BLG312 – Bilgisayar İşletim Sistemleri, Bahar 2017

ÖDEV-1

Emre KÖSE

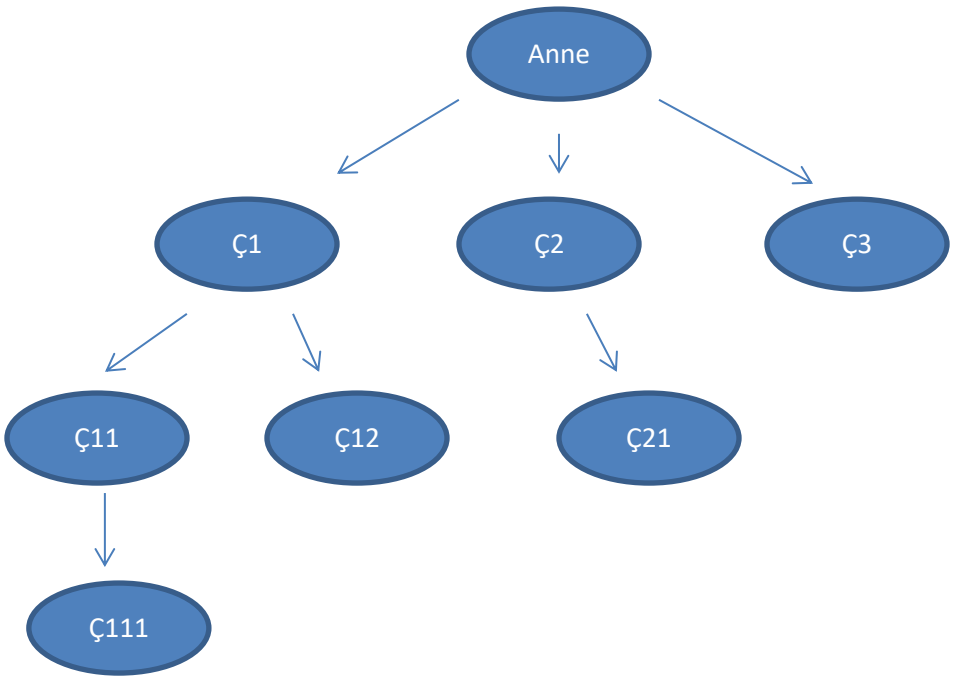
150130037

Soru 1 : Aşağıda yer alan kod parçasını inceleyin, yürütün ve verilen soruları yanıtlayın.

```
 main()
{
int i = 1;
int sonuc = 0;
for (i = 1; i <= 3; i++)
{ sonuc = fork();
if (sonuc == 0)
{
printf("Cocuk proses %d. \n",i);
 }
else {
printf("Anne proses (i=%d). \n",i); } } }
```

a) Kaç tane proses vardır? Bunlardan kaçı anne proses ve kaçı çocuk prosestir?

Kod incelendiğinde kodu içeren başlangıç programı aslında bir prosestir. Bu prosesten üretilen 3 çocuk proses olacaktır. Bu üç çocuk prosesten ilki(Ç1) iki defa fork yaparak 2 ayrı çocuk proses daha oluşturur(Ç11,Ç12). Ç11 in i değeri ikidir bu nedenle o da bir fork daha kullanarak Ç111 çocuk prosesini oluşturur. İkinci(Ç2) çocuk prosesin i değeri 2 ile başlar, bu nedenle o bir defa fork çağrısı kullanarak bir çocuk proses oluşturmuş olur(Ç21). İlk prosesin sonuncu çocuğunun(Ç3) i değeri üçtür bu nedenle o herhangi bir fork çağrısı yapmadan sonlanır.



Bu proseslerden Ç1, Ç11, Ç2 prosesleri çocuk proses olarak doğsalarda, sonrasında fork fonksiyonunu kullanarak anne proses konumuna geçmişlerdir. Sonuç olarak programda toplamda 4 adet anne 4 adet de çocuk proses olmak üzere 8 proses oluşmuş olur.

b) Fork çağrısı kaç kere yürütülecektir? Printf çağrısı kaç kere yürütülecektir?

Yukarıdaki şekilde görüldüğü gibi toplamda 7 adet ok vardır. Bu oklar aslında kullanılan fork çağrıları sonucu oluşan çoçukları göstermektedir. Toplamda ise 7 defa fork çağrısı kullanılmış olduğu anlaşılmaktadır.

Printf fonksiyonun kaç kere kullanıldığını sonuç değeri döndürecek olan fork çağrısını yapanın anne prosesin mi yoksa bu fork çağrısı sonucu oluşan çocuk prosesin mi olacağını kontrol etmek yeterli olacaktır. Anne proseste bu değer sıfırdan büyük, çocuk proseste sıfıra eşit olacaktır. Sonuç olarak ise toplamda 7 defa "`printf("Cocuk proses %d. \n",i);`" 7 defa da "`printf("Anne proses (i=%d). \n",i);`" fonksiyonu çalışır. Toplamda ise 14 defa printf fonksiyonu çalışmış olur.

c) Program nasıl bir çıkış üretecektir, örnek verin? Farklı çıkışlar üretilebilir mi, neden?

