ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
Τμήμα Πληροφορικής και Τηλεπικοινωνιών
1η Εργασία - Τμήμα: Αρτίων Αριθμών Μητρώου
K22: Λειτουργικά Συστήματα – Χειμερινό Εξάμηνο '15
Ημερομηνία Ανακοίνωσης: Τρίτη 6 Οκτωβρίου 2015
Ημερομηνία Υποβολής: Πέμπτη 29 Οκτωβρίου 2015 Ώρα 23:59

Εισαγωγή στην Εργασία:

Ο στόχος αυτής της εργασίας είναι να εξοικειωθείτε με το `Linux/Unix` και τα βασικά εργαλεία προγραμματισμού και ανάπτυξης λογισμικού στο εν λόγω περιβάλλον.

Θα πρέπει να υλοποιήσετε μια δομή που να μπορεί να εισαγάγει/προσπελάσει/τροποποιεί εγγραφές που βρίσκονται αποθηκευμένες στην κυρίως μνήμη. Το κόστος εισαγωγής μίας εγγραφής πρέπει να είναι $O(n)$ όπου n είναι ο αριθμός εγγραφών που υπάρχουν ήδη αποθηκευμένες στην δομή σας. Το κόστος προσπέλασης πρέπει να είναι $O(1)$ και εξειδικευμένες επερωτήσεις εύρους να έχουν ένα γραμμικό κόστος $O(k)$ όπου k είναι ο αριθμός των εγγραφών ενδιαφέροντος. Προφανώς κάτι τέτοιο μπορούμε να το επιτύχουμε με κάποιο είδος κατακερματισμού (hashing) σε συνδυασμό με μια γραμμική δομή που μπορεί να δώσει γρήγορες απαντήσεις 'εύρους' (range questions). Ο κατακερματισμός που θα πρέπει να χρησιμοποιήσετε σε αυτή την άσκηση είναι ο *Γραμμικός Κατακερματισμός* ή γνωστός και σαν *Linear Hashing* [1, 2, 4].

Οι εγγραφές που θα χρησιμοποιήσετε έχουν να κάνουν με την δαπάνη πελατών σε μια εταιρεία σουπερμάρκετ. Η εφαρμογή σας πρέπει να δίνει την δυνατότητα σε υπαλλήλους τις εταιρείας όχι μόνο να βλέπουν τα ποσά που έχουν δαπανηθεί από συγκεκριμένους πελάτες αλλά και να επιτυγχάνουν εισαγωγές/ενημερώσεις εγγραφών, καθώς επίσης, και να κάνουν επερωτήσεις για ομάδες χρηστών με βάση το ποσό δαπάνης που έχουν μέχρι στιγμής.

Μερικές βασικές προϋποθέσεις για την παραπάνω εφαρμογή είναι οι εξής:

1. Η βασική δομή των δεδομένων –που μπορεί να είναι μία λίστα- οργανώνεται με βάση το ποσό δαπάνης που στην συγκεκριμένη δομή παίζει το ρόλο του κλειδιού.
2. Η παραπάνω δομή δέχεται εισαγωγές, επερωτήσεις και προσαυξήσεις στο ποσό δαπάνης.
3. Η εγγραφή κάθε πελάτη αποτελείται από ποσό δαπάνης, το `customerid` (μοναδικό) καθώς επίσης και το όνομα, επίθετο, διεύθυνση, ταχυδρομικό κώδικα, και πόλη διαμονής του πελάτη.
4. Η δομή γραμμικού κατακερματισμού θα χρησιμοποιείται ώστε να επιτύχουμε $O(1)$ χρόνο προσπέλασης στην βασική δομή που διαθέτει τις εγγραφές.
5. Η εφαρμογή θα πρέπει να μπορεί και να διαβάσει δεδομένα στην εκκίνηση της από ένα αρχείο εισόδου.
6. Το πρόγραμμα σας θα πρέπει –όποτε αυτό απαιτείται– να ελευθερώνει όλη την μνήμη που έχει δεσμεύσει. Το ίδιο ισχύει για τον τερματισμό της εφαρμογής.

