

Παράλληλος προγραμματισμός 2018

Προγραμματιστική άσκηση #2

Παρασκευή Μέμου

Π2013092

Το πρόγραμμα αναπτύχθηκε σε περιβάλλον Windows 10, ενώ μεταγλωττίστηκε κι εκτελέστηκε στο εργαλείο Cygwin το οποίο είναι ένα περιβάλλον προσομοίωσης του UNIX/Linux και παρέχει τη δυνατότητα να δημιουργεί προγράμματα κάνοντας χρήση των βιβλιοθηκών και των υπηρεσιών του UNIX ενώ το ίδιο το εκτελέσιμο του προγράμματος τρέχει σε Windows. Για να κάνουμε compile το πρόγραμμα εκτελούμε την εντολή «*gcc -O2 -Wall quicksort.c -o quicksort*», ενώ για να εκτελέσουμε το πρόγραμμα τρέχουμε την εντολή «*./quicksort*». Ο υπολογιστής στον οποίο τρέξαν τα προγράμματα διαθέτει τον επεξεργαστή [i7-4510U](https://www.notebookcheck.net/Intel-Core-i7-4510U-Notebook-Processor.115083.0.html), ο οποίος έχει τις ακόλουθες μνήμες cache.

Level 1 Cache	128 KB
Level 2 Cache	512 KB
Level 3 Cache	4 MB

Πηγή: <https://www.notebookcheck.net/Intel-Core-i7-4510U-Notebook-Processor.115083.0.html>

Το μέγεθος της κυκλικής ουράς για τα μηνύματα ορίζεται με την παράμετρο *CIRCULAR_QUEUE_SIZE* στον κώδικα, ενώ το συνολικό πλήθος των μηνυμάτων δηλώνεται με την παράμετρο *N*. Η κυκλική ουρά είναι η μεταβλητή *global_buffer* και είναι τύπου *circular_queue*. Η δομή *circular_queue* παρατίθεται στον επόμενο πίνακα. Οι μεταβλητές *front*, *rear* είναι δείκτες στο πρώτο και τελευταίο στοιχείο της ουράς αντίστοιχα, η μεταβλητή *capacity* τίθεται ίση με την παράμετρο *CIRCULAR_QUEUE_SIZE*, ενώ ο πίνακας *array* είναι μεγέθους *CIRCULAR_QUEUE_SIZE* και περιέχει μεταβλητές τύπου *quicksort_data*.

```
typedef struct circular_queue {  
    int front, rear;  
    unsigned capacity;  
    quicksort_data *array;  
} circular_queue;
```

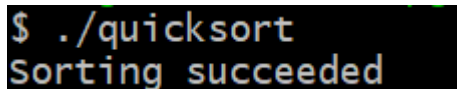
Η δομή `quicksort_data` είναι το είδος των μηνυμάτων που αποφασίστηκε να στέλνεται στην ουρά και παρατίθεται στον επόμενο πίνακα. Η μεταβλητή `a` είναι ο αρχικός δείκτης και το `pivot` η `pivot` τιμή για τον `quicksort`.

```
typedef struct quicksort_data {  
    double *a;  
    unsigned pivot;  
} quicksort_data;
```

Αρχικά, η κύρια ρουτίνα του προγράμματος δημιουργεί τυχαίους πραγματικούς αριθμούς και τους αποθηκεύει στο δυναμικό πίνακα `a`. Έπειτα αρχικοποιείται η κυκλική ουρά με τη βοήθεια και της συνάρτησης `enqueue`. Έπειτα, δημιουργούνται 4 νήματα και για κάθε ένα από αυτά καλείται η συνάρτηση `quicksort_thread_function` με όρισμα τη μεταβλητή `thr_data`. Η μεταβλητή αυτή είναι τύπου `thread_data`, που περιέχει δύο μεταβλητές, το πλήθος των στοιχείων που έχουν ταξινομήσει και το αν πρέπει να τερματίσουν ή όχι.

Το κάθε νήμα παίρνει μηνύματα με τη συνάρτηση `get_job_from_circular_queue` (η συνάρτηση αυτή καλεί τη μέθοδο `dequeue` για να αφαιρέσει τα δεδομένα από την ουρά) και τα αναθέτει στη μεταβλητή `q`. Έπειτα ελέγχει αν το στοιχείο `pivot` είναι μικρότερο ή ίσο από την τιμή που έχει η παράμετρος `CUTOFF`, ώστε να καλέσει τη μέθοδο `insertion_sort`. Διαφορετικά καλεί τη μέθοδο `perform_quicksort` ώστε να ταξινομήσει τα στοιχεία που έλαβε και προσθέτει επιπλέον μηνύματα στην κυκλική ουρά με τη μέθοδο `add_jobs_to_circular_queue`. Η μέθοδος αυτή καλεί τη συνάρτηση `enqueueer` για να εισάγει στοιχεία στην κυκλική ουρά.

Στη συνέχεια της κύριας ρουτίνας, όσο δεν έχουν ταξινομήσει όλα τα στοιχεία περιμένουμε κάνοντας χρήση της συνάρτησης `pthread_cond_wait`. Έπειτα, ειδοποιούμε όλα τα νήματα να τερματίσουν και καταστρέφουμε τις μεταβλητές που χρησιμοποιήθηκαν. Στο τέλος της κύριας ρουτίνας συμπεριλήφθηκε ένας έλεγχος ώστε να επιβεβαιωθεί ότι τα στοιχεία στον πίνακα είναι ταξινομημένα. Όπως φαίνεται στο παρακάτω screenshot, ο αλγόριθμος ταξινόμηση επιτυχώς τα στοιχεία.



```
$ ./quicksort  
Sorting succeeded
```

Οι συναρτήσεις `circular_queue_full`, `circular_queue_empty`, `enqueue`, `dequeue` αφορούν καθαρά την κυκλική ουρά και οι υλοποιήσεις τους δεν περιέχουν συναρτήσεις της βιβλιοθήκης `pthread`. Το ίδιο συμβαίνει και με τις συναρτήσεις `insort` και `perform_quicksort` που μας δόθηκαν έτοιμες. Αντίθετα, οι υπόλοιπες συναρτήσεις `get_job_from_circular_queue`,

`perform_insertion_sort`, `add_jobs_to_circular_queue`, `quicksort_thread_function` περιέχουν συναρτήσεις της βιβλιοθήκης `pthread`, όπως και η κύρια ρουτίνα. Σε όποια σημεία τα νήματα χρησιμοποιούσαν κοινές μεταβλητές χρησιμοποιήθηκαν οι μέθοδοι `pthread_mutex_lock` και `pthread_mutex_unlock` ώστε οι μεταβλητές να έχουν πάντα έγκυρες τιμές, ενώ χρησιμοποιήθηκαν και άλλες συναρτήσεις για την επικοινωνία μεταξύ των νημάτων.