ÖRNEK1

```
night@night-VirtualBox:~/Desktop$ gcc deneme.c -o odev
night@night-VirtualBox:~/Desktop$ ./odev
Anne proses (i=1).
Anne proses (i=2).
Cocuk proses 1.
Cocuk proses 2.
Anne proses (i=3).
Anne proses (i=2).
Anne proses (i=3).
Anne proses (i=3).
Cocuk proses 3.
Cocuk proses 3.
Cocuk proses 2.
Anne proses (i=3).
Cocuk proses 3.
Cocuk proses 3.
```

ÖRNEK2

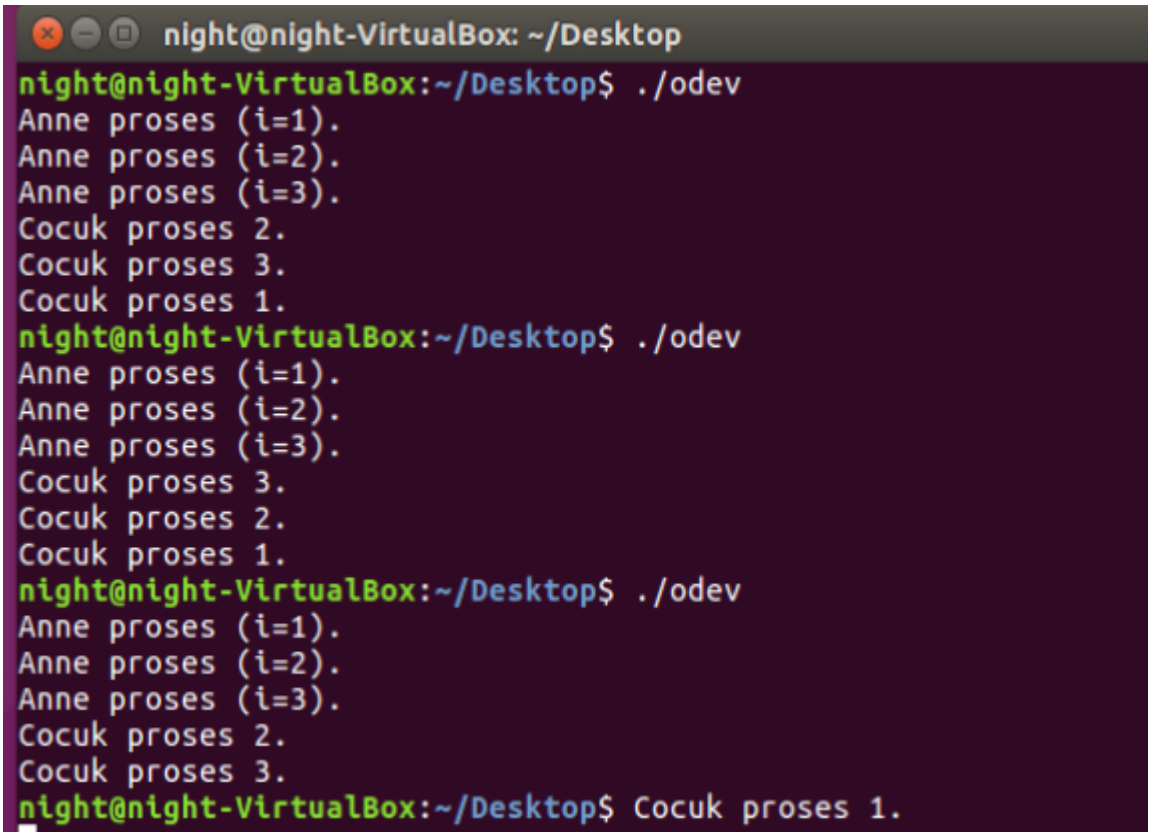
```
night@night-VirtualBox:~/Desktop$ ./odev
Anne proses (i=1).
Anne proses (i=2).
Cocuk proses 1.
Anne proses (i=3).
Anne proses (i=2).
Anne proses (i=3).
Cocuk proses 3.
Cocuk proses 2.
Cocuk proses 3.
Anne proses (i=3).
Cocuk proses 2.
Anne proses (i=3).
Cocuk proses 3.
night@night-VirtualBox:~/Desktop$ Cocuk proses 3.
```

Yukarıda görülen örneklerde anne proses (i=1) den sonra çıktılar hakkında herhangi bir tahminde yapmak mantıksız olacaktır. İşletim sistemleri kendi uygun gördüğü prosesi önce çalıştırdığından, çıktıların sırası hakkında yorum yapamayız. Sadece bu çıktılarda ve denediklerim anne proses (i=1) ile başlayıp, toplamda 14 kez printf fonksiyonu ve de önceki soruda anlatıldığı üzere bunların 7 sinin anne 7 sinin çocuk proseste kullanıldığı görülmüştür.

d) Verilen kodu, toplam 4 proses (1 anne ve 3 çocuk) üretmesini sağlayacak şekilde değiştirin ve ürettiği çıkışı ekrana basarak gösterin.

Burada istenen, elimizdeki kodda oluşan çocuk proseslerin anne proses haline dönüşmesini engellemek olacaktır. Bu da 'sonuc' değişkeninde 0 olarak dönecek çocuk proseslerde if yapısının içine return 0; komutunu eklemekle yapılabilir.

```
if (sonuc == 0)
{
printf("Cocuk proses %d. \n",i);
return 0;  // eklendi
 }
```



Program üç defa derlenmiştir. Toplamda 6 adet printf fonksiyonu kullanıldığını görüyoruz. Her fork çağrısında tek olan anne i nin 1, 2 ve 3 değerleri için ve her fork çağrısında çocuk proseslerde sonuc = 0 ile çalışan 3 adet printf fonksiyonun farklı sıralarda çalıştıklarını görmekteyiz.