Για την εργασία συνεργαστήκαμε δύο άτομα! Αναφέρονται τα όνοματα από κάτω(μαζί με ΑΜ),καθώς και τι έκανε ο καθένας μας.

ΔΗΜΗΤΡΙΟΣ ΣΟΥΜΗΣ(ΑΜ:1115201600157)

Στην operations_in_tree.c ακολουθούνται τα στάδια που περιγράφονται στο πρώτο βήμα της εκφώνησης.

Αρχικά στη συνάρτηση a() με την χρήση της stat() ελέγχεται αν υπάρχει το δοθέν path-κατάλογος στην πηγή και το ανάλογο στον προορισμό.Εάν υπάρχει στην πηγή αλλά όχι στον προορισμό, δημιουργείται και στον προορισμό.Έπειτα,κάνουμε insert στο backup tree το path του καινούριου κατάλογου και επιστρέφουμε από τη συνάρτηση τη θέση του στο δέντρο. Εάν υπάρχει και στον προορισμό και είναι file, αποσυνδέεται το file με τη χρήση της unlink() και δημιουργείται ο κατάλογος με mkdir. Αρχικά διαγράφουμε το file και από το backup tree,κάνουμε insert στο backup tree το path του καινούριου κατάλογου και επιστρέφουμε από τη συνάρτηση τη θέση του στο δέντρο. Εάν υπάρχει και στον προορισμό και είναι κατάλογος απλά επιστρέφουμε από τη συνάρτηση τη θέση του στο backup tree.

Στη συνέχεια στη συνάρτηση b() αρχικά ελέγχουμε με την stat() άν ένα path δεν υπάρχει στην πηγή αλλά υπάρχει στον προορισμό και αυτό το path είναι κατάλογος και καλούμε την recursion_for_b() η οποία διαγράφει έναν κατάλογο διαγράφοντας πρώτα αναδρομικά όλα τα περιεχόμενα του. Επίσης ενημερώνεται το backup tree διαγράφοντας αντίστοιχα τον συγκεκριμένο κατάλογο.

Επιπρόσθετα στη συνάρτηση c() αρχικά ελέγχουμε με την stat() άν ένα path υπάρχει στην πηγή αλλά δεν υπάρχει στον προορισμό και αυτό το path είναι file. Αν ο προορισμός του source inode είναι NULL τότε γίνεται link() το όνομα του αρχείου στο source που το παίρνουμε από την λίστα ονομάτων του inode με το path στον προορισμό και προσθέτουμε το path στο backup tree. Αλλιώς δημιουργούμε το αρχείο στον προορισμό και αντιγράφουμε τα περιεχόμενα του και προσθέτουμε το path στο backup tree.Τέλος σε κάθε περίπτωση επιστρέφουμε τη θέση του αρχείου στο backup tree.

Στη συνέχεια στη συνάρτηση d() αρχικά ελέγχουμε με την stat() άν ένα path δεν υπάρχει στην πηγή αλλά υπάρχει στον προορισμό και αυτό το path είναι file. Αποσυνδέουμε το αρχείο με unlink και διαγράφουμε το path από το backup tree.

Τέλος στη συνάρτηση e() ελέγχουμε με την stat() αν υπάρχει το path και στην πηγή και στον προορισμό και ότι είναι file. Αν ο προορισμός του inode του source είναι ίδιο με το inode του προορισμού δεν χρειάζεται καμία ενέργεια.Αν όχι, αποσυνδέουμε το αρχείο, διαγράφουμε το path από το backup tree, καλούμε την συνάρτηση c() η οποία επιστρέφει τη δομή inode του προορισμού και αναθέτουμε την τιμή αυτή στον προορισμό του inode του source.

Στο αρχείο inotify.c έχει χρησιμοποιηθεί το δείγμα που δίνεται στη σελίδα του μαθήματος για το πως λειτουργεί η inotify. Σε κάθε λήψη event ελέγχουμε πως δεν έχει length 0 και ότι δεν είναι ακριβώς ίδιο με το προηγούμενο event και εκτελούμε ό,τι ζητείται στο table of events ενημερώνοντας πάντα τα δέντρα (όποια έξτρα πληροφόρηση υπάρχει υπό τη μορφή σχολίων).

Σημείωση:Το event IN_DELETE_SELF έχει len 0.

Περνάμε σαν ορίσματα στη συνάρτηση 2 πίνακες οι οποίοι περιέχουν υπό τη μορφή strings όλα τα directory paths της πηγής και του προορισμού. Για κάθε κατάλογο της πηγής αναθέτουμε και έναν watch descriptor με την inotify_add_watch(). Η όλη διαδικασία λήψης event τερματίζει μόλις ο χρήστης πατήσει κατά την διάρκεια της εκτέλεσης Ctrl+C μέσω ενός signal δηλαδή.