Μέθοδοι προσπέλασης σαν και αυτές που αναφέρονται παραπάνω είναι πολύ κοινές σε υλοποιήσεις συστημάτων ανάκτησης πληροφορίας, μηχανών αναζήτησης, και πληροφοριακών συστημάτων.

Διαδικαστικά:

Το πρόγραμμά σας θα πρέπει να γραφτεί σε *C* (ή *C++* αν θέλετε αλλά χωρίς την χρήση STL/Templates) και να τρέχει στις μηχανές `Linux workstations` του τμήματος.

Το πρόγραμμα σας (source code) πρέπει να αποτελείται από **τουλάχιστον** δυο (και κατά προτίμηση πιο πολλά) διαφορετικά αρχεία. Η χρήση του separate compilation είναι *επιτακτική!*

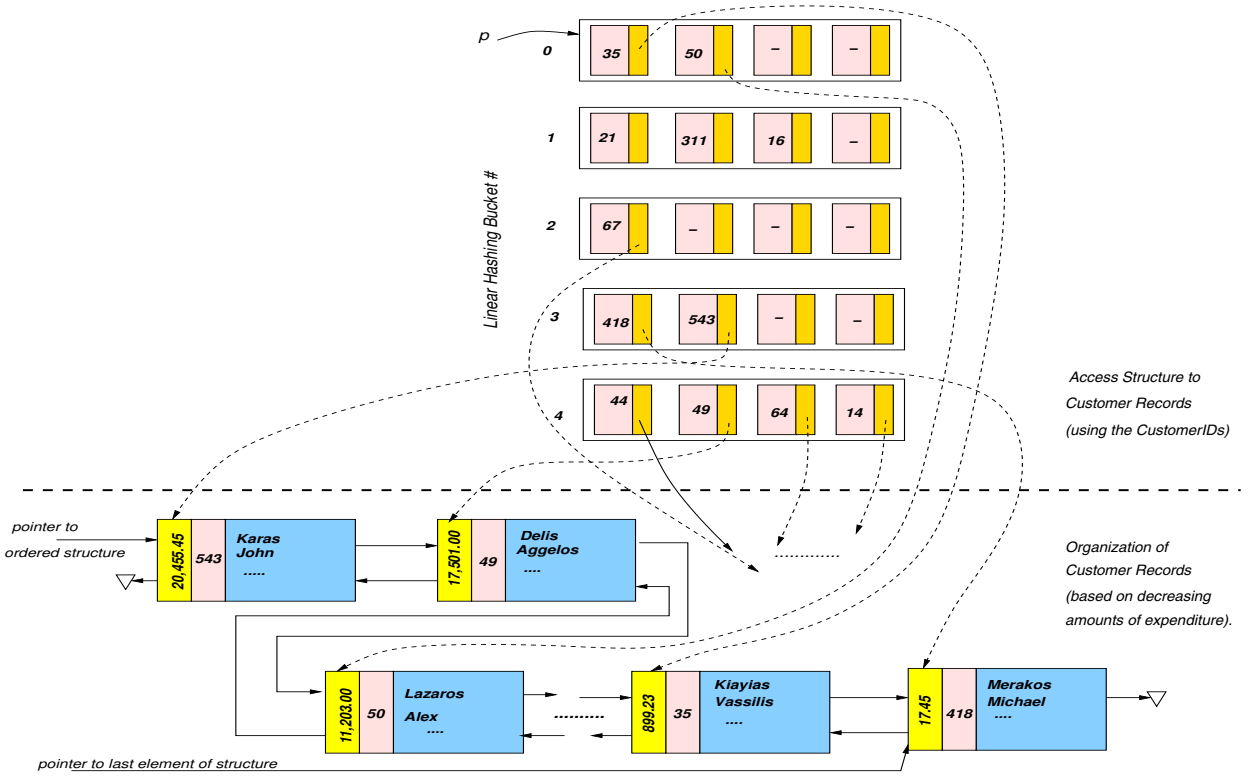
Παρακολουθείτε την ιστοσελίδα του μαθήματος `http://www.di.uoa.gr/~ad/k22/` για επιπρόσθετες ανακοινώσεις αλλά και την ηλεκτρονική-λίστα (η-λίστα) του μαθήματος στο URL `https://piazza.com/uoa.gr/fall2015/k22sectiondelis/home`

