Αρχείο Κατακερματισμού

Υλοποιήθηκε από τους:
Αναστασόπουλο Δημήτρη με ΑΜ: 1115201100030
Δημόπουλο Γρηγόρη με ΑΜ: 1115201100198

Στο πρώτο block του αρχείου κατακερματισμού φυλάσσεται το struct HT_info με τις πληροφορίες για το ευρετήριο αλλά φυλάσσεται και πληροφορία για το αν το αρχείο είναι αρχείο κατακερματισμού ή όχι, στα 4 πρώτα byte του block. Στο δεύτερο block φυλάσσεται το ευρετήριο με μέγιστο ολικό βάθος 6. Το ευρετήριο είναι της μορφής (πιθανές τιμές που επιστρέφει η hash function) (αριθμός block που αποθηκεύονται οι εγγραφές που η hash function επιστρέφει αυτή την τιμή για αυτές). Η hash function επιστρέφει integer και όχι αριθμούς στο δυαδικό. Το ευρετήριο είναι 512 byte άρα 512/8=64 sqrt(64)=6 άρα το ολικό βάθος του ευρετηρίου μας φτάνει ως 6. Τα επόμενα block του αρχείου μπορεί να είναι είτε block τύπου bucket είτε block τύπου chane. Τα block τύπου bucket στα πρώτα 4 byte του block αποθηκεύουν το τοπικό βάθος ενώ στα 4 επόμενα αποθηκεύουν το πλήθος των εγγραφών που έχουν μέσα (μέγιστο πλήθος εγγραφών 7).Τα τελευταία 4 byte έχουν των αριθμό του πρώτου block αλυσίδας αν υπάρχει αλλιώς -1.Τα byte απο 504 ως 508 έχουν μια ένδειξη που αρχικοποιείται με -1 ή 0 και χρησιμοποιείται στην HT_GetAllEntries και υποδεικνύει αν έχει τυπωθεί το block ή όχι. Τα block τύπου chane είναι ίδια με τα bucket αλλά στο πεδίο του τοπικού βάθους έχουν -1. Επίσης, χαρακτηριστικό της υλοποίησης μας είναι ότι όταν μια εγγραφή είναι ολόιδια στο πεδίο ευρετηρίου με όλες της άλλες σε ένα γεμάτο bucket δεν θα γίνει διπλασιασμός ευρετηρίου ή split του bucket αλλά θα μπει η εγγραφή στην αλυσίδα του bucket αυτού. Επίσης εγγραφή μπαίνει στην αλυσίδα όταν το ολικό βάθος πρόκειται να πάει πάνω από 6, δηλαδή οι αλυσίδες χρησιμεύουν και σαν κάδοι υπερχείλισης. Όσο αναφορά την υλοποίηση του αλγορίθμου του επεκτατού κατακερματισμού, αυτή έγινε όπως έχει αναφερθεί στο μάθημα.

Πέρα από την υλοποίηση όλων των βασικών συναρτήσεων που είναι το ζητούμενο της άσκησης έχουν δημιουργηθεί και οι παρακάτω βοηθητικές συναρτήσεις:

void print_record(Record record) : τυπώνει ένα Record. (**Δημήτρη Α**.)

int Last_Chane(int fileDesc,int chane) : επιστρέφει το τελευταίο block μιας αλυσίδας.**(Γρηγόρη Δ.)**

void Chane_Rec(void * block,void next,int i,int fileDesc,Record * record)**: παίρνεις στο record την ί εγγραφή στο block με δείκτη block.**(Γρηγόρη Δ.)**

int Chane_Count(int block_num,int * counter,int fileDesc): επιστρέφει των αριθμό των block μιας αλυσίδας χωρίς το τελευταίο και επιστρέφει των αριθμό των εγγραφών του τελευταίου block. **(Γρηγόρη Δ.)**

void * Rec_Key(char * fieldName,Record *record): επιστροφή δείκτη στην τιμή του πεδίου που προσδιορίζεται από το fieldname του record.**(Γρηγόρη Δ.)**

int compare(void* block, char* fieldName, void* value, int use): χρησιμοποιείται για συγκρίσεις. (**Δημήτρη Α**.)

int AllSame(void * block_ptr,char * fieldName,void * value): κοιτάει αν οι εγγραφές ενός block είναι ίδιες με το value στο πεδίο που προσδιορίζεται από το fieldname του record.**(Γρηγόρη Δ.)**

i**nt Mod_Class(int gen_depth,HT_info* header_info,Record record)**: περιπτωσιολογία τύπων για κλήση της συνάρτησης κατακερματισμού. **(Γρηγόρη Δ.)**

i**nt HT_Function(int depth, int akeraio_pedio, char* string_pedio, int length)**: συνάρτηση κατακερματισμού. (**Δημήτρη Α**.)

Ο καταμερισμός της εργασίας για το αρχείο κατακερματισμού έγινε ως εξής:
Open-Close-InsertEntry από τον **Γρηγόρη Δ.**
Create-GetAllEntries από τον **Δημήτρη Α**.

Καθ’ όλη την διάρκεια της εκπόνησης της εργασίας εννοείται πως υπήρξε συζήτηση, συνεργασία και υποστήριξη και από τις δύο μεριές.
Το πρόγραμμα γράφτηκε και δοκιμάστηκε σε περιβάλλον Ubuntu και στους υπολογιστές του εργαστηρίου sun της σχολής μέσω Putty.