BİLGİSAYAR İŞLETİM SİSTEMLERİ

III. ÖDEVİ

UNIX işletim sisteminde C ile SEMAFOR ve ORTAK BELLEK ALANI KULLANIMI

Beycan Kahraman
040020337
Yrd. Doç. Dr. A. Şima UYAR
14.04.2006

AMAÇ :

UNIX işletim sisteminde C programlama dili ile Semafor ve Ortak Bellek Alanı Kullanımının Öğrenilmesi

SİSTEM:

Programımı hazırlarken TUBITAK ve UEKAE tarafından geliştirilen PARDUS Linux işletim sisteminde, GCC 3.4.4 derleyicisini kullandım.

PROGRAM:

Uygulamada gördüğümüz örneklerle büyük oranda benzerlik taşıyan bu ödevde; semafor uygulamasından farklı olarak çocukların içinde öncelikle, son durumdaki i değerini öğrendik. Daha sonra da bu numaraya karşılık düşen bellek gözündeki değerleri sıralayıp, sonsem semafor değerini bir azaltarak anne programa geri döndük. Aşağıda çocuk proses için çalıştırılan kod görülmektedir.

```
        pause();
        sonsem = semget(SEMKEY, 1,0);
        pba = shmget ( KEYSHM, sizeof(int)*COCUK_SAYISI*SAYI,0 );
        dizi = (int *) shmat ( pba, 0, 0 );

// i. BÖLÜMÜ SIRALAMA
        printf("    %d. Çocuk : 'Sıralıyorum.'\n", i+1 );
        mergeSort ( SAYI*i, SAYI*(i+1)-1, dizi );
// ----------------

        sem_signal(sonsem,1);
```

Anne program çocuk programa göre daha kapsamlı olmakla beraber, burada öncelikle sonsem semaforunu oluşturduk. Böylece çocuklar kendi bölgelerinin sıralamasını bitirmeden anne dizileri birleştirmeye asla kalkmayacaktır. Görüldüğü üzere bu amacı gerçekleştirmek için en az bir semafor kullanmamız gerekmektedir.

Yine benzer şekilde, ortak bellek alanı oluşturulduktan sonra, buraya bir işaretçi atanıp dizideki tüm değerler rasgele olarak değiştirilmektedir. Böylece çocuk proseslerin yaptığı değişiklikler daha açık olarak görülebilecektir.

Daha sonra çocuk prosesler uyandırılıp, bunların ölmesi beklenir. Çocuk proseslerin sonlandırılmasını anlamak için önceden kullanıcılar için ayrılmış Sinyal12() fonksiyonunu kullandık.

Tüm çocuklar sonlandırıldığında ise sonsem semafor değeri tekrar sıfıra düştüğünden anne proses çalışmasına devam edebilmektedir. Bu durumda öncelikle çocuklardan gelen ve grup olarak sıralanmış diziyi ekrana bastırdık. Böylece çalışan çocuk proseslerin istenenen işlemi uygun şekilde gerçekleyip gerçeklemediği görülmüş oldu.

Daha sonra da, bu oluşturulan 5 farklı sıralanmış diziye 4 kez MERGE işlemi uygulanarak istenen sıralanmış büyük dizi elde edilmiş oldu. Buralarda kullanılan sıralama işlemlerinde Mergesort() ve yardımcı Merge() fonksiyonlarından yaralanılmıştır.

İstenen dizi elde edildikten sonra tekrar ekrana basılmakta ve oluşan ilk dizi ile karşılaştırılabilmektedir.

