Θέματα προετοιμασίας για τις εξετάσεις εργαστηρίου στο μάθημα «Κατανεμημένα και Παράλληλα Συστήματα»

Τμήμα Πληροφορικής και Τηλεπικοινωνιών – Πανεπιστήμιο Ιωαννίνων @ Άρτα 2020

Γκόγκος Χρήστος

A. PThreads

Θέμα 1

Γράψτε ένα πρόγραμμα σε C που να υπολογίζει τη μεγαλύτερη τιμή στον πίνακα {23, 11, 18, 90, 16, 22, 34, 52, 19, 41, 88, 72} χρησιμοποιώντας 4 νήματα pThreads. Το πρώτο νήμα να υπολογίζει το μέγιστο από τις 3 πρώτες τιμές, το δεύτερο νήμα από τις 3 επόμενες κ.ο.κ. Το κύριο νήμα να λαμβάνει τα αποτελέσματα, να υπολογίζει το συνολικό μέγιστο και να τον εμφανίζει. Γράψτε τις εντολές μεταγλώττισης και εκτέλεσης του προγράμματος.

Λύση: https://github.com/chgogos/ceteiep_pdc/blob/master/exams_preparation/prepare_pthreads01.c

Θέμα 2

Γράψτε ένα πρόγραμμα σε C που να δημιουργεί έναν πίνακα με 1.000.000 τυχαίες ακέραιες τιμές στο διάστημα [0,99] και να χρησιμοποιεί 10 νήματα pThreads έτσι ώστε να τον ταξινομήσει χρησιμοποιώντας τον πίνακα συχνοτήτων για τις τιμές του πίνακα. Γράψτε τις εντολές μεταγλώττισης και εκτέλεσης του προγράμματος.

Λύση: https://github.com/chgogos/ceteiep_pdc/blob/master/exams_preparation/prepare_pthreads02.c

Θέμα 3

Γράψτε ένα πρόγραμμα σε C που να δημιουργεί έναν πίνακα με 1.000.000 τυχαίες ακέραιες τιμές στο διάστημα [0,10.000] και να χρησιμοποιεί 10 νήματα pThreads έτσι ώστε να τον ταξινομήσει με τον αλγόριθμο rank_sort. Ο αλγόριθμος rank_sort μετρά το πλήθος των τιμών που είναι μικρότερες από την τρέχουσα τιμή και η τιμή αυτή προσδιορίζει τη θέση της τρέχουσας τιμής στην ταξινομημένη λίστα. Γράψτε τις εντολές μεταγλώττισης και εκτέλεσης του προγράμματος.

Λύση: https://github.com/chgogos/ceteiep_pdc/blob/master/exams_preparation/prepare_pthreads03.c

Θέμα 4

Τι θα εμφανίσει κατά την εκτέλεσή του ο ακόλουθος κώδικας;

```c
#include <pthread.h>
#include <stdio.h>

#define T 5

int gl = 0;
intptr_t flag = 0;

void *work(void *tid) {
  intptr_t id = (intptr_t)tid;

  int lo = 0;
  static int st = 0;
```

```
  while (flag != id)
    ;
  ++lo;
  ++st;
  ++gl;
  printf("THREAD ID: %ld, Static: %d, Global: %d Local:%d\n", id, st, gl, lo);
  flag++;
}

int main() {
  pthread_t threads[T];

  for (intptr_t i = 0; i < T; i++)
    pthread_create(&threads[i], NULL, work, (void *)i);

  for (int i = 0; i < T; i++)
    pthread_join(threads[i], NULL);
  printf("Global: %d\n", gl);
  return 0;
}
```

Λύση: https://github.com/chgogos/ceteiep_pdc/blob/master/exams_preparation/prepare_pthreads04b.c

Θέμα 5

Το πρόγραμμα που δίνεται υπολογίζει το εσωτερικό γινόμενο δύο διανυσμάτων με 1.000.000 τυχαίες τιμές το καθένα. Μετατρέψτε το πρόγραμμα έτσι ώστε να έτσι να χρησιμοποιεί 5 pThreads νήματα για να επιτύχει το ίδιο αποτέλεσμα. Χρησιμοποιήστε mutex για αμοιβαίο αποκλεισμό της ενημέρωσης της μεταβλητής inner_product.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define N 1000000

int main() {
  double *a;
  double *b;
  double inner_prod = 0.0;
  a = (double *)malloc(sizeof(double) * N);
  b = (double *)malloc(sizeof(double) * N);
  srand(time(NULL));
  for (int i = 0; i < N; i++) {
    a[i] = (double)rand() / (double)RAND_MAX;
    b[i] = (double)rand() / (double)RAND_MAX;
  }