- Υπεύθυνοι για την άσκηση αυτή (ερωτήσεις, αξιολόγηση, βαθμολόγηση κλπ.) ο κ. Γιάννης Χρόνης `i.chronis+AT-di` και ο κ. Μάνος Αγγελογιαννόπουλος `eangelog+AT-di`.
- Στην διάρκεια των μαθημάτων των δύο πρώτων εβδομάδων κυκλοφορούμε hard-copy λίστα στην τάξη στην οποία θα πρέπει να δώσετε το όνομά σας και το `Unix` user-id σας.

  Με αυτό το τρόπο μπορούμε να γνωρίζουμε ποιος/-α προτίθεται να υποβάλλει την πρώτη άσκηση και να προβούμε στις κατάλληλες ενέργειες για την τελική υποβολή της άσκησης.
- Με βάση την παραπάνω λίστα, θα σας γράψουμε στο σύστημα `piazza.com` και πιο συγκεκριμένα στο `https://piazza.com/uoa.gr/fall2015/k22sectiondelis/home`. Μόλις αποδεχτείτε ένα 'κωδικό' που θα σας σταλεί, μπορείτε να κάνετε ερωτήσεις και να δείτε απαντήσεις ή/και διευκρινήσεις που δίνονται σχετικά με την άσκηση.
- Η χρήση του `piazza.com` για το Χειμερινό Εξάμηνο του 2015 παραμένει δωρεάν και για αυτό τεχνικά θέματα υποστήριξης που μπορεί να προκύψουν απο την λειτουργία του μπορούν να λυθούν σε χρονικό διάστημα ημερών.

Η Σύνθεση Δομής που θα Δημιουργήσετε:

Το Σχήμα 1 δείχνει μια μερική αλλά αντιπροσωπευτική κατάσταση της σύνθετης δομής που θα δημιουργήσετε και ονομάζεται twa (tiny warehouse).



Σχήμα 1: Παράδειγμα της Σύνθετης Δομής για την Εφαρμογή *twa*

Οι εγγραφές πελατών είναι οργανωμένες με την βοήθεια μιας αμφίδρομης λίστας (κάτω τμήμα του Σχήματος 1).

Οι εγγραφές είναι ταξινομημένες σε σχέση με το πόσο που έχει δαπανηθεί σε *φθίνουσα* σειρά. Προφανώς προσαυξήσεις στο ποσό ενός πελάτη μπορεί να φέρουν αναδιάταξη στην σειρά.

Κάθε εγγραφή έχει όλα τα στοιχεία πληροφορίας για τον πελάτη και περιέχει το `customerid` (κλειδί για πελάτη), επίθετο, όνομα, διεύθυνση, Τ.Τ., πόλη και φυσικά ποσό δαπάνης του εν–λόγω πελάτη μέχρι στιγμής.

Η προσπέλαση στις εγγραφές επιτυγχάνεται με την βοήθεια linear hashing που βασίζεται στην χρήση κουβάδων (buckets) [1, 2, 3] χρησιμοποιώντας το κλειδί `customerid` και την τεχνική load-factor για να αποφασίζουμε πότε διαμοιράζεται (δηλ. γίνεται split) ένας κουβάς. Κάθε κουβάς έχει ένα μέγιστο σταθερό αριθμό από λ εγγραφές. Όταν υπάρχει υπερχείλιση, ένα νέο bucket την φορά παραχωρείται από την μνήμη και με αυτό το τρόπο, αποφεύγουμε μεγάλα κομμάτια της μνήμης να δεσμεύονται άσκοπα (όπως π.χ., συμβαίνει στο extensible hashing[3]). Το επάνω τμήμα του Σχήματος 1 δίνει την εικόνα του linear hashing σε μια συγκεκριμένη χρονική στιγμή.

