adı olduğunun kontrolünün yapılması gerekecek. POSIX tabanlı işletim sistemlerinde bulunan **stat** isimli sistem çağrısı sayesinde dosyalar/dizinler hakkında bilgi edinebiliriz. Eğer aranılan dosya/dizin mevcut değilse stat fonksiyonu hata kodu olarak -1 döndürecektir. Stat sistem çağrısı, ilk parametre olarak dosya adını, ikinci parametre olarak ise stat tipinde bir veri yapısının işaretçisini istemektedir. Eğer dosya/dizin mevcutsa, ikinci parametreyle gönderilen veri yapısının içeriğine, bu dosya/dizin hakkında bilgi yüklemektedir. Stat sistemini kullanabilmek için programda **sys/stat.h** kütüphanesi bulunmalıdır. Aşağıda verilen bir ismin, dosya/dizin veya mevcut olmadığını yazdıran program parçası bulunmaktadır.

```
#include <sys/stat.h>
...
struct stat bilgiler;
int sonuc=stat(dosya, &bilgiler);
if(sonuc==-1) {
     printf("Aranılan isim yok\n");
}
else if(bilgiler.st_mode & S_IFDIR) {
     printf("Verilen isim bir dizin\n");
}
else {
     printf("Verilen isim bir dosya\n");
}
```

1. Kısım

Projenin birinci kısmında bir sunucu oluşturarak gelen bağlantıları kabul etmemiz, bu bağlantılardan gelen verileri işleyerek kolaylıkla kullanabileceğimiz hale çevirmemiz gerekmektedir. Birinci kısımın sonunda her istek için aşağıdaki değerlere sahip olmalıyız.

1. **İstenilen kaynağın adı**

```
char kaynak[1024]={};
```

2. **Başlıklar ve başlıklara ait değerler**

```
int  basliksayisi=0;
char basliklar[100][1024]={}, degerler[100][1024]={};
```

Sistemin doğruluğunu test etmek için bu değerleri ekrana yazdırınız(**altta verilen örnek resme bakınız**). Daha sonra kendi sunucunuza bir web tarayıcısı tarafından bağlanarak (örneğin: http://localhost:7000/) gelen isteği kontrol ediniz. Kullandığınız tarayıcıya göre gelen bilgiler değişse de, istenilen kaynak "/" olmalı ve en azından sunucunun bilgisinin **Host** başlığıyla iletilmesi gerekmektedir. Bu bilgileri doğru topladığınızdan emin olmanız çok önemli, aksi takdirde bir sonraki kısımda daha büyük problemlere yol açabilir.

Not: Bu programı C veya C++ ile yazabilirsiniz. Ancak C++ C'ye göre daha katı olduğundan, deney 4'teki programı kullanmanız halinde hatalar alabilirsiniz.

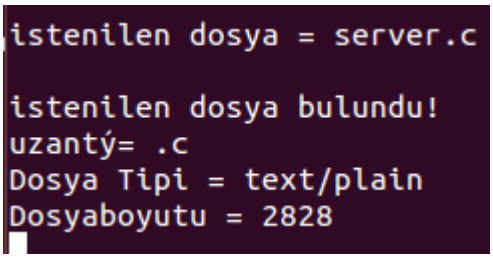
2. Kısım

Bu kısımda gelen isteğe yanıt hazırlanarak geri iletim yapılacaktır. Bunun için gelen istekteki kaynak, dosya sisteminden bulunarak okunmalıdır. Ancak gönderilen kaynak isteği sunucunun adresine göre gönderilmektedir. Yani bize gelen /index.html isteği dosya sistemi üzerinde önceden ayarlanmış bir yerden bakılmalıdır. Aksi takdirde /index.html dosya sisteminin en başında bulunan index.html'i belirtir ve biz normal kullanıcılar olarak dosya sisteminin bu kısmında değişiklik yapamayız. Kolay bir yöntem olarak dosyanın başındaki "/" işaretini kaldırıp dosyayı şu an bulunduğumuz dizinden aratabiliriz. Sistemdeki dosyayı herhangi bir fonksiyonla açıp okuyabilirsiniz (open, fopen, **ifstream**). Ancak eğer **fopen** yada **ifstream** kullanılıyorsa, dosyanın

```
bind: tamam!
listen: tamam!
accept: tamam!
GET / HTTP/1.1
Host: localhost:1508
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:28.0) Gecko/20100101 Firefox
/28.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

ikili dosya olduğunun belirtilmesi gerekmektedir (örn: **fopen** için yalnızca "r" yerine "rb").

1. kısımda başlıklardan elde ettiğimiz '**istenen dosya**'yı alıp stat() sistem çagrısına gönderip, dosyanın olup olmadığına bakabiliriz eğer dosya varsa fopen veya benzeri bir fonksiyonla açılıp



bunu okumamız gerekir (**Alttaki örnek resime bakınız**).

Gönderilecek olan dosya bilgisinin dışında cevap ve cevap başlıkları da gönderilmelidir. Daha önce de belirtildiği gibi tarih, dosya içeriği ve dosya boyutu başlık olarak iletilmelidir. İstenilen tarih düzeni için aşağıda verilen kod kullanılabilir.

