Bilgisayar İşletim Sistemleri
Ödev - I

Ad-Soyad	: Gökhan Karaca
Numara	: 040070234
Teslim Tarihi	: 14.03.2011
Öğretim Görevlisi	: Şima Uyar

Soru 1-)

Soruda istenilenlere göre prog1.c isimli program çalıştırıldığında aşağıdaki çıktıyı alıyoruz.

```
karaca@Karaca:~/Masaüstü/odev$ gcc prog1.c
karaca@Karaca:~/Masaüstü/odev$ ./a.out
Anne: Proses Kimlik Numaram: 2212
Anne: Cocugumun Proses Kimlik Numarasi: 2213
   Cocuk: Proses Kimlik numaram: 2213
   Cocuk: Annemin proses kimlik numarasi 2212
Anne: Sonlaniyorum...
```

Programı inceleyecek olursak: Program çalıştığında anne için çalışıyor olacak ve fork() komutu ile bir çocuk process yaratacak.fork() ile çocuk process yaratıldığı için geri gelen değer 0 olacak(f=fork() => f=0 olacak).İlk başta anne process için f>0 olduğu için else bloğuna gireceğiz ve bu else bloğundaki kodlar çalışmaya başlayacak.Else bloğundaki iki printf komutu ile anne process in kimliği ve çocuğunun kimlikleri ekrana bastırılacak ve sonra wait satırına gelindiğinde çocuk process e geçecek ve f=0 olması nedeniyle else if(f=0) bloğuna giriş yapılacak ve çocuk process e ilişkin bilgiler ekrana basılacak.Sonrasında tekrar anne process işlemine geri dönüldü ve Sonlaniyorum... mesajı ekrana basıldı.

Anne process in kimlik numarası 2212. Çocuğunun yaratılması anneden sonra olduğu için çocuk bir sonraki numarayı 2213 ü aldı(2213a boşta olduğu için).

Soru 2-)

Anne process in annesinin kimlik numarasını da ekrana yazmak için aşağıdaki kodu else bloğunun içine ekledik ve aşağıdaki ekran görüntüsünü elde ettik.

Eklenen kod satırı = *printf("Anne-Anne: Proses Kimlik Numaram: %d \n", getppid());*

```
karaca@Karaca:~/Masaüstü/odev$ gcc prog1.c
karaca@Karaca:~/Masaüstü/odev$ ./a.out
Anne-Anne: Proses Kimlik Numaram: 2103
Anne: Proses Kimlik Numaram: 2738
Anne: Cocugumun Proses Kimlik Numarasi: 2739
   Cocuk: Proses Kimlik numaram: 2739
   Cocuk: Annemin proses kimlik numarasi 2738
Anne: Sonlaniyorum...
```

Ekrana çıkan bu numara programı başlatan process in kimlik numarasıdır.Yani biz prog1.c yi koşturmadan önce var olan ve ./a.out komutunu yazmadan önce var olan ve biz ./a.out komutunu yazınca prog1 i çalıştıran process in kimliği.2103 numaralı process Unix terminaline aittir.Bu process prog1.c yi koşturmuştur.Bu yüzden onun annesidir ve bu kimlik

numarası terminale aittir.Yeni bir terminal ekranı açsaydık ve yine bu programı koştursaydık,2103 yerine başka bir process numarası gelecekti ama yine terminale ait process numarası olacaktı.

Soru 3-)

prog1.c den Wait(Null); satırını kaldırınca aşağıdaki ekran görüntüsü elde edildi.

```
karaca@Karaca:~/Masaüstü/odev$ gcc prog1.c
karaca@Karaca:~/Masaüstü/odev$ ./a.out
Anne: Proses Kimlik Numaram: 2979
Anne: Cocugumun Proses Kimlik Numarasi: 2980
Anne: Sonlaniyorum...
   Cocuk: Proses Kimlik numaram: 2980
karaca@Karaca:~/Masaüstü/odev$    Cocuk: Annemin proses kimlik numarasi 1
```

Kaldırmış olduğumuz wait satırı anne process çalışırken çalışan bloğun içerisindeydi.Anne process koştuğunda çocuk process oluşmuş olacak ve paralel çalışacaklar fakat wait(Null) satırı kaldırıldığı için anne process çocuk process in bitmesini beklemeden sonlanacak.Çocuk process tüm işlemlerini bitirmeden annesi sonlandığı için annesinin kimlik bilgisini kaybetti.

Soru 4-)

