ΥΣΒΔ

Εργασία 2

Επεκτατός Κατακερματισμός

Ονοματεπώνυμα Φοιτητών:

Παπαδήμα Θεοδοσία 1115202000162

Παπακώστας Ευάγγελος 1115201800152

Φωτιάδης Ευάγγελος 1115201900301

Contents

hash_file.h..... 2

hash_file.c.....3

main.c.....7

Για να τρέξουμε το πρόγραμμα.....7

hash_file.h

Προστέθηκαν οι ακόλουθες δομές:

1. HT_info, όπου αποθηκεύονται τα μεταδεδομένα του αρχείου

```
typedef struct {
    bool is_ht;
    int fileDesc;
    int global_depth;
    int max_records;
    int* ht_array;
    int ht_array_length;
    int ht_array_head;
    int ht_array_size;
    int num_blocks;
} HT_info;
```

2. και HT_block_file, όπου αποθηκεύονται τα δεδομένα κάθε μπλοκ

```
typedef struct {
    int num_records;
    int local_depth;
    int max_records;
    int next_block;
    int indexes_pointed_by;
} HT_block_info;
```

hash_file.c

Στο αρχείο έχει οριστεί, αρχικά, ως κοινόχρηστος ένας πίνακας τύπου int (hash_table), που αντιπροσωπεύει τον πίνακα κατακερματισμού.

Συναρτήσεις:

- intToBinary

Παίρνει έναν ακέραιο και μετατρέπει κάθε ψηφίο του σε δυαδικό του σε πίνακα ακεραίων, ώστε να παίρνουμε αργότερα στο πρόγραμμα τα σημαντικά bits.

- binaryToInt

Κάνει το αντίθετο → μετατρέπει έναν δυαδικό σε ακέραιο

- hash & hash2

Οι συναρτήσεις κατακερματισμού

- DirtyUnpin

Κάνει dirty & unpin ένα μπλοκ

- checkOpenFiles

Ελέγχει πόσα αρχεία έχουμε ανοιχτά

- HT_Init

Αρχικοποιεί τα στοιχεία του hash_table σε NULL.

- HT_CreateIndex

1) Δημιουργεί ένα νέο αρχείο με το όνομα filename
2) Ανοίγει το αρχείο
3) Και στην 1η κενή (-1) θέση που θα βρει, τοποθετεί το fd του

4) Αναθέτει το πρώτο μπλοκ στο αρχείο όπου αποθηκεύονται τα μεταδεδομένα του αρχείου:
    - δημιουργεί το μπλοκ
    - παίρνει τα δεδομένα του
    - αρχικοποιεί το ht_info
    - το αντιγράφει στη μνήνη
5) Γράφει το μπλοκ με το ht_info στον δίσκο
6) Φτιάχνει 2 μπλοκ που είναι δική μας παραδοχή, ότι πάντα ξεκινάμε με τόσα, όσο και να είναι το όρισμα depth.
7) Τοποθετεί σε αυτά το ht_block_info τους.
8) Ενώνει το ht_array με το αντίστοιχο πίνακα ht_array_global της main, ο οποίος είναι παράλληλος με τον fd_array. (fd_array[0] - ht_array[0] - αφορούν το ίδιο αρχείο με fd = fd_array[0])
9) Κλείνει το αρχείο αφήνοντας κενή τη θέση του fd στον πίνακα

- HT_OpenIndex

1) Ελέγχει αν μπορούμε να ανοίξουμε κι άλλο αρχείο - αν δεν έχουμε τερματίζει
2) Αν έχουμε, το ανοίγει
3) Βρίσει μια κενή θέση στο fd_array για να το βάλει και επιστρέφει τη θέση αυτή στο indexDesc όπως ζητείται
4) Ελέγχει αν το αρχείο είναι όντως HT file
5) Κάνει unpin το μπλοκ και επιστρέφει HT_OK.

- HT_CloseFile

  Καλεί την BF_CloseFile, απελευθερώνοντας τη θέση του αρχείου στον πίνακα με τα fds και κλείνει το αρχείο.

- HT_InsertEntry

1) Παίρνει το πρώτο μπλοκ (μπλοκ 0) του αρχείου και τα μεταδεδομένα του.
2) Υπολογίζει, με την συνάρτηση κατακερματισμού, σε ποιον κάδο πρέπει να εισαχθεί η νέα εγγραφή με τον εξής τρόπο:
    a) Μετατρέπει την hashed τιμή σε binary
    b) Παίρνει τα σημαντικά της bits

c) Τα μετατρέπει σε δεκαδικό αριθμό
και έτσι βρίσκει τον index στου οποίου το μπλοκ
πρέπει να εισαχθεί η εγγραφή

