## Λειτουργικά Συστήματα,

## Αναφορά Πρώτης Εργασίας

Ψαλίδας Ζήσης, ΑΜ: 3369

Βάρελης Παναγιώτης, ΑΜ: 3388

Έγινε τροποποίηση στον τρόπο εκτέλεσης του προγράμματος. Πλέον ο σωστός τρόπος κλήσης του είναι:

./kiwi-bench write <count> read <number of threads> <count>

Το πρόγραμμα λειτουργεί με έναν μόνο γραφέα και πολλαπλούς αναγνώστες και δίνει προτεραιότητα στον γραφέα.

Καλό θα ήταν να γίνει πρώτα ένα write, προκειμένου να υπάρχουν κλειδιά για να τα βρει το read. Το πρόγραμμα έχει δοκιμαστεί μέχρι 1.000.000 κλειδιά.

Αρχεία που τροποποιήθηκαν:

/Kiwi-source

|——-/bench

|——-bench.c

|——-kiwi.c

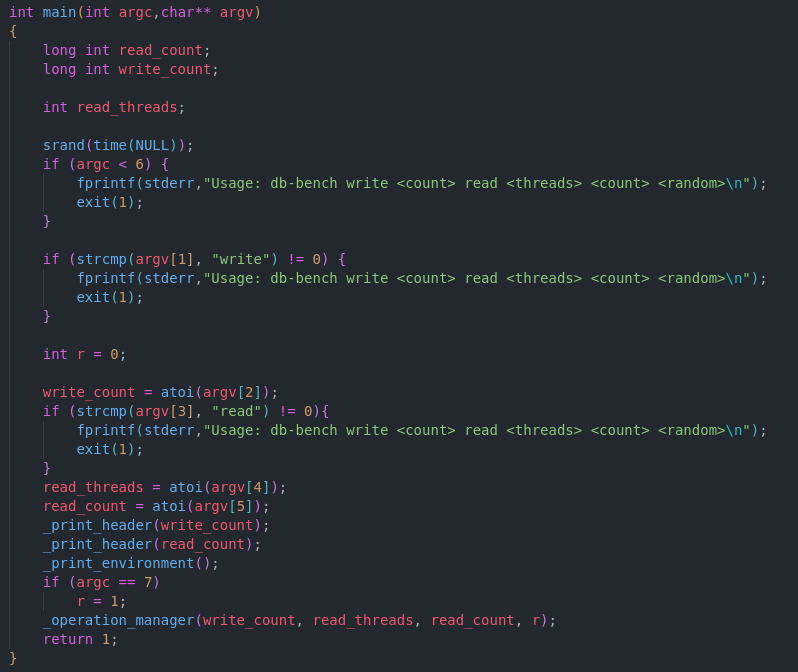
|——-Makefile

|——-/engine

|——-db.c

|——-db.h

**bench.c**



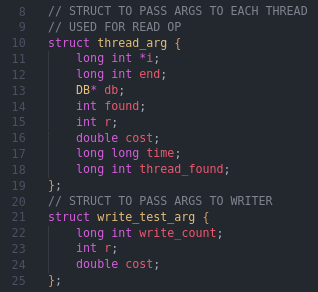
Έγινε αλλαγή στον τρόπο κλήσης του προγράμματος, όπως αναφέρθηκε και πριν.

