BLGM 344 – DENEY 4 *
SUNUCU PROGRAMLAMA

Amaçlar

1. Hata işleme
2. TCP sunucusuyla ilgili pratik yapma

Not: Bu deney, Deney 3'ün devamıdır. Gerektiğinde bir önceki deney dökümanını okuyunuz.

Hata Ayıklama

POSIX sistem çağrılarında, hatalı bir durumla karşılaşılması durumunda negatif bir değer geri döndürülür. Böylece bir hata olduğunu anlayabiliriz. Statik olarak tanımlanmış **errno** değeriyle hatanın koduna ulaşabilirsiniz. Bu hata kodları kullandığımız sistemler içerisinde tanımlıdır. Örneğin, **connect()** çağrısı bağlantı hatası yaşadığında **errno** değerine **ECONNREFUSED** sabitini atayabilir. Biz de bu değeri test ederek hedefteki sunucuya erişemediğimizi anlayabiliriz. Bunun dışında, **perror()** kütüphane fonksiuonu da hata çıktığında kullanılabilir. Bu fonksiyon bizim verdiğimiz bir yazıyla birlikte sistem tarafından tespit edilen hatayı yazdıracaktır. Örneğin:

```
int sonuc;
sonuc = connect(soket, &sunucu, sizeof(struct sockaddr_in));
if(sonuc<0)
{
   perror("Bağlantı hatası");
   exit(0);
}
```

fork()

Birden fazla işlemin aynı anda yapılmasını sağlamak için **fork()** sistem çağrısı kullanılabilir. **fork()** sistem çağrısını kullanarak herhangi bir işlem (ana işlem) bir ya da daha fazla alt işlem oluşturabilir. Her alt işlem, ana işlemin, **fork()** sistem çağırısından önceki açtığı tüm dosyalara erişebilir.

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

Oluşturulan alt işlem, ana işlemin sahip olduğu tüm dosya haklarına da sahip olur. **fork()** çağrısından önceki tüm değişkenler, **fork()** işlemi anındaki değerleriyle birlikte yeni oluşturulan alt işleme kopyalanır. **fork()** sistem çağrısının ağ programlamadaki temel kullanımı birden fazla istemciye aynı anda cevap vermektir. Yeni oluşturulan alt işlem gelen istekle ilgilenirken, ana işlem yeni gelecek olan çağrılarla ilgilenebilir. Genel kullanımı aşağıdaki gibidir:

```
if(fork()==0) //yeni işlemde yapılacak olan işlem
{
     .......
     exit (0);
}
```

Deney 1

Bu deneyde Deney 3'de yazmış olduğumuz programa ek olarak hata kontrolleri ekleyeceğiz. Her yapılan sistem çağrısının sonucunu kontrol ederek, hata çıkması durumunda, **perror()** aracılığıyla ortaya çıkan hatayı yazdırınız.

Ek çalışma: Eğer **connect()** çağrısı hata verirse programınızın bağlanılacak olan sunucu ve kapı numarasını kullanıcıdan sormasını sağlayınız.

Deney 2

Bu deneyde basit bir sunucu yazmanız istenmektedir. Bu sunucu gelen bağlantıları kabul etmeli. Bağlantı sağlandığında, sunucu istemciye "Adınız nedir?" yazısını göndermeli ve gelen cevabı beklemeli. Daha sonra gelen cevaba göre, "Merhaba *isim*." şeklinde cevap göndermelidir. Bu programın testini yaparken, daha önce yazmış olduğunuz istemciyi kullanabilirsiniz. Bu kısımda yapmanız gereken işlemler aşağıdaki gibi olmalıdır.

1. Soket oluştur (**socket()**) ve yerel bağlantısını yap (**bind()**)
2. Dinlemeye başla (**listen()**)
3. Sonuz döngüye girerek (while(1))
    1. "Bağlantı için bekliyor." , mesajını yazdır (printf())
    2. Bağlantı kabul et (**accept()**)
    3. Yeni bir işlem oluştur (**fork()**) ve yeni işlemde
        1. Veri gönder (**send()**)
        2. Veri al (**recv()**)
        3. Yeni mesajı oluştur (sprintf())
        4. Mesajı gönder (**send()**)
        5. Yeni bağlantıyı kapat (**close()**)
        6. Çık (return)

Sorular

1. connect() sistem çağrısı hangi hataları verebilir? (man connect)
2. bind() sistem çağrısı hangi hataları verebilir?
3. recv() sistem çağrısı hangi hataları verebilir?
4. Yazmış olduğumuz sunucu **aynı anda** birden fazla istemciye cevap verebilir mi? Deneyiniz.
5. Yazdığımız sunucunun birden fazla istemciye (art arda) cevap vermesini nasıl sağladık?