  for (int i = 0; i < N; i++) {
    inner_prod += a[i] * b[i];
  }

  printf("%.2f\n", inner_prod);
```

```
  free(a);
  free(b);
}
```

https://github.com/chgogos/ceteiep_pdc/blob/master/exams_preparation/inner_product.c

Λύση: https://github.com/chgogos/ceteiep_pdc/blob/master/exams_preparation/prepare_pthreads05.c

B. OpenMP

Θέμα 1

Γράψτε ένα πρόγραμμα σε C που χρησιμοποιώντας το OpenMP να δημιουργεί 4 νήματα και να εμφανίζει τα ακόλουθα μηνύματα:

- Το νήμα 0 να εμφανίζει το μήνυμα “Hello”.
- Το νήμα 1 να εμφανίζει το μήνυμα “Hi”.
- Το νήμα 2 να εμφανίζει το μήνυμα “Geia”.
- Το νήμα 3 να εμφανίζει το μήνυμα “Hallo”.

Γράψτε τις εντολές μεταγλώττισης και εκτέλεσης του προγράμματος.

Λύση: https://github.com/chgogos/ceteiep_pdc/blob/master/exams_preparation/prepare_omp01.c

Θέμα 2

Γράψτε ένα πρόγραμμα σε C που χρησιμοποιώντας το OpenMP να πραγματοποιεί τα ακόλουθα. Να δέχεται ως όρισμα γραμμής εντολών έναν ακέραιο αριθμό x και να εμφανίζει το άθροισμα των ριζών όλων των ακεραίων αριθμών από το 1 μέχρι και το x, χρησιμοποιώντας 2 νήματα. Γράψτε τις εντολές μεταγλώττισης και εκτέλεσης του προγράμματος.

Λύση: https://github.com/chgogos/ceteiep_pdc/blob/master/exams_preparation/prepare_omp02.c

Θέμα 3

Γράψτε ένα πρόγραμμα σε C που χρησιμοποιώντας το OpenMP να γεμίζει έναν πίνακα με 1.000.000 τυχαίες ακέραιες τιμές από το 1 μέχρι και το 5 και να εμφανίζει τον πίνακα συχνοτήτων, Οι υπολογισμοί να πραγματοποιηθούν μοιράζοντας την εργασία σε 2 νήματα. Γράψτε τις εντολές μεταγλώττισης και εκτέλεσης του προγράμματος.

Λύση: https://github.com/chgogos/ceteiep_pdc/blob/master/exams_preparation/prepare_omp03.c

Θέμα 4

Τι πρόκειται να εμφανίσει ο ακόλουθος κώδικας κατά την εκτέλεσή του.

```c
#include <omp.h>
#include <stdio.h>

#define N 4

int main() {
  int a = 0;
  int b = 0;
#pragma omp parallel for default(none) shared(a) private(b) num_threads(4)
  for (int i = 0; i < N; i++) {
    a++;
    b++;
  }
  printf("%d %d\n", a, b);
```

```
  return 0;
}
```

Λύση: https://github.com/chgogos/ceteiep_pdc/blob/master/exams_preparation/prepare_omp04.c

Θέμα 5

Το πρόγραμμα που δίνεται υπολογίζει το εσωτερικό γινόμενο δύο διανυσμάτων με 1.000.000 τυχαίες τιμές το καθένα. Μετατρέψτε το πρόγραμμα έτσι ώστε να έτσι να χρησιμοποιεί 5 pThreads νήματα για να επιτύχει το ίδιο αποτέλεσμα. Χρησιμοποιήστε mutex για αμοιβαίο αποκλεισμό της ενημέρωσης της μεταβλητής inner_product.

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define N 1000000

int main() {
  double *a;
  double *b;
  double inner_prod = 0.0;
  a = (double *)malloc(sizeof(double) * N);
  b = (double *)malloc(sizeof(double) * N);
  srand(time(NULL));
  for (int i = 0; i < N; i++) {
    a[i] = (double)rand() / (double)RAND_MAX;
    b[i] = (double)rand() / (double)RAND_MAX;
  }

  for (int i = 0; i < N; i++) {
    inner_prod += a[i] * b[i];
  }

  printf("%.2f\n", inner_prod);
  free(a);
  free(b);
}
```

https://github.com/chgogos/ceteiep_pdc/blob/master/exams_preparation/inner_product.c

Λύση: https://github.com/chgogos/ceteiep_pdc/blob/master/exams_preparation/prepare_omp05.c

Θέμα 6

Τι πρόκειται να εμφανίσει ο ακόλουθος κώδικας κατά την εκτέλεσή του; Τι θα αλλάξει αν προστεθεί στην εντολή #pragma το schedule(static,1) ;

```c
#include <omp.h>
#include <stdio.h>

#define N 8

int main() {
#pragma omp parallel for num_threads(4)
```

```c
for (int i = 0; i < N; i++) {
    int my_rank=omp_get_thread_num();
    if (my_rank==3)
      printf("%d ", i);
  }

  return 0;
}
```

Λύση: https://github.com/chgogos/ceteiep_pdc/blob/master/exams_preparation/prepare_omp06.c

Γ. MPI

Θέμα 1

Γράψτε ένα πρόγραμμα σε C που χρησιμοποιώντας το MPI να δημιουργεί δύο διεργασίες και η πρώτη διεργασία να στέλνει τον αριθμό 42 στη δεύτερη η οποία και θα τον εμφανίζει. Γράψτε τις εντολές μεταγλώττισης και εκτέλεσης του προγράμματος. Γράψτε τις εντολές μεταγλώττισης και εκτέλεσης του προγράμματος.

