

# Genel: SÃ¼reÃŒlerin HaberleÅmesi: Soket Programlama

SÃ¼reÃŒlerin HaberleÅmesi: Soket Programlama  
BarÅÅ ÅÅMÅEK - 1998-11-10

```
/*  
 * Bu belgenin telif haklarÅ BarÅÅ ÅÅÅek'e aittir.  
 * KÅk: http://www.acikkod.org  
 * SÃ¼rÃ¼m No: 6  
 * Ålk baskÅ: 1998-11-10  
 * Son gÃ¼ncelleme: 2004-08-01  
 * Bu dÅkÅman AÅÅkkod.ORG Belge YazÅm ve DaÅÅtÅm LisansÅ ile  
 * daÅÅtÅlmaktadÅr.  
 */
```

ÅÅindekiler

[1. Soket KavramlarÅ](#)

[2. TCP/IP ProtokolÅ](#)

Å Å Å [2.1. AÅ EriÅim KatmanÅ](#)

Å Å Å [2.2. AÅ KatmanÅ](#)

Å Å Å [2.3. TaÅÅma KatmanÅ](#)

Å Å Å [2.4. Uygulama KatmanÅ](#)

[3. Uygulamadaki Temel Prensipler](#)

Å Å Å [3.1. Soket TÅrleri](#)

Å Å Å [3.2. Veri DÅnÅÅmleri](#)

Å Å Å [3.3. Sistem ÅaÅrÅlarÅ](#)

[4. Sunucu Soket ProgramÅ](#)

Å Å Å [4.1. Sistem ÅaÅrÅlarÅnÅn KullanÅmÅ](#)

Å Å Å [4.2. sunucu.c](#)

[5. Åstemci Soket ProgramÅ](#)

[6. GeliÅmiÅ G/Å](#)

Ä Ä Ä [6.1.](#) Bloksuz G/Ä†

Ä Ä Ä [6.2.](#) select()

[7.](#) SÄ¼k Sorulan Sorular

[8.](#) Belge TarihÄŒesi

## 1. Kavramlar

Soketler, aynÄ± veya farklı hostlar Ä¼zerindeki sÄ¼reÄŒlerin haberleÄŒmesini saÄŒlayan bir haberleÄŒme (interprocessing) yÄŒntemidir. Soket, soyut bir tanÄ±mla haberleÄŒme uÄŒ noktalarÄ±dÄ±r. POSIX'in saÄŒladÄ±ÄŒÄ± programlama API'si sayesinde programcÄ± soketlere yazarken veya okurken yine write,read gibi sistem ÄŒaÄŒrÄ±larÄ±nÄ± kullanabilir.

*Ästemci (Client):* Hizmet isteyen soket programlara denir.

*Sunucu (Server):* Hizmet veren soket programdÄ±r.

*Port:* Bir bilgisayarda birden ÄŒok soket bulunabilir. ÄrneÄŒin hem telnet soketi, hem de ftp soketi aÄŒÄ±k olabilir. Soketleri birbirinden ayÄ±rt etmek ve istemciyi sunucudaki uygun program ile buluÄŒturmak iÄŒin her soket programÄ±n PORT denilen bir numarasÄ± vardÄ±r. ÄrneÄŒin FTP protokolÄ±nÄ±n port numarasÄ± 21, Telnet sunucunun port numarasÄ± ise 23'tÄ¼r. Standart servislerin port numaralarÄ± /etc/services dosyasÄ±nda tanÄ±mlÄ±dÄ±r. 1-1024 arasÄ±ndaki portlar yalnÄ±z root tarafÄ±ndan kullanÄ±labilir, normal kullanÄ±cÄ±lar bu portlarÄ± bind edemezler.

*IP NumarasÄ±:* TCP/IP protokolÄ¼ hostlarÄ±, sadece kendisine ait olan bir IP numarasÄ± ile tanÄ±mlar. Bilgisayarlar birbiri ile IP

# Genel: SÄ¼reÄŒlerin HaberleÄŒmesi: Soket Programlama

numarasÄ±nÄ± kullanarak haberleÄŒirler. Bir istemci soket program, Ä¶nce IP numarasÄ±nÄ± kullanarak sunucunun bulunduÄŒu bilgisayara, sonra PORT numarasÄ±nÄ± kullanarak hizmet istediÄŒi sunucu program ile temasa geÄŒer. IPV4 standardÄ±na gÄ¶re IP'ler 192.168.1.10 gibi [0-255].[0-255].[0-255].[0-255] formatÄ±na sahiptir. IPV6 standartÄ± 128bit adres kullanÄ±r.

*Sarmalama(Encapsulation):* Her katman kendisine gelen pakete bir baÄŒÄ± ekler ve bir sonraki protokole aktarÄ±r. Buna encapsulation (sarmalama) denir. KarÄŒÄ± tarafta paket aÄŒÄ±lÄ±rken yine her katman kendisi ile ilgili baÄŒÄ± aÄŒar. Ä±rneÄŒin fiziksel katman (mesela ethernet aygÄ±tÄ±) gÄ¶nderen sistemin fiziksel katmanÄ±nÄ± eklediÄŒi baÄŒÄ± aÄŒar. Zaten diÄŒer baÄŒÄ±klarÄ± yorumlayamaz. ÄŒaÄŒda sarmalanmÄ±ÄŒ bir paket gÄ¶rÄ¼lmektedir.

```
+-----+
|Ä± Ä± Ä± Ä± Ä± EthernetÄ± Ä± Ä± Ä± Ä± Ä±
|
|+-----+|
||Ä± Ä± Ä± Ä± Ä± Ä± IPÄ± Ä± Ä± Ä± Ä± Ä± Ä±
||
||+-----+||
|||Ä± Ä± Ä± Ä± Ä± Ä± UDPÄ± Ä± Ä± Ä± Ä± Ä±
||| | | | | |
|||+-----+|||
||||Ä± Ä± Ä± Ä± Ä± DNSÄ± Ä± Ä± Ä± Ä± ||||
||||+-----+||||
||||Ä± Ä± Ä± Ä± Ä± VeriÄ± Ä± Ä± ||||
||||+-----+||||
```

## 2. TCP/IP ProtokolÄ¼

TCP/IP, "Transaction Control Protocol and Internet Protocol" iÄŒin bir kÄ±saltmadÄ±r. TCP/IP, DARPA(Defense Advanced Research Projects Agency) projesi olarak 1970'lerde Amerika'daki bazÄ± devlet kurumlarÄ±nÄ± birbirine baÄŒlamak amacÄ± ile geliŒtirildi. Daha sonra BSD(Berkeley Software Distribution) UNIX'e eklenerek uygulama alanÄ± buldu. TCP/IP'nin iÄŒerdiÄŒi protokollerin tanÄ±mlarÄ± RFC(Request For Comments)'lerde yapÄ±lmÄ±Œtir. RFC'lere ulaŒmak iÄŒin: <http://www.faqs.org/rfcs/>

TCP/IP 4 katmanlÄ± bir modeldir: 1. Uygulama KatmanÄ± 2. TaÄŒma KatmanÄ± 3. Internet KatmanÄ± 4. AÄŒ EriŒim KatmanÄ±. Her bir katman OSI modelindeki bir veya daha fazla katmanÄ±n iÄŒlevine sahiptir. 1982'de BSD UNIX ile TCP/IP uygulamalarÄ± geliŒtirildi.

### 2.1. AÄŒ EriŒim KatmanÄ±

Bu katman TCP/IP paketlerini fiziksel aÄŒa bÄ±rakmak ve aynÄ± zamanda fiziksel aÄŒdan gelen paketleri almakla gÄŒrevlidir. OSI modelindeki Fiziksel katman ve Veri-BaÄŒ katmanÄ±nÄ± karŒlaŒtÄ±r.

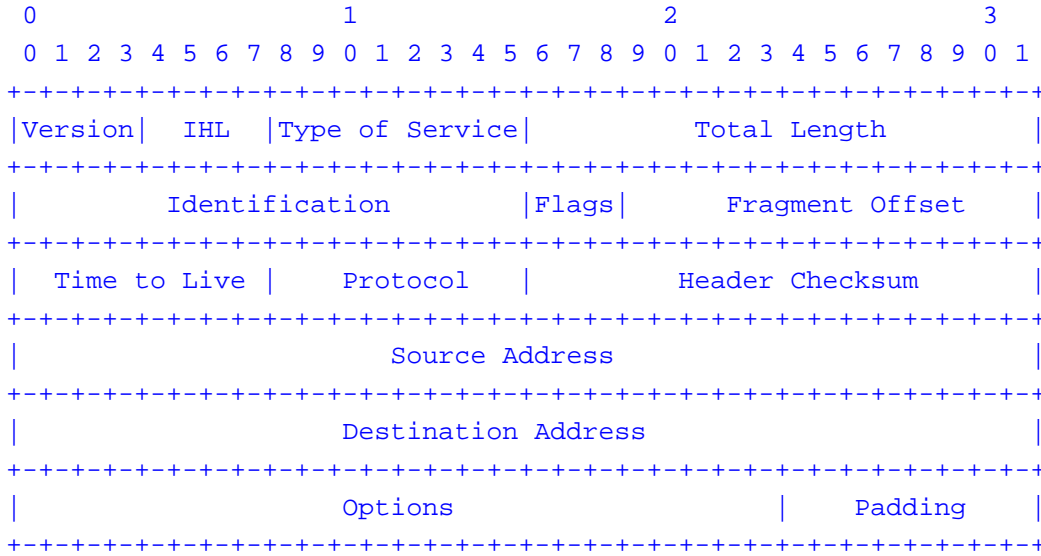
### 2.2. AÄŒ KatmanÄ±

Bu katman adresleme, paketleme ve yÄŒlendirme fonksiyonlarÄ±nÄ± yerine getirir. IP, ARP, ICMP ve IGMP protokolleri, bu katmana ait ÄŒekirdek protokollerdir.

*Internet Protocol (IP):* Adres bilgilerini ve paket yÄŒlendirme iÄŒin bazÄ± kontrol bilgilerini iÄŒerir. RFC 791'de tanÄ±mlanmÄ±Œ olup en ÄŒnemli internet protokolÄ¼dür. Ä°ki ÄŒnemli gÄŒrevi vardÄ±r: 1. AÄŒlar arasÄ±nda baÄŒlantÄ±sÄ±z datagram daÄŒtÄ±mÄ± yapmak, 2. Fregmantasyon ve veri katmanÄ±na yardÄ±mcÄ± olarak deÄŒiŒik

MTU(maximum-transmission unit) deÄŒerleri ile datagramlarÄ± yeniden oluŒturmak. IP paketinin baÄŒIÄ±k yapÄ±sÄ± aÄŒaÄŒdaki gibi tanÄ±mlanmÄ±Œtir:

# Genel: SÄ¼reÄŒlerin HaberleÄŒmesi: Soket Programlama



*Version:* KullanÄ±lan internet baÄŒliÄŒÄ±nÄ±n biÄŒimini, versiyonunu gÄŒsterir.

*IP Header Length (IHL):* Datagram baÄŒliÄŒÄ±nÄ±n 32 bit olarak boyutunu gÄŒsterir. DoÄŒru bir IP baÄŒliÄŒÄ± iÄŒin baÄŒliÄŒk boyutu en az 5 olmalÄ±.

*Type of Service:* Ä°stenilen hizmet kalitesi ile ilgili soyut parametreler sunar. Ä±rneÄŒin bazÄ± aÄŒlar, ÄŒnceliÄŒi destekler. TrafiÄŒin bir kÄ±smÄ±na ÄŒncelik verilebilir.

