ΠΑΝΕΠΙΣΤΗΜΙΟ
ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

Συστήματα Παράλληλης και Κατανεμημένης Επεξεργασίας

Ενότητα: ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ No:10

Δρ. Μηνάς Δασυγένης

mdasyg@ieee.org

Τμήμα Μηχανικών Πληροφορικής και Τηλεπικοινωνιών

Εργαστήριο Ψηφιακών Συστημάτων και Αρχιτεκτονικής Υπολογιστών

http://arch.icte.uowm.gr/mdasyg

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ψηφιακά Μαθήματα του Πανεπιστημίου Δυτικής Μακεδονίας**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Περιεχόμενα

1. Σκοπός της άσκησης 4
2. Παραδοτέα 4
3. Δημιουργία του πρώτου προγράμματος με threads. 4
   3.1 Δημιουργία & Καταστροφή νημάτων. 4
   3.2 Πέρασμα παραμέτρων σε νήματα: 5
      3.2.1 Μια ακέραια τιμή 5
      3.2.2 Πολλαπλές τιμές..... 6
   3.3 Συγχρονισμός διεργασιών με pthread_join()..... 9

1. Σκοπός της άσκησης

- Νήματα POSIX.
- Διεργασίες.
- Συγχρονισμός με pthread_join.

2. Παραδοτέα

(A) 4 ερωτήσεις
(C) 4 ασκήσεις

3. Δημιουργία του πρώτου προγράμματος με threads.

3.1 Δημιουργία & Καταστροφή νημάτων.

1. Κατασκευάστε ένα αρχείο με όνομα **c1.c**.
2. Τοποθετήστε το `#include <pthread.h>` και ότι άλλα include χρειάζονται.
3. Ορίστε μια συνάρτηση **`void *PrintHello(void *threadid)`**
    1. Η συνάρτηση αυτή θα εκτυπώνει το string **hello world** και θα ακολουθεί ο αριθμός του `threadid`.
    2. Στη συνέχεια θα καλείται η συνάρτηση τερματισμού thread.
4. Στο κυρίως πρόγραμμα ορίστε ένα `#define NUM_THREADS 5`
5. Δημιουργήστε ένα βρόχο επαναλήψεων, το οποίο δημιουργεί τόσα threads όσα το προηγούμενο define. Όλα τα thread θα εκτελούν τη συνάρτηση `PrintHello()`. Η παράμετρος που θα στέλνουμε κάθε φορά θα είναι ο αριθμός του thread. Μπορείτε να χρησι-μοποιήσετε τη δήλωση `pthread_t threads[NUM_THREADS];` ώστε να μην απαιτείται κάθε φορά να δηλώνετε ένα ένα όλα τα **`pthread_t`** που χρειάζεστε.
6. **(A1)** Ελέγξτε την τιμή που επιστρέφεται κάθε φορά που δημιουργείται ένα νέο νήμα. Ποια είναι η τιμή επιτυχίας και ποια η αποτυχίας; Αν η τιμή δεν είναι σωστή *(δείτε το man page)* τότε θα εμφανίζεται ένα μήνυμα αποτυχίας σχετικά με το ποιο thread δε μπόρεσε να δημιουργηθεί.
7. **(A2)** Το threadid είναι δείκτης void. Τι σημαίνει αυτό, και γιατί το χρησιμοποιούμε;
8. Κάντε compile με το gcc με την παράμετρο ότι θα χρησιμοποιήσετε τη βιβλιοθήκη `pthread` (“Search the library named library when linking.”).

   (A3) Ποια είναι η πλήρης εντολή που χρησιμοποιήσατε;

9. Επιβεβαιώστε την ορθή λειτουργία.
10. Αλλάξτε την τιμή του **define** των **threads** σε 50 και επιβεβαιώστε την ορθή λειτουργία.

(C1) Να παραδώσετε το ανωτέρω αρχείο.

