ΥΛΟΠΟΙΗΣΗ ΣΥΣΤΗΜΑΤΩΝ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ
ΧΕΙΜΕΡΙΝΟ ΕΞΑΜΗΝΟ 2017-2018
2Η ΕΡΓΑΣΙΑ

ΣΥΜΜΕΤΕΧΟΝΤΕΣ ΣΤΗΝ ΟΜΑΔΑ:

ΕΥΑΓΓΕΛΙΟΥ ΠΑΝΑΓΙΩΤΗΣ AM:1115201500039
ΜΑΣΤΟΡΑΚΗΣ ΕΥΣΤΑΘΙΟΣ-ΑΝΔΡΕΑΣ AM:1115201500092
ΣΠΙΘΑΣ ΕΥΑΓΓΕΛΟΣ AM:1115201500147

Εντολές για μεταγλώττιση και εκτέλεση am_main συναρτήσεων:

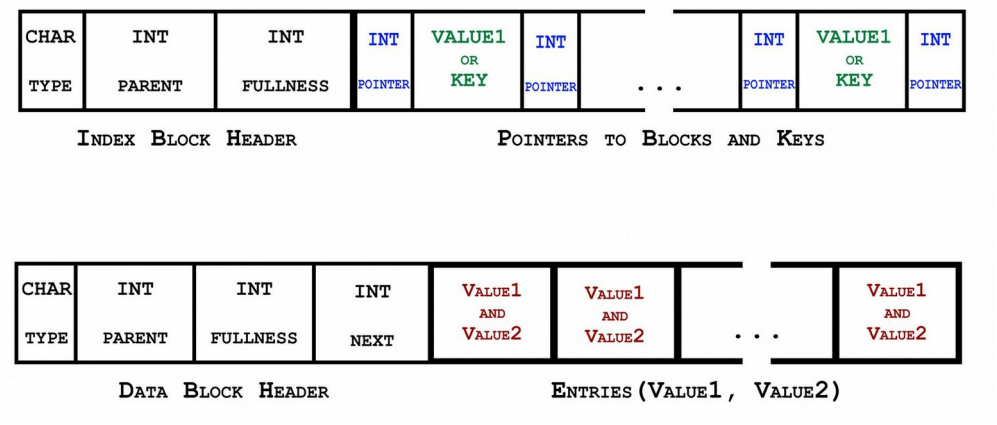
- Για am_main1 -> **make am_main1** και **./build/am_main1**
- Για am_main2 -> **make am_main2** και **./build/am_main2**
- Για am_main3 -> **make am_main3** και **./build/am_main3**
- Για διαγραφή των αρχείων που δημιουργήθηκαν καθώς και των εκτελέσιμων προγραμμάτων τρέχουμε την εντολή **make clean**

Το πρόγραμμα τρέχει σωστά και έχουν υλοποιηθεί όλα όσα αναφέρονται στην εκφώνηση. Σύμφωνα με τον valgrind δεν υπάρχει κανένα error και κανένα memory leak (εκτός από το leak που λέει ότι ένα block είναι still reachable και το οποίο αναφέραμε και στο e-class στην εργασία 1).

Συναρτήσεις Επιπέδου Ευρετηρίου AM (Access Method)

Δομές που χρησιμοποιήθηκαν (structs.h):

- **struct file_info:** Πληροφορίες για κάθε ανοιχτό αρχείο, τις οποίες αποθηκεύουμε στον global πίνακα file_info **open_files.
- **struct entryCoordinates:** Η θέση μιας entry στο B+ δέντρο, δηλαδή ο αριθμός του block που βρίσκεται καθώς και σε ποια θέση του συγκεκριμένου block είναι.
- **struct scan_info:** Πληροφορίες για κάθε ανοιχτή σάρωση, τις οποίες αποθηκεύουμε στον global πίνακα scan_info **open_scans.
- **struct entry:** Αναπαράσταση μιας entry δηλαδή το σύνολο (key,value).
- **struct dataBlockHeader:** Η κεφαλίδα ενός block δεδομένων.
- **struct indexBlockHeader:** Η κεφαλίδα ενός block ευρετηρίου.
  Παρατηρήσεις: 1) Έχουμε κάνει την παραδοχή ότι το fileName θα έχει μέγεθος το πολύ 40 χαρακτήρες (μαζί με το τερματικό \0). Το μέγεθος αυτό μπορεί να αλλαχθεί αλλάζοντας απλώς το πεδίο char fileName[40] του struct file_info.



