

Matris Çarpımı

Ege Üniversitesi Bilgisayar Mühendisliği
İşletim Sistemleri 2

Ödev 2

Umut BENZER 05-06-7670 <http://www.ubenzer.com/>

0.1
20 Mayıs 2010



Sorunun Cevabı

Processler, kendi adres sahalarına sahiptir (vfork hariç) ve bu yüzden birbirlerinden tamamen bağımsızdırlar. Birinde oluşan bir hata, bellek alanı paylaşılmadığından dolayı diğerini etkilemez. Ancak tüm processlerin birbirinden bağımsız olması bazı dezavantajları da beraberinde getirir. Örneğin iki process'in birbiriyle erişebilmesi için pipe veya benzeri mekanizmalar gerekir ki, bunları kurmak ve yönetmek program yazarı için bir külfetken, bu mekanizmalar kernel seviyesine indiği için aynı zamanda birer yavaşlama sebebidir.

Threadler ise aynı bellek bölgesinin elemanlarıdır. Bu yüzden (isteğe bağlı olarak) bazı değişkenler threadler arasında paylaşılabilir. Bunun getirdiği birkaç problem vardır. En başta senkronizasyon sorunu gelir. Bu sorun mutex ya da benzer yapılarla halledilebilir. Bir başka sorun, hata ayıklamanın zorluğudur. Bir threadde oluşabilecek bir hata, aynı bellek bölgesinde çalışıklarından ve izole olmadıklarından başka bir threadi etkileyebilir.

Multiprocessing ve çok işlemcili, çok çekirdekli ortamlarda eş zamanlı işletim konusunda çok processli yazılımlar ile çok threadli yazılımlar arasında bir işletim farkı bulunmamaktadır.

Eğer kod yazımım esnasında fark ettiklerime gelirse, multi thread yazılım geliştirmek, multi process yazılıma göre daha kolay. Paylaşılan bellek yönetimi, mutexleri ne zaman kullanacağımızı iyi bildikten sonra hiç problem oluşturmuyor.

Kaynak Kod

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <errno.h>

/* Mutex ve cond deęişkenleri */
pthread_mutex_t condition_mutex = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t condition_cond = PTHREAD_COND_INITIALIZER;

pthread_mutex_t creating_threads_mutex = PTHREAD_MUTEX_INITIALIZER;
int creating_threads = 0;

int *sharedMatrisA;
int *sharedMatrisB;
int *sharedMatrisC;
int m;
int n;
int k;

void *matrisHesapla(int satirno) {

    printf("Thread #%d init.\n",satirno+1);
    fflush(stdout);

    /* Hala thread yaratma asamasındaysak bekle. */
    if(creating_threads > 0) {
        printf("Thread #%d digerlerinin yaratilmasini beklemek uzere kitlenecek.\n",satirno+1);
        fflush(stdout);
        /* Yaratılınca bir bekle önce. */
        pthread_mutex_lock( &condition_mutex );
        printf("Thread #%d mutex locked.\n",satirno+1);
        fflush(stdout);
        pthread_cond_wait( &condition_cond, &condition_mutex );
        printf("Thread #%d cond wait ended.\n",satirno+1);
        fflush(stdout);
        pthread_mutex_unlock( &condition_mutex );

        printf("Thread #%d calismaya devam ediyor.\n",satirno+1);
        fflush(stdout);
    } else {
        pthread_mutex_unlock(&creating_threads_mutex);
        printf("Thread #%d beklemeye f&#305;rsat bulamadan tüm threadler ya-
rat&#305;lm&#305;&#351;. \n",satirno+1);
        fflush(stdout);
    }

    printf("Thread #%d hesaplamaya basladi.\n",satirno+1);
    int ic1,ic2;
    for(ic1=0; ic1<k; ic1++) {
        for(ic2=0; ic2<n; ic2++) {
            /* Her thread kendi satırına yazdığı için problem olmaz, mutexe gerek yok. */
            sharedMatrisC[(satirno*k) + ic1] += sharedMatrisA[(satirno*n) + ic2] *
sharedMatrisB[(ic2*k)+ic1];
        }
    }
    printf("Thread #%d hesaplamayı bitirdi ve sonlandı.\n",satirno+1);
}

int main() {

    /* Acilis yazilari BASLANGICI */
    printf("Umut BENZER\n");
    printf("05-06-7670\n");
    printf("Ege Universitesi Bilgisayar Muhendisligi 3. Sinif\n");
    printf("http://www.ubenzer.com\n");
    printf("Threads\n\n");
    /* Acilis yazilari SONU */