**Makefile**

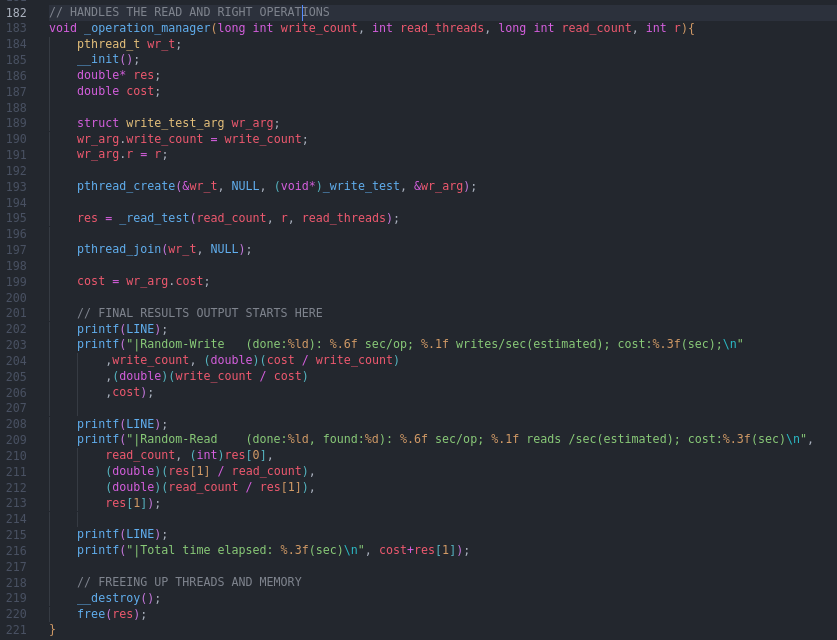


Προσθέσαμε το flag –lrt, για να μπορούμε να χρησιμοποιήσουμε σεμαφόρους.

**kiwi.c**



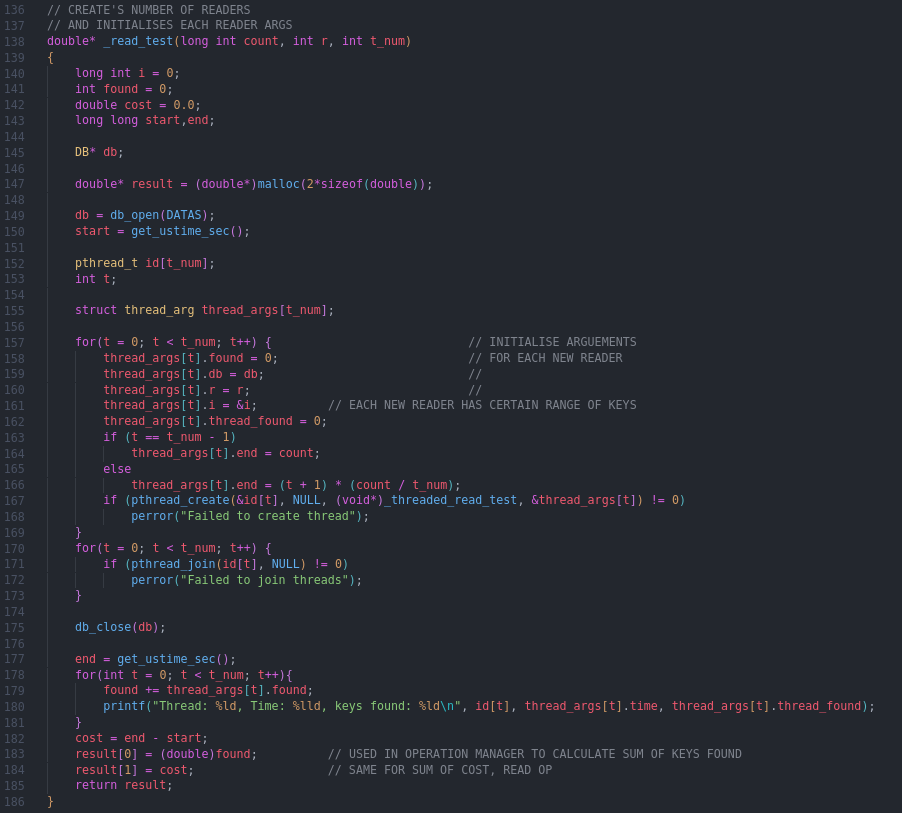
Δημιουργήσαμε δύο structs, τα οποία κρατάνε τα arguments για τις συναρτήσεις που καλούμε μέσω threads.