3) Ελέγχει αν το μπλοκ που υπολογίσαμε παραπάνω (target_block) έχει αρκετό χώρο για την εγγραφή.
4) Αν έχει, τότε εισάγει την εγγραφή ως εξής:
   - την αντιγράφει στη τελευταία θέση εγγραφών του μπλοκ
   - ενημερώνει το HT_block_info του μπλοκ και το αντιγράφει στο τέλος του μπλοκ
   - γράφει το μπλοκ στον δίσκο
   - κάνει κατάλληλη απελευθέρωση μνήμης
5) Αν δεν έχει χώρο, τότε εισάγει ξανά τις παλιές εγγραφές μαζί με την καινούργια:
   - αποθηκεύει σε έναν πίνακα τις παλιές
   - Αν το τοπικό βάθος είναι μικρότερο του ολικού:
     α) δημιουργεί & αρχικοποιεί ένα νέο μπλοκ, αυξάνει τα βάθη και αλλάζει τους δείκτες του ht_array σε δύο ίσα υποσύνολα ώστε το ένα να δείχνει στο target_block και το άλλο στο καινούργιο μπλοκ.
     β) ξαναγράφει τα μεταδεδομένα στο νέο μπλοκ και στο μπλοκ 0 για να έχουν τις ενημερωμένες πληροφορίες οι επόμενες αναδρομικές κλήσεις της συνάρτησης
     γ) για κάθε εγγραφή πο υπάρχει ήδη, αντιγράφει σε έναν πίνακα τις εγγραφές που διαβάζει, ώστε να τις ξανα-εισάγει.
     δ) ξανα-εισάγει τις εγγραφές (παλιές→πίνακας + καινούργια) αναδρομικά

   - Αλλιώς αν το τοπικό βάθος είναι ίσο με το ολικό:
     α) διπλασιάζει το μέγεθος του ht_array, κρατώντας κι ένα αντίγραφο του παλιού
     β) για κάθε εγγραφή στον παλιό εξάγει τα πρώτα d σημαντικά bits
     γ) αντιγράφει τον δείκτη του κάδου από τον παλιό στον καινούργιο πίνακα, ώστε κάθε καινούρια σχέση με d+1 bits να δείχνει όπου έδειχνε η θέση του παλιού πίνακα με τα ίδια πρώτα d bits.
     δ) απελευθερώνει τη μνήμη, ενημερώνει τα βάθη και περνάει τα ανανεωμένα δεδομένα στον δίσκο
     ε) εισάγει ξανά την τελευταία εγγραφή, αναδρομικά ώστε να διαχειριστούν την περίπτωση πλέον οι παραπάνω περιπτώσεις.

6) Απελευθερώνει την κατάλληλη μνήμη και επιστρέφει HT_OK.

- HT_PrintAllEntries

1) Παίρνει το πρώτο μπλοκ (μπλοκ 0) του αρχείου και τα μεταδεδομένα του.
2) Παίρνει τον συνολικό αριθμό μπλοκ στο αρχείο
3) Ξεκαρφιτσώνει το μπλοκ 0
4) Για όλα τα υπόλοιπα μπλοκ του αρχείου
    a) Αν έχουμε περάσει στο όρισμα id την τιμή NULL, τότε
    - τα φέρνει στη μνήμη ένα ένα τα μπλοκ του αρχείου προσπερνώντας το μπλοκ 0
    - παίρνει τα δεδομένα του
    - εκτυπώνει ΟΛΕΣ τις εγγραφές του
    - γράφει στο δίσκο το μπλοκ που μόλις διάβασε.
    - αποδεσμεύει τη μνήμη του.

    b) Αν δώσαμε όρισμα ένα συγκεκριμένο id,
        i) Ακολουθεί την ίδια διαδικασία με την InsertEntry() ώστε να φτάσει στο μπλοκ όπου “θα έπρεπε” να βρίσκεται η εγγραφή
        ii) Παίρνει τα δεδομένα του
        iii) Αναζητά σειριακά τις εγγραφές του ελέγχοντας αν κάθε μια είναι αυτή που αναζητούμε
        iv) Αν την βρει την εκτυπώνει, αλλιώς ανδ εν υπάρχει, εμφανίζει αντίστοιχο μήνυμα

- HT_HashStatistics

1) Παίρνει το πρώτο μπλοκ (μπλοκ 0) του αρχείου και τα μεταδεδομένα του.
2) Παίρνει τον συνολικό αριθμό μπλοκ στο αρχείο
3) Αρχικοποιεί τις μεταβλητές min_records, max_records, total_records και avg_records.
4) Για κάθε μπλοκ πέραν του μπλοκ 0:
    - το φέρνει στη μνήμη

- κάνει τους κατάλληλους ελέγχους του αριθμού των εγγραφών στο μπλοκ με τις μεταβλητές που αρχικοποιήθηκαν παραπάνω και ενημερώνει τις τιμές τους, όπου ικανοποιούνται οι συνθήκες ελέγχου
- γράφει το μπλοκ στον δίσκο

5) Εμφανίζει τα αποτελέσματα
6) Κάνει unpin το μπλοκ, απελευθερώνει τη μνήμη που είχε δεσμεύσει και επιστρέφει HT_OK.

main.c

Η main αρχικά θέτει ως στρατηγική αντικατάστασης την LRU και στη συνέχεια καλεί τις συναρτήσεις που υλοποιήθηκαν στο hash_file.c.

Αρχικοποιούμε τους πίνακές μας και καλούμε τις συναρτήσεις HT μία μία κάνοντας τα αντίστοιχα prints.

Σημειώσεις:
Δεν έχουμε υλοποιήσει την αποθήκευση του πίνακα κατακερματισμού κάθε αρχείου σε μπλοκς μέσα στο ίδιο αρχείο, γι' αυτό και χρησιμοποιούμε τα global arrays. Επίσης, δεν έχει τεσταριστεί η δημιουργία πολλαπλών αρχείων.
Έχουμε βάλει ένα print για πριν την κλήση κάθε συνάρτησης και καλούμε την HT_PrintAllEntries για όλες αλλά και για 1 τυχαία εγγραφή.

Για να τρέξουμε το πρόγραμμα

make clean (αν το έχουμε τρέξει ήδη μία φορά)

make hp

make run