Kod üzerinde aşağıdaki düzenleme yapıldı ve sonraki resimdeki görüntü elde edildi.

```
else if (f==0)
  {
     printf("    Cocuk: Proses Kimlik numaram: %d \n", getpid());
     printf("    Cocuk: Global degiskenin degeri : %d   , bu degeri 2 ile
degistiriyorum \n",Global);
     Global=2;
     printf("    Cocuk: Global degiskenin degeri : %d \n",Global);
     sleep(1);
     printf("    Cocuk: Annemin proses kimlik numarasi %d \n", getppid());
     printf("    Cocuk: Global degiskenin degeri : %d \n",Global);
     exit(0);
  }
  else
  {
     printf("Anne: Proses Kimlik Numaram: %d \n", getpid());
     printf("Anne: Cocugumun Proses Kimlik Numarasi: %d \n", f);
     printf("Anne: Global degiskenin degeri : %d   , bu degeri 1 ile
degistiriyorum \n",Global);
     Global=1;
```

```
    printf("Anne: Global degiskenin degeri : %d \n",Global);
    wait(NULL);
    printf("Anne: Sonlaniyorum...\n");
    printf("Anne: Global degiskenin degeri : %d \n",Global);
    exit(0);
}
```

```
Anne: Proses Kimlik Numaram: 3281
Anne: Cocugumun Proses Kimlik Numarasi: 3282
Anne: Global degiskenin degeri : 0  , bu degeri 1 ile degistiriyorum
Anne: Global degiskenin degeri : 1
   Cocuk: Proses Kimlik numaram: 3282
   Cocuk: Global degiskenin degeri : 0  , bu degeri 2 ile degistiriyorum
   Cocuk: Global degiskenin degeri : 2
   Cocuk: Annemin proses kimlik numarasi 3281
   Cocuk: Global degiskenin degeri : 2
Anne: Sonlaniyorum...
Anne: Global degiskenin degeri : 1
```

Değiştirmiş olduğumuz global değişkenin değeri fork işlem çağrısından önceki haliyle anne ve çocuk process lere gitti.Sonrasında yapılan değişiklikler anne ve çocuk için farklı kaldı.Birbirleriyle bağlantısı koptu bu global değerin.Anne ve çocuk process ler faklı adres ve veri uzayını kullandıkları için bağlantıları yok.Bunu anne sonlandıktan sonra ekrana global değerini 1 olarak yazmasından anlıyoruz.

Soru 5-)

Global değişkenimizi malloc fonksiyonunu kullanarak tanımladığımızda.Yani değişkenizmiz için hafızayı kendimiz aldıktan sonra programımızı çalıştırırsak aşağıdaki ekran görüntüsünü aldık.

```
karaca@Karaca:~/Masaüstü/odev$ gcc prog1.c
karaca@Karaca:~/Masaüstü/odev$ ./a.out
Anne: Proses Kimlik Numaram: 1890
Anne: Cocugumun Proses Kimlik Numarasi: 1891
Anne: Global degiskenin degeri : 0  , bu degeri 1 ile degistiriyorum
   Anne: Global adresi : 0x804a03c
Anne: Global degiskenin degeri : 1
   Cocuk: Proses Kimlik numaram: 1891
   Cocuk: Global degiskenin degeri : 0  , bu degeri 2 ile degistiriyorum
   Cocuk: Global adresi : 0x804a03c
   Cocuk: Global degiskenin degeri : 2
   Cocuk: Annemin proses kimlik numarasi 1890
   Cocuk: Global degiskenin degeri : 2
Anne: Sonlaniyorum...
Anne: Global degiskenin degeri : 1
```

Çıktıları değerlendirecek olursak; anne process imiz ve çocuk process imiz global değişken olarak aynı bellek gözünü kullanmışlardır.Bir önceki

soruda anne ve çocuk processlerimiz birbirinden farklı değerlerde global değişkenlerine sahiplerdi.Bu programda da farklı değerlere sahipler fakat aynı bellek gözü üzerinde işlem yapmaktalar.

Soru 6-)

➔p1.c yi çalıştırdıktan sonra çıkan ekran görüntüsü şu şekildedir.

```
karaca@Karaca:~/Masaüstü/odev$ ./a.out
0: Value= 1
Main: Created 3 procs.
2: Value= 1
karaca@Karaca:~/Masaüstü/odev$ 1: Value= 1
```

1 numaralı değerin program sonlandıktan sonra çıkmasının nedeni anne processin çocuklarını beklememesidir.Çıktıları yorumlayacak olursak: For döngüsü içerisinde toplamda 3 tane çocuk process yaratılmıştır ve bu çocuk processleri içerisinde my_function isimli fonksiyon çağrılmıştır.Bu fonksiyon içerisinde global temp değişkeninin değeri 1 artılmaya çalışılmıştır.Fakat çocuk process ler oluşturulurken temp değerinin ilk değerini kendi space lerine kaydetmişlerdir ve 1 artırma işlemini hep başlangıç değeri üzerinden kendi spacelerinde işleme sokmuşlardır.Bu yüzden sürekli 0+1 işleminin sonucu olan 1 ekrana yazılmıştır.

➔p2.c yi çalıştırdıktan sonra çıkan ekran görüntüsü şu şekildedir.

```
karaca@Karaca:~/Masaüstü/odev$ ./cikti
main(): Created 3 threads.
0: Value= 1
1: Value= 2
2: Value= 3
```

Çıktıları yorumlayacak olursak: For döngüsü içerisinde toplamda 3 tane iplik yaratılmıştır ve bu ipliklerin yaratılması sırasında my_function isimli fonksiyon gönderilmiştir.Bu fonksiyon içerisinde global temp değişkeninin değeri 1 artılmaya çalışılmıştır.p1.c in aksine bu sefer temp in değeri 1.2.3 şeklinde artmıştır bunun nedeni ipliklerin aynı adres ve bellek uzayını kullanmalarıdır.Her iplik my_function fonksiyonuna girdiğinde aynı temp değişkeni üzerinde işlem yapacak ve bir önceki ipliğin artırdığı değerin üzerine +1 ilave edecek.Toplamda 3 iplik olduğu için temp değişkeninin son değeri 3 olacak.