*Total Length:* BaÄŒliÄŒk ve veri bilgisi ile birlikte toplam datagram boyutunu gÄŒsterir. 16 bittir, buradaki deÄŒer byte olarak gÄŒsterir. Yani IP paketi en fazla 64K boyutunda olabilir.

*Identification:* GÄŒnderen tarafÄ±ndan yazÄ±lÄ±r. Datagram parÄŒsalarÄ±nÄ± biraraya getirmeye yardÄ±mcÄ± olur.

*Flags:* Paketin parÄŒalanabileceÄŒini veya parÄŒalanamayacaÄŒÄ±nÄ± gÄŒsterir.

*Fragment Offset:* Bu paketin datagram iÄŒerisinde nereye ait olduÄŒunu tanÄ±mlar.

*Time to Live:* SÄ¼rekli azalan tam sayÄ±dÄ±r. Paketin hangi noktada yok edileceÄŒini belirtir. Paketin sonsuza tek aÄŒda kalmasÄ±na engel olur.

*Protocol:* IP'nin iÄŒi bittikten sonra paketi hangi Ä¼st protokol alacaÄŒÄ±nÄ± gÄŒsterir.

# Genel: SÄ¼reÄŒlerin HaberleÄŒmesi: Soket Programlama

*Header Checksum:* IP baÄŒiÄŒÄ±nÄ±n bozulmadÄŒÄ±na emin olmak iÄŒin tutulan deÄŒer

*Source Address:* GÄŒnderen noktayÄ± gÄŒsterir.

*Destination Address:* AlÄ±cÄ± noktayÄ± gÄŒsterir.

*Options:* IP, gÄ¼venlik gibi deÄŒiÄŒik seÄŒenekleri destekler.

*Data:* Äest katmana verilecek veriyi tutar.

## 2.3. TaÄŒma KatmanÄ±

Bu katman transparan bir Äekilde verinin hosttan hosta taÄŒnmasÄ±nÄ± saÄŒlar. AkÄŒ kontrolÄ¼nÄ¼ ve hata dÄ¼zeltmeyi saÄŒlar. Veri transferinin bittiÄŒinden emin olur. TCP ve UDP

protkolleri bu katmana aittir. ÄÄ katmanÄ± baÄŒlantÄ± yÄŒnelimli (connection oriented) baÄŒlantÄ± saÄŒlamaz. TaÄŒma katmanÄ± bunu saÄŒlar. ÄÄ katmanÄ± ulaÄŒan paketlerin, gÄŒnderildiÄŒi sÄ±rada ulaÄŒtÄŒÄ±nÄ± da garanti etmez. TaÄŒma katmanÄ± her

paketi numaralandÄ±rarak bunu basitÄŒe ÄŒÄŒzer. Hata oluÄŒtuÄŒu durumda paketi yeniden ister. BÄŒylece oluÄŒabilecek hatalarÄ±n ÄŒnÄ¼ne kesilir.

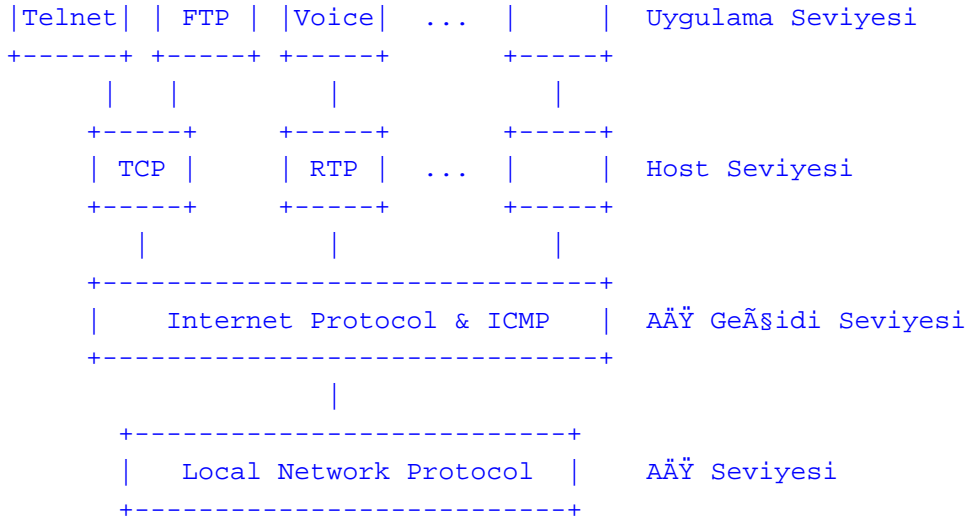
*Transmission Control Protocol (TCP):* TCP, IP ortamÄ±nda uÄŒtan uca gÄ¼venli haberleÄŒme sunan baÄŒlantÄ± yÄŒnelimli(connection oriented) bir protokoldÄ¼r. RFC 793'de tanÄ±mlanmÄŒÄ±r.

Uygulama katmanÄ±nÄ±n hemen altÄ±nda bulunur. AynÄ± zamanda sÄ¼reÄŒler arasÄ± haberleÄŒme(interprocess communication) prokolÄ¼dÄ¼r. Ä°ki sÄ¼reÄŒ arasÄ±nda sanal bir devre oluÄŒturur. Telnet, TCP kullanan popÄ¼ler uygulamalardan birisidir.

TCP, zarar gÄŒrmÄŒÄŒ, kaybolmuÄŒ veya sÄ±rasÄ± bozulmuÄŒ veriyi kurtarabilir. AktarÄ±lan her sekizlik iÄŒin sÄ±ra numarasÄ± tutar ve alÄ±cÄ± noktadan olumlu ACK(Acknowledge-aldÄŒÄ±nÄ± bildirmek) bekler. EÄŒer ACK, bir zamanaÄŒmÄ±(timeout) sÄ¼resi iÄŒerisinde gelmezse veri yeniden aktarÄ±lÄ±r. AlÄ±cÄ± taraf verileri sÄ±ralÄ± almamÄŒÄŒ veya geciken ACK'lerden dolayıÄ± birden fazla almÄŒÄŒ olabilir. TCP bunlarÄ± dÄ¼zeltir. Her bir segmente bir kontrol toplamÄ±(checksum) eklenerek alÄ±cÄ± tarafÄ±n aldÄŒÄŒ verinin doÄŒru olup olmadÄŒÄ±nÄ± denetlemesi saÄŒlanÄ±r. TCP'nin diÄŒer protokoller ile iliÄŒkisi:

+-----+ +-----+ +-----+ +-----+

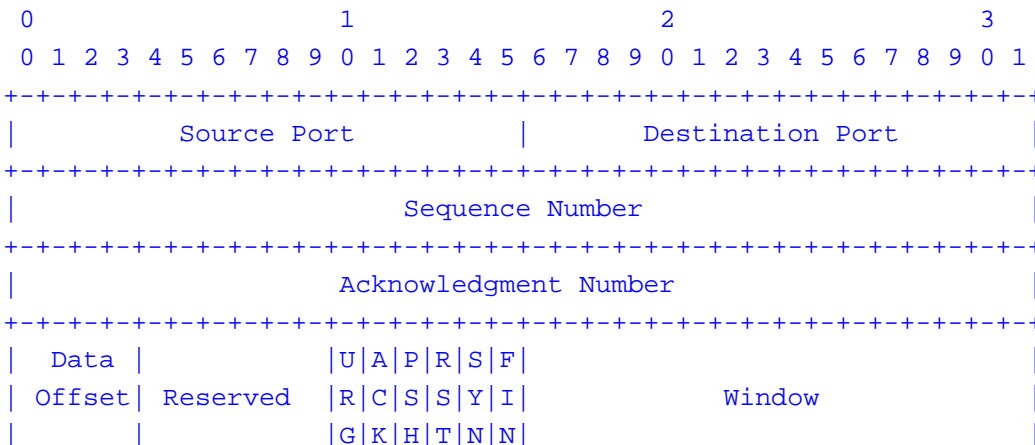
# Genel: SÄ¼reÄŒlerin HaberleÄŒmesi: Soket Programlama



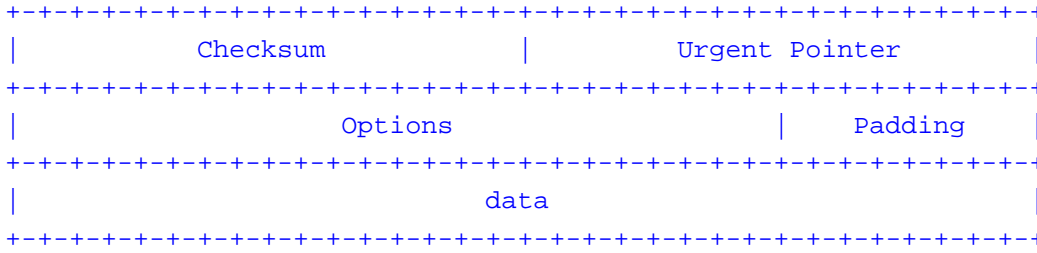
TCP, veri stream'lerini birbirinden ayÄŒrt etmek iÄŒin port tanÄŒmlayÄŒcÄŒ kullanÄŒr. Her TCP birbirinden baÄŒÄŒmsÄŒz port tanÄŒmlayÄŒcÄŒ sunar. Bu nedenle port tanÄŒmlayÄŒcÄŒlar tek olmayabilir. Bu nedenle soket oluÄŒturulurken internet adresi de kullanÄŒlÄŒr.

Bir baÄŒlantÄŒ tamamen uÄŒlardaki soketler arasÄŒnda oluÄŒur. Yerel bir soket pek ÄŒok dÄŒÄŒ soket ile baÄŒlantÄŒ yapabilir. BaÄŒlantÄŒ iki yÄŒnlÄŒ veri taÄŒÄŒmada (full duplex) kullanÄŒlabilir.

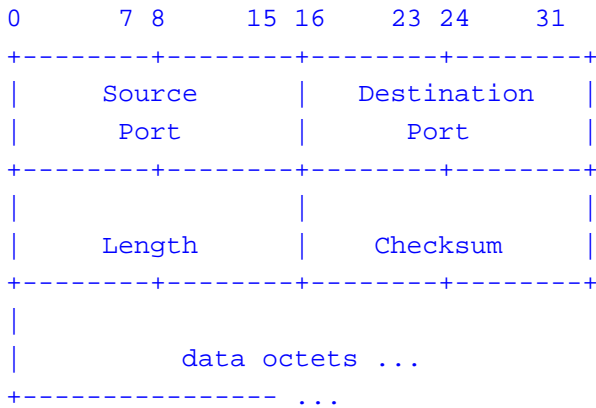
TCP segmentleri internet datagramlarÄŒ olarak gÄŒnderilir. ÄŒÄŒnkÄŒ altÄŒnda IP protokolÄŒ vardÄŒr. TCP segmentleri, IP tarafÄŒndan paketlenip gÄŒnderilir. TCP baÄŒÄŒÄŒ aÄŒÄŒdaki gibidir:



# Genel: SÄ¼reÄŒlerin HaberleÄŒmesi: Soket Programlama



*User Datagram Protocol (UDP)*: RFC 768'de tanÄ±mlanmÄŒtÄ±r. Bu protokol uygulamalar iŒin en az protokol yÄ¼kÄ¼ ile haberleŒme olanaÄŒÄ± saÄŒlar. Protokol iŒlemi gerŒekleŒtirmeye yÄ¶neliktir. DaÄŒÄ±tÄ±m ve gÄ¼venliÄŒi temin etmez. Checksum deÄŒeri tutar ancak paket bozulmuŒsa yeniden paketi daÄŒÄ±tmaz. BaÄŒiÄ±k yapÄ±sÄ± Ä¼u ÄŒekildedir:



DNS ve tftp bu protokolÄ¼ kullanan en popÄ¼ler uygulamalardÄ±r.

