T.C.

19 MAYIS ÜNİVERSİTESİ

SİSTEM PROGRAMLAMA DERSİ 1.PROJE ÖDEVİ

HAZIRLAYAN:

20060358 ZEYNEP SILA KAYMAK

PROJENİN GİTHUB LİNKİ: https://github.com/zeynepsila/sistem-programlama-proje-odevi

PROJE: Multi-threaded Matrix Multiplication.

PROBLEM: A ve B matrislerinin çarpımı olan C matrisinin threadler kullanılarak hesaplanması amaçlanmıştır.

PLATFORM VE DİL:

Proje yapılırken VirtualBox sanal makine üzerinden UBUNTU kullanılmıştır. Bu projenin tüm kısımları C dilinde yazılmıştır.

THREAD NEDİR?

Bir process'in birden fazla işi aynı anda yapmasını sağlayan yapılara thread denir. Bir process bünyesinde bir ya da birden fazla thread barındırabilir. Thread'ler aynı anda sadece tek bir iş yapabilir. Kısaca N adet thread N adet iş yapabilir diyebiliriz. Thread'ler aynı zamanda ligth-weight process (hafif siklet proses) olarak da nitelendirilebilir. Thread kavramı Türkçe'deki en uygun karşılığı iş parçacıklarıdır. Bir process içerisinde birden fazla thread çalıştırılmasına ise multi-threading denir.

KODLARIN AÇIKLAMASI VE KULLANILAN METOTLAR:

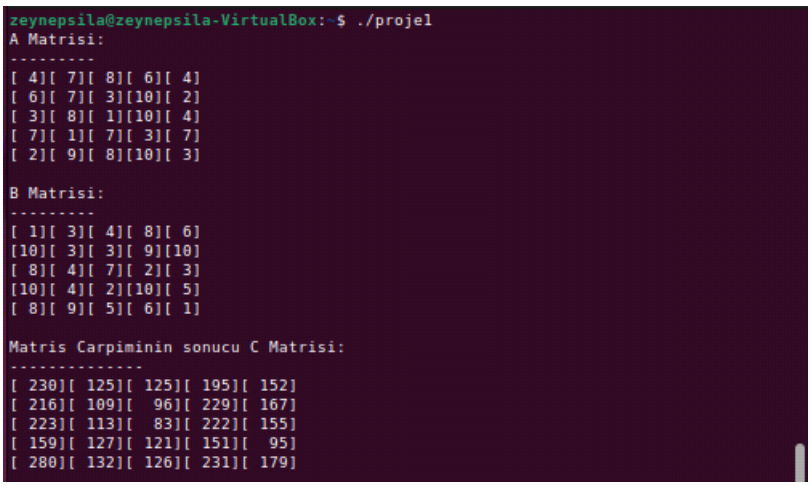
- #include <stdio.h>: stdio.h dosyası giriş ve çıkış işlemlerini yerine getiren fonksiyonları barındıran bir kütüphane dosyasıdır. printf() fonksiyonunu kullanabilmek için koda eklenmiştir.
- #include <stdlib.h>: stdlib.h başlık dosyasında, dinamik bellek yönetimi, rasgele sayı üretimi, çevre ile iletişim, tamsayı aritmetiği, arama, sıralama ve dönüştürme gibi çeşitli genel amaçlı fonksiyonlar tanımlanmıştır.
- #include <time.h>: Zaman bilgilerini depolamak için kullanılan yapıdır. İşlemcinin bir saniyedeki tik sayısını gösterir.