3.2 Πέρασμα παραμέτρων σε νήματα:

3.2.1 Μια ακέραια τιμή

1. Αντιγράψτε το προηγούμενο αρχείο στο **c2.c**. Κάντε τις παρακάτω τροποποιήσεις.
2. Μέσα στη συνάρτηση των thread το printf θα τροποποιηθεί στο
   `printf("Thread %d: %s\n", taskid, messages[taskid]);`
   όπου `taskid` είναι ο αριθμός του thread και messages ένας πίνακας που έχει οριστεί στο κυρίως πρόγραμμα.
3. Στο κυρίως πρόγραμμα, ορίστε τα εξής:
```
messages[0] = "English: Hello World!";
messages[1] = "French: Bonjour, le monde!";
messages[2] = "Spanish: Hola al mundo";
messages[3] = "Klingon: Nuq neH!";
messages[4] = "German: Guten Tag, Welt!";
messages[5] = "Russian: Zdravstvytye, mir!";
messages[6] = "Japan: Sekai e konnichiwa!";
messages[7] = "Latin: Orbis, te saluto!";
```
4. Έξω από κάθε συνάρτηση ορίστε τη global μεταβλητή:
   `char *messages[NUM_THREADS];`
5. Με αυτόν τον τρόπο όλα τα νήματα θα έχουν πρόσβαση σε αυτόν τον κοινόχρηστο πίνακα.
6. Μέσα στο βρόχο που δημιουργούνται τα threads
   (από `t=0` έως `t<NUM_THREADS`) θα δεσμεύουμε μια περιοχή μνήμης, θα αντιγράφουμε σε αυτήν την περιοχή μνήμης την τιμή που θέλουμε να στείλουμε σε κάθε νήμα και θα στέλνουμε ως παράμετρο στα νήματα τη διεύθυνση αυτής της περιοχής μνήμης:

```
taskids[t] = (int *) malloc(sizeof(int));
*taskids[t] = t;
```

(A4) Στην πρώτη εντολή από αυτές γιατί χρησιμοποιούμε (int *) μπροστά από τη malloc; Τι θα συνέβαινε αν δε χρησιμοποιούσαμε αυτό;

**** Γιατί πρέπει να χρησιμοποιήσουμε αυτές τις 2 γραμμές; Αν στέλναμε κατευθείαν τη μεταβλητή t σε κάθε thread τότε θα στέλναμε τη διεύθυνση**

μνήμης της μεταβλητής `t` , η οποία έχει δεδομένα που τροποποιούνται συνέχεια στο βρόχο *(αρχικά τα δεδομένα είναι 0 μετά είναι 1, κ.ο.κ.)*. Κατασκευάζουμε μια μεταβλητή για κάθε νήμα λοιπόν, η οποία δεν αλλάζει καθώς αλλάζει ο δείκτης του βρόχου. ***

7. Η παράμετρος που θα στέλνουμε σε κάθε thread είναι η
   `(void *) taskids[t]`
8. Ορίστε το `NUM_THREADS` σε `8`.
9. Προσοχή στη συνάρτηση νήματος:

   a. η συνάρτηση στον ορισμό της έχει ως παράμετρο δείκτη:
   `void * parameter`

   b. προκειμένου να διαβάσουμε την τιμή που βρίσκεται στη διεύθυνση μνήμης που δείχνει ο δείκτης *(που αντιστοιχεί σε έναν ακέραιο αριθμό)* θα πρέπει να ορίσουμε ότι θα χρησιμοποιήσουμε ένα δείκτη ακεραίων ως εξής:
   `int *ptr;`

   c. Μετά θα τοποθετήσουμε στο δείκτη που έχουμε δηλώσει τον pointer που έχουμε δεχθεί ως παράμετρο της συνάρτησης, με το κατάλληλο typecasting σε int:
   `ptr=(int *) parameter;`

   d. Θα πρέπει να δηλώσουμε μια ακέραια μεταβλητή για να τοποθετήσουμε αυτό που θα διαβάσουμε ως:
   `int index;`

   e. Για να διαβάσουμε την ακέραια τιμή που δείχνει ο pointer `*ptr` αρκεί να δώσουμε `index=*ptr;`

   f. Την τιμή index που είναι μια ακέραια τιμή θα χρησιμοποιήσουμε στο `printf()`.

10. Κάντε compile και επιβεβαιώστε την ορθή λειτουργία.

(C2) Να παραδώσετε το ανωτέρω αρχείο.

3.2.2 Πολλαπλές τιμές

1. Αντιγράψτε το προηγούμενο αρχείο στο **c3.c**
2. Στη συνάρτηση δημιουργίας του νήματος, μπορούμε να στείλουμε μια μεταβλητή. Αν έχουμε πολλές παραμέτρους που πρέπει να στείλουμε μπορούμε να δημιουργήσουμε μια μεταβλητή τύπου struct, να τοποθετήσουμε όλες τις μεταβλητές σε αυτή τη δομή και να στείλουμε αυτή τη μεταβλητή στο νήμα. Για να λειτουργήσει αυτό σωστά θα πρέπει να είναι

γνωστό και στο νήμα και στο κυρίως πρόγραμμα το συγκεκριμένο struct. Για αυτό το λόγο ορίζουμε έξω από τις συναρτήσεις την παρακάτω δομή (struct):

```
struct thread_data{
int thread_id;
int  sum;
char *message;
};
```

Με το παραπάνω struct μπορούμε να στείλουμε με την ίδια μεταβλητή *(τύπου struct thread_data),* ταυτόχρονα δύο ακέραιους *(thread_id,sum)* και έναν δείκτη προς μια σειρά χαρακτήρων.

