Λειτουργικά Συστήματα – Χειμερινό '23 Άρτιοι

4η Άσκηση - Readme

Περιγραφή Προγράμματος και συναρτήσεων

Όλες οι συναρτήσεις που απαιτούνται για την εργασία βρίσκονται στο πηγαίο αρχείο *Compare.cpp*. Οι συναρτήσεις που είναι ορατές στον χρήστη, δηλαδή την *main* του *project* είναι η *Compare* και η *MergeFiles*. Η λειτουργία των συναρτήσεων σε ένα υψηλό επίπεδο αφαίρεσης είναι :

- *Compare* : δέχεται ως ορίσματα δύο συμβολοσειρές ($char*$) που αντιστοιχούν σε δύο ιεραρχίες φακέλων και εκτυπώνει ποια αρχεία είναι μοναδικά σε κάθε φάκελο