2η Εργασία στα
Λειτουργικά Συστήματα

Ονοματεπώνυμο: Χριστοδούλου Δημήτρης
Α.Μ. : 1115201500179

Folder Structure

Οι φάκελοι της εργασίας είναι οργανωμένοι ως εξής:

- exe, που περιέχει τα εκτελέσιμα μετά την εντολή make.
- inc, ο οποίος περιέχει τα header αρχεία της εργασίας.
- src, ο οποίος περιέχει τον πηγαίο κώδικα της εργασίας.
- tests, που περιέχει το αρχείο tests.c, το οποίο χρειάζεται για κάποια unit tests.
- tmp, στον οποίο δημιουργούνται τα FIFOs που χειρίζεται το πρόγραμμα.

Compile - Execute - Dependencies

Χρησιμοποιήστε την εντολή make για να κάνετε compile όλα τα εκτελέσιμα που έχουν καθοριστεί από την εκφώνηση.

Χρησιμοποιήστε την εντολή make clean για να καθαρίσετε το project από όλα τα .o αρχεία, όλα τα εκτελέσιμα αρχεία και όλα τα FIFOs που υπάρχουν στον φάκελο temp.

Συνιστάται η χρήση της make clean μετά από κάθε εκτέλεση του προγράμματος, καθώς εάν μείνουν pipes στον φάκελο και δημιουργηθούν καινούρια με ίδια ονόματα είναι πολύ πιθανό να υπάρχει πρόβλημα στην εκτέλεση.

Το πρόγραμμα **χρειάζεται** την βιβλιοθήκη cunit για να κάνει compile, καθώς τρέχει κάποια unit tests. Για να εγκαταστήσετε την βιβλιοθήκη χρησιμοποιήστε την εντολή **make cunit**.

Για να εκτελέσετε το πρόγραμμα **χωρίς την εκτέλεση** unit tests, γράφετε:

./exe/myfind [arguments], όπου [arguments] οι απαιτούμενες από την εκφώνηση παράμετροι με οποιαδήποτε σειρά.

Για να εκτελέσετε το πρόγραμμα **με την εκτέλεση** unit tests, γράφετε:

./exe/myfind [arguments] -test, όπου [arguments] οι απαιτούμενες από την εκφώνηση παράμετροι με οποιαδήποτε σειρά.

Σύντομη εξήγηση προγράμματος

Το αρχείο tree.c κατασκευάζει κάποια δεδομένα που χρησιμοποιούνται από τους υπόλοιπους κόμβους και τα στέλνει με την exec στο rootNode.

Αυτό με τη σειρά του περιέχει τις συναρτήσεις για τον υπολογισμό των διάφορων στατιστικών και καλεί τον πρώτο splitter-merger κόμβο, ο οποίος χρησιμοποιεί την exec αναδρομικά για να κατασκευάσει το δέντρο μέχρι το h-1 ύψος. Ταυτόχρονα κατασκευάζονται τα FIFOs και γίνεται κλήση της read, η οποία σταματάει την εκτέλεση των splitter-mergers, μέχρι να διαβάσουν αποτελέσματα από τα leaf nodes.

Τέλος, τα leaf nodes διαβάζουν από το αρχείο και μόλις βρούν εγγραφή που ταιριάζει με το query input το αποστέλλουν προς τα πάνω, δηλαδή προς τον πατέρα splitter-merger. Πριν τελειώσουν, στέλνουν στο root node ένα signal SIGUSR2 και ο root node το χειρίζεται αυξάνοντας μία global μεταβλητή.

Πολυπλοκότητες διαφόρων διεργασιών

Το πρόγραμμα σε γενικές γραμμές δεν είναι αλγοριθμικά απαιτητικό, επομένως δεν υπάρχουν πολλά τμήματά του, των οποίων η πολυπλοκότητα να απαιτεί ανάλυση.

Η αναζήτηση του ερωτήματος στο αρχείο έχει πολυπλοκότητα O(n/d), όπου n είναι το πλήθος των εγγραφών στο αρχείο και d είναι το πλήθος των διεργασιών leaf nodes. Αυτό συμβαίνει καθώς οι διεργασίες αυτές τρέχουν συγχρόνως, με αποτέλεσμα την βελτίωση της χρονικής πολυπλοκότητας.

Η μεταφορά των δεδομένων στον κόμβο ρίζα έχει πολυπλοκότητα O(m*(h-1)), όπου m το πλήθος των εγγραφών μετά την εφαρμογή της επερώτησης και h το ύψος του δέντρου των διεργασιών. Αυτό συμβαίνει καθώς κάθε εγγραφή πρέπει να περάσει από h-1 splitter-merger κόμβους, κάτι που καθιστά την πολυπλοκότητα αυτής της διεργασίας λίγο πιο απαιτητική.

Σχόλια

Μπορείτε να δείτε την σταδιακή ανάπτυξη του κώδικα μέσα από αυτό το link στο Github