BLGM 344 – DENEY 3 *
AĞ PROGRAMLAMAYA GİRİŞ

Amaçlar

1. TCP protokolünün öğrenilmesi
2. Ağ programlamada kullanacağımız sistem komutlarının öğrenilmesi
3. Ağ programlamanın kavranması

TCP ile veri alışverişi

TCP bağlantı tabanlı bir protokoldür, yani, istemci sunucuya bağlanır ve bu bağlantı sağlandıktan sonra veri transferi başlayabilir. TCP bağlantılarında istemci ve sunucunun gerçekleştireceği adımlar farklıdır. Ağ tabanlı veri transferi yapılabilmesi için öncelikle bir soket açılması (**socket()**), sonra da bu sokete bir kapı numarası (port number) ve host adresinin (IP address) atanması (**bind()**) gerekmektedir.

İstemcinin yapacağı işlemler sunucuya göre daha basittir. Önceden bahsettiğimiz gibi bir kapı numarasına bağlı bir soket oluşturulması gerekmektedir. Ancak buradaki kapı numarası önemsiz olduğu için işletim sistemine bırakılabilir. Böylece işletim sistemi tarafından bu sokete rastgele bir kapı numarası atanır. Daha sonra bu soket kullanılarak, sunucuya bağlantı sağlanabilir (**connect()**). Sunucuya bağlanabilmek için sunucunun IP adresini ve kapı numarasını bilmemiz gerekmektedir. Eğer belirtilen adreste istenen kapı numarasını kullanan bir sunucu yoksa, bağlatı çağrısı hata verecektir.

Sunucu soketi oluşturduktan sonra bu soketi istemci tarafından da bilinen bir kapı numarasına ve IP adresine bağlar. Sunucu bağlantıları kabul edeceğini dinle çağrısıyla (**listen()**) belirtir. Bu işlemin ardından istemci sunucuya bağlanabilir, ancak, veri transferinin başlayabilmesi için sunucunun isteği kabul etmesi gerekmektedir (**accept()**). Kabul etme işlemi başarılı olursa *yeni* bir soket yaratılarak sunucuyla istemci arasında veri transferinin başlayabilmesini sağlanır. Bağlantı bekleyen soket hala açık ve bağlantı bekleme durumundadır, ve istenirse bir başka istemcinin bağlantı isteği de kabul edilebilir. Aynı anda birden fazla istemciye yanıt vermek için, sunucu birden fazla işlem (**fork()**) veya iş parçası (**pthread**) kullanabilir.

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

Sunucu veya istemci, bağlantı sağlandıktan sonra veri gönderebilir (**send()**) veya alabilir (**recv()**). Veri transferini veya dinleme işlemini sonlandırmak için kapat (**close()**) çağrısı kullanılabilir.

socket()

socket() sistem çağrısında iki parametre ile istenilen soket tipi belirtilir. Internet Protocol için AF_INET, TCP bağlatısı için SOCK_STREAM (akış soketi) kullanılması gerekmektedir. Eğer döndürülen değer negatifse, bir hata çıktığını belirtmektedir. Eğer herhangi bir sorun yoksa bu sistem çağrısı soket tanımlayıcısı döndürür. Örnek:

```
int soket = socket(AF_INET, SOCK_STREAM, 0);
```

bind()

bind() sistem çağrısı verilen soket tanımlayıcısını verilen yerel adres ve kapı numarasına bağlar. Parametre olarak önce soket tanımlayıcısı, sonra yerel adres tanımlayıcısı (domain, port ve IP adres içerir) ve soket adres tanımlayısıcının boyutunu kabul eder. Soket adres tanımlayıcı olarak **sockaddr_in** (soket adresi, internet) kullanacağız. Adres veya kapı numarası bizim için önemli değilse **INADDR_ANY** değerini kullanabiliriz. Çağrı esnasında hata çıkarsa sonuç olarak -1 döndürülür. En sık karşılaşılan hata belirtilen kapının numarasının kullanılıyor olmasıdır. Eğer **close()** çağrısı kullanılmamışsa, bir program kapandıktan bir süre sonra o programın kullandığı kapı numarası serbest duruma düşer.

```
struct sockaddr_in yerel = {AF_INET, INADDR_ANY, INADDR_ANY};
bind(soket, &yerel, sizeof(sockaddr_in));
```

connect()

connect() sistem çağrısı belirtilen adrese bağlantı sağlar. Yalnızca bağlantı tabanlı iletişim sistemlerinde kullanılır. Sunucunun bilgileri **bind()** sistem çağrısındaki gibi **sockaddr_in** veri yapısı kullanılarak sağlanır, ancak, sunucunun bilgileri tam olarak verilmelidir. **INADDR_ANY** **connect()** ile kullanılamaz. Çağrı esnasında hata çıkarsa sonuç olarak -1 döndürülür. Genel olarak üç hata karşımıza çıkar: hedef bilgisayarın adı/IP'si yanlış olabilir, belirtilen kapı numarası yanlış ya da bu kapıyı dinleyen bir program olmayabilir, son olarak da bilgisayarın güvenlik duvarı bağlantıya engel olabilir.

```
struct sockaddr_in sunucu =
     {AF_INET, htons(80), inet_addr("194.27.78.195")};
connect(soket, &sunucu, sizeof(sockaddr_in));
```

listen()