Η συνάρτηση \_operation\_manager αρχικά αρχικοποιεί τα semaphores και τα mutexes καλώντας την συνάρτηση \_init, την οποία δημιουργήσαμε στο αρχείο db.c. Μετά καλεί με thread την \_write\_test και της περνάει τα arguments που χρειάζεται μέσω struct τύπου write\_test\_arg. Έπειτα καλεί την συνάρτηση \_read\_test η οποία είναι υπεύθυνη για την δημιουργία των read threads. Όταν έχουν τελειώσει όλα τα threads, καταστρέφει τα semaphores και mutexes, και τυπώνει τα αποτελέσματα.

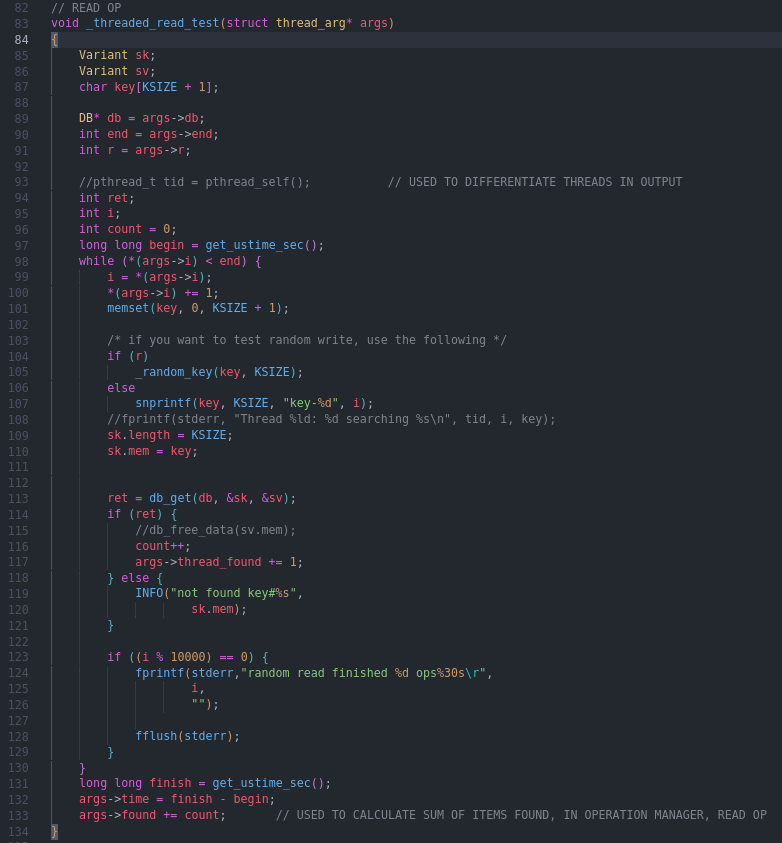


Η \_write\_test πλέον καλείται από thread. Στην συνάρτηση δίνουμε ένα struct με τα arguments που θα χρειαστεί, ενώ ταυτόχρονα κρατάμε την τιμή του cost για να το χρησιμοποιήσουμε μετά τη λήξη του thread. Δεν έχει γίνει κάποια άλλη σημαντική αλλαγή από τον αρχικό κώδικα της \_write\_test.



Η συνάρτηση \_read\_test δημιουργεί μία λίστα από structs τύπου thread\_args, τα οποία έχουν τα arguments για κάθε κλήση της συνάρτησης \_threaded\_read\_test. Έπειτα αναθέτει τιμές στα arguments και δημιουργεί t\_num αριθμό από threads για διάβασμα.

Αφού τελειώσουν όλα τα read threads, υπολογίζουμε το άθροισμα των εγγραφών που βρέθηκαν και το επιστρέφουμε.