    FILE *fp;
```

```

if((fp=fopen("input.txt","r")) == NULL) {
    printf ("Dosya acilamadi.");
    exit(-1);
}

fscanf(fp,"%d %d %d",&m, &n, &k);

int matrisA[m][n];
int matrisB[n][k];
int matrisC[m][k];
sharedMatrisA = &matrisA;
sharedMatrisB = &matrisB;
sharedMatrisC = &matrisC;

int i;
int j;

for(i=0;i<m;i++) {
    for(j=0;j<k;j++) {
        matrisC[i][j] = 0;
    }
}

/* A MATRISINI OKU */
for(i=0;i<m;i++) {
    for(j=0;j<n;j++) {
        fscanf(fp,"%d",&matrisA[i][j]);
    }
}

/* B MATRISINI OKU */
for(i=0;i<n;i++) {
    for(j=0;j<k;j++) {
        fscanf(fp,"%d",&matrisB[i][j]);
    }
}

fclose(fp);

/* EKRANA MATRISLERI YAZDIR */
printf("Matris A:\n");
for(i=0;i<m;i++) {
    for(j=0;j<n;j++) {
        printf("%d\t", matrisA[i][j]);
    }
    printf("\n");
}

printf("\nMatris B:\n");
for(i=0;i<n;i++) {
    for(j=0;j<k;j++) {
        printf("%d\t", matrisB[i][j]);
    }
    printf("\n");
}

printf("\n");
pthread_t threads[m];
fflush(stdout);

creating_theads = 1;
printf("Threadler yarat&#305;l&#305;yor...\n");
fflush(stdout);
for(i=0;i<m;i++) {
    printf("Thread #%d/%d yarat&#305;ld&#305;.\n",i+1,m);
    fflush(stdout);
    pthread_create(&threads[i], NULL, &matrisHesapla, i);
}

pthread_mutex_lock(&creating_theads_mutex);
creating_theads = 0;
pthread_mutex_unlock(&creating_theads_mutex);

```

```

printf("Thread yaratimi tamamlamdi. Simdi beklemelerin bitmesi icin mesaj gonderilecek.\n");
fflush(stdout);

/* Threadler çalışabilirler */
pthread_mutex_lock( &condition_mutex );
pthread_cond_broadcast( &condition_cond );
pthread_mutex_unlock( &condition_mutex );

printf("Beklemede olabilecek threadlere isleme baslamasini soyleyen mesaj gonderildi.\n");

fflush(stdout);
for(i=0;i<m;i++) {
    printf("Main thread #%d/%d'nin bitmesini bekliyor...\n",i+1,m);
    pthread_join(threads[i],NULL);
}

printf("\nMatris C:\n");
for(i=0;i<m;i++) {
    for(j=0;j<k;j++) {
        printf("%d\t", matrisC[i][j]);
    }
    printf("\n");
}

printf("\n");
printf("...ve bize ayrilan surenin sonuna geldik...\n");
printf("...bir sonraki odevde gorusmek uzere, esen kalin efendim...\n");
exit(0);
}

```

Örnek Matris Dosyası

20 13 2

```

11 9 9 8 8 1 3 4 5 3 2 4 2 1 2 3 2 5
8 7 8 7 6 6 5 5 4 3 2 1 3 4
6 5 9 9 0 9 9 8 6 5 6 7 4 3
8 7 6 5 4 3 2 2 3 3 5 5 6 7 8 9
10 9 9 8 8 1 3 4 5 3 2 4 2 1 2 3 2 5
8 7 8 7 6 6 5 5 4 3 2 1 3 4
6 5 9 9 0 9 9 8 6 5 6 7 4 3
8 7 6 5 4 3 2 2 3 3 5 5 6 7 8 9
10 9 9 8 8 1 3 4 5 3 2 4 2 1 2 3 2 5
8 7 8 7 6 6 5 5 4 3 2 1 3 4
6 5 9 9 0 9 9 8 6 5 6 7 4 3
8 7 6 5 4 3 2 2 3 3 5 5 6 7 8 9
10 9 9 8 8 1 3 4 5 3 2 4 2 1 2 3 2 5
8 7 8 7 6 6 5 5 4 3 2 1 3 4
6 5 9 9 0 9 9 8 6 5 6 7 4 3
8 7 6 5 4 3 2 2 3 3 5 5 6 7 8 9

```

```

6 7
1 2
2 2 2
2 3 1
2 1
1 2 2
4 5
6 7
9 5
3 2 1
1 1
0 0
1 2 1 2

```