Ορίστε έναν πίνακα που έχει τόσες γραμμές όσα τα νήματα, και του οποίου κάθε γραμμή είναι τύπου `struct thread_data`, ως εξής:

```
struct thread_data thread_data_array[NUM_THREADS];
```

Με αυτό το τρόπο θα στέλνουμε σε κάθε thread ένα struct το οποίο έχει πολλαπλές μεταβλητές. Μόνο με αυτό τον τρόπο μπορούμε να στείλουμε πολλαπλές παραμέτρους σε ένα νήμα.

3. Τροποποιήστε τη συνάρτηση `printhello()` ώστε να δέχεται ως όρισμα το (`void *threadarg`) και ότι θα χρησιμοποιήσει το

   ```
   struct struct thread_data *my_data;
   ```

4. Ορίστε ως ακέραια τιμή μια μεταβλητή **`taskid`**, η οποία θα φέρει κάθε φορά τον ακέραιο αριθμό αριθμού thread που βρίσκεται μέσα στο struct thread_data.

5. Στη συνέχεια μέσα στη συνάρτηση printhello, διαβάστε όλες τις τιμές του struct ως εξής:

   ```
   my_data = (struct thread_data *) threadarg;

   taskid = my_data->thread_id;
   ```

6. Και μέσα στη συνάρτηση printhello, εκτυπώστε με printf το κάθε στοιχείο, ώστε στο τέλος να εμφανίζονται τα μηνύματα ως εξής:

```
...

Thread 4: German: Guten Tag, Welt!  Sum=10
Thread 5: Russian: Zdravstvytye, mir!  Sum=15
Thread 6: Japan: Sekai e konnichiwa!  Sum=21
Thread 7: Latin: Orbis, te saluto!  Sum=28

....
```

7. Στο κυρίως πρόγραμμα στη δημιουργία των threads θα πρέπει να τοποθετήστε τις γραμμές που γράφουν δεδομένα στο struct όπως:

```
thread_data_array[t].thread_id = t;

thread_data_array[t].sum = sum;

thread_data_array[t].message = messages[t];
```

8. Το sum *(που το έχετε ορίσει ως ακέραιο μέσα στο main() για να συμφωνεί με το struct)* να το υπολογίζετε κάθε φορά ως:

```
sum = sum + t + 1;
```

9. Η κλήση της δημιουργίας νημάτων θα έχει για παράμετρο το

```
(void *) &thread_data_array[t];
```

10. Κάντε compile και επιβεβαιώστε την ορθή λειτουργία.

(C3) Να παραδώσετε το ανωτέρω αρχείο.

ΠΡΟΣΟΧΗ:

Η παρακάτω δομή είναι λάθος, επειδή στέλνει τη διεύθυνση της μεταβλητής t η οποία βρίσκεται σε κοινή μνήμη και προσβάσιμη από κάθε νήμα. Καθώς το loop προχωράει η τιμή του t αλλάζει (αρχικά t=0, μετά t=1..) οπότε μέχρι να διαβάσει το νήμα την τιμή έχει ήδη αλλάξει αυτή και μας είναι άχρηστη. Για αυτό το λόγο προσέχουμε αν θα στείλουμε τη διεύθυνση της μεταβλητής με το & μπροστά ή την τιμή της μεταβλητής χωρίς το & από μπροστά

Λάθος:
```
int rc;long t;
for(t=0; t<NUM_THREADS; t++)
{
printf("Creating thread %ld\n", t);
rc = pthread_create(&threads[t], NULL, PrintHello,
(void *) &t);
     ...
}
```

Σωστό:
```
int rc;long t;
for(t=0; t<NUM_THREADS; t++)
{
 printf("Creating thread %ld\n", t);
 rc = pthread_create(&threads[t], NULL, PrintHello, (void
*) t );
...
}
```

3.3 Συγχρονισμός διεργασιών με pthread_join()

Θα κατασκευάσουμε μια διαδικασία που θα την εκτελεί κάθε νήμα προκειμένου να κάνει κάποιους υπολογισμούς. Το κυρίως πρόγραμμα θα περιμένει να ολοκληρωθούν όλοι οι υπολογισμοί *(να τερματίσουν δηλαδή όλα τα νήματα)*. Όταν κάνουν join όλα τα νήματα, τότε θα ολοκληρώνεται και το κυρίως πρόγραμμα.