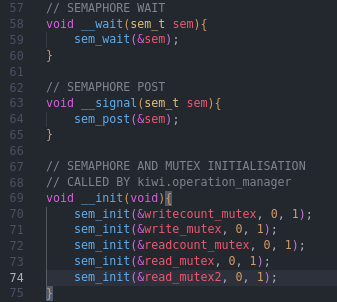
Η συνάρτηση \_threaded\_read\_test χρησιμοποιείται για να κατακερματίσουμε το διάβασμα εγγραφών από την βάση δεδομένων. Όλα τα thread έχουν ένα κοινό counter, το i και λειτουργούν μέχρι το i να φτάσει το count που δόθηκε από τον χρήστη. Στην αρχή κάθε επανάληψης, το thread παίρνει την τιμή του i και την αυξάνει κατά ένα, για το επόμενο thread που θα πάει να το διαβάσει. Ο υπόλοιπος κώδικας είναι ίδιος με αυτόν την \_read\_test που μας δόθηκε.

**db.h**

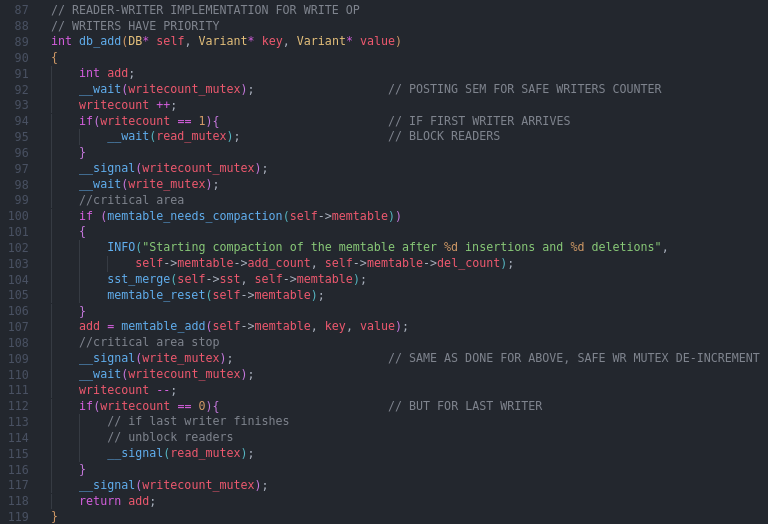


Προσθέσαμε το semaphore.h, για να μπορέσουμε να χρησιμοποιήσουμε semaphores στο πρόγραμμα και δηλώσαμε δύο καινούριες συναρτήσεις του db.c, οι οποίες καλούνται στο kiwi.c.

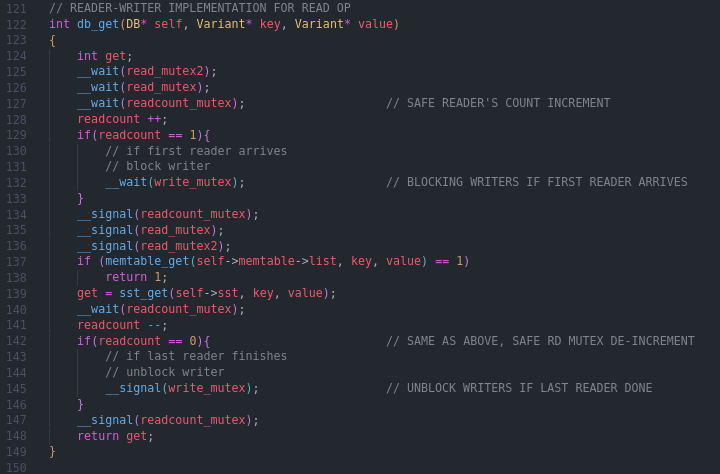
**db.c**



Δημιουργήσαμε 4 καινούριες συναρτήσεις, οι \_\_wait και \_\_signal είναι απλά βοηθητικές ενώ οι \_\_init και \_\_destroy αρχικοποιούν και δημιουργούν τα semaphores και mutexes.