- #include <sys/types.h>: typedef ile kendi türlerimizi tanımlamamız için koda eklenmiştir.
- #include <unistd.h>: dosyası, POSIX standardıyla uyumluluk için gerekli tanımların bulunduğu dosyaları içerir.
- #include <string.h>: Karakter dizisi kütüphanesi (<string.h>) karakter dizileriyle alakalı pek çok kullanışlı fonksiyonu bünyesinde bulundurur. Bunlardan başlıcaları karakter dizisi kopyalama, karakter dizisi birleştirme, karakter dizilerini karşılaştırma gibi fonksiyonlardır.
- #include <pthread.h>: Bilgisayar bilimlerinde geçen lif (thread, iplik, sicim) kavramının C dili ile kodlanabilmesi için genellikle UNIX türevi işletim sistemelerinde geliştirilen programlama kütüphanesidir.
- fill_matrix() metodu: Verilen matrise 1'den 10'a kadar rastgele bir sayıyla doldurmaya yarayan fonksiyon.
- print_matrix() metodu: Verilen matrisi yazdırmaya yarayan fonksiyon.
- matrix_page metodu: Verilen matrisi mmap() kullanarak bir belleğe eşler.
- matrix_unmap(): Verilen matrisin bellekten eşlemesini kaldırır.
- __attribute__ ((noreturn)) void row_multiply(void *row_args): Verilen satır için tüm indeksleri hesaplamaya yarar. ((noreturn)) ise derleyiciye bu fonksiyonun herhangi bir dönüşü olmayacağını bildirmek için eklenir.
- pthread_exit(0);: işlevi çağıran diziyi sonlandırır. pthread_exit() işlevi çağırana geri dönemez .

MAİN METOTUNUN AÇIKLANIŞI:

Main metodunda öncelikle matrislerin bellek boyutu hesaplanır. Ardından matrix_a, matrix_b ve matrix_c bir belleğe eşlenir. fill_matrix() fonksiyonu ile her iki matrisin içi de 1-10 arası rastgele sayılarla doldurulur ve iki matris de yazdırılır. Program önce thread verileri için malloc() fonksiyonu ile bellekten yer ayırır. pthread_create() ile matris oluşturulur ve pthread_join() ile her threadin yaptığı iş toplanır. Matris çarpımının sonucu yani C matrisi yazdırılır ve bellekten ayrılan alanlar geri verilir.

PROJEYE AİT ÇIKTI GÖRÜNTÜLERİ:



PROJEYE AİT KOD GÖRÜNTÜLERİ:

```c
/*
                Multi-threaded Matrix Multiplication
*/

#include <stdio.h> // giris cikis islemlerinin yapilmasini saglar
#include <stdlib.h> //dinamik bellek yönetimi,rastgele sayi üretimi vb fonksiyonlarin yapilmasini saglar
#include <time.h> // zaman bilgilerini depolamak icin kullanilir
#include <sys/types.h> //typedef ile kendi türlerimizi tanimlamamizi saglar
#include <unistd.h> //POSIX standartıyla uyumluluk icin gerekli tanimlari icerir
#include <sys/mman.h>
#include <string.h> //karakter dizileriyle alakali fonksiyonlari icerir
#include <pthread.h> //thread kavraminin c ile kodlanabilmesini saglar

// matris boyutlarini kontrol eder
#define MATRIX_SIZE 5
 // normal bir diziyi iki boyutluymus gibi ele alir
#define array(arr, i, j) arr[(int) MATRIX_SIZE * (int) i + (int) j]

void fill_matrix(int *matrix);
void print_matrix(int *matrix, int print_width);
int *matrix_page(int *matrix, unsigned long m_size);
void matrix_unmap(int *matrix, unsigned long m_size);
__attribute__ ((noreturn)) void row_multiply(void *row_args);

