BLG312 – Bilgisayar İşletim Sistemleri

ÖDEV-2

Emre KÖSE

150130037

Proseslerin oluşturulması:

Problem olarak durumu incelediğimizde prosesler arasında ortak alan paylaşımının kullanılması ve bu proseslerin ana prosesin altında iki çocuk proses ile yapılması isteniyor.

Bu nedenle döngü içinde prosesler oluşturma aşamasında yani fork aşamasında döngü sonucunda oluşan çocukların bir daha fork yapmaması için her çocuk proseste programın devam ettirilmemesi "break" ile döngüden çıkılması sağlanmıştır.

```
for(i=0; i<2; i++){
order = i;
f = fork();

if(f == -1){
    printf("Error in Fork \n");
    exit(1);
}
if(f == 0){
    break;
}

}
```

Prosesler için ortak kullanım yaratma:

Programda karışık olarak dizi üretilmesi ve bu dizinin bellirli bir değerden küçük olanları ile büyük olanları yanı sıra dizinin eleman sayısı, M değeri, x ve y değerlerinin de bu ayrılan alan içine yazılması istenmiştir. Bu nedenlerden ötürü aslında 4 * sizeof(int) boyutu sadece n, M, x ve y değerlerini integer olarak tutabilmek için gereklidir. Sonrasında diziyi ve parçalanan dizileri de tutabilmek için 2*n*sizeof(int) kadar bir miktarda alan gereklidir. Tüm bu alanlar proseslerin kullanabilmesi için ana proses içinde aşağıdaki gibi ayrılmıştır.

```
memory_id1 = shmget(KEYSHM1, sizeof(int)*(2*n+4),0700|IPC_CREAT);
ptr_n = (int *)shmat(memory_id1, 0, 0);
```

Prosesler arası senkronizasyon:

Problemin çözümünde alt dizilerin oluşturulması olayını çocuk prosesler tarafından yürütülmesi istenmiştir. Proseslerin fork kullanıldıktan sonra hangisinin daha önce kullanılacağı bilinmediğinden, 1.çocuğun küçük olan dizi elemanlarını bulmadan önce 2.çocuğun çalıştırılmaması istenmiştir ve son olarak da anne proses ortak memory alanını ayırmadan ve de diziyi oluşturmadan önce çoçuk proseslerin çalışmaması gerektiği için öncelikli anne prosesin belli bir zamana kadar ilk olarak çalışması zorunludur. (NOT: İkinci çocuk proses ile birinci çocuk proses arasında senkronizasyon olması gerektiği ancak birincisinin önce çalışmak zorunda olmadığı düşüncesindeyim. Örneğin, önce 2.çocuk çalışırsa 2n + 4 boyutundaki adresten direk şu adrese gidilebilir: n + 4 + n - y). Bu nedenlerden ötürü 3 farklı semafor yapısı kullanılarak çözüme ulaşılmıştır.

Birinci semafor değeri senk_var oluşturulduktan sonra sem_wait ile değeri 2 azaltılarak çocuk proseslerin görevini tamamladıklarında bu değeri birer artırarak anne prosesin görevinin son kısmını yapması için kullanılmıştır. Amacı anne prosesin çocukların bittiğinden haberdar olduktan sonra ayrılan alanların ve semafor pointerlarının serbest bırakma işleminin gerçekleştirilmesidir. senk_var3 semaforu ise anne prosesin ortak alanı ayırmadan çocukların çalışması durumunu önlemek amacıyla kullanılmıştır. Yani amaç önce anne prosesin çalışmasını sağlamaktır.

```
senk_var = semget(Semafor_K1,1,0700|IPC_CREAT); // çocukların bittiğinin bildirimini al
semctl(senk_var,0,SETVAL,0);



senk_var3 = semget(Semafor_K3,1,0700|IPC_CREAT); // anne proses yer ayırma işlemini yap
semctl(senk_var2,0,SETVAL,0);
```

senk_var2 semaforu ise 1.çocuk bitmeden ikincisinin çalışması durumunu önlemek amacıyla 1.çocuk prosesinin içinde oluşturulmuş ve kullanılmıştır. Amaç ise iki çocuk arasındaki senkronizasyondur.

```
  senk_var2 = semget(Semafor_K2,1,0700|IPC_CREAT);   //   1.çocuk bitmed
  semctl(senk_var2,0,SETVAL,0);
```

Semafor işaretçilerinin iadesi ve paylaşılan bellek alanlarının boşaltılması:

Çocuk prosesler sıralama görevlerini tamamladıktan sonra senk_var semafor değeri ile bittiklerini anne prosese bildirirler anne ise programın artık görevinin tamamlandığını anlar ve de ayrılan bu dinamik belleğin iade işlemini gerçekleştirir.

```
shmdt(ptr_n);

semctl(senk_var, 0, IPC_RMID, 0);
semctl(senk_var2, 0, IPC_RMID, 0);
semctl(senk_var3, 0, IPC_RMID, 0);

shmctl(memory_id1, IPC_RMID, 0);
```