Προγραμματισμός Συστήματος project 1

Το πρόγραμμα αυτό συγκρίνει τα 2 αρχεία που θα του δοθούν σαν ορίσματα στη γραμμή εντολών και παρουσιάζει τις διαφορές τους αν υπάρχουν όπου αυτές υπάρχουν όπως ζητήθηκε στην εκφώνηση. Το πρόγραμμα κάνει compile μέσω του makefile που δημιούργησα για τις απαιτήσεις του project. Στη συνέχεια εκτελείται παίρνοντας 4 ορίσματα στη γραμμή εντολής όπως ζητήθηκε στην εκφώνηση. Αντιμετώπισα ένα πρόβλημα γράφοντας το συγκεκριμένο πρόγραμμα και αυτό παρουσιάστηκε στο ότι το tree fan-out ήταν μεταβλητού μεγέθους. Δεν κατάφερα να το λύσω και άλλαξα την υλοποίηση μου έτσι ώστε το δέντρο που θα χρησιμοποιηθεί για τη σύγκριση είναι δυαδικό γνωρίζοντας σαφώς ότι θα επηρεάσει τη βαθμολογία μου για το project. Το όρισμα για το tree fan-out το παίρνει κανονικά αλλά δεν το χρησιμοποιεί πουθενά. Αρχικά στη main διαβάζεται το πρώτο αρχείο που δόθηκε από τη γραμμή εντολών. Σύμφωνα με το page-size που δόθηκε στη γραμμή εντολών διαβάζεται και ένα ανάλογο κομμάτι του αρχείου (page), υπολογίζεται το digest της με τις συναρτήσεις κατακερματισμού που δόθηκαν και στη συνέχεια γίνεται η εισαγωγή του κάθε digest στο δέντρο σύγκρισης που δημιουργείται δυναμικά για το πρώτο αρχείο. Αμέσως μετά την συνάρτηση εισαγωγής καλείται η συνάρτηση ελέγχου (check) τόσες φορές όσες και το ύψος της ρίζας προκειμένου να μην υπάρχουν κόμβοι με παιδιά τα οποία το digest τους είναι υπολογισμένο ενώ το δικό τους digest δεν έχει υπολογιστεί και αποθηκευτεί. Στη συνέχεια όταν δεν υπάρχουν πλέον στοιχεία να εισαχθούν καλούνται οι συναρτήσεις check και last check για την ολοκλήρωση του δέντρου. Η last check υπολογίζει και αποθηκεύει το digest κόμβων οι οποίοι έχουν ένα παιδί και συνεπώς η check δεν μπορούσε να τους γεμίσει. Αφού δημιουργηθεί το δέντρο του πρώτου αρχείου γίνονται οι ανάλογες ενέργειες για το δεύτερο αρχείο. Στη συνέχεια και δεδομένου ότι έχουν δημιουργηθεί τα δύο αυτά δέντρα καλείτε η συνάρτηση σύγκρισης (compare) στην οποία αν υπάρχουν διαφορές στα δύο αρχεία εμφανίζει τις σελίδες στις οποίες υπάρχουν οι διαφορές αυτές. Τέλος τα δύο δέντρα καταστρέφονται προκειμένου να ελευθερωθεί η μνήμη που χρησιμοποιήθηκε στην πορεία εκτέλεσης του προγράμματος.
Αναλυτικά:
-κάθε κόμβος. Έχει ένα αριστερό και ένα δεξί παιδί τα οποία είναι NULL αν ο κόμβος είναι φύλλο. Έχει ένα unsigned int που είναι το digest του και 3 int.
Done: για done=0 το digest του κόμβου δεν έχει υπολογιστεί, για done=1 το δdigest έχει υπολογιστεί.
Page: η σελίδα της οποίας το digest βρίσκεται στον κόμβο. 0 αν ο κόμβος δεν είναι φύλλο.
Level: το επίπεδο του κόμβου το οποίο μεγαλώνει όσο ανεβαίνουμε στο δέντρο. 0 αν ο κόμβος είναι φύλλο.
Αναλυτικά για τις συναρτήσεις:

dhmiourgia(): δημιουργεί την πρώτη ρίζα του δέντρου δεσμεύοντας μνήμη για αυτήν και αρχικοποιώντας το level της με 0 και το done με 0.

eisagwgh(): αρχικά ελέγχει αν ο κόμβος (current) είναι κόμβος επιπέδου 1 (αν δηλαδή τα παιδιά του είναι κόμβοι φύλλα) και αν δεν έχει υπολογιστεί το digest του. Αν ισχύουν αυτά δημιουργείται ο κόμβος που θα εισαχθεί στο δέντρο, εισάγεται στο κενό παιδί της current και η εισαγωγή τελειώνει. Αν όχι τότε γίνεται ένας έλεγχος για τον υπολογισμό του digest της ρίζας του δέντρου. Αν υπολογιστεί (done=1) τότε δημιουργείται νέα ρίζα για το δέντρο και το αριστερό της παιδί δείχνει την παλιά ρίζα. Στη συνέχεια παίρνοντας το level του επιπέδου κάτω από το επίπεδο της ρίζας το χρησιμοποιώ στην while για να δημιουργηθούν αν δεν υπάρχουν τόσοι κόμβοι μέχρι να φτάσω στο επίπεδο ένα, να χρησιμοποιήσω τον κόμβο αυτό σαν νέο current και να καλέσω την eisagwgh ξανά για την εισαγωγή του digest.

Check(): ελέγχει αν το επίπεδο του κόμβου που δόθηκε σαν όρισμα είναι 1 ή 0. αν είναι 1 τότε δεν χρειάζεται ο έλεγχος γιατί έχει ήδη υπολογιστεί από την eisagwgh και αν είναι 0 δεν χρειάζεται κάποιος υπολογισμός οπότε επιστρέφει. Αν δεν επιστρέψει ελέγχει αν το digest του κόμβου είναι

υπολογισμένο. Αν είναι τότε επιστρέφει. Αν όχι ελέγχει αν το αριστερό του παιδί έχει υπολογισμένο digest. Αν έχει τότε ελέγχει το δεξί του παιδί. Αν αυτό είναι NULL τότε επιστρέφει καθώς δεν μπορεί να υπολογιστεί το digest του κόμβου. Αν δεν είναι ελέγχει αν το digest του δεξιού του παιδιού είναι υπολογισμένο. Αν είναι υπολογίζει το digest του κόμβου σύμφωνα με τα digest των παιδιών του. Αν το digest του δεξιού του παιδιού δεν είναι υπολογισμένο τότε καλείται η check με όρισμα το δεξί παιδί του κόμβου. Αν το digest του αριστερού παιδιού του κόμβου δεν είναι υπολογισμένο καλείται η check με όρισμα το αριστερό παιδί του κόμβου.

last_check(): ελέγχει αν ο κόμβος που δόθηκε σαν όρισμα έχει υπολογισμένο digest. Αν έχει τότε επιστρέφει καθώς δεν χρειάζεται να υπολογίσει κάτι. Αν δεν είναι ελέγχει αν το αριστερό παιδί του κόμβου έχει υπολογισμένο digest. Αν όχι καλείται η last_check έχοντας ως όρισμα το αριστερό παιδί του κόμβου. Αν είναι υπολογισμένο ελέγχει αν το δεξί του παιδί είναι NULL. Αν είναι υπολογίζει το digest του κόμβου λαμβάνοντας υπόψιν το digest του αριστερού παιδιού αλλιώς εάν το done του δεξιού παιδιού του κόμβου είναι 0 καλείται η last_check για το δεξί αυτό παιδί.

Compare(): η compare ουσιαστικά υλοποιεί τον αλγόριθμο της σελίδας 9 της εκφώνησης.

Katastrofh(): η συνάρτηση αυτή απελευθερώνει αναδρομικά τη μνήμη που δέσμευσαν τα 2 δέντρα.