Λύση: https://github.com/chgogos/ceteiep_pdc/blob/master/exams_preparation/prepare_mpi01.c

Θέμα 2

Γράψτε ένα πρόγραμμα σε C που χρησιμοποιώντας το MPI να δημιουργεί 7 διεργασίες. Οι διεργασίες με αριθμούς από το 1 μέχρι και το 6 να υπολογίζουν το άθροισμα των ακεραίων από το 1 μέχρι την τιμή που βρίσκεται στην θέση του πίνακα {50, 45, 33, 17, 19, 28} που υποδηλώνεται από τον αριθμό της διεργασίας (π.χ. η διεργασία 1 θα πρέπει να υπολογίζει το άθροισμα από το 1 μέχρι και το 50, η διεργασία 2 θα πρέπει να υπολογίζει το άθροισμα από το 1 μέχρι το 45, κ.ο.κ.). Τα αποτελέσματα να αποστέλλονται στη διεργασία 0 η οποία θα τα αθροίζει και θα εμφανίζει το τελικό αποτέλεσμα. Γράψτε τις εντολές μεταγλώττισης και εκτέλεσης του προγράμματος.

Λύση: https://github.com/chgogos/ceteiep_pdc/blob/master/exams_preparation/prepare_mpi02.c

Θέμα 3

Επιλύστε το προηγούμενο θέμα χρησιμοποιώντας τη συνάρτηση MPI_Reduce.

Λύση: https://github.com/chgogos/ceteiep_pdc/blob/master/exams_preparation/prepare_mpi03.c

Θέμα 4

Τι θα εμφανίσει ο ακόλουθος κώδικας όταν εκτελεστεί για 2 διεργασίες.

```c
#include <stdio.h>
#include <mpi.h>
#include <stdlib.h>
#define N 10

int main(void)
{
  int comm_sz, my_rank;

  MPI_Init(NULL, NULL);
  MPI_Comm_size(MPI_COMM_WORLD, &comm_sz);
  MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
```

```
  int a[N] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

  int local_n = N / comm_sz;
  int local_a[local_n];
  MPI_Scatter(a, local_n, MPI_INT, local_a, local_n, MPI_INT, 0, MPI_COMM_WORLD);

  if (my_rank == 1)
  {
    for (int i = 0; i < local_n; i++)
    {
      printf("%d", local_a[i]);
    }
  }

  MPI_Finalize();
  return 0;
}
```

Λύση: https://github.com/chgogos/ceteiep_pdc/blob/master/exams_preparation/prepare_mpi04.c

Θέμα 5

Το πρόγραμμα που δίνεται υπολογίζει το εσωτερικό γινόμενο δύο διανυσμάτων με 1.000.000 τυχαίες τιμές το καθένα. Μετατρέψτε το πρόγραμμα έτσι ώστε να έτσι να χρησιμοποιεί 5 διεργασίες για να επιτύχει το ίδιο αποτέλεσμα.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define N 1000000

int main() {
  double *a;
  double *b;
  double inner_prod = 0.0;
  a = (double *)malloc(sizeof(double) * N);
  b = (double *)malloc(sizeof(double) * N);
  srand(time(NULL));
  for (int i = 0; i < N; i++) {
    a[i] = (double)rand() / (double)RAND_MAX;
    b[i] = (double)rand() / (double)RAND_MAX;
  }

  for (int i = 0; i < N; i++) {
    inner_prod += a[i] * b[i];
  }

  printf("%.2f\n", inner_prod);
```

```
  free(a);
  free(b);
}
```

https://github.com/chgogos/ceteiep_pdc/blob/master/exams_preparation/inner_product.c

Λύση: https://github.com/chgogos/ceteiep_pdc/blob/master/exams_preparation/prepare_mpi05.c

Δ. Java concurrency

Θέμα 1

Το ακόλουθο πρόγραμμα δημιουργεί έναν πίνακα 1.000.000 θέσεων με τυχαίες ακέραιες τιμές στο διάστημα [0,99]. Με τη χρήση 10 νημάτων υπολογίστε και εμφανίστε τον πίνακα συχνοτήτων.

```java
import java.util.Random;

public class Exams01 {
  final static int N = 1_000_000;
  final static int M = 100;

  static int a[] = new int[N];
  static int frequency[] = new int[M];

  public static void main(String[] args){
    Random random = new Random(1729);
    for (int i = 0; i < N; i++) {
      a[i] = random.nextInt(M);
    }

    for (int i = 0; i < N; i++) {
      frequency[a[i]]++;
    }

    for (int i = 0; i < M; i++) {
      System.out.printf("%d --> %d\n", i, frequency[i]);
    }
  }
}
```

https://github.com/chgogos/ceteiep_pdc/blob/master/exams_preparation/PrepareJavaConcurrency01a.java

Λύση: https://github.com/chgogos/ceteiep_pdc/blob/master/exams_preparation/PrepareJavaConcurrency01b.java