ΜΑΧΖΟΥΜΠ ΑΔΑΜ(ΑΜ:1115201600099)

Για τα inodes χρησιμοποιούμε μια απλή δομή λίστας.Η create_inode δημιουργεί ενα νέο κόμβο η list_insert τον προσθέτει η check_list τσέκαρει αν υπάρχει ήδη το inode στη λίστα,ώστε να μη προστεθεί ξανά το ίδιο.Η rm_inode_list βρίσκει το inode και αν υπάρχει μόνο ένα όνομα που δείχνει σ αυτό το διαγράφει αλλιώς απλα διαγράφει το όνομα,που της δώσαμε.Στη πρώτη περίπτωση αν είμαστε σε backup tree inode αποσυνδέουμε το αντίστοιχο destination του Slist.Για τα ονόματα του inode έχουμε μια απλή λίστα με απλές συναρτήσεις διαγραφής και push.

Για την υλοποιήση του δέντρου χρησιμοποιούνται κόμβοι οι οποίοι :

α) δείχνουν στον επομένο κόμβο άρα στον ίδιο φάκελο.

β) δείχνουν σε directories και σε files(άρα έχει νόημα να χρησιμοποιείται μόνο εάν ο κόμβος αντιπροσοπεύει directory ώστε να δείχνει σε αρχεία και υποφακέλους του).

Στην εισαγωγή ενός κόμβου ψάχνουμε κάθε φορά το subpath δηλαδή αν εμείς βρισκόμαστε στο home/4i στο δέντρο και θέλουμε να βρούμε το home/4i/a/k/p.c,ψάχνουμε μέσω της strstrt αρχικά το home/4i/a kai μετά τον υποφάκελο k.Όταν βρούμε το φάκελο τότε ανάλογα με το αν είναι αρχείο η directory προστίθεται στη

κατάλληλη λίστα κόμβων του φακέλου.Με παρόμοια λογική λειτουργεί η find και η delete, Η τελευταία επιίσης διαγράφει και όλους τους υποφακέλους και αρχεία αν ο επιθυμητό κόμβος είναι φάκελος,ενώ καλεί και την rm_node_list.Η modify διαγράφει το προηγούμενο inode και τροποποιεί τα στοιχεία του επιθυμητού κόμβου.Οι συναρτήσεις traverse απλά διασχίζουν τους κόμβους του δέντρου,η μια για να βρεί όλα τα ονόματα των φακέλων και η άλλη για να κάνει print όλο το δέντρο.

Τα δέντρα αρχικοποιούνται ώστε να αντιπροσωπεύουν τους φακέλους που δόθηκαν στο terminal με τη χρήση της opendir,readdir και stat ώστε να ανοιχτούν όλοι οι φάκελοι και να βρεθούν όλα τα αρχεία και να προστεθούν στα δέντρα.Μέτα γίνεται συγχρονισμός των δέντρων ως εξής :

Αν έχουμε και στους δύο κόμβους NULL επιστρέφουμε

Αν έχουμε NULL source κόμβο τότε οι next κόμβοι του backup διαγράφονται(ανδρομική κλήση της συνάρτησης) ,αφού είναι πλεονάζοντες.Αυτό γίνεται μέσω της d,b και οι διαγραφές γίνονται στο σύστημα αρχείων αλλα και στο bakup δέντρο.

Αν έχουμε στο bakcup NULL τότε λείπουν κόμβοι απο τον backup άρα προσθέτουμε μέσω των κατάλληλων συναρτήσεων(a,c) στο σύστημα αρχείων αλλα και στο bakup δέντρο.

Δεν έχουμε πουθενά null και έχουμε ίδιο όνομα και είναι αρχεία τότε καλείται η e για να τσεκάρει τα inodes.Αν είναι φάκελοι δε διαγράφεται τίποτα.

Στις παραπάνω περιπτώσεις γίνεται ανδρομική κλήση της συναρτης στους επομένους κόμβους και των δύο αλλά και στους υποφακέλους και στα αρχεία αν οι δύο κόμβοι είναι φάκελοι.

Σε περίπτωση μη ίδιων ονομάτων διαγράφεται ο κόμβος του backup και προστίθεται σε αυτόν ο κόμβος του source,αν όμως δεν έχουμε ίδιο όνομα σημαίνει ότι δε προστέθηκε ο τελευταίος ακριβώς στο ίδιο σημείο των δέντρων με το source όποτε πρέπει να εξέτασουμε τον ίδιο backup κόμβο με τον επόμενο του source και μέτα αν είναι φάκελοι να εξεταστούν οι επομένοι(αν δε γινόταν αυτό παρατηρούσαμε μη διαγραφή κάποιων φακέλων και προσθήκη αρχείων σε λάθους υποφάκελους σε συγκεκριμένες περιπτώσεις) καθώς και των υποφακέλων και των αρχείων τους.

Σε κάθε περίπτωση συνδέονται τα inodes,να τονίσουμε επίσης οτί στην αρχή μπορεί να υπάρχουν ίδια όνοματα στους δύο φακέλους αλλά δε συνδέονται οπότε ο backup tree

στην ουσία φτιάχνεται από την αρχή εκτός των φακέλων του!

Compile command : make

Clean command : make clean

Count command : make count

Run command : ./mirr (source) (backup)

,οπού (source),(bakcup) αναγκαία ορίσματα και αντιπροσωπεύουν τα path του source directory και του backup directory αντίστοιχα!