Soket Programlama

1. Giriş

Soketler iki yönlü haberleşme aygıtlarıdır. Aynı makinedeki farklı süreçler arasında ya da uzak makinelerdeki süreçlerin haberleşmesinde kullanılabilir. Uzak makinelerdeki süreçler arasında da haberleşmeyi olanaklı kılan tek yöntemdir.

Soket oluşturulacağı zaman, üç parametrenin belirtilmesi gerekir: *haberleşme stili, namespace* ve *protokol*. *Haberleşme stili* soketin iletilen veriye nasıl davranacağını belirler. Aynı zamanda taraflar arasındaki paket iletiminin nasıl gerçekleştirileceğini belirler. *Namespace*, soket adreslerinin nasıl yazılacağını belirler. Soket adresi soket bağlantısının bir ucunu işaretler. Örneğin lokal namespace için soket adresleri dosya isimleri iken, internet namespace için soket adresi ağa bağlı olan uzak makinenin IP adresi ve port numarasıdır. Son parametre olan *protokol* verinin nasıl iletileceğini belirler.

Soket programlamada yer alan temel metodlar;

- `socket` – soket oluşturur.
- `close` – soketi yok eder.
- `connect` – iki soket arasında bağlantı oluşturur.
- `bind` – sunucu soketini bir adres ile etiketler.
- `listen` – bir soketi durumları bekleyecek şekilde konfigüre eder.
- `accept` – bağlantıyı kabul eder ve bu bağlantı için yeni bir soket oluşturur.

Soketleri oluşturma ve yok etme: `socket` ve `close` fonksiyonları soketleri oluşturur ve yok eder. Bir soket oluştururken üç parametre bildirilir, namespace, haberleşme stili, ve protokol.

- Namespace parametresi için `PF_` (Protocol Families'in kısaltılması) öneki ile başlayan sabitler kullanılır. `PF_LOCAL` veya `PF_UNIX` lokal namespace'i tanımlarken, `PF_INET` Internet namespace'inin kullanılacağını gösterir.

- Haberleşme stili parametresi için `SOCK_` öneki ile başlayan sabitler kullanılır. `SOCK_STREAM` TCP tabanlı haberleşme stilini tanımlarken, `SOCK_DGRAM` UDP tabanlı haberleşme stilini ifade eder.
- Üçüncü parametre protokol ise veriyi iletme ve almak için düşük seviyeli mekanizmayı belirtir. Genellikle 0 olarak kullanımı tercih edilir. `soket` çağrısının başarılı olması durumunda soket için olan bir dosya tanımlayıcısı geri döner. `read`, `write` komutları kullanılarak sokete yazılabilir ya da okunabilir. Soket 'in kullanımı sona erdiğinde ise `close` komutu ile yok edilir.

Bağlantının çağırılması:
İki soket arasında bağlantının oluşturulabilmesi için client tarafında `connect` fonksiyonunun kullanılması gerekir. Bu fonksiyon hangi server soketin adres bilgisini belirtir. Client bağlantının ilklendirmesini gerçekleştirirken, server bağlantıları kabul etmek için beklemededir.

2. Lokal Soketler

Aynı bilgisayardaki süreçleri bağlayan soketler, `PF_LOCAL` ve `PF_UNIX` sabitleri ile beraber lokal namespace'i kullanırlar. Bunlar lokal soketler yada UNIX domen soketleri olarak adlandırılırlar. Dosya isimleri ile belirtilen soket adresleri bağlantılar oluşturulacağı zaman kullanılırlar.

Soketin ismi sockaddr_in yapısında belirtilir. Sun_family alanı `AF_LOCAL` olarak tanımlanır. Bu sabit değer soketin lokal olduğunu gösterir. Sun_path alanı kullanılacak olan dosya ismidir. Sockaddr_in yapısının boyu `SUN_LEN` makrosu kullanılarak bulunabilir. Herhangi bir dosya ismi soket oluşturmada kullanılabilir. Ancak sürecin dizine yazma hakkının olması gerekir. Bir sokete bağlanılabilmesi için, sürecin dosya için okuma hakkına sahip olması gerekir. Farklı bilgisayarlardaki süreçler aynı dosya sistemini paylaşsalar dahi, yalnızca aynı makinedeki süreçler lokal namespace soketleri yardımı ile aralarında haberleşebilir. Lokal namespace için kullanılabilecek protokol 0 değeri ile gösterilir.
İki program kullanılarak soket konusuna örnek verilecektir. Sunucu programı lokal namespace soketi oluşturur ve bağlantılar için dinlemeye başlar. Bir bağlantı talebi aldığında, bağlantıdan gelen mesajı okur ve bağlantı kapanıncaya kadar yazar. Eğer gelen mesajlardan biri "quit" dir, sunucu programı soketi kaldırır ve bağlantıyı sonlandırır. Socket-server programı soket dosyasının yerini parametre olarak komut satırından almaktadır.

3.Uygulma

Deney kapsamında TCP soket haberleşme kullanarak basit bir istemci-sunucu uygulaması gerçekleştirilecektir. Bunun için

Uyulama kaba kodu;

Client tarafı;

1- *Soketin oluşturulması*. `socket` metodu kullanılarak oluşturulacaktır. Namespace, haberleşme stili, ve protokol parametreleri kullanılarak çağrısı yapılmalıdır. Lokal namespace,

```
socket (PF_LOCAL, SOCK_STREAM, 0);
```

2- *Soketin adres bilgisinin alınması.*

```
name.sun_family = AF_LOCAL;
strcpy (name.sun_path, socket_name);
```

3-*Soket bağlantısı kur.* `connect` metodu kullanılarak bağlantı sağlanacaktır.

```
connect (socket_fd,(struct sockaddr *) &name, SUN_LEN
(&name));
```

4- *Komut satırındaki mesajı sokete yaz.*

write_text (socket_fd, message);

5- *Bağlantıyı sonlandır.* `close` metodu kullanarak gerçekleştirilecektir.

```
close (socket_fd);
```

Server tarafı;

Temel işlem: Soketten metnin okunması ve ekrana yazılması gerçekleştirilecektir. Soket kapanana kadar devam edecek. Client'dan "quit" mesajı geldiği durumda 0 döndürecek, diğer durumlarda 0 olmayan bir değer döndürecek şekilde kodlamasını gerçekleştiriniz.

1-Soketin oluşturulması

```
socket_fd = socket (PF_LOCAL, SOCK_STREAM, 0);
```

2-Soketin server soket olduğunun belirtilmesi.

```
  name.sun_family = AF_LOCAL;
  strcpy (name.sun_path, socket_name);
  bind (socket_fd,(struct sockaddr *) &name, SUN_LEN
(&name));
```

3-Bağlantıların dinlenmesi

```
  listen (socket_fd, 5);
```

4-Bir Client dan "quit" mesajı gelene kadar serverın bağlantıları kabul etmesi.

```
    struct sockaddr_un client_name;
    socklen_t client_name_len;
    int client_socket_fd;
    int addrlen;
```

Tanımlarına göre

- Bağlantının kabul edilmesi

```
client_socket_fd = accept (socket_fd,(struct sockaddr
*) &client_name, &addrlen);
```

5- Soketin kapatılması.

```
 close (socket_fd);
 unlink (socket_name);
```