listen() sistem çağırısı verilen soketin bağlantı kabul ettiğini belirtir. İlk parametre soketin kendisi, ikinci ise aynı anda kaç bağlantı isteğinin beklemede olacağını belirtir. Genelde bağlantı isteği geldiğinde kısa süre içerisinde kabul edileceğinden, küçük bir değerin kullanılması herhangi bir sorun yaratmayacaktır. Çağrı esnasında hata çıkarsa sonuç olarak -1 döndürülür.

```
listen(soket, 5);
```

accept()

accept() çağrısı gelen bir bağlantı isteğini kabul etmek için kullanılır. Bu çağrı bir bağlantı isteği yoksa bağlantı gelinceye kadar bekler. Bu çağrının kullanılabilmesi için **listen()** sistem çağrısının bu soket üzerinde çağırılmış olması gerekmektedir. Çağrı esnasında hata çıkarsa sonuç olarak -1 döndürülür. Çağrı başarılı olursa, yeni oluşturduğu soketi geri döndürecektir. İstemciyle bu soketi kullanarak iletişim sağlamanız gerekmektedir. Orijinal soket hala bağlantı beklemeye devam edecektir. Ayrıca **accept()** çağrısı gelen bağlantı hakkındaki bilgileri de bize sockaddr_in yapısıyla iletir.

```
int uzunluk;
struct sockaddr_in istemcibilgisi;
int istemci = accept(soket,
     (struct sockaddr *)&istemcibilgisi, &uzunluk);
```

send()

TCP bağlantısı sağlamış olan bir soket kullanılarak karşıdaki bilgisayara veri gönderilebilir. Bu işlem için **send()** çağrısı kullanılmalıdır. **send()** çağrısı, kullanılacak olan soketi, gönderilecek veriyi, verinin uzunluğunu ve gönderim ayarlarını parametre olarak alır. Çağrı esnasında hata çıkarsa sonuç olarak -1 döndürülür. Eğer gönderim başarılıysa gönderilen veri miktarını geri döndürür. Gönderilen veri miktarı göndermek istenilen veriden az olabilir. Bu durumda kalan veri bir başka çağrıyla tekrar gönderilmelidir. Örnek:

```
char data[]="merhaba";
int gonderilen = send(soket, data, strlen(data)+1, 0);
```

recv()

recv() sistem çağrısı TCP bağlantısı üzerinden veri okumak için kullanılır. Kullanılacak soketi, doldurulacak veri alanını, bu alanın büyüklüğünü ve ayarları parametre olarak alır. Eğer bekleyen veri yoksa bu çağrı veri gelene kadar bekler. Birden fazla **send()** çağrısıyla gönderilen veriler **recv()** ile okunurken birleşik de okunabilir. Bu sebepten, okunan verinin birden fazla paket içerip içermediğinin kontrol edilmesi gerekebilir. Çağrı esnasında hata çıkarsa sonuç olarak -1 döndürülür. Eğer çağrı başarılıysa okunan verinin boyutunu verir.

Not: gönderilen veride dizgi sonu yoksa, alınan veriye otomatik olarak eklenmez. Bu yüzden gerektiği durumlarda okunan verinin sonuna '\0' karakterini yerleştirmeniz gerekebilir.

```
char buf[1024];
int alinan = recv(soket, buf, 1024, 0);
```

close()

Açılan bir soket bağlantısı **close()** kullanarak kapatılabilir. Parametre olarak soket tanımlayıcısını alır.

Deney

Bu deneyde basit bir istemci yazmanız istenmektedir. Bu istemci 194.27.78.195 IP adresinde bulunan 1500 numaralı kapı üzerinde çalışan sunucuya bağlanmalı, daha sonra sunucunun gönderdiği veriyi ekrana yazmalıdır. Bunun ardından kullanıcıdan bir yazı okuyarak bunu sunucuya geri göndermelidir. Daha sonra da sunucudan gelen veriyi göstererek soketi kapatıp çıkmalıdır. Aşağıda gerçekleştirilmesi gereken adımlar listelenmiştir:

1. Soket oluştur (**socket()**) ve yerel bağlantısını yap (**bind()**)
2. Sunucuya bağlan (**connect()**)
3. “Sunucuya bağlandı.”, mesajını yazdır (printf()
4. Sunucudan veri oku (**recv()**)
5. Okunan veriyi ekrana yaz (printf())
6. Kullanıcıdan veri oku (scanf())
7. Kullanıcıdan okunan veriyi sunucuya gönder (**send()**)
8. Sunucudan gelen veriyi oku (**recv()**)
9. Bağlantıyı kapat (**close()**) ve çık (return)

Yazdığınız bu istemciyi birkaç kere deneyiniz.

Sorular

1. Bir soket bağlantısı hangi verilerle ayırt edilir? İpucu: 5 adet veri kullanılmaktadır.
2. Hangi sistem çağrıları hem sunucu hem de istemci tarafından kullanılabilir?
3. connect() sistem çağrısı hangi tip soketlerde kullanılabilir?
4. Bir sunucu aynı anda birden fazla istemciye yanıt verebilir mi?
5. İstemcinin kullandığı **connect()** çağrısına karşılık sunucunun çağırmış olduğu **accept()** çağrısı sayesinde bağlantı sağlanır. Bu iki sistem çağrısı arasında, birinin sunucu birinin ise istemci tarafında çalışması haricindeki, en önemli fark nedir?
6. sockaddr_in veri yapısında **in** ne anlama gelmektedir?
7. INADDR_ANY hangi durumda ne anlama gelmektedir?