



ΙΟΝΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ – ΠΑΡΑΛΛΗΛΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

ΕΡΓΑΣΙΑ 2

Ονοματεπώνυμο: Τασιόπουλος Μανώλης

ΑΜ : Π2015046

Διδάσκοντας : Μιχάλης Στεφανιδάκης

Αριθμός σελίδων : 6

ΠΕΡΙΕΧΟΜΕΝΑ

1. Περιγραφή.....σελ.3
2. Thread Pool σελ.4
3. Thread_Func().....σελ.5
4. Αποτελέσματα και έξοδοισελ.6

ΠΕΡΙΓΡΑΦΗ

QUICKSORT.C

-Το "[quicksort.c](#)" είναι ένα πρόγραμμα γραμμένο σε C κώδικα το οποίο υλοποιεί τον αλγόριθμο QuicSort() με διάφορες παραλλαγές, αξιοποιώντας τη βιβλιοθήκη <[pthread.h](#)>.

Αποτελείται από τα εξής χαρακτηριστικά:

- **Έναν πίνακα που περιέχει τα Threads**

Τα threads αποθηκεύονται σε
έναν πίνακα που ονομάζεται
"[thread pool](#)" και αναλαμβάνουν
την επεξεργασία διαφόρων
πακέτων με τη χρήση μιας ουράς
εργασιών

- **→ Ουρά εργασιών**

Η ουρά αυτή είναι υπεύθυνη
για την προώθηση και την
αποθήκευση των πακέτων
εργασίας τα οποία επεξεργάζονται
στα threads του thread poll

-Στη συνέχεια θα δούμε πιο αναλυτικά τον τρόπο με τον οποίο μπορούν να χρησιμοποιηθούν αυτά τα δύο χαρακτηριστικά για την ταξινόμηση ενός πίνακα N θέσεων

THREAD POOL

- Το `thread_pool` είναι μια μεταβλητή της συνάρτησης `main()` και θα χρησιμοποιηθεί για την αποθήκευση των `thread` που θα χρειαστούν για την ταξινόμηση του πίνακα.

Η λειτουργίες της μεταβλητής αυτής είναι οι εξής:

- Έχει μέγεθος που ορίζεται από τη σταθερά "**NUM_THREADS**"
- Δημιουργείται με τη συνάρτηση **`pthread_create()`**
- Ως ορίσματα περνιούνται το `thread_pool`, η συνάρτηση **`threrad_func()`** και ο πίνακας για την ταξινόμηση

-Η `thread_func()` είναι μία συνάρτηση η οποία επικοινωνεί με την `main()` και ελέγχει ποια πακέτα θα επεξεργαστούν και με ποιο τρόπο.

-Ξεκινώντας τα `threads` ελέγχουν την ουρά για πακέτα με τη χρήση της συνάρτησης `receive()`. Η συνάρτηση αυτή επιστρέφει:

- By reference έναν τύπο **`struct`** που έχουμε ορίσει για τα πακέτα μας
- Έναν δείκτη `start` για την αρχή του πίνακα
- Έναν δείκτη `end` για το τέλος του πίνακα

-Κατά τη λειτουργία της η `receive()`:

- **Κλειδώνει** το `mutex` που είναι απαραίτητο για το συγχρονισμό
- Ελέγχει τον αριθμό των μηνυμάτων με τη μεταβλητή **`msg_count`**
- Αν δεν υπάρχει άλλο μήνυμα στην ουρά **ξεκλειδώνει** το `mutex`
- Περιμένει σήμα για το **conditional variable `msg_in`**
- Αν εντοπιστούν οι τιμές που του μηνύματος **αποθηκεύονται** στα ορίσματα
- Στέλνει σήμα `msg_out` και **ξεκλειδώνει** το `mutex`

-Υπάρχει περίπτωση να έχουν κολλήσει τα `threads` αν δεν υπάρχει αρκετός χώρος στην ουρά, οπότε και για αυτό ξεκλειδώνουμε τον **`mutex`**.

THREAD_FUNC()

-Η συνάρτηση "thread_func()" περιμένει να λάβει ένα σήμα μηνύματος. Μόλις εντοπίσει ένα μήνυμα ελέγχει την μεταβλητή του struct που το αποτελεί και περιέχει τον τύπο του. Το πρόγραμμα αναγνωρίζει 3 τύπους μηνύματος [**WORK** = 0, **FINISH** = 1, **STOP** = 2]

Αναλυτικότερα για τον τύπο:

- **WORK:** Για το μήκος του κάθε κομματιού πίνακα που λαμβάνει υπολογίζεται το μήκος του με τη διαφορά (end – start).
 - **Αν** είναι μικρότερο από ένα Threshold που ορίσαμε
 - Τότε ταξινομείται με τη συνάρτηση **Insertion Sort()**
 - Τοποθέτηση του τύπου σε **FINISH**
 - **Αλλιώς**
 - Ταξινομείται με τη συνάρτηση **Quick Sort()**
 - Διαχωρισμός του πίνακα στο **pivot** με τη συνάρτηση **partition()**
 - **Τοποθέτηση** στην ουρά τα πακέτα για τα δύο υποσύνολα
- **FINISH:** Αν το thread λάβει ένα τέτοιο μήνυμα τότε το προωθεί στην ουρά εργασίας ώστε να ενημερώσει το thread για την κατάσταση της εκτέλεσης
- **STOP:** Αν λάβει το thread αυτό το μήνυμα τότε σταματάει να ελέγχει για καινούργιους τύπους μηνύματος και το τοποθετεί στην ουρά προκειμένου να σταματήσει η εκτέλεση και να αρχίσει ο έλεγχος των μηνυμάτων

-Μόλις ολοκληρωθεί η διαδικασία του ελέγχου των τύπων των μηνυμάτων τότε η εκτέλεση του προγράμματος επιστρέφει στη συνάρτηση **main()** για να ολοκληρωθεί ο έλεγχος της σωστής ταξινόμησης του πίνακα

ΑΠΟΤΕΛΕΣΜΑΤΑ ΚΑΙ ΕΞΟΔΟΙ

-Το πρόγραμμα “[quicksort.c](#)” μπορεί να ταξινομήσει έναν πίνακα N θέσεων με τη χρήση ενός αριθμού threads, μία ουρά εργασίας και τους αλγόριθμους **QuickSort** και **InsertionSort**.

Τα δεδομένα που χρησιμοποιούνται είναι:

1. ΑΡΙΘΜΟΣ THREADS = 4
2. ΜΗΚΟΣ ΟΥΡΑΣ ΕΡΓΑΣΙΑΣ = 1000000
3. ΜΗΚΟΣ ΠΙΝΑΚΑ ΤΑΞΙΝΟΜΗΣΗΣ = 100
4. THRESHOLD = 10

Δημιουργία πίνακα και προσθήκη τυχαίων τιμών

```
C:\Users\JeStEr\Desktop\Ex2>quicksort
Allocate new array with size [100]

Fill the array[100]: 0.001251, 0.563585, 0.193304, 0.808741, 0.585009, 0.479873, 0.350291, 0.895962, 0.822840, 0.746605, 0.174108, 0.85894
3, 0.710501, 0.513535, 0.303995, 0.014985, 0.091403, 0.364452, 0.147313, 0.165899, 0.988525, 0.445692, 0.119083, 0.004669, 0.008911, 0.377
880, 0.531663, 0.571184, 0.601764, 0.607166, 0.166234, 0.663045, 0.450789, 0.352123, 0.057039, 0.607685, 0.783319, 0.802606, 0.519883, 0.3
01950, 0.875973, 0.726676, 0.955901, 0.925718, 0.539354, 0.142338, 0.462081, 0.235328, 0.862239, 0.209601, 0.779656, 0.843654, 0.996796, 0
.999695, 0.611499, 0.392438, 0.266213, 0.297281, 0.840144, 0.023743, 0.375866, 0.092624, 0.677206, 0.056215, 0.008789, 0.918790, 0.275887, 0
.272897, 0.587909, 0.691183, 0.837611, 0.726493, 0.484939, 0.205359, 0.743736, 0.468459, 0.457961, 0.949156, 0.744438, 0.108280, 0.59904
8, 0.385235, 0.735008, 0.608966, 0.572405, 0.361339, 0.151555, 0.225105, 0.425153, 0.802881, 0.517106, 0.989990, 0.751549, 0.345561, 0.168
981, 0.657308, 0.491897, 0.063540, 0.699759, 0.504807,
```

```
Finnish
create thread pool
return
```

Πρόσδος των μηνυμάτων και έλεγχος για τη σωστή ταξινόμηση του πίνακα

```
create thread pool

---Messages state---

Messages completed: 5 / 100
Messages completed: 12 / 100
Messages completed: 20 / 100
Messages completed: 30 / 100
Messages completed: 36 / 100
Messages completed: 39 / 100
Messages completed: 49 / 100
Messages completed: 59 / 100
Messages completed: 64 / 100
Messages completed: 69 / 100
Messages completed: 78 / 100
Messages completed: 87 / 100
Messages completed: 93 / 100
Messages completed: 100 / 100

SUCCESS array is sorted
```

```
C:\Users\JeStEr\Desktop\Ex2>
```

```
C:\Users\JeStEr\Desktop\Ex2>
```

```
SUCCESS array is sorted
```

```
Messages completed: 100 / 100
```