Bu iki tÄ¼rÄ¼n Ä¶zellikleri ve aralarÄ±ndaki temel farklarÄ± ÄŒÄ¶yle sÄ±ralayabiliriz:

1. Stream soketler verileri sÄ±ralÄ± gÄ¶nderir, datagram soketleri sÄ±ralÄ± gÄ¶ndermeyebilir. (TCP protokolu, paketleri sÄ±ralÄ± gÄ¶ndermeyi garanti eder. UDP garanti etmez. TCP paketlerin baÄŒiÄ±k bilgisinde sÄ±ra numarasÄ± vardÄ±r, UDP'de yoktur. TCP, her zaman sÄ±radaki paketi ister. Ä±rneÄŒin 4 numaralÄ± paket yerine 5 numaralÄ± paket eline ulaŒÄ±rsa karÄŒÄ± tarafa bunu bildirir ve 4'Ä¼ ister. 4'Ä¼ alÄ±nca da 5'ten Ä¶nceye koyar.)

## Genel: SÄ¼reÄŒlerin HaberleÄŒmesi: Soket Programlama

2. Stream soketler gÄ¼venlidir, Datagram soketler gÄ¼vensizdir. (TCP protokolu gÄ¼venliÄŒi garanti eder, UDP garanti etmez. Ä±Ä¼nkÄ¼ TCP acknowledgement ile denetim yapar. Yani bir paketi gÄ¼nderdiÄŒi zaman, karÄŒÄ± taraf paketi aldÄ±ÄŒÄ±nÄ± haber vermeden o paketi gÄ¼ndermiÄŒ saymaz kendini ve tekrar gÄ¼nderir. ayrÄ±ca paketin doÄŒru gidip gitmediÄŒini anlamak iŒin baÄŒlÄ±k bilgisinde checksum -kontrol bilgisi- tutar. UDP'de checksum tutar ancak checksum yanlÄ±ÄŒsa aynÄ± paketi tekrar istemez.)

3. Stream soketler, iÄŒlem bitene kadar kesintisiz bir baÄŒlantÄ± kurar. Datagram soketler ise baÄŒlantÄ± kurmaz. Sadece veri gÄ¼ndereceÄŒi zaman baÄŒlantÄ± kurar ve iÄŒi bitince baÄŒlantÄ±yÄ± koparÄ±r.

Bu iki arasÄ±ndaki farkÄ± anlatmak iŒin postacÄ± ve telefon benzetmesini vereceÄŒim. Mektup insanlar arasÄ±nda haberleÄŒmeyi saÄŒlayan bir yÄ¼ntemdir. PostacÄ± mektuplarÄ± posta kutusuna bÄ±rakÄ±p gider. KiÄŒi ise mektuplarÄ± mÄ¼sait olduÄŒu herhangi bir an (belki 1 saat sonra, belki 1 gÄ¼n, belki 1 hafta) alÄ±r ve okur. CevabÄ±nÄ± yine posta kutusuna atar ve postacÄ± bir sÄ¼re sonra mektuplarÄ± alÄ±p karÄŒÄ±ya taÄŒÄ±r. Telefon Ä¼rneÄŒinde ise, bir taraf diÄŒer tarafa telefon aŒsar. Aradaki baÄŒlantÄ± kurulduktan sonra insanlar baÄŒlantÄ± kopmadan karÄŒÄ±lÄ±klÄ± konuÄŒurlar. Posta Ä¼rneÄŒinde baÄŒlantÄ±nÄ±n sÄ¼rekliliÄŒi gibi bir ÄŒey sÄ¼z konusu deÄŒildi. Telefon gÄ¼rÄ¼ÄŒmesinde sÄ¼zlerin sÄ±ralÄ± gitmesi sÄ¼z konusu. Yani sÄ¼zler birbirine karÄ±ÄŒmaz. Ancak postada ise durum farklÄ±. Mektuplar karÄŒÄ± tarafta sÄ±ralÄ± okunmayabilir. Ä¼rneÄŒin posta kutusunda 5 mektup birikince mektuplarÄ±nÄ± okur. Telefonda ise karÄŒÄ±lÄ±klÄ± sÄ¼rekli konuÄŒulur ve sÄ¼ylenen karÄŒÄ±ya iletilir.

UDP'nin bu kadar tez avantajÄ±na raÄŒmen neden daha ÄŒok kullanÄ±ldÄ±ÄŒÄ± bu ÄŒemalardan aŒÄ±kÄŒa gÄ¼rÄ¼lmektedir. TCP bir veri karÄŒÄ±ya 6x32+Veri boyu kadar bir paket olarak gitmektedir. Yani her paket fazladan 192 bit baÄŒlÄ±k (header) bilgisi taÄŒÄ±maktadÄ±r. Oysa UDP paketleri 64 bitlik baÄŒlÄ±k (header) bilgisine sahiptir.

UDP kullanmanÄ±n en Ä¼nemli nedeni az protokol yÄ¼kÄ¼dÄ¼r. Video sunucu gibi realtime veri akÄ±ÄŒÄ± gerektiren bir uygulama iŒin TCP fazla yÄ¼k getirir ve gÄ¼rÄ¼ntÄ¼ realtime oynamaz. Bu nedenle multicast uygulamalarÄ±nda Datagram soketler kullanÄ±lÄ±r. AyrÄ±ca video ve ses gÄ¼rÄ¼ntÄ¼lerinde genelde az bir veri kaybÄ± sesi veya gÄ¼rÄ¼ntÄ¼yÄ¼ bozmaz. Bu nedenle sÄ±kÄ± paket kontrolüne gerek yoktur. EÄŒer iyi bir fiziksel baÄŒlantÄ±nÄ±z varsa hata oranÄ± dÄ¼ÄŒÄ¼k olacaktÄ±r ve bu nedenle TCP'nin yaptÄ±ÄŒÄ± hatalÄ± paket kontrol iÄŒlemleri fazladan yÄ¼k olacaktır.

# Genel: SÄ¼reÄŒlerin HaberleÄŒmesi: Soket Programlama

UDP her ne kadar kendisi paket gÄ¼venliÄŒini denetlemese de bunu yazÄ¼lÄ¼mcÄ¼ yapabilir. Ä¼rneÄŒin TCP bir paketi gÄ¼nderdiÄŒinde karÄŒÄ¼ tarafÄ¼n onu aldÄ¼ÄŒÄ¼nÄ¼ anlamak iÄŒin acknowledgement bekler. UDP bunu yapmaz. Fakat bunu soket yazÄ¼lÄ¼mcÄ¼sÄ¼ yapabilir. YazÄ¼lÄ¼mcÄ¼, gÄ¼nderilen her pakete bir cevap bekleyerek bunu saÄŒler.

## 2.4. Uygulama KatmanÄ¼

TCP/IP protokolÄ¼nÄ¼n en Ä¼stÄ¼nde yer alÄ¼r. TaÄŒÄ¼ma katmanÄ¼nÄ¼n saÄŒladÄ¼ÄŒÄ¼ UDP ve

TCP protokollerini kullanarak veri aktarÄ¼mÄ¼ yapabilirler. Telnet, FTP, SMTP, HTTP uygulama katmanÄ¼ protokolleridir.

## 3. Uygulamadaki Temel Prensipler

Soketler her zaman iki uca sahiptir: AlÄ¼cÄ¼ ve gÄ¼nderici. BÄ¼tÄ¼n mesajlar ve protokol gereÄŒi olan baÄŒliÄ¼klar nihayetinde fiziksel katmandan, mantÄ¼ksal 1 ve 0'a karÄŒÄ¼Ä¼k gelen elektriksel sinyaller olarak gÄ¼nderilir.

Soket program ya istemci, yada sunucudur. ProgramlarÄ¼ daha karÄ¼Ä¼k olmakla beraber bazÄ¼ soket programlar her iki gÄ¼revi de yapmaktadÄ¼r. Sunucu program ile istemci program arasÄ¼nda ÄŒsalÄ¼Ä¼ma olarak bazÄ¼ farklar vardÄ¼r. AÄŒÄ¼Ä¼daki tablo her iki tarafta olaylarÄ¼n nasÄ¼l gittiÄŒini gÄ¼stermektedir:

Ä¼stemci	Sunucu
	Soket oluÄŒtur, <code>socket()</code>
Soket oluÄŒtur, <code>socket()</code>	Adres bilgilerini yerleÄŒtir <code>struct sockaddr_in</code>
	Soket adÄ¼nÄ¼ adresi ile iliÄŒkilendir <code>bind()</code>
	Soketi dinlemeye geÄŒ, <code>bind()</code>
BaÄŒlantÄ¼ yap, <code>connect()</code>	BaÄŒlantÄ¼yÄ¼ kabul et, <code>accept()</code>
Veri gÄ¼nder, <code>send()</code>	Veri al, <code>recv()</code>
Veri al, <code>recv()</code>	Veri gÄ¼nder, <code>send()</code>

# Genel: SÄ¼reÄŒlerin HaberleÄŒmesi: Soket Programlama

```
| ... | ... |
| DiÄŒer iÄŒlemler | DiÄŒer iÄŒlemler |
| ... | ... |
|-----|-----|
| Soketi kapat, close() | Soketi kapat, close() |
|-----|-----|
```

## 3.1. Soket TÄ¼rleri

TanÄ¼mlÄ¼ pek ÄŒok soket tÄ¼rÄ¼ vardÄ¼r. Ancak u dÄŒkÄ¼manda en ÄŒok kullanÄ¼lan 3 tÄ¼rden bahsedilecektir.

**SOCK\_STREAM:** SÄ¼ralÄ¼, gÄ¼venli, iki yollu, baÄŒlantÄ¼ yÄŒnelimli (connection oriented) veri akÄ¼Ä¼ saÄ¼lar. Veri alÄ¼Ä¼verÄ¼inden ÄŒnce baÄŒlantÄ¼ (connect) yapÄ¼lmÄ¼Ä¼ olmalÄ¼. Paketin kaybolmadÄ¼Ä¼nÄ¼ veya birden ÄŒok gelmediÄ¼ini uygulama katmanÄ¼na garanti eder. Kabul edilebilir bir sÄ¼re iÄŒerisinde veri transferi saÄ¼lanamazsa baÄŒlantÄ¼ yok varsayÄ¼lÄ¼r.

**SOCK\_DGRAM:** Datagram baÄŒlantÄ¼sÄ¼ saÄ¼lar. Datagramlar, baÄŒlantÄ¼sÄ¼z (connectionless), gÄ¼venilir olmayan paketlerdir.

**SOCK\_RAW:** TCP/IP ye raw olarak (herhangi bir format ve sÄ¼nÄ¼r olmadan) ulaÄ¼mayÄ¼ saÄ¼lar.

Raw soketler baÄŒÄ¼ baÄŒÄ¼na bir konu olup detaylÄ¼ bilgi iÄŒin Murat Balaban'Ä¼n yazdÄ¼Ä¼ <http://www.acikkod.org/yayingoster.php?id=34> dÄŒkÄ¼manÄ¼ndan faydalanabilirsiniz. Datagram ve stream soketleri [2.3.](#) TaÄŒÄ¼ma KatmanÄ¼ kÄ¼smÄ¼nda tartÄ¼Ä¼ldÄ¼.