1. Δημιουργήστε το αρχείο **c4.c**
2. Κάντε include το header file των μαθηματικών σχέσεων γιατί θα χρησιμοποιήσουμε τριγωνομετρικές συναρτήσεις.

***** Όταν χρησιμοποιούμε μαθηματικές συναρτήσεις στη C *(gcc)* θα πρέπει κατά το compile να κάνουμε link με τη μαθηματική βιβλιοθήκη. Αυτό γίνεται με το να προσθέσουμε το διακόπτη -lm στην εντολή gcc …. ******

3. Κάντε include όλα τα υπόλοιπα header file που θα χρησιμοποιήσετε.
4. Ορίστε στο define τον αριθμό των threads σε 4.
5. Ορίστε μια διαδικασία που θα εκτελούν τα threads ως computethread με:

   ```
   void *computethread(void *t)
   ```

   1. Η διαδικασία θα εκτυπώνει ένα μήνυμα ότι ξεκινάει το νήμα με αριθμό **XXX** *(ο αριθμός που δίνεται στην παράμετρο)*.
   2. Στη συνέχεια θα εκτελεί ένα βρόχο από i=0 έως $i=10^7$ με την πράξη: **(μέσα στη συνάρτηση έχει οριστεί `double result=`0 . 0)**
      ```
      result = result + sin(i) * tan(i);
      ```
   3. Όταν υπολογιστεί ο αριθμός result το νήμα εκτυπώνει ότι "Το νήμα με αριθμό XXXX, τελείωσε. Το result είναι YYYYY"

6. Στην κυρίως συνάρτηση ***(main).***

   1. Ορίστε για κάθε thread τον τύπο `pthread_t` *(σε μορφή πίνακα όπως το είχατε κάνει πριν).*
   2. Ορίστε τη ρύθμιση των threads attr , ως:
      ```
      pthread_attr_t attr;
      ```
   3. Αρχικοποιήστε το `attr` με τη χρήση της συνάρτησης `pthread_attr_init`.
   4. Ορίστε ότι το `attr` θα είναι JOINABLE με τη χρήση της συνάρτησης `pthread_attr_setdetachstate()`, δηλαδή η κατάσταση να είναι `PTHREAD_CREATE_JOINABLE`.

5. Δημιουργήστε ένα βρόχο for από **t=0** έως **t<NUM_THREADS** το οποίο θα εκτυπώνει ένα μήνυμα :

   "MAIN: Creating thread Number ..." και θα δημιουργεί το νήμα με παράμετρο το **(void *)t** και το χαρακτηριστικό **&attr**.

6. Να υπάρχει έλεγχος ότι δημιουργείται το thread. Αν δε μπορεί να δημιουργηθεί το thread τότε να εμφανίζεται σχετικό μήνυμα.
7. Όταν ολοκληρωθεί το παραπάνω loop, τότε δε θα χρειάζεται πια το attribute και μπορούμε να το απομακρύνουμε με την κλήση της συνάρτησης **pthread_attr_destroy**.
8. Δημιουργήστε ένα νέο βρόχο for από **t=0** έως **t<NUM_THREADS** το οποίο θα κάνει κλήση της συνάρτησης **pthread_join** με πρώτη παράμετρο το **pthread_t** του κάθε thread, και δεύτερη παράμετρο το **&status**.
9. Στο status θα τοποθετείται η τιμή που επιστρέφει το κάθε νήμα όταν τερματίζει.
10. Να ελέγξετε αν εκτελέστηκε κάθε φορά με επιτυχία η **pthread_join** με μια δομή **if**. Αν δεν έχει εκτελεστεί σωστά να εκτυπώνεται το αντίστοιχο μήνυμα.
11. Το main να εκτυπώνει στο τέλος κάθε join που γίνεται μαζί με την τιμή που επιστρέφει

    "Main: completed join with thread number XXXX having a status of YYYYY".

12. Χρησιμοποιήστε σε αυτήν την άσκηση τη συνάρτηση **pthread_self()** για να εκτυπώνεται το thread ID.
13. Κάντε compile και επιβεβαιώστε την ορθή λειτουργία.

(C4) Να παραδώσετε το ανωτέρω αρχείο.