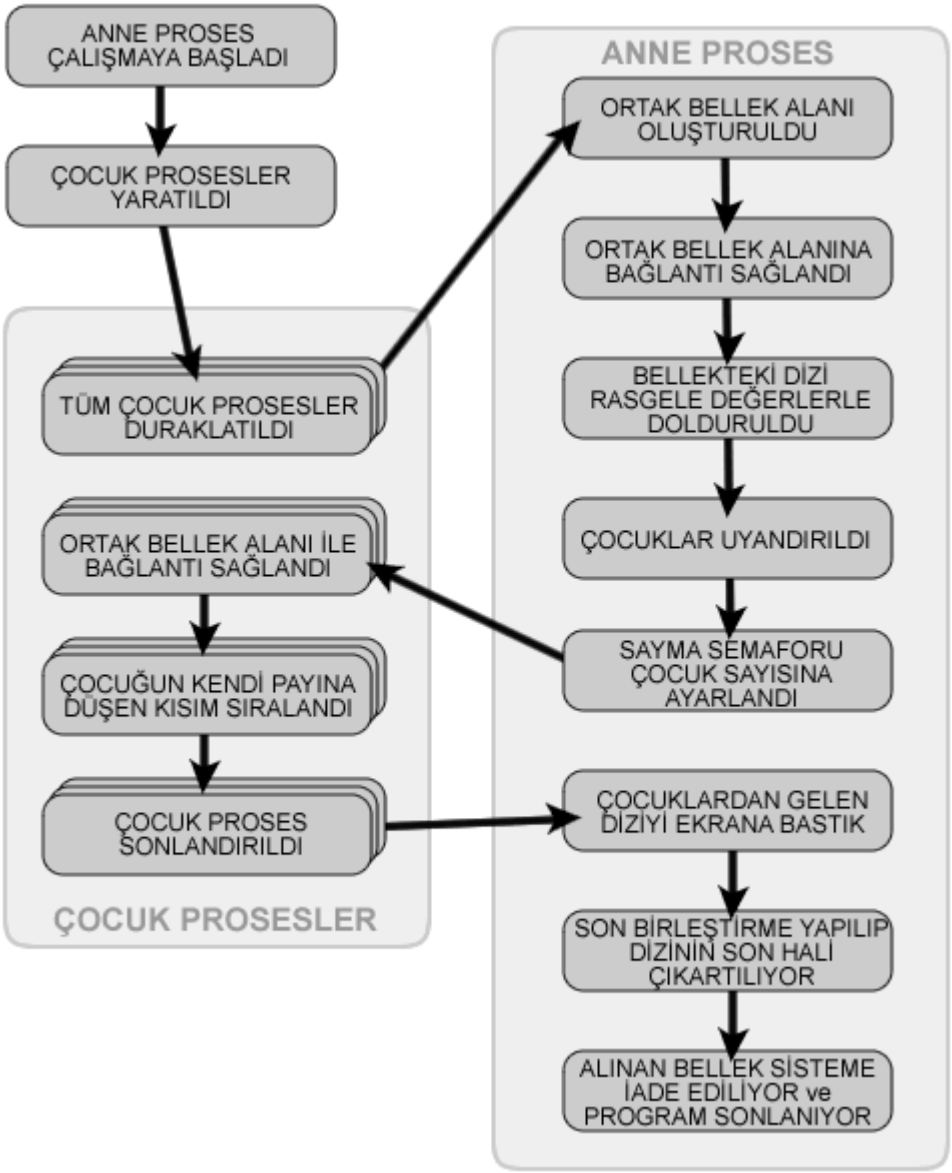
Programın son adımında ise alının ortak bellek alanı ve semafor sisteme iade edilmekte ve programdan çıkılmaktadır. Anne için hazırlanan kod parçası aşağıdaki şekilde verilmiştir.

```
        sonsem = semget ( SEMKEY, 1, 0700|IPC_CREAT );
        semctl ( sonsem, 0, SETVAL, 0 );

// DİZİ OLUŞTURMA
        printf("    Anne     : 'Diziyi Oluşturuyorum.'\n", i+1);
        pba = shmget ( KEYSHM, sizeof(int)*COCUK_SAYISI*SAYI, 0700|IPC_CREAT );
        dizi = (int *) shmat ( pba, 0, 0 );

        for ( i=0; i<COCUK_SAYISI*SAYI; ++i )
            dizi[i] = rand()%MAX_RANDOM;
// ----------------

        sleep(1);

        printf("    Anne     : 'Çocukları Oluşturuyorum.'\n", i+1);
        for ( i=0; i<COCUK_SAYISI; ++i )
            kill(cocuk[i], 12);

        sem_wait(sonsem,COCUK_SAYISI);
        printf("    Anne     : 'Çocuklar Öldü.'\n", i+1);

// EKRANA YAZMA 1 ( Çocuklardan gelen hali )
        printf("    Anne     : 'Çocuklardan Dönen Hali:'\n", i+1);
        for ( i=0; i<COCUK_SAYISI*SAYI; ++i ){
            printf ( "%d ", dizi[i] );
            if ( i%SAYI == SAYI-1 )
                printf ( "\n" );
        }
// ----------------

// DİZİYİ BİRLEŞTİRME
        printf("    Anne     : 'Birleştiriyorum.'\n", i+1);
        for ( i=1; i<COCUK_SAYISI; ++i )
            merge ( 0, i*SAYI-1, (i+1)*SAYI-1, dizi );
// ----------------

// EKRANA YAZMA II ( Sıralanmış Hali )
        printf("    Anne     : 'Sıralanmış Hali:'\n", i+1);
        for ( i=0; i<COCUK_SAYISI*SAYI; ++i ){
            printf ( "%d ", dizi[i] );
            if ( i%SAYI == SAYI-1 )
                printf ( "\n" );
        }
// ----------------

        printf("    Anne     : 'Çıkıyorum.'\n", i+1);
        shmctl(pba,IPC_RMID,0);
        semctl(sonsem,0,IPC_RMID,0);
```

Bunların dışında kullanılan Mergesort ve Merge fonksiyonlarına program kodundan ulaşılabilir.

AKIŞ DİYAGRAMI:

Hazırladığım programın akış diyagramı aşağıdaki gibi çizilebilir.



SONUÇ:

Ortak bellek alanı ve semaforları kullanarak prosesleri hatasız ve daha etkili şekilde kullanmayı amaçladığımız bu ödevde istenenleri uygun şekilde derleyip çalıştırabildim. Bilgisayarda kendi hazırladığımız bu ödev ile, bu konudaki bilgilerimiz pekişmiş oldu. Bu ödev de diğer iki ödevde olduğu gibi konuyu anlamamızda büyük yarar sağladı.