## 3.2. Veri DÄŒnÄ¼Ä¼mleri

Ä¼nsanlarÄ¼n soldan saÄ¼ya veya saÄ¼dan sola alfabelere sahip olmalarÄ¼ gibi iÄŒlemciler de byte'larÄ¼ saklarken ÄŒnemli byte'Ä¼n solda veya saÄ¼da olmasÄ¼na gÄŒre sÄ¼nÄ¼flÄ¼rÄ¼lÄ¼r. Buna endianness da denir. Arap rakamlarÄ¼nda olduÄ¼u gibi (Ä¼ngilizce veya TÄ¼rkÄŒede kullandÄ¼Ä¼mÄ¼z rakamlar) ÄŒnemli byte'Ä¼n solda olduÄ¼u sÄ¼ralamaya

# Genel: SÄ¼reÄŒlerin HaberleÄŒmesi: Soket Programlama

big-endian denir. Ä–nemli byte'Ä±n en saÄŒda olduÄŒü sÄ±ralama ise Little Endian olarak adlandÄ±rÄ±lÄ±r.

BÄ¼tÄ¼n iÄŒlemciler kendi sÄ±ralamasÄ±nÄ± seÄŒmiÄŒtir. i386 ve klonu olan iÄŒlemciler little endian'dÄ±r. Sun Sparc, Motorola 68K ve PowerPC big endian kullanÄ±r. Java Sanal Ä°ÄŒlemcisi (Java VM) de big endian kullanÄ±r.

FarklÄ± iki iÄŒlemcisi olan makineler birbirileri ile haberleÄŒecekleri zaman (IPC), bu veri dÄŒnÄ¼ÄŒÄ¼mÄ¼nÄ¼ yapmazlar ise haberleÄŒemezler.

ÄŒ protokolleri de kendi sÄ±ralamasÄ±nÄ± seÄŒmelidir. Aksi takdirde iki farklÄ± mimarideki bilgisayar IPC yaparak birbirileri ile haberleÄŒecekleri zaman anlaÄŒamayacaklardÄ±r. TCP/IP big endian sÄ±ralamasÄ±nÄ± kullanÄ±r. Bunun anlamÄ± ÄŒü: Herhangi bir paket (IP adresi, paket uzunluÄŒü, kontrol deÄŒeri gibi) gÄŒnderileceÄŒi zaman en ÄŒnemli byte'Ä± ÄŒnce gÄŒnderilir ve alÄ±nÄ±r.

Ä°ki tÄ¼r byte sÄ±ralamasÄ± vardÄ±r. En ÄŒnemli byte'Ä±n ÄŒnde geldiÄŒi sÄ±ralama ki buna *Network Byte SÄ±ralamasÄ±* denir ve ÄŒnemli byte'Ä±n sonra geldiÄŒi dÄ±ralama. Buna da *Host Byte SÄ±ralamasÄ±* denir. Bu ikisi arasÄ±ndaki dÄŒnÄ¼ÄŒmÄ¼ler aÄŒaÄŒÄ±daki dÄŒrt fonksiyon tarafÄ±ndan yapÄ±lmaktadÄ±r:

```
#include <netinet/in.h>
```

```
uint32_t htonl(uint32_t hostlong); /* Host to Network Long */
```

```
uint16_t htons(uint16_t hostshort); /* Host to Network Short */
```

```
uint32_t ntohl(uint32_t netlong); /* Network to Host Long */
```

```
uint16_t ntohs(uint16_t netshort); /* Network to Host Short */
```

ÄŒer iki host da little endian ise veri transferinden ÄŒnce network byte sÄ±rasÄ±na ÄŒevirilmeli. AlÄ±nan tarafta tekrar little endian'a ÄŒevrilir.

# Genel: SÄ¼reÄŒlerin HaberleÄŒmesi: Soket Programlama

ÄŒu senaryoyu dikkatle inceleyelim inceleyelim: Intel bir makine internet Ä¼zerinden SPARC makine ile haberleÄŒecek. Veri olarak 192.168.1.254 gibi bir IP adresini dÄ¼ÄŒÄ¼nelim. Intel iÄŒlemci little endian olarak bunu 0xFE01A8C0(254 1 168 192) ÄŒeklinde tutar ve FE-01-A8-C0 sÄ¼rasÄ¼yla bunu gÄ¼nderir. SPARC makine bunu aynÄ¼ sÄ¼rada alacak ve bu sÄ¼rada kullanacak. Ä¼Ä¼nkÄ¼ big endian kullanÄ¼r ve Ä¼nemli olan en baÄ¼ta gelir. DolayÄ¼sÄ¼yla SPARC bu IP adresini 254.1.168.192 olarak algÄ¼lar.

ProgramcÄ¼ hostlarda hangi sÄ¼ralamanÄ¼n kullanÄ¼ldÄ¼ÄŒÄ¼nÄ¼ bilmek zorunda deÄ¼il. YukarÄ¼daki dÄ¼nÄ¼ÄŒÄ¼rÄ¼cÄ¼ fonksiyonlar kendi iÄŒlemcileri ile ilgili dÄ¼nÄ¼ÄŒÄ¼mleri otomatik olarak yapacaktÄ¼r.

Veriler aÄ¼ Ä¼zerinde network byte sÄ¼rasÄ¼ ile dolaÄ¼yacaktÄ¼r. Veriyi alan host makina ntohl ile bu veriyi kendi anladÄ¼ÄŒÄ¼ sÄ¼raya ÄŒevirecektir.

## 3.3. Sistem Ä¼ÄŒÄ¼rÄ¼larÄ¼

Adres atamasÄ¼:

Gerek sunucu, gerekse istemci internet adres bilgilerini tutmak iÄŒin sockaddr\_in yapÄ¼sÄ¼nÄ¼ kullanÄ¼r. Bu yapÄ¼nÄ¼n aÄŒÄ¼lÄ¼mÄ¼ Ä¼yü ÄŒekildedir:

```
struct sockaddr_in {  
  
    short  
    sin_family;  
  
    unsigned short  
    sin_port;  
  
    struct in_addr  
    sin_addr;  
  
    char  
    sin_zero[8];  
}
```

## Genel: SÄ¼reÄŒlerin HaberleÄŒmesi: Soket Programlama

Bu yapÄ± iŒerisinde bir baÄka yapÄ± daha vardÄr. Bu yapÄ± in\_addr yapÄ±sÄ±dÄr ve aŒÄ±mÄ± ÅŒyledir:

```
struct in_addr {
```

```
union {
```

```
struct { u_char s_b1, s_b2, s_b3, s_b4; } S_un_b;
```

```
struct { u_short s_w1, s_w2; } S_un_w;
```

```
u_long S_addr;
```

```
} S_un;
```

```
}
```

# Genel: SÄ¼reÄŒlerin HaberleÄŒmesi: Soket Programlama

Bu yapÄ±larÄ± doÄŒrudan kullanmayacaÄŒÄ±z; o nedenle rahat olun. Ancak ilk yapÄ±nÄ±n temel yapÄ±mÄ±z olduÄŒunu unutmayÄ±n. Bu yapÄ± iÄŒerisinde IP adreslerini ve port numaralarÄ±nÄ± saklayacaÄŒÄ±z.Ä

Ä°nsanlar bir baÄŒlantÄ± gerÄŒekleÄŒtirecekleri zaman genelde host isimlerini kullanÄ±rlar. Oysa internetteki hiÄŒbir makine host ismini kullanmaz. "cclub.ktu.edu.tr" bir host ismidir. 193.140.168.77 bu hostun sahip olduÄŒu IP adresidir. Host isimlerini IP adreslerine ÄŒeviren sistemlere DNS (Domain Name Server) denir.Ä Ä°ÄŒletim sistemi bize DNS iÄŒlemlerini yapmak iÄŒin kÄ¼tÄ¼phane sunmaktadÄ±r. IP dÄŒnÄ¼ÄŒmÄ¼ yapmak iÄŒin kÄ¼tÄ¼phane ÄŒaÄŒrÄ±larÄ±ndan gethostbyname() 'i kullanabiliriz.

```
#include <netdb.h>
struct hostent* host;
host = gethostbyname("cclub.ktu.edu.tr");
```

Soket OluÄŒturma:

Ä°ki programÄ±n haberleÄŒmesi iÄŒin ÄŒncelikle her iki tarafta da soket aÄŒÄ±lmasÄ± gerekir.

```
int socket(int domain, int type, int protocol);
```

socket() fonksiyonunun ikinci parametresi soketin tipidir. Buraya:Ä

SOCK\_STREAMÄ

SOCK\_DGRAMÄ

SOCK\_RAWÄ

SOCK\_SEQPACKETÄ

SOCK\_RDMÄ

SOCK\_PACKET

# Genel: SÄ¼reÄŒlerin HaberleÄŒmesi: Soket Programlama

gibi soket tÄ¼rlerini yazabiliriz. Ä°lk ikisini yukarÄ±da anlattÄ±m. DiÄŒerlerinin ne anlama geldiÄŒi ÄŒu aÄŒamada ÄŒnemli deÄŒildir. YalnÄ±zca biz SOCK\_STREAM tipini kullandÄ±ÄŒÄ±mÄ±z iÄŒin ikinci parametre olarak bunu verdiÄŒimizi bilin. Yani bu bir stream sunucu soket uygulamasÄ±dÄ±r.

ÄœÄŒÄ¼ncÄ¼ parametresi protocol tipini belirler. Hali hazÄ±rda 5 protokol tÄ¼rÄ¼ vardÄ±r :Ä

AF\_UNIX (UNIX internal protocols)Ä

AF\_INET (ARPA Internet protocols)Ä

AF\_ISO (ISO protocols)Ä

AF\_NS (Xerox Network Systems protocols)Ä

AF\_IMPLINK (IMP "host at IMP" link layer)Ä

Soket ile ilgili tanÄ±mlamalarÄ±n baÄŒliÄ±k dosyalarÄ± sys/types.h ve <sys/socket.h> dir.

Veri AlÄ±ÄŒ-veriÄŒi:

Birbiri ile baÄŒlantÄ±sÄ± yapÄ±lmÄ±ÄŒ iki soket arasÄ±nda artÄ±k veri alÄ±ÄŒ-veriÄŒine baÄŒlayabiliriz. Buradaki veriler karÄŒÄ±ya taÄŒÄ±nmasÄ± gereken gerÄŒek veriler olabileceÄŒi gibi, karÄŒÄ±ya yapmasÄ± gereken iÄŒleri bildiren bir komut listesi olabilir. Her ÄŒey, iki program arasÄ±nda tarafÄ±nÄ±zdan tanÄ±mlanmÄ±ÄŒ protokol ÄŒerÄŒevesinde olacaktÄ±r.

write() ve read() fonksiyonlarÄ± aynÄ± zamanda dosya iÄŒlemleri iÄŒin de kullanÄ±lÄ±r.Ä send() ve recv() ise bu fonksiyonlarÄ±n soketler iÄŒin ÄŒzelliÄŒtirilmiÄŒ halidir.

```
int send(int s, const void *msg, int len, unsigned int flags);
int recv(int s, const void *msg, int len, unsigned int flags);
```

## 4. Sunucu Soket ProgramÄ±

# Genel: SÄ¼reÄŒlerin HaberleÄŒmesi: Soket Programlama

## 4.1. Sistem ÄŒtaÄŒrÄ¼larÄ¼nÄ¼n KullanÄ¼mÄ¼

Bu kadar teorik bilgiden sonra bir uygulama yapmaya geÄŒebiliriz. 2222 numaralÄ¼ portta ÄŒalÄ¼ÄŒan basit bir sunucu programÄ¼ Ä¼zerinde olaylarÄ¼ inceleyelim. Bundan sonra Ä¼zerinde ÄŒalÄ¼ÄŒacaÄŒÄŒmÄ¼z kodlar sunucu iÄŒin yazÄ¼lmaktadÄ¼r. Ä¼stemci program olarak standart telnet programÄ¼nÄ¼ kullanacaÄŒÄ¼z. Mimari olarak sunucu yazÄ¼lÄ¼mlerÄ¼n, istemci yazÄ¼lÄ¼mlardan farklÄ¼ olduÄŒunu unutmayÄ¼n. AÄŒaÄŒÄ¼daki kodlarÄ¼ gcc C derleyicisi kullanarak yazdÄ¼m. Ancak bu kodlar tamamen standart olup Windows altÄ¼ndeki Visual C derleyicisi ile de ÄŒalÄ¼ÄŒabilir. DeÄŒiÄŒik platformlarda sorun yatmasÄ¼n diye kod iÄŒerisinde TÄ¼rkÄŒe karakter kullanmadÄ¼m.