```
time_t tt=time(NULL);
char tarih[100];
strftime(tarih, 100, "%a, %d %b %Y %H:%M:%S GMT", gmtime(&tt));
```

Şimdilik dosya içerik tipi olarak "text/plain" göndermeniz yeterlidir. Dosya boyutu olarak da dosyadan okunan veri miktarını kullanabilirsiniz. Sisteminizin testi için web tarayıcısından programınızın bulunduğu dizinde varolan bir dosyayı istebilirsiniz.

Örn:

```
http://localhost:7000/websunucu.c
```

Programınızın bu isteğe vermesi gereken yanıt aşağıdaki gibidir (dosya boyutunun 11240 byte olduğunu varsayıyoruz):

```
HTTP 1.1 200 OK
Date: Sat, 24 Apr 2013 09:23:00 GMT
Content-Type: text/plain
Content-Length: 11240

#include <stdio.h>
...
```

Tarayıcının ekranında kaynak kodlarınızın eksiksiz ve düzgün bir şekilde görünmesi gerekmektedir.

3. Kısım

Bazı durumlarda bir dizin adı da istek olarak gönderilebilir. Böyle durumlara da yanıt verebilmemiz gerekir. Genel olarak bir dizin istendiğinde, o dizinde bulunan index.html dosyası yanıt olarak gönderilir. Yapacağımız projede siz de bu yöntemi kullanmalısınız. Gönderilen isteğin bir dosya mı yoksa dizin mi olduğunu **stat** sistemini kullanarak öğrenebilirsiniz.

Eğer web sunucusundan istenilen sayfa mevcut değilse, web sunucusu bu durumu hata mesajı göndererek bildirmelidir. Bu durumda gönderilmesi gereken yanıt **`404 Not Found`** olmalıdır. Bu yanıt, istek yanıtı gibi tarih, içerik tipi ve içerik uzunluğu başlıklarıyla, **404** hatasını anlatan bir içeriğe sahip olmalıdır. Aşağıda örnek bir 404 hata bilgisi verilmiştir.

```
HTTP/1.1 404 Not Found
Date: Sat, 24 Apr 2013 09:23:00 GMT
Content-Type: text/plain
Content-Length: 3
...
```

Gönderdiğiniz bulunamadı mesajının kullanıcı tarafından anlaşılabilir olması gerekmektedir.

Programınızın her işlemden sonra hata/başarılı mesajı vermesini sağlayınız. Deney 4'te bu konu hakkında örnek mevcuttur. Ayrıca her bağlanan istemcinin istediği sayfayı, verilen yanıtın tipini (200 yada 404) ve belirtilmiş ise istemcinin bilgilerini (User-Agent başlığı) yazdırınız. Örneğin:

```
Web sunucusu başlıyor...
Soket istemi: başarılı
Bağlama: başarılı
Dinleme: başarılı

Yeni bağlantı
Kabul: başarılı
İstenen sayfa: /
Yanıt: 200 OK
İstemci: Opera/9.80 (X11; Linux i686) Presto/2.12.388
Version/12.14
Bağlantı kesildi
```

```
404 Hatasına örnek;
```

```
bind: tamam!
listen: tamam!
accept: tamam!
GET /yokboylebirserver.c HTTP/1.1
Host: localhost:1508
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:28.0) Gecko/20100101 Firefox
/28.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive


istenilen dosya = yokboylebirserver.c
Istenilen Dosya Bulunamadi!
```

Aşağıda doğru bir şekilde yazılan programın vermesi gereken örnek konsol çıktısı verilmiştir (tamamen aynı çıktıya sahip olmanız zorunlu değildir, fakat sizden istenilenleri doğru yaptığınız taktirde benzer bir sonuca ulaşacaksınız).

```
bind: tamam!
listen: tamam!
accept: tamam!
GET /server.c HTTP/1.1
Host: localhost:1508
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:28.0) Gecko/20100101 Firefox
/28.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive


istenilen dosya = server.c

istenilen dosya bulundu!
uzantý= .c
Dosya Tipi = text/plain
Dosyaboyutu = 2828
```