```

```c
// tüm matrisler icin integer isaretcileri tanimlanir
static int *matrix_a, *matrix_b, *matrix_c;

//her thread icin bagimsiz degisken yapisi
typedef struct arg_struct
{
  int *a;
  int *b;
  int *c;
  int row;
} thr_args;

//verilen matri 1'den 10'a kadar rastgele integer sayilar ile doldurur
void fill_matrix(int *matrix)
{
  for (int i = 0; i < MATRIX_SIZE; i++)
  {
    for (int j = 0; j < MATRIX_SIZE; j++)
    {
      array(matrix, i, j) = rand() % 10 + 1;
    }
  }
  return;
}

//verilen matrisi yazdirir
void print_matrix(int *matrix, int print_width)
{
  for (int i = 0; i < MATRIX_SIZE; i++)
  {
    for (int j = 0; j <  MATRIX_SIZE; j++)
    {
      printf("[%*d]", print_width, array(matrix, i, j));
    }
    printf("\n");
  }
  printf("\n");
  return;
}

//verilen matrisi mmap() kullanarak bir bellek sayfasina esler
int *matrix_page(int *matrix, unsigned long m_size)
{
  matrix = mmap(0, m_size, PROT_READ | PROT_WRITE,
    MAP_SHARED | MAP_ANONYMOUS, -1, 0);
  // eger mmap() basarisizca cikar
  if (matrix == (void *) -1)
  {
    exit(EXIT_FAILURE);
  }
  memset((void *) matrix, 0, m_size);
  return matrix;
```

```c
}

//verilen matrisin bellek sayfasindan eslemesini kaldirir
void matrix_unmap(int *matrix, unsigned long m_size)
{
  /// eger munmap() basarisizca cikar
  if (munmap(matrix, m_size) == -1)
  {
    exit(EXIT_FAILURE);
  }
}

//verilen satir icin tüm indeskleri hesaplamaya yarar
//noreturn ise derleyiciye bu fonksiyonun herhangi bir donusu olmayacagini bildirmek icin eklenir
__attribute__ ((noreturn)) void row_multiply(void *row_args)
{
  thr_args *args = (thr_args *) row_args;
  for(int i = 0; i < MATRIX_SIZE; i++)
  {
    for (int j = 0; j < MATRIX_SIZE; j++)
    {
      int add = array(args->a, args->row, j) * array(args->b, j, i);
      array(args->c, args->row, i) += add;
    }
  }
  pthread_exit(0); //islevi cagiran diziyi sonlandirir
}

int main(void)
{
  //matrislerin bellek boyutu hesaplanir
  unsigned long m_size = sizeof(int) * (unsigned long) (MATRIX_SIZE * MATRIX_SIZE);

  // matrix_a, matrix_b ve matrix_c bir bellek sayfasina eslenir
  matrix_a = matrix_page(matrix_a, m_size);
  matrix_b = matrix_page(matrix_b, m_size);
  matrix_c = matrix_page(matrix_c, m_size);

  //her iki matris de 1-10 arası rastgele sayilarla doldurulur
  fill_matrix(matrix_a);
  fill_matrix(matrix_b);

  //her iki matris de yazdirilir
  printf("A Matrisi:\n---------\n");
  print_matrix(matrix_a, 2);
  printf("B Matrisi:\n---------\n");
  print_matrix(matrix_b, 2);

  //threadler icin malloc() fonksiyonu ile diziler tahsis edilir
  pthread_t *thrs;
  thr_args *args;
```

```
128   if ((thrs = malloc(sizeof(pthread_t) * (unsigned long) MATRIX_SIZE)) ==
    NULL ||
129       (args = malloc(sizeof(thr_args) * (unsigned long) MATRIX_SIZE)) == NULL)
130   {
131     exit(EXIT_FAILURE);
132   }
133
134   //threadler olusturulur
135   for (int i = 0; i < MATRIX_SIZE; i++)
136   {
137     args[i] = (thr_args) {
138       .a = matrix_a,
139       .b = matrix_b,
140       .c = matrix_c,
141       .row = i
142     };
143     pthread_create(&thrs[i], NULL, (void *) &row_multiply, (void *)
    &args[i]);
144   }
145
```

```
146   //her threadin yaptigi is toplanir
147   for (int j = 0; j < MATRIX_SIZE; j++)
148     pthread_join(thrs[j], NULL);
149
150   //her thread icin ayrilan alanlar bosaltilir
151   if (thrs != NULL)
152   {
153     free(thrs);
154     thrs = NULL;
155   }
156   if (args != NULL)
157   {
158     free(args);
159     args = NULL;
160   }
161
162   //carpma istemini sonucu olarak C matrisi yazdirilir
163   printf("Matris Carpiminin sonucu C Matrisi:\n--------------\n");
164   print_matrix(matrix_c, 4);
165
166   //bellek sayfalarindan ayrilan alanlar geri verilir
167   matrix_unmap(matrix_a, m_size);
168   matrix_unmap(matrix_b, m_size);
169   matrix_unmap(matrix_c, m_size);
170 }
```