void AM_Init:

- Αρχικοποίηση του BF επιπέδου χρησιμοποιώντας την πολιτική LRU.
- Δημιουργεί και αρχικοποιεί τους global πίνακες open_files και open_scans μέσω των βοηθητικών συναρτήσεων Open_Files_Init() και Open_Scans_Init().
- Δέσμευση του κατάλληλου χώρου για τα πεδία της global μεταβλητής entryToReturn που είναι τύπου struct entry και η οποία θα χρησιμοποιηθεί προκειμένου να επιστραφούν οι τιμές που πρέπει από την AM_FindNextEntry.

  Παρατηρήσεις: 1) Το μέγεθος που δεσμεύουμε για τα πεδία της global μεταβλητής entry είναι 255 διότι τα πεδία value και key μπορεί να είναι είτε τύπου ‘c’ είτε ‘i’ είτε ‘f’ οπότε δεσμεύουμε για το μέγιστο αριθμό bytes που θα πρέπει να “φυλαχθούν” δηλαδή το μέγιστο μήκος ενός πεδίου με τύπο ‘c’ που είναι 255.

int AM_CreateIndex:

- Ελέγχει τα ορίσματα που παίρνει ώστε να μην έχουν λάθος τιμές μέσω της βοηθητικής συνάρτησης checkTypeLength.
- Αρχικοποιεί το πρώτο block του αρχείου που θα αποτελεί το αναγνωριστικό ότι πρόκειται για AM_FILE καθώς και μερικές πληροφορίες για το B+ δέντρο που αναπαρίσταται μέσω του AM_FILE.
- Το αναγνωριστικό λοιπόν αποτελείται από το *“AM_FILE”* ακολουθούμενο από τον *τύπο και το μέγεθος του πρώτου πεδίου*, *τον τύπο και το μέγεθος του δεύτερου πεδίου* , τον αριθμό του block το οποίο αποτελεί την *ρίζα* του B+ δέντρου (εφόσον είμαστε στο στάδιο της δημιουργίας ενός νέου AM_FILE η ρίζα αρχικά θα έχει την τιμή 0 και έπειτα θα ανανεώνεται κάθε φορά που αλλάζει), τον *μέγιστο αριθμό από pointers* που χωράνε σε ένα block ευρετηρίου, τον *μέγιστο αριθμό από entries* που χωράνε σε ένα block δεδομένων και το *υπόλοιπο μπλοκ* που απομένει γεμίζεται με τον *αριθμό 1*.

int AM_DestroyIndex:

- Εφόσον ελέγξει ότι δεν υπάρχει έστω και μια instance του αρχείου ανοιχτή τότε διαγράφει το αρχείο.

int AM_OpenIndex:

- Ελέγχει αν το αρχείο υπάρχει.
- Το ανοίγει.
- Ελέγχει αν το αναγνωριστικό , δηλαδή το πρώτο block του αρχείου , είναι αυτό που πρέπει να έχει ένα AM_FILE και το οποίο παρουσιάστηκε παραπάνω.
- Εφόσον περάσει από όλους τους ελέγχους , βρίσκει την πρώτη διαθέσιμη θέση στο global πίνακα με τα ανοιχτά αρχεία και εισάγει τις κατάλληλες πληροφορίες.

int AM_CloseIndex:

- Εφόσον ελέγξει ότι δεν υπάρχει ανοιχτή σάρωση πάνω στην συγκεκριμένη instance του αρχείου, κλείνει την συγκεκριμένη instance του.

int AM_InsertEntry:

- Όταν πάει να εισαχθεί η πρώτη τιμή στο αρχείο θα πρέπει να δημιουργηθούν και τα πρώτα μπλοκ. Αυτό θα γίνει με την συνάρτηση Base_Insert. Αυτή η συνάρτηση αρχικά δημιουργεί τη ριζά και έπειτα δημιουργεί το πρώτο μπλοκ δεδομένων, στο οποίο θα γραφτεί η πρώτη εγγραφή. Ο πρώτος δείκτης στην ριζά θα δείχνει στο πρώτο μας μπλοκ δεδομένων.
- Στη συνέχεια εφόσον υπάρχει τουλάχιστον μια τιμή οι εισαγωγές θα γίνονται με την συνάρτηση Insert_Data. Στην InsertEntry θα κληθεί η getDataBlock η οποία θα βρει το μπλοκ δεδομένων στο οποίο θα πρέπει να εισαχθεί η τιμή.
- Η Insert_Data θα πάει να εισάγει τη τιμή στο μπλοκ το οποίο επέστρεψε η getDataBlock. Αν υπάρχει χώρος για να γίνει η εισαγωγή τότε θα χρησιμοποιηθεί η findFirstEntry που θα επιστρέψει το κατάλληλο offset για το που θα μπει η τιμή. Αν είναι μεταξύ άλλων τιμών τότε όσες θα πρέπει να είναι δεξιά θα μετακινηθούν κατάλληλα με τη χρήση της memmove. Αλλιώς αν πρέπει να μπει στο τέλος θα εισαχθεί κατευθείαν εκεί.
- Σε περίπτωση που το μπλοκ είναι γεμάτο και δεν έχει χώρο τότε το μπλοκ θα σπάσει και οι τιμές θα ισομοιραστούν. Αν όμως υπάρχουν πολλές κοινές τιμές ίδιες με την τιμή στην οποία θα γίνει το σπάσιμο θα γίνει κατάλληλη διαδικασία για να μοιραστούν.
- Αφού διαχωριστούν τα μπλοκ, το νέο μπλοκ που φτιάχτηκε θα εισαχθεί στο ευρετήριο. Αφού γίνει αυτό θα πραγματοποιηθεί νέα αναζήτηση και εισαγωγή της τιμής με τον ίδιο τρόπο που έγινε και στην InsertEntry.
- Στην περίπτωση που σπάει ένα μπλοκ δεδομένων θα πρέπει να γίνει εισαγωγή του δείκτη και της πρώτης τιμής κλειδιού του νέου μπλοκ στο ευρετήριο. Αυτό θα γίνει με τη χρήση της συνάρτησης Insert_Index.
- Η Insert_Index έχει παρόμοια συμπεριφορά με την Insert_Data. Αν υπάρχει χώρος στο μπλοκ ευρετηρίου τότε θα βρεθεί η κατάλληλη θέση με την FindFirstKey και θα εισαχθεί με παρόμοιο τρόπο όπως γίνονταν οι εισαγωγές στην Insert_Data.
- Τώρα σε περίπτωση που και αυτό είναι γεμάτο θα πρέπει να σπάσει. Η τιμή κλειδιού, η οποία θα είναι ενδιάμεσα στους δείκτες που διαχωρίζονται, και ο δείκτης του νέου μπλοκ θα εισαχθούν με την Insert_Index στον πάτερα του αρχικού μπλοκ αφού εισαχθεί το αρχικό κλειδί που θέλαμε να εισάγουμε.

  Παρατηρήσεις: 1)Διατήρηση αύξουσας σειράς στα value1 του data block και index block: Χρησιμοποιείται η findFirstEntry/findFirstKey, η οποία μας πληροφορεί για το ποια είναι η κατάλληλη θέση του entry στο συγκεκριμένο μπλοκ. Αν η τιμή επιστροφής είναι -1 τότε απλά κάνουμε εισαγωγή αμέσως μετά το τελευταίο entry. Αλλιώς, πρέπει να 'σπρώξουμε' (memmove) τα υπάρχοντα entries ανάλογα με την τιμή επιστροφής έτσι ώστε να διατηρείται η αύξουσα σειρά των τιμών value1.
  2)Αποφυγή σπασίματος block πάνω σε entries με ίδια value1: Αν κατά το σπάσιμο ενός μπλοκ δεδομένων διαπιστωθεί ότι ο διαχωρισμός θα γίνει πάνω σε entries με ίδιο value1, τότε γίνεται έλεγχος για το αν είναι πιο συμφέρον να μεταφέρουμε όλα αυτά τα ίδια κλειδιά 'δεξιά' στο καινούριο μπλοκ, ή να μείνουν στο 'αριστερό' μπλοκ. Αυτό γίνεται με χρήση δύο μεταβλητών, οπού η κάθε μια κινείται αριστερά και δεξιά του αρχικού breakpoint και μετράνε πόσα ίδια value1 υπάρχουν στη σειρά. Τελικά θα επιλεγεί εκείνη που έχει τα περισσότερα duplicate value1 στη μεριά της. Για παράδειγμα, αν αριστερά του breakpoint υπάρχουν 2 ακόμα ίδια κλειδιά ενώ δεξιά συναντάμε τρία τότε τα πολλαπλά ίδια κλειδιά θα πάνε στο 'δεξί' (καινούριο) μπλοκ.
  3)Περίπτωση που ένα μπλοκ είναι γεμάτο με ίδια κλειδιά: Σε τέτοια περίπτωση, όπως έχει διευκρινιστεί, δε περιμένουμε παραπάνω ίδια κλειδιά από όσα θα χώραγαν σε ένα μπλοκ, οπότε κάθε entry που οδηγείται κατά την εισαγωγή του σε μπλοκ το οποίο είναι γεμάτο με ίδια κλειδιά θα μπει σε καινούριο block. Αν τύχει και το γεμάτο αυτό μπλοκ είναι το πρώτο πρώτο μπλοκ δεδομένων και γίνει εισαγωγή ενός entry με μικρότερο κλειδί τότε δημιουργείται καινούριο μπλοκ, μεταφέρονται όλα τα entries με ίδια κλειδιά σε αυτό και το entry με μικρότερο κλειδί εισάγεται στο άδειο πλέον αρχικό μπλοκ.

int AM_OpenIndexScan:

- Βρίσκει την πρώτη διαθέσιμη θέση στον global πίνακα με τις ανοιχτές σαρώσεις.
- Αρχικοποιεί τα πεδία που αναφέρονται στην τελευταία εγγραφή που διαβάστηκε σε -1, ώστε να γνωρίζει η AM_findNextEntry ότι είναι η πρώτη φορά που καλείται για την συγκεκριμένη ανοιχτή σάρωση και να κάνει τις κατάλληλες ενέργειες.
- Εισάγει τα στοιχεία που αρχικοποιήσαμε στην διαθέσιμη θέση του πίνακα που βρήκαμε.

void *AM_FindNextEntry:

- Αν τα πεδία της τελευταίας εγγραφής που διαβάστηκε έχουν την τιμή -1 σημαίνει ότι είναι η πρώτη φορά που καλείται η συνάρτηση για την ανοιχτή σάρωση που αντιστοιχεί στον αριθμό scanDesc οπότε κάνουμε τις παρακάτω ενέργειες:
  - Καλούμε την συνάρτηση InitFirstEntry ώστε να αρχικοποιηθεί κατάλληλα η πρώτη εγγραφή που θα διαβάσουμε.
  - Επομένως αν ο τελεστής είναι EQUAL,GREATER_THAN ή GREATER_THAN_OR_EQUAL καλούμε τις συναρτήσεις getDataBlock και findFirstEntry οι οποίες θα μας υποδηλώσουν την θέση της πρώτης εγγραφής με τιμή στο κλειδί της ίση με την τιμή value της ανοιχτής σάρωσης ή αν δεν υπάρχει κάποια εγγραφή με τέτοια τιμή θα μας επιστρέψει το που θα έπρεπε να ήταν αυτή η εγγραφή.
  - Αν ο τελεστής είναι NOT_EQUAL,LESS_THAN ή LESS_THAN_OR_EQUAL πάντα θα πρέπει να ξεκινάμε από την εγγραφή με την μικρότερη τιμή κλειδιού οπότε αρχικοποιούμε τις πληροφορίες της τελευταίας εγγραφής που διαβάστηκε σε 2 για τον αριθμό του block (αφού πάντα το block με αριθμό 2 θα έχει την εγγραφή με την μικρότερη τιμή κλειδιού) και την θέση της εγγραφής με τον αριθμό 0 , αφού θα είναι η πρώτη εγγραφή στο μπλοκ (η αρίθμηση των θέσεων των εγγραφών ξεκινάει από το 0).
- Αν τα πεδία έχουν την τιμή -2 σημαίνει ότι υπήρξε κάποιο error την τελευταία φορά που κλήθηκε η AM_FindNextEntry.
- Αν έχουν οποιαδήποτε άλλη τιμή σημαίνει ότι έχουμε ήδη πληροφορίες για την τελευταία εγγραφή που διαβάστηκε οπότε θα ορίσουμε την τιμή της θέσης της τελευταίας εγγραφής να δείχνει στην επόμενη θέση του ίδιου block.
- Εφόσον ορίσαμε τις κατάλληλες τιμές για τα πεδία της τελευταίας εγγραφής (είτε αρχικοποιήσαμε είτε κάναμε ++) καλούμε την συνάρτηση GetEntry η οποία θα μας επιστρέψει την εγγραφή που αντιστοιχή στο lastEntry.blockNum και lastEntry.entryNum (είτε στο ίδιο block , είτε στο επόμενο block , αλλάζοντας τα πεδία στις κατάλληλες τιμές) ή θα επιστρέψει -1 αν δεν υπάρχει αυτή η εγγραφή.
- Έπειτα θα συγκρίνουμε την εγγραφή που επιστράφηκε με την χρήση της συνάρτησης EntryCompare, και αν ικανοποιεί την συνθήκη που ορίζεται από τον τελεστή και την τιμή value της ανοιχτής σάρωσης την επιστρέφουμε.

  Παρατηρήσεις: 1)Η συνάρτηση EntryCompare αν ο τελεστής είναι NOT_EQUAL, GREATER_THAN ή GREATER_THAN_OR_EQUAL θα προσπεράσει όσες τιμές δεν ικανοποιούν την συνθήκη μέχρι να βρει κάποια τιμή που την ικανοποιεί ή μέχρι να μην υπάρχουν άλλες εγγραφές.
  2) Για την επιστροφή της τιμής του δεύτερου πεδίου της εγγραφής που ικανοποιεί την συνθήκη της ανοιχτής σάρωσης χρησιμοποιείται η global μεταβλητή entryToReturn που τα πεδία της έχουν δεσμευτεί καταλλήλως όπως αναφέρθηκε παραπάνω. Αυτό σημαίνει ότι η findNextEntry επιστρέφει κάθε φορά έναν δείκτη που δείχνει πάντα στην ίδια διεύθυνση μνήμης (με διαφορετικό περιεχόμενο κάθε φορά ανάλογα με την τιμή της value) και από εκεί και πέρα είναι ευθύνη της main αν θέλει να κρατήσει το περιεχόμενο της διεύθυνσης επειδή την επόμενη φορά θα αλλάξει και θα έχει το καινούριο περιεχόμενο της επόμενης findNextEntry,πράγμα το οποίο είναι αποδεκτή λύση όπως επισημάνθηκε και στο eclass που σας ρωτήσαμε.

- Βοηθητικές συναρτήσεις:
  - compare: Η συνάρτηση αυτή χρησιμοποιείται σε πολλά σημεία στον κωδικά μας και ο σκοπός της είναι να συγκρίνει τις τιμές στις οποίες δείχνουν οι left και right δείκτες,ανεξαρτήτως του τύπου τους. Για να το κάνει αυτό χρησιμοποιεί και την strcmp.
  - getDataBlock: Η συνάρτηση αυτή ξεκινά από το μπλοκ που υποδηλώνεται από το index και βάση του ορίσματος key ψάχνει στα μπλοκ ευρετηρίου να βρει το μπλοκ δεδομένων στο οποίο θα πρέπει να συναντήσουμε την εισαγωγή με value1 ίσο με το key.
  - findFirstEntry: Η συνάρτηση αυτή ξεκινά με από το μπλοκ δεδομένων που υποδηλώνεται από το blockNum και βάση του ορίσματος key ψάχνει στα συγκεκριμένο μπλοκ να βρει την θέση στην οποία θα πρέπει να βρίσκεται η εισαγωγή με value1 ίσο με το key.
  - findFirstKey: Όμοια με την findFirstEntry αλλά για μπλοκ ευρετηρίου.
  - IsDataBlock: Επιστρέψει αν το μπλοκ που δίνεται σαν όρισμα είναι μπλοκ δεδομένων ή όχι.

int AM_CloseIndexScan:

- Κλείνει την σάρωση που αντιστοιχεί στην τιμή scanDesc.

void AM_PrintError:

- Τυπώνει τα κατάλληλα μηνύματα σφαλμάτων.

void AM_Close:

- Αποδεσμεύει την μνήμη για τους global πίνακες με την χρήση των βοηθητικών συναρτήσεων Open_Files_Destroy και Open_Scans_Destroy.
- Αποδεσμεύει την μνήμη για την global μεταβλητή entryToReturn.