Ο αριθμός των αναμενόμενων εισαγωγών/προσαυξήσεων εγγραφών στην δομή δεν είναι γνωστός εξ΄ αρχής. Οι δύο δομές του Σχήματος 1 μπορούν να μεγαλώσουν και να τροποποιηθούν κατά βούληση.

Η συγκεκριμένη μορφή που παίρνουν ο πίνακας καταμερισμού και η δομή αποθήκευσης έχουν να κάνουν με επιλογές σχεδιασμού που θα πρέπει να πάρετε και να περιγράψετε στο σύντομο αρχείο σχεδιασμού που θα υποβάλετε μαζί με τον κώδικα σας. Συνολικά ωστόσο οι δομές που τελικά θα χρησιμοποιήσετε θα πρέπει να είναι *δυναμικές*.

Γραμμή Κλήσης της Εφαρμογής:

Η εφαρμογή μπορεί να κληθεί με τον παρακάτω αυστηρό τρόπο:
./twa -l DataFile -b Bucketentries -f LoadFactor -p OperationsFile -c config-file
όπου

- *twa* είναι το εκτελέσιμο,
- *DataFile* είναι το (δυαδικό) αρχείο που έχει τις εγγραφές που θα πρέπει *αρχικά* να εισαχθούν στις δομές της εφαρμογής,
- *Bucketentries* είναι ο μέγιστος αριθμός κλειδιών (και σχετικών δεικτών) που κάθε κουβάς του γραμμικού κατακερματισμού μπορεί να αποθηκεύσει.
- *LoadFactor* είναι ο δεκαδικός αριθμός που προσδιορίζει το παράγοντα φορτίου για την δομή hashing.
- *OperationsFile* είναι ένα αρχείο σειριακής εισόδου με λειτουργίες που θα πρέπει να εφαρμοστούν στην δομή (δηλ. ερωτήσεις, εισαγωγές, προσαυξήσεις ποσών σε συγκεκριμένες εγγραφές, κλπ.). Τέτοιες λειτουργίες μπορούν να εισαχθούν και από το prompt της εφαρμογής,
- *config-file* είναι ένα προαιρετικό configuration αρχείο που μπορείτε να το χρησιμοποιήσετε ώστε να παραμετροποιήσετε εσείς όπως επιθυμείτε την εφαρμογή σας.

Οι σημαίες *-l/-p/-f/-b/-c* μπορούν να χρησιμοποιηθούν με οποιαδήποτε σειρά στην γραμμή εκτέλεσης του προγράμματος και δεν μπορείτε να κάνετε αλλαγές στη ονομασία τους. Από τις παραπάνω in-line παραμέτρους μόνο *-b/-f* είναι υποχρεωτικές για την επιτυχή εκτέλεση του προγράμματος. Ωστόσο όλες θα πρέπει να υλοποιηθούν.

Περιγραφή της Διεπαφής του Προγράμματος:

Η εφαρμογή μέσω ενός prompt επιτρέπει στον χρήστη να αλληλεπιδρά με την δομή και να ανασύρει, αποθηκεύει, ή υπολογίζει διάφορες πληροφορίες με τις παρακάτω εντολές (η μορφή των εντολών είναι αυστηρή και θα πρέπει να χρησιμοποιηθούν όπως ακριβώς ορίζονται παρακάτω):

1. `l(oad) DataFile`: εισήγαγε στην δομή τις εγγραφές που περιγράφονται στο αρχείο `DataFile`. Το

πρόγραμμά διαβάζει το παραπάνω (δυαδικό) αρχείο που δεν έχει προκαθορισμένο μέγεθος όσον αφορά στον αριθμό εγγραφών. Οι εγγραφές εισάγονται μία μία και οι δομές μεγαλώνουν δυναμικά. Ταυτόχρονα με την δημιουργία του πίνακα κατακερματισμού δημιουργείται και η λίστα με τις εγγραφές.