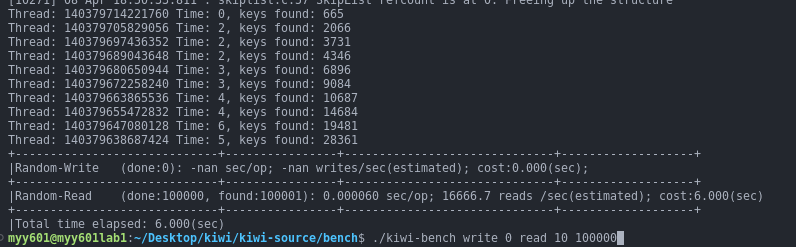
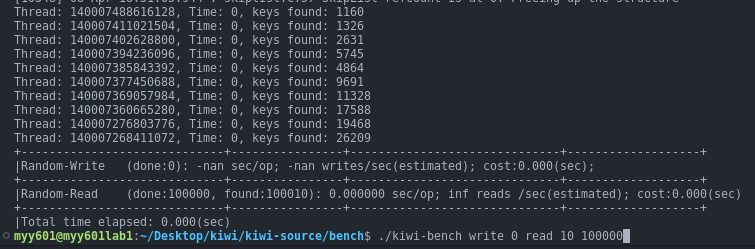


Αρχικά κάνουμε ασφαλή αύξηση του writecount. Στέλνουμε σήμα στα readers ότι υπάρχει writer που εκτελεί δουλειά, προκειμένου να περιμένουν. Αφού τελειώσει το memtable\_add, Στέλνουμε πάλι σήμα στους readers για να συνεχίσουν να δουλεύουν, αν έχουν τελειώσει όλοι οι writers. Στην συγκεκριμένη εφαρμογή, οι writers έχουν προτεραιότητα.

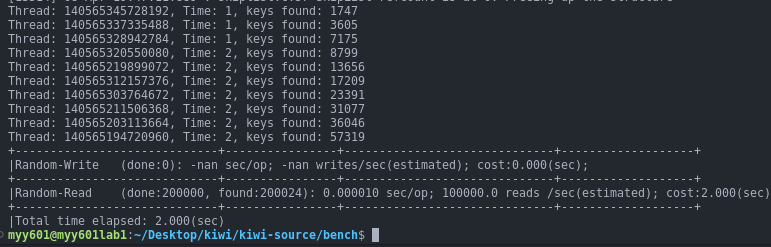
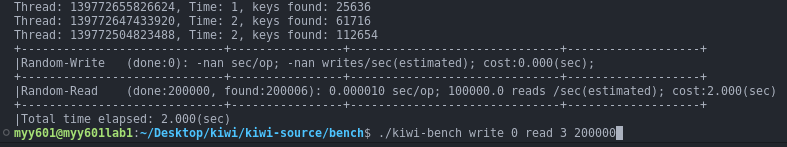
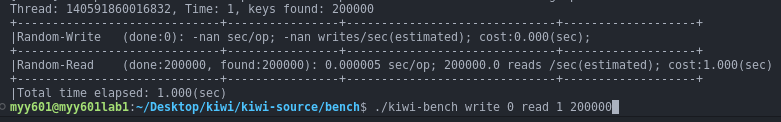


Αρχικά συγχρονίζουμε τους readers μεταξύ τους και το reader με τους writers και έπειτα αυξάνουμε το readcount. Στέλνουμε σήμα στους readers ότι μπορούν να διαβάσουν και στον writer να σταματήσει να γράφει. , εκτελούμε το sst\_get και στέλνουμε σήμα στους writers για να συνεχίσουν να γράφουν, όταν τελειώσει ο τελευταίος reader.

**Στατιστικά**



Το print επηρεάζει πολύ τους χρόνους, ειδικά για μεγάλα νούμερα,οπότε το κλείσαμε. Επίσης η μέτρηση χρόνου δεν είναι και πολύ αξιόπιστη, μιας και γίνεται με δευτερόλεπτα, που είναι ακέραια μέτρηση.



Κατά μέσο όρο δεν παρατηρήσαμε κάποια σημαντική αλλαγή στον χρόνο εκτέλεσης.