Ä¼nce port numaramÄ¼zÄ¼ belirleyelim:

```
#define PORT 2222
```

SÄ¼ra soket oluÄŒturmada:

```
int sockfd;  
sockfd=socket(AF_INET,SOCK_STREAM,0);  
if (sockfd<0) {  
    perror("socket");  
    exit(1);  
}
```

Burada verdiÄŒim kodlar tek baÄŒÄ¼na ÄŒalÄ¼ÄŒmaz. Daha kolay anlaÄŒÄ¼lsÄ¼n diye programdan parÄŒalar bir araya getiriyorum. En sonunda ÄŒalÄ¼ÄŒabilir kodun tamamÄ¼nÄ¼ vereceÄŒim. Bu iÄŒlemlere baÄŒlamadan ÄŒnce bazÄ¼ header (.h) dosyalarÄ¼nÄ¼n yÄ¼klenmesi gerekir. BunlarÄ¼ da kodun tamamÄ¼nda gÄŒretilirsiniz.

YukarÄ¼daki kodu incelersek: Soket oluÄŒturma iÄŒlemini socket() fonksiyonu ile yapÄ¼yoruz.

socket() fonksiyonu geriye bir tamsayÄ¼ deÄŒer dÄŒndÄ¼rÄ¼r. Hata oluÄŒumunda bu deÄŒer -1'dir. EÄŒer hata oluÄŒmazsa geri dÄŒnen deÄŒer

## Genel: SÃ¼reÃ¼Ålerinin HaberleÅ¼mesi: Soket Programlama

soketin tanÃ¼mlenmiÅ¼ numarasÃ¼dÃ¼r. Bu numara en baÅ¼ta bahsettiÅ¼imiz gibi dosya tanÃ¼mlenmiÅ¼sÃ¼dÃ¼r. Bu numarayÃ¼ kodda gÃ¼rÃ¼ldÃ¼Å¼ gibi sockfd deÅ¼iÅ¼kenine atadÃ¼m. Bu iÅ¼lemden sonra soketi takip etmek iÅ¼in artÃ¼k sockfd deÅ¼iÅ¼kenini kullanacaÅ¼Ã¼m. OluÅ¼turulan her soket bu Å¼ekilde ayrÃ¼ bir numara alÃ¼r.

Hata kontrolÃ¼ yapmaya Å¼izen gÃ¼sterin. Aksi halde zamanÃ¼zÃ¼n bÃ¼yÃ¼k bir kÃ¼smÃ¼nÃ¼ programdaki hatalarÃ¼ ayÃ¼klamak ile geÅ¼irirsiniz.

Å¼imdi port numarasÃ¼nÃ¼ atamasÃ¼nÃ¼ yapalÃ¼m:

```
if(bind(sockfd,(struct sockaddr *)&my_addr,sizeof(struct sockaddr)) == -1) {  
    perror("bind");  
    exit(1);  
}
```

bind() fonksiyonu soketi isimlendirir. socket() ile oluÅ¼turulmuÅ¼ bir soket

iÅ¼in isim uzayÃ¼nda (bellekte) vardÃ¼r ancak bir isme sahip deÅ¼ildir.

Bind iÅ¼lemi

belirtilen port numarasÃ¼nÃ¼ ve gerekli sistem kaynaklarÃ¼nÃ¼ iÅ¼letim sisteminden ister. EÅ¼er

bu iÅ¼lemi yapmazsanÃ¼z iÅ¼letim sistemi port havuzundan herhangi bir port atar. Port

numarasÃ¼nÃ¼ belirlemezseniz, diÅ¼er insanlar sizin programla haberleÅ¼ecek programlar yazamaz. bind() fonksiyonu aynÃ¼ zamanda sizi bir veya daha

fazla aÅ¼ arabirim kartÃ¼ndan (Network Interface Card) yalÃ¼tÃ¼r. AÅ¼

Ã¼zerindeki her host bir NIC'a sahiptir ve her NIC en az bir IP adresine

sahiptir. addr.sin\_addr deÅ¼eri olarak INADDR\_ANY yazmakla iÅ¼letim

sistemine host Ã¼zerinde bulunan mÃ¼mkÃ¼n bÃ¼tÃ¼n IP adreslerinden baÅ¼lantÃ¼ kabul edeceÅ¼inizi sÃ¼ylÃ¼yorsunuz. Bunu

istemiyorsanÃ¼z makinenin sahip olduÅ¼u IP numaralarÃ¼ndan birini, hizmet vermek

Ã¼zere seÅ¼ebilirsiniz. EÅ¼er Ã¼zerinde bulunduÅ¼unuz makine bir firewall ise

belirlenmiÅ¼ bir IP'yi kullanmak yararÃ¼nÃ¼za olacaktÃ¼r. Bu Å¼ekilde firewall'un yalÃ¼zca bir yÃ¼zÃ¼ hizmet sunacaktÃ¼r.

```
if (listen(sockfd, 5) == -1) {  
    perror("listen");  
    exit(1);  
}
```



# Genel: Sereşlerin Haberleşmesi: Soket Programlama

hatırlayın. Sanki bir şeyler hatırlar gibi olduk değil mi?

```
int fd, client_size;
struct sockaddr_in client_addr;
client_size = sizeof(struct sockaddr_in);
fd = accept(sockfd, (struct sockaddr *)&client_addr, &client_size);
printf("Merhaba %s", inet_ntoa(client_addr.sin_addr));
```

Sunucumuza bir bağlantı yapıldığında sunucu ekrana "Merhaba 193.140.168.54" gibi bir şey yazacak. Gördüğümüz gibi bağlantı yapanın adresini de alacak.

## 4.2. sunucu.c

```
// sunucu.c - Stream soket sunucu
// Baris Simsek, <simsek at acikkod org>
// http://www.acikkod.org

#include <stdio.h>

#include <string.h>
#include <netdb.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <sys/wait.h>

#define PORT 3333
#define LISTQUEUE 5

main(int argc, char *argv[])
{
    int sockfd, new_fd;
    struct sockaddr_in server_addr, client_addr;
    int client_size;
    char buffer[1024];

    printf("%s %d portu uzerinde calismaya basladi...
", argv[0], PORT);

    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
        perror("socket");
        exit(1);
```

# Genel: SÄ¼reÄŒlerin HaberleÄmesi: Soket Programlama

```
}

memset(&server_addr, 0, sizeof(server_addr));
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(PORT);
server_addr.sin_addr.s_addr = INADDR_ANY;

if (bind(sockfd, (struct sockaddr *)&server_addr, sizeof(server_addr)) == -1) {
    perror("bind");
    exit(1);
}

if (listen(sockfd, LISTQUEUE) != 0) {
    perror("listen");
    exit(1);
}

while(1) {
    client_size = sizeof(struct sockaddr_in);
    if((new_fd=accept(sockfd,(struct sockaddr*)&client_addr,&client_size)) == -1) {
        perror("accept");
        exit(1);
    }

    printf("%s sunucumuza baglandi...
",inet_ntoa(client_addr.sin_addr));

    memset(&buffer, 0, sizeof(buffer));
    strcpy(buffer,"Merhaba ");
    strcat(buffer,(char *) inet_ntoa(client_addr.sin_addr));
    strcat(buffer," :");

    if (send(new_fd,&buffer, strlen(buffer), 0) == -1) perror("send");
    if (recv(new_fd,&buffer, strlen(buffer)-1, 0) == -1) perror("recv");
    printf("Alinan yanit: %s
", buffer);

    close(new_fd);
}
close(sockfd);
while (waitpid(-1,NULL,WNOHANG) > 0);
return 0;
}
```

Evet. Ä°Äte harika dÄ¼nyaya adÄ±m

# Genel: SÄ¼reÄŒlerin HaberleÄŒmesi: Soket Programlama

attÄ±k ve "Merhaba" dedik. Bunu ilk denediÄŒimde gÄŒzlerime inanamamÄ±ÄŒtÄ±m. BaÄŒka bir bilgisayardaki kiÄŒiye merhaba demiÄŒtim. O zamanlar chat programlarÄ± bu kadar populer deÄŒildi. Bu gerÄŒekten harika bir ÄŒey.

ÄŒimdi programÄ±mÄ±zÄ± linux altÄ±nda "GNU C Compiler" gcc ile nasÄ±l derleyeceÄŒimizi gÄŒrelim.

```
cclub:~> gcc -o sunucul sunucu.c
```

Bu komutla sunucu.c dosyasÄ±na yazdÄ±ÄŒÄ±mÄ±z kodlarÄ± derleyip sunucu1 isimli ÄŒalÄŒabilir dosyayÄ± ÄŒrettik. BakalÄ±m neler oluyor:

```
cclub:~> ./sunucul
./sunucul 2222 portu uzerinde calismaya basladi...
193.140.168.54 sunucumuza baglandi...
```

Tahmin edeceÄŒiniz gibi bunlar sunucunun ekranÄ±na yazanlar. "193.140.168.54 sunucumuza baglandi..." mesajÄ±, ceng hostundan baÄŒlantÄ± yapÄ±ldÄ±ÄŒÄ±nda ekrana basÄ±lmÄ±ÄŒtÄ±r. Peki istemcinin ekranÄ±na neler dÄŒinÄŒiyor:

```
ceng:~# telnet 193.140.168.77 2222
Trying 193.140.168.77...
Connected to 193.140.168.77.
Escape character is '^]'.
Merhaba 193.140.168.54 :)
```

Merhaba 193.140.168.54 :) mesajÄ±, sunucu tarafÄ±ndan istemcimize (telnet) gÄŒnderilen mesajdÄ±r. Onun ÄŒstÄŒindeki mesajlarÄ± merak etmeyin. OnlarÄ±n bizimle ilgisi yok. Telnet'in ÄŒrettiÄŒi mesajlardÄ±r. Windows makineden "BaÄŒlat-ÄŒtalÄ±ÄŒtÄ±r" kullanarak aynÄ± komutu verebilirsiniz.

## 5. ÄŒstemci Soket ProgramÄ±

ÄŒimdi istemci bir soket programÄ±n genel yapÄ±sÄ±na bakalÄ±m. BaÄŒlangÄŒÄŒ aÄŒamasÄ±nda iÄŒlemler sunucu program ile aynÄ± olacaktÄ±r. Ancak asÄ±l iÄŒ yapÄ±lan kÄ±sÄ±mda hem mimari, hem de fonksiyonlar deÄŒiÄŒecek. ÄŒÄŒnkÄŒ bir taraf hizmet verme, ÄŒteki taraf ise hizmet alma durumunda olacaktÄ±r.

ÄŒimdi geliÄŒtireceÄŒimiz istemci yukardaki sunucu iÄŒin geliÄŒtirilmemiÄŒtir. Bu programÄ±mÄ±z 3333 numaralÄ± portta ÄŒalÄŒÄ±mÄ±n. Bu

## Genel: SÃ¼reÃ¼Ålerinin HaberleÅmesi: Soket Programlama

istemci program iÅinin bir de sunuu program yazdÄ±m. Sunucu programÄ±n ÅalÄ±ÅmasÄ±nÄ± yukarÄ±da anlattÄ±Åmdan tekrar bu program iÅinin de anla. Bu programÄ±mÄ±z 3333 numaralÄ± portta ÅalÄ±ÅsÄ±n. Bu istemci program iÅinin bir de sunuu program yazdÄ±m. Sunucu programÄ±n ÅalÄ±ÅmasÄ±nÄ± yukarÄ±da anlattÄ±Åmdan tekrar bu program iÅinin de anlatmayacaÅÄ±m. DoÅrudan kodunu vermekle yetineceÅÄ±m.

Äncelikle port numaramÄ±zÄ± belirleyelim:

```
#define PORT 3333
```

Sunucunun ÅalÄ±ÅtÄ±ÅÄ± bilgisayar istemciye parametre olarak verilecek:

```
int main(int argc, char *argv[])
{
    if (argc != 2) {
        printf("Kullanimi : %s hostname ",argv[0]);
        exit(1);
    }
}
```

Sunucunun Belirlenmesi:

Parametre olarak alÄ±nan host ismini ÅÅzÄ±mlÄ±yoruz. Adres yapÄ±sÄ±nda olan server\_addr deÅiÅkenine PORT numarasÄ±nÄ± ve IP adresini yazÄ±yoruz.

```
struct hostent *h_name;

if(argc > 1) {
    h_name = gethostbyname(argv[1]);
}
else {
    printf("Kullanimi: %s hostname ",argv[0]);
    exit(1);
}

bzero((char *) &serv_addr, sizeof(serv_addr));
serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = *(u_long *) h_name->h_addr;
serv_addr.sin_port = htons(PORT);

printf("Sunucu adresi: %s ",inet_ntoa(serv_addr.sin_addr));
```

# Genel: SÄ¼reÄŒlerin HaberleÄmesi: Soket Programlama

Sunucuya BaÄlanma:

ArtÄ±k hizmetkarÄ±mÄ±za, yani sunucumuza baÄlanÄ±p selam verme vakti geldi.

```
printf("Mesajinizi girin: ");
fgets(buf, sizeof(buf)+1, stdin);

if((send(sockfd, buf, sizeof(buf), 0)) <= 0) perror("send");
if((recv(sockfd, buf, sizeof(buf), 0)) <= 0) perror("recv");
printf("Sunucu'dan gelen mesaj: %s", buf);

shutdown(sockfd, 2);
close(sockfd);
```

Aynen sunucu programda olduÄ¼ü gibi soket aÄŒtÄ±k. Soket tÄ¼rÄ¼mÄ¼z yine SOCK\_STREAM. Sonra connect()fonksiyonunu kullanarak aÄŒtÄ±ÄÄ±mÄ±z soket Ä¼zerinden bir baÄlantÄ± yaptÄ±k. connect() fonksiyonu sys/types.h baÄlÄ±k dosyasÄ±nda aÄÄÄ±daki gibi tanÄ±mlanmÄ±ÄtÄ±r:

```
int connect(int s, const void *addr, int addrlen);
```

Ä°lk parametre soket tanÄ±mlayÄ±cÄ±sÄ±, ikinci parametre baÄlantÄ± kurulacak sunucunun adres bilgilerini iÄŒeren server\_addr yapÄ±sÄ±, son parametre ise adres yapÄ±sÄ±nÄ±n boyutudur. ArtÄ±k sunucumuzla aramÄ±zda stream bir soket baÄlantÄ±sÄ± kurmuÄ¼ olduk. Ä°stemci tarafÄ±nda connect() yapÄ±ldÄ±ÄÄ± zaman, sunucu tarafÄ±nda accept() yapÄ±ldÄ±r. Yani yapÄ±lan baÄlantÄ± kabul edilir. Daha Ä¼nce yazdÄ±ÄÄ±mÄ±z sunucu programdaki accept() iÄlemi, telnet programÄ±nÄ±n connect() isteÄ¼ini karÄÄ±lÄ±yordu. Ä¼imdi bizim istemimiz bu isteÄ¼i gÄ¼nderecek. Dikkat ederseniz ilkel bir telnet programÄ± yazÄ±yoruz.

Sunucuya Bilgi GÄ¼nderme:

Sonraki kÄ±smÄ±, anlaÄ¼Ä±lmasÄ± aÄŒÄ±sÄ±ndan basit tuttum. Ekrandan bir mesaj alÄ±p bunu send() fonksiyonu ile sunucuya gÄ¼nderiyoruz. Bu fonksiyon, dosyalara yazmak iÄŒin kullandÄ±ÄÄ±mÄ±z write() fonksiyonu gibidir. send() ile sokete veri gÄ¼nderiyoruz. Soket sunucu ile baÄlantÄ±lÄ± olduÄ¼üandan veri, sunucuya ulaÄ¼Ä±r. Sunucu ise her send() isteÄ¼ine recv() ile karÄÄ±lÄ±k verir. send() yada recv() iÄlemleri her iki tarafta da yapÄ±labilir. Sunucudan gelen her send() isteÄ¼ine karÄÄ±k da istemci de bir recv() vardÄ±r. Ä¼rneÄ¼in ftp programÄ±nda karÄÄ±lÄ±k send() ve recv() iÄlemleri vardÄ±r. Siz Ä¼nce istemci tarafÄ±ndan 'get

## Genel: SÄ¼reÄŒlerin HaberleÄŒmesi: Soket Programlama

dosya' komutunu gÄ¼ndererek dosya isteÄŒinizi belirtiyorsunuz. Sunucu bu komutu okuyup yorumluyor ve size dosyayÄ± ftp protokolüne uygun paketler halinde gÄ¼nderiyor. ArtÄ±k siz okumaya baÄŒlyÄ±yorsunuz. Paketleri alÄ±p birleÄŒtirip yerel diske yazÄ±yorsunuz. Bu aÄŒamadan sonra istediÄŒinizi yaptÄ±rabilirsiniz.

istemci.c

AÄŒaÄŒÄ±daki bir SMTP sunucuya baÄŒlanÄ±p smtp komutlarÄ± gÄ¼nderir ve cevaplarÄ±nÄ± alÄ±r. Tam olarak bir SMTP istemci deÄŒildir. SMTP protokolünün detaylari iÄŒin RFC 821'e bakabilirsiniz.

```
/*
 * istemci.c - SMTP sunucuyu kontrol eder.
 *
 * Baris Simsek, <simsek at acikkod org>
 * http://www.acikkod.org/
 */

#include <stdio.h>
#include <string.h>
#include <netdb.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/wait.h>
#include <sys/socket.h>

#define SMTP_PORT 25

int main(int argc, char *argv[])
{
    struct hostent *h_name;
    int sd;
    char cmd[512];
    struct sockaddr_in serv_addr;

    printf("checkd - SMTP sunucuyu kontrol eder.
");
    printf("http://www.acikkod.org/

");

    if(argc < 2) {
        printf("KullanÄ±mÄ±: %s hostname
",argv[0]);
        exit(0);
    } else {
        h_name = gethostbyname(argv[1]);
```

# Genel: SÄ¼reÄŒlerin HaberleÄmesi: Soket Programlama

```
}

bzero((char *) &serv_addr, sizeof(serv_addr));
serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = *(u_long *) h_name->h_addr;
serv_addr.sin_port = htons(SMTP_PORT);

printf("> Soket aÄŒlÄ±yor... ");
if( (sd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
    printf("hata!
");
    perror("socket");
    exit(1);
} else printf("tamam
");

printf("> %s baÄŒlanÄ±yor... ", inet_ntoa(serv_addr.sin_addr));
if(connect(sd, (struct sockaddr*)&serv_addr, sizeof(serv_addr)) < 0) {
    printf("hata!
");
    perror("connect");
    exit(2);
} else printf("tamam
");

printf("> KarÄŒlama mesajÄ± alÄ±nÄ±yor... ");
memset(cmd, 0, 256);
if((recv(sd, cmd, 256, 0)) <= 0) {
    printf("hata!
");
    perror("recv");
    exit(3);
} else printf("tamam
");

printf(" %s
", cmd);

printf("> Selam veriliyor... ");
memset(cmd, 0, 256);
strcpy(cmd, "helo world
");
if((send(sd, cmd, 256, 0)) <= 0) {
    printf("hata!
");
    perror("send");
    exit(4);
} else printf("tamam
");
```

# Genel: SÄ¼reÄŒlerin HaberleÄŒmesi: Soket Programlama

```
printf("> Cevap alÄ±nÄ±yor... ");
memset(cmd, 0, 256);
if((recv(sd, cmd, 256, 0)) <= 0) {
    printf("hata!
");
    perror("recv");
    exit(5);
} else printf("tamam
");

printf("  %s
", cmd);

printf("SMTP sunucunuz ÄŒsalÄ±ÄŒiyor.
");

return 0;
}
```

## 6. GeliÄŒmiÄŒ G/ÄŒ

### 6.1. Bloksuz G/ÄŒ

GiriÄŒ/ÄŒÄ±kÄ±ÄŒ (I/O); bir dosyaya, pipe'a, terminale veya aÄŒ aygÄ±tÄ±na yazmak veya bu aygÄ±tlardan okumak gibi iÄŒlemleri iÄŒermektedir.

Okunacak veri hazÄ±r deÄŒilse veya yazÄ±lacak veri o an kabul edilmiyorsa bu iÄŒlemleri yapan sÄ¼reÄŒler bloklanacaktır. Bloksuz G/ÄŒ (Nonblocking I/O), verinin veya aygÄ±tÄ±n hazÄ±r olmamasÄ± gibi durumlarda bloklanmayÄ± tamamen ortadan kaldÄ±ran bir ÄŒzelliktir. Bu konunun detaylarÄ± dÄŒkÄ¼manÄ±n kapsamÄ± dÄ±ÄŒndedir. (man 2 fcntl)

### 6.2. select()

Soket program aynÄ± anda birden fazla soketten veri okumak veya yazmak durumunda kalabilir. Bunu tek soket tanÄ±mlayÄ±cÄ± ile saÄŒleyamazsÄ±nÄ±z. ÄŒÄ¼nkÄ¼ soketiniz blok durumuna geÄŒtiÄŒinde (ÄŒrneÄŒin accept(), baÄŒlantÄ± gelene kadar programÄ±n blok olmasÄ±na neden olur) kodunuzun geri kalan kÄ±smÄ± ÄŒsalÄ±ÄŒmaz, bloktan ÄŒÄ±kmayÄ± bekler.

ÄŒÄŒyle bir senaryoyu hayal edelim. ÄŒki adet soket tanÄ±mlayÄ±cÄ± var ve bu ikisi ile karÄŒÄ± uÄŒtan dosya veri alacaksÄ±nÄ±z. 1. uÄŒ henÄ¼z veriyi hazÄ±rlamadÄ± ancak 2. uÄŒ hazÄ±rladÄ± varsayalÄ±m. EÄŒer program 1. soket tanÄ±mlayÄ±cÄ± ile ilk ÄŒnce 1. uÄŒtan

## Genel: SÄ¼reÅŒlerin HaberleÅŒmesi: Soket Programlama

veri ÅŒekmeye ÅŒalÄ±ÅŒÄ±rsa blok olacaktÄ±r. 2. ucun verisi hazÄ±r olduÄŸu halde veri alÄ±namayacaktÄ±r. Oysa select() ile bu iki uÅŒ kontrol edilip hazÄ±r olan uÅŒtan -senaryomuzda 2. uÅŒ- veri okunsa idi program blok olmayacak ve verisi hazÄ±r olan iÅŒlerini yapmaya devam edecekti. Bu ÅŒekilde bloklanma riski taÅŒÄ±mayan bir program yazÄ±labilir.

Tek bir sÄ¼reÅŒ iÅŒerisinde bloksuz G/ÅŒ kullanarak bu sorun ÅŒÄ±zÄ±labilir. BÄ¼tÄ¼n dosya tanÄ±mlayÄ±cÄ±lar bloksuz G/ÅŒ yapacak ÅŒekilde set edilir. EÅŒer veri hazÄ±r deÅŒilse read() hemen sonlanÄ±r. AynÄ± iÅŒlemi ikinci dosya tanÄ±mlayÄ±cÄ±sÄ± iÅŒin de yaparÄ±z. Belli bir sÄ¼re sonra tekrar ilk tanÄ±mlayÄ±cÄ±yÄ± kontrol ederiz. Buna 'polling' deniliyor. Bunun dezavantajÄ±, gereksiz yere CPU zamanÄ± harcamaktadÄ±r.

AynÄ± problemi ÅŒÄŒlemek iÅŒin kullanÄ±labilecek bir diÅŒer teknik de "asen kron G/ÅŒ". Bu yÄŒntemde, dosya tanÄ±mlayÄ±cÄ±mÄ±z G/ÅŒ iÅŒin hazÄ±r olduÄŸunda ÅŒekirdek, G/ÅŒ yapacak sÄ¼reci haberdar edecektir. Ancak burada standart problemleri ve sinyalleri iÅŒleme (signal handling) ile ilgili problemler vardÄ±r.

Bunlardan daha iyi bir teknik ise G/ÅŒ ÅŒoÅŒullama (I/O multiplexing) olarak adlanÄ±lmaktadÄ±r. ÅŒlgilendiÄŸimiz tanÄ±mlayÄ±cÄ±larÄ±n eklendiÄŸi bir kÄ¼me vardÄ±r ve tanÄ±mlayÄ±cÄ±lardan biri G/ÅŒ iÅŒin hazÄ±r olmadÄ±kÅŒa ÅŒÄ±kmayan bir sistem ÅŒaÅŒrÄ±sÄ± yapÄ±lÄ±r. Sistem ÅŒaÅŒrÄ±sÄ±ndan ÅŒÄ±kÄ±ldÄ±ÄŸÄ±nda hangi tanÄ±mlayÄ±cÄ±larÄ±n G/ÅŒ iÅŒin hazÄ±r olduÄŸu sorulabilir.

select birden fazla soket tanÄ±mlayÄ±cÄ±nÄ±n durumunu takip eden ve BSD4.2 ile gelen bir sistem ÅŒaÅŒrÄ±sÄ±dÄ±r. Burada ÅŸunu belirtmeliyim ki, select() yalnÄ±zca soketler iÅŒin deÅŒil genel olarak dosya ve G/ÅŒ iÅŒlemleri iÅŒin kullanÄ±lan bir sistem ÅŒaÅŒrÄ±sÄ±dÄ±r.

```
#include <sys/time.h>
#include <sys/types.h>
```

```
#include <unistd.h>
```

```
int select(int n, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct
timeval *timeout);
```

n: En bÄ¼yÄ¼k soket tanÄ±mlayÄ±cÄ±nÄ±n bir fazlasÄ± (tanÄ±mlayÄ±cÄ±lar 0'dan baÅŒladÄ±ÄŸÄ±ndan)

readfds: Okumak iÅŒin izlenecek (hazÄ±r olup olmadÄ±ÄŸÄ± bakÄ±lacak) soket tanÄ±mlayÄ±cÄ±

## Genel: SÄ¼reÄŒlerin HaberleÄŒmesi: Soket Programlama

writelfds: Yazmak iÄŒin izlenecek soket tanÄ±mlayÄ±cÄ±

exceptfds: Ä°stisnai durumlar iÄŒin izlenecek soket tanÄ±mlayÄ±cÄ±

timeval, aÄŒaÄŒÄ±daki ÄŒekilde bir veri yapÄ±sÄ±dÄ±r:

```
struct timeval {  
    Ä long tv_sec; /* seconds */  
    Ä long tv_usec; /* microseconds */  
};
```

timeout eÄŒer NULL olarak verilirse bir dosya hazÄ±r olana kadar blokda bekler. HazÄ±r olmazsa sonsuza kadar bekleyecektir. Tabi sinyal(signal) gonderilirse sonlanÄ±r. EÄŒer sinyal ile sonlandÄ±rÄ±lÄ±rsa select() -1 dÄŒÄ±ndÄ¼recek ve errno=EINTR olacaktÄ±r.

'0' a setlenirse hiÄŒ beklemez. BÄ¼tÄ¼n belirtilen tanÄ±mlayÄ±cÄ±lar test edilir ve ÄŒaÄŒrÄ±dan hemen ÄŒÄ±kÄ±lÄ±r. Bu, select iÄŒinde blocklanmadan birden fazla tanÄ±mlayÄ±cÄ±yÄ± test etmek iÄŒin kullanÄ±lÄ±r.

0'dan bÄ¼yÄ¼k bir deÄŒere setlenirse o deÄŒer kadar bekler. Belirtilen tanÄ±mlayÄ±cÄ±lardan biri hazÄ±r olduÄŒunda veya zamanaÄŒÄ±mÄ±na uÄŒradÄ±ÄŒÄ±nda sistem ÄŒaÄŒrÄ±sÄ± return yapar.

GÄŒizlenecek tanÄ±mlayÄ±cÄ±lar tanÄ±mlayÄ±cÄ± setine (descriptor set) eklenir. TanımlayÄ±cÄ± seti, fd\_set veri yapÄ±sÄ± ÄŒeklinde kayÄ±t edilmiÄŒ veridir. fd\_set, her tanÄ±mlayÄ±cÄ± iÄŒin bir bit tutar ve geÄŒerli boyutu 1024 bittir (sys/types.h iÄŒinde tanÄ±mlÄ±). Set Ä¼zerinde iÄŒlem yapmak iÄŒin bazÄ± makrolar tanÄ±mlanmÄ±ÄŒtÄ±r:

fd\_set aÄŒaÄŒÄ±daki gibi tanÄ±mlanÄ±r:

```
fd_set dset;
```

Seti sÄ±fÄ±rlamak iÄŒin:

```
FD_ZERO(&dset);
```

Takip etmek istediÄŒimiz tanÄ±mlayÄ±cÄ±larÄ± eklemek iÄŒin:

# Genel: SÄ¼reÄŒlerin HaberleÄŒmesi: Soket Programlama

```
FD_SET(fd, &dset);
```

select() seti deÄŒiÄŒtirmektedir. select sistem ÄŒaÄŒrÄ±sÄ±ndan sonra bir tanÄ±mlayÄ±cÄ±nÄ±n hala sette olup olmadÄ±ÄŒÄ±nÄ± test etmek iÄŒin:

```
if (FD_ISSET(fd, &dset)) {  
    ...  
}
```

Bir tanÄ±mlayÄ±cÄ±yÄ± setten ÄŒÄ±kartmak iÄŒin:

```
FD_CLR(fd, &dset);
```

Bu bilgiler Ä±ÄŒÄ±nda bir uygulama geliÄŒtirebiliriz. Standart giriÄŒ (stdin), dosya tanÄ±mlayÄ±cÄ±sÄ± 0 olan bir dosyadÄ±r. EÄŒer bir PC kullanÄ±yorsanÄ±z klavye standart giriÄŒtir. AÄŒaÄŒÄ±daki kod, standart giriÄŒi izlemektedir. EÄŒer standart giriÄŒ okumak iÄŒin hazÄ±rsa (bunun anlamÄ± klavyeden birÄŒey yazÄ±p enter'a basmÄ±ÄŒsak) hazÄ±r olduÄŒunu ekrana basacak. FD\_ISSET ile de verinin hazÄ±r olduÄŒunu gÄŒireÄŒiz (HazÄ±r olduÄŒunda FD\_ISSET, true olacaktÄ±r. EÄŒer hazÄ±r deÄŒilse false olacaktÄ±r.).

```
/*  
 * select.c - I/O multiplexing  
 *  
 * Baris Simsek, <simsek at acikkod org>  
 * http://www.acikkod.org  
 * 07/07/2004  
 *  
 */  
  
#include <stdio.h>  
  
#include <sys/time.h>  
#include <sys/types.h>  
#include <unistd.h>  
  
int  
main(void) {  
    fd_set dset;  
    struct timeval tv;  
    int ret;  
  
    FD_ZERO(&dset);  
    FD_SET(0, &dset); /* stdin'i gozlemeye aldik */
```

# Genel: SÄ¼reÄŒlerin HaberleÄŒmesi: Soket Programlama

```
tv.tv_sec = 10; /* 10 sn giris icin bekle */
tv.tv_usec = 0;

ret = select(1, &dset, NULL, NULL, &tv);

if (ret == -1)
    perror("select()");
else if (ret) {
    printf("Standart input okumak iÄŒsin hazır.
");
    if(FD_ISSET(0, &dset))
        printf("Standart input icin dset true
");
    }
else {
    printf("10 saniye icinde standart giristen veri girilmedi.
");
    if(!FD_ISSET(0, &dset))
        printf("Standart input icin dset false
");
    }
return 0;
}
```

## 7. SÄ¼k Sorulan Sorular

### 7.1. Kitap ÄŒnerebilir misiniz?

W. Richard Stevens'Ä±n "[UNIX Network Programming - Volume 1](#)" kitabÄ± bu alanda baÄŒlÄ±ca referans kitaptÄ±r.

### 7.2. Soketler NasÄ±l ÄŒalÄ±ÄŒr?

Soketler (ÄŒzelikle connection oriented soketler) dosyalar veya [PIPE](#) gibi ÄŒalÄ±ÄŒr. Pipe'dan farkÄ± iki yÄŒnlÄ¼ olmasÄ±dÄ±r. Dosyalardan farkÄ± ise beklediÄŒiniz kadar veri okuyamayabilir veya istediÄŒiniz kadar veri yazamayabilirsiniz.

### 7.3. select() veri hazÄ±r dediÄŒi halde, 0 byte neden okunur?

select() verinin hazÄ±r olduÄŒunu sÄŒyledikten sonra karÄŒÄ± taraf baÄŒlantÄ±yÄ± koparmÄ±ÄŒtÄ±r. Bu da read() 'in 0 dÄŒndÄ¼mesine neden olur.

## 7.4. Soket seÄŒeneklerini nasÄ¼l deÄŒiyiÄŒtiririm?

```
int setsockopt(int s, int level, int optname, const void *optval, socklen_t optlen);
```

```
int flag = 1;
int result = setsockopt(sock, /* socket affected */
                        IPPROTO_TCP, /* seÄŒeneÄŒyi TCP seviyesinde set et. */
                        TCP_NODELAY, /* seÄŒenek */
                        (char *) &flag,
                        sizeof(int)); /* seÄŒenek deÄŒerinin bÄ¼yÄ¼klÄ¼ÄŒÄ¼ */
```

SeÄŒenekler hakkÄ¼nda detay iÄŒin "man 2 setsockopt"

## 7.5. SO\_KEEPALIVE seÄŒeneÄŒyi neden kullanÄ¼lÄ¼r?

Oturum aÄŒmÄ¼ÄŒ iki bilgisayar arasÄ¼nda belli bir sÄ¼re (RFC1122 de 2 saat olarak tanÄ¼mlandÄ¼) veri alÄ¼ÄŒveriÄ¼yi olmazsa, karÄ¼Ä¼ tarafÄ¼ yoklamak iÄŒin (ACK isteyerek) kullanÄ¼lÄ¼r. Zira karÄ¼Ä¼ taraf ulaÄ¼Ä¼lamaz durumda olabilir. Bu durumu algÄ¼lamak iÄŒin kullanÄ¼lÄ¼r.

## 7.6. Soketi tam olarak nasÄ¼l kapatÄ¼rÄ¼m?

Soket kapatÄ¼lÄ¼ktan sonra "netstat -na" ile bakÄ¼ldÄ¼ÄŒÄ¼nda sisteminizde TIME\_WAIT'de duran soketler hala gÄ¼zÄ¼kebilir. Bu normaldir. Ä¼Ä¼nkÄ¼ TCP, bÄ¼tÄ¼n verinin karÄ¼Ä¼lÄ¼klÄ¼ transfer edildiÄ¼yinden emin olmak istiyor. Soket kapatÄ¼ldÄ¼ÄŒÄ¼nda her iki taraf da baÄ¼ka veri transferi olmayacaÄ¼Ä¼ konusunda anlaÄ¼mÄ¼ÄŒ demektir. BÄ¼yle bir anlaÄ¼madan sonra socket rahatlaÄ¼la kapatÄ¼labilir. Ancak burada iki problem sÄ¼z konusu olacak. Bunlardan biri son ACK'in ulaÄ¼Ä¼p ulaÄ¼madÄ¼ÄŒÄ¼ bilinemeyecek. (Bu ACK iÄŒin tekrar ACK istense bunun da ulaÄ¼Ä¼p ulaÄ¼madÄ¼ÄŒÄ¼ belli olmayacak. KÄ¼sÄ¼r dÄ¼ngÄ¼ olur.). EÄ¼er bu ACK aÄ¼da kaybolmuÄ¼ ise sunucu bunu bekliyor olacaktÄ¼r. Bir diÄ¼er problem de eÄ¼er bir paket router'in birinde herhangi bir nedenle bekliyorsa ve alÄ¼cÄ¼ taraf bunu belli bir sÄ¼re iÄŒerisinde alamamÄ¼ÄŒsa paketi yeniden talep edecektir. Ancak diÄ¼er paket gerÄŒekte kaybolmamÄ¼ÄŒtÄ¼r ve belli bir sÄ¼re aÄ¼da yeniden ortaya ÄŒÄ¼kacaktÄ¼r. Bu kaybolma ve yeniden ortaya ÄŒÄ¼kmasÄ¼ sÄ¼resi iÄŒerisinde baÄ¼lantÄ¼ koparsa ve aynÄ¼ host aynÄ¼ porttan yeni bir baÄ¼lantÄ¼ aÄŒarsa gÄ¼ndereceÄ¼yi paketin sÄ¼ra numarasÄ¼ aÄ¼daki ile Ä¼st Ä¼ste

# Genel: SÄ¼reÄŒlerin HaberleÄŒmesi: Soket Programlama

binecektir. Ä±Ä¼nkÄ¼ eski oturumdan kalma bir paket yeni oturumda transfer edilmiÄŒtir. Bundan kurtulmak iÄŒin TIME\_WAIT durumu ortadan kalkmadan yeni bir oturum aÄŒmamalÄ±.

BÄ¼tÄ¼n bunlar dÄ¼ÄŒÄ¼nde TIME\_WAIT'in programcÄ± iÄŒin bir yardÄ±mcÄ± olduÄŒu anlaÄŒılır. Ancak TIME\_WAIT olduÄŒu sÄ¼rece programÄ±nÄ±z aynÄ± soketi yeniden bind() edemeyecektir. 7.4. anlatÄ±ldÄ¼ÄŒÄ± ÄŒekilde SO\_REUSEADDR seÄŒeneÄŒi set edilerek bu sorunu ÄŒÄŒzebilirsiniz. Ä±te yandan TIME\_WAIT'te bekleyen soketler bir sÄ¼re sonra (Linux'lerde bu 60 sn.dir.) close() olacaktÄ±r. sysctl ile bu sÄ¼re deÄŒiÄŒtirilebilir.

close() doÄŒru kapatma yÄŒntemi ise de shutdown() daha kullanÄ±lÄ±dÄ±r. Ä±Ä¼nkÄ¼ tek yÄŒnliÄ¼ soketi kapama olanaÄŒÄ± da sunar.

```
int shutdown(int s, int how);
```

Ä°kinci parametre ile kapa yÄŒntimÄ¼ verebilirsiniz:

SHUT\_RD: Veri alÄ±mÄ± kesilecektir.

SHUT\_WR: Veri gÄŒnderimi kapatÄ±lacaktÄ±r.

SHUT\_RDWR: Ä°ki yÄŒnliÄ¼ veri alÄ±ÄŒveriÄŒi durdurulacaktÄ±r. (close)

close(), o sÄ¼reÄŒ iÄŒin soketi kapatÄ±r ancak eÄŒer socketi baÄŒka bir sÄ¼reÄŒle paylaÄŒıyorsa socket hala aÄŒÄ±k duracaktÄ±r. shutdown() bÄ¼tÄ¼n sÄ¼reÄŒler iÄŒin soketi kapatÄ±r.

## 7.7. String halindeki bir adresi internet adresine nasÄ±l ÄŒevirim?

```
struct in_addr *atoaddr(char *address) {
    struct hostent *host;
    static struct in_addr saddr;

    /* Ä±nce IP formatÄ±nda deniyoruz. */
    saddr.s_addr = inet_addr(address);
    if (saddr.s_addr != -1) {
        return &saddr;
    }
}
```

## Genel: SÃ¼reÃ§lerin HaberleÄ±mesi: Soket Programlama

```
/* IP formatÄ±nda deÄ±yilse FQDN olarak deniyoruz. */
host = gethostbyname(address);
if (host != NULL) {
    return (struct in_addr *) *host->h_addr_list;
}
return NULL;
}
```

### 7.8. Soketlerde dinamik buffer kullanmanÄ±n bir yolu var mÄ±?

Soketten okuyacaÄ±nÄ±z veri miktarÄ± belli olmadÄ±Ä±nda bÃ¼yle bir ihtiyaÅ± doÄ±uyor. Bu durumda malloc() ile mÄ±mkÄ±n olan en bÃ¼yÃ¼k tampon belleÄ±i ayÄ±rsÄ±nÄ±z. Okunan verinin bÃ¼yÃ¼klÃ¼Ä±ne gÃ¼re tampon bellek realloc() ile yeniden boyutlandÄ±rÄ±lÄ±r. Zaten pek Å§ok UNIX'de malloc() fiziksel bellekten yer ayÄ±rmaz. Sadece adres uzayÄ±nÄ± belirler. Tampona veri yazdÄ±Ä±nda gerÅ§ek bellek sayfalarÄ± kullanÄ±lÄ±r. Bu nedenle bÃ¼yÃ¼k buffer ayÄ±rmakla gereksiz kaynak kullanÄ±mÄ±na neden olunmaz.

### 7.9. "address already in use" hatasÄ±nÄ± neden alÄ±rÄ±m?

Port kullanÄ±lyordur veya sunucu bir programÄ± henÃ¼z sonlandÄ±rdÄ±nÄ±z ve socket TIME\_WAIT'dedir. Ä±kinci durum iÅ§in 7.4. ve 7.5. sorularÄ±nÄ±n Å§Ã¼zÃ¼mlerine bakÄ±nÄ±z. Birinci durum iÅ§in aynÄ± portta Å§alÄ±Å±yan diÄ±er socket programÄ± durdurmanÄ±z gerekmektedir.

### 7.10. ProgramÄ±mÄ± nasÄ±l daemon yapabilirim?

En kolay yolu inetd ile kullanmanÄ±zdÄ±r. DiÄ±er bir yÃ¼ntem ise fork() ederek isteklere cevap vermektir. Detay icin <http://www.enderunix.org/docs/daemontr.html> Programın temel iskeleti asagidaki yapiya cevirilmeli:

```
ret = fork ();
if (ret == -1) { /* fork hata verdi */
    perror ("fork()");
    exit (3);
}
if (ret > 0) exit(0); /* Ana sÃ¼reÃ§ Å§Ã¼kar */
if (ret == 0) { /* Alt sÃ¼reÃ§ devam eder */
    close (STDIN_FILENO);
    close (STDOUT_FILENO);
    close (STDERR_FILENO);
    if (setsid () == -1) exit(1);
    /* Alt sÃ¼rece ait iÃ¼ler */
}
```

}

## 7.11. AynÄ± anda birden fazla soketi nasÄ±l dinlerim?

select() kullanÄ±n. Hangi socket veri iÄŒin hazÄ±r ise onun, kullanmanÄ±za olanak saÄŒlar. [6.2.](#) select() bÄŒIÄ¼mÄ¼ne bakÄ±nÄ±z.

## 7.12. 1024 ten kÄ¼ÄŒÄ¼k portlarÄ± neden bind edemiyorum?

GÄ¼venlik nedenleri ile 1024'ten kÄ¼ÄŒÄ¼k portlarÄ± yalnÄ±zca yetkili kullanÄ±cÄ± (root) aÄŒabilir.

## 8. Belge TariÄŒesi

VI.sÄ¼rÄ¼mde "7. SÄ±k Sorulan Sorular" bÄŒIÄ¼mÄ¼ eklendi. (1 AÄŒustos 2004)

V.sÄ¼rÄ¼mde Soket TÄ¼rleri ve Veri DÄŒnÄ¼ÄŒmleri baÄŒIÄ±klarÄ± eklendi. Sarmalama(Encapsulation) tanÄ±mÄ± eklendi. (8 Temmuz 2004)

IV.sÄ¼rÄ¼mde GeliÄŒmiÄŒ G/ÄŒ baÄŒIÄ±ÄŒÄ± eklendi. (28 Haziran 2004)

III.sÄ¼rÄ¼mde RFC'ler gÄŒzden geÄŒirilerek yeniden dÄ¼zenlenmiŒtir. Ä°lk sÄ¼rÄ¼mlerdeki bazÄ± yanlÄ±ÄŒlar giderildi. (5 Haziran 2004)

II.sÄ¼rÄ¼m, Linux KullanÄ±cÄ±larÄ± DerneÄŒi Liste Ä¼yeleri iÄŒin yeniden gÄŒzden geÄŒirildi. (2001)

Ä°lk sÄ¼rÄ¼m: 10 KasÄ±m 1998 (KTÄœ Bilgisayar KlubÄ¼ Dergisi iÄŒin yazÄ±ldÄ±.)

HatalarÄ± ÄŒekinmeden bana bildirebilirsiniz. b\$

Yazar: Erkan Kaplan

Son GÄ¼ncelleme: 2005-02-22 02:15