2. `i(nsert) customerid lastname firstname stree number postcode city amount`: εισήγαγε στην δομή ένα πελάτη με τα προφανή παραπάνω στοιχεία που ζει στη πόλη `city` και έχει την δαπάνη `amount`. Το `customerid` λειτουργεί σαν κλειδί για την εισαγωγή σχετικών της εγγραφής στο πίνακα κατακερματισμού.

   Αν η εγγραφή (πελάτης) υπάρχει ήδη τότε το πόσο που είναι αποθηκευμένο στην εγγραφή προσαυξάνεται με το `amount`. Τα υπόλοιπα στοιχεία της εισαγωγής (δηλ. `lastname, firstname, street`, κλπ .) αγνοούνται.
3. `q(uery) customerid`: ανέσυρε και τύπωσε όλη την εγγραφή του πελάτη με κλειδί `customerid`.
4. `t(op) k`: παρουσίασε τις εγγραφές για τους `k` πρώτους πελάτες που εμφανίζονται στην λίστα εγγραφών.
5. `b(ottom) k`: παρουσίασε τις εγγραφές για τους `k` πελάτες που έχουν δαπανήσει τα μικρότερα ποσά.
6. `a(verage)`: παρουσίασε μέσο όρο του ποσού που έχει δαπανηθεί από όλους τους πελάτες μέχρι στιγμής.
7. `r(ange) custid1 custid2`: παρουσίασε όλους τους πελάτες που εμφανίζονται μεταξύ `custid1` και `custid2` σε *αύξουσα* σειρά δαπάνης. Εδώ, θα πρέπει να χρησιμοποιήσετε το γεγονός οτι οι εγγραφές είναι διατεγμένες σε *φθίνουσα* σειρά σε σχέση με το ποσό που ο κάθε πελάτης έχει δαπανήσει.
8. `p(ercentile) custid`: για τον πελάτη `custid` δώσε την εκατοστιαία μονάδα που βρίσκεται όσον αφορά στο ποσό δαπάνης (π.χ., 92%).
9. `e(xit)`: το πρόγραμμα απλά τερματίζει αφού ελευθερώσει πρώτα όλο το χώρο που έχει καταλάβει στην μνήμη.

Κάθε εγγραφή που αποθηκεύεται στην δομή(-ες) αποτελείται από:

- Αριθμός πελάτη (μοναδικός ακέραιος αριθμός ή μοναδική σειρά χαρακτήρων)
- Όνομα συνδρομητή (σειρά χαρακτήρων)
- Επίθετο συνδρομητή (σειρά χαρακτήρων)
- Οδός (σειρά χαρακτήρων)
- Νούμερο (σειρά χαρακτήρων ή ακέραιος)
- Πόλη Κατοικίας (σειρά χαρακτήρων)
- Ταχυδρομικός Τομέας (5–ψήφιο νούμερο)
- Ποσό δαπάνης (πραγματικός αριθμός)

Η μορφή εξόδου της κάθε μία από τις παραπάνω εντολές δίνεται με λεπτομέρεια στην σελίδα του μαθήματος.

Χαρακτηριστικά του Προγράμματος που Πρέπει να Γράψετε:

1. Δεν μπορείτε να κάνετε pre-allocate οποιοσδήποτε χώρο αφού η δομή(-ές) θα πρέπει να μπορεί(-ουν) να μεγαλώσει(-ουν) χωρίς ουσιαστικά κανέναν περιορισμό όσον αφορά στον αριθμό των εγγραφών που μπορούν να αποθηκεύσουν. Η χρήση στατικών πινάκων/δομών που δεσμεύονται στην διάρκεια της συμβολομετάφρασης του προγράμματος σας δεν είναι αποδεκτές επιλογές.
2. Εκτός τις βασικές δομές μπορείτε να χρησιμοποιήσετε οποιαδήποτε άλλη δομή σας δίνει την δυνατότητα να απαντήσετε όλα τα ερωτήματα με πολυπλοκότητα *O(1)* και όσον αφορά τα ερωτήματα εύρους *O(κ)*. Θα πρέπει να περιγράψετε στην υποβληθείσα γραπτή εξήγηση πώς πετυχαίνετε κάτι τέτοιο.
3. Πριν να τερματίσει η εκτέλεση της εφαρμογής, το περιεχόμενο των δομών απελευθερώνεται συνολικά.
4. Αν πιθανώς κάποια κομμάτια του κωδικά σας προέλθουν από κάποια δημόσια πηγή, θα πρέπει *να δώσετε αναφορά στη εν λόγω πηγή* είτε αυτή είναι βιβλίο, σημειώσεις, Internet URL κλπ. και να εξηγήσετε πως ακριβώς χρησιμοποιήσατε την εν λόγω αναφορά.

5. Έχετε πλήρη ελευθερία να επιλέξετε το τρόπο με τον οποίο τελικά θα υλοποιήσετε βοηθητικές δομές.

Τι πρέπει να Παραδοθεί:

1. Μια σύντομη και περιεκτική εξήγηση για τις επιλογές που έχετε κάνει στο σχεδιασμό του προγράμματος σας (1-2 σελίδες σε ASCII κειμένου είναι αρκετές).
2. Οπωσδήποτε ένα Makefile (που να μπορεί να χρησιμοποιηθεί για να γίνει αυτόματα το compile του προγράμματος σας). Πιο πολλές λεπτομέρειες για το (Makefile) και πως αυτό δημιουργείται δίνονται στην ιστοσελίδα του μαθήματος.
3. Ένα tar-file με όλη σας την δουλειά σε έναν κατάλογο που πιθανώς να φέρει το όνομα σας και θα περιέχει όλη σας την δουλειά δηλ. source files, header files, output files (αν υπάρχουν) και οτιδήποτε άλλο χρειάζεται.

Άλλες Σημαντικές Παρατηρήσεις:

1. Οι εργασίες είναι **ατομικές**.
2. Το πρόγραμμα σας θα πρέπει να τρέχει σε `Linux` αλλιώς **δεν θα βαθμολογηθεί**.
3. Αν και αναμένεται να συζητήσετε με φίλους και συνεργάτες το πως θα επιχειρήσετε να δώσετε λύση στο πρόβλημα, αντιγραφή κώδικα (οποιαδήποτε μορφής) είναι κάτι που **δεν επιτρέπεται** και δεν πρέπει να γίνει. Οποιοσδήποτε βρεθεί αναμεμειγμένος σε αντιγραφή κώδικά **απλά παίρνει μηδέν** στο μάθημα. Αυτό ισχύει για **όσους εμπλέκονται ανεξάρτητα** από το ποιος έδωσε/πήρε κλπ.
4. Προγράμματα που **δεν χρησιμοποιούν** separate compilation χάνουν αυτόματα 10% του βαθμού.
5. Σε καμιά περίπτωση τα Windows *δεν είναι επιλέξιμη* πλατφόρμα για την παρουσίαση αυτής της άσκησης.

Αναφορές

[1] G. Boetticher. Video Presentation *"Linear Hashing - Part 1"*. `https://www.youtube.com/watch?v=Yw1ts57uL7c`, February 2011.

[2] G. Boetticher. Video Presentation *"Linear Hashing - Part 2"*. `https://www.youtube.com/watch?v=RNoboXmu3zA`, February 2011.

[3] J. Chronis, Linear Hash with Load Factor Note, `http://cgi.di.uoa.gr/~ad/k22/LinearHashingBy-J.Chronis.pdf`, September 2015.

[4] D. Zhang et al. Linear Hashing. Technical report, `http://cgi.di.uoa.gr/~ad/MDE515/e_ds_linearhashing.pdf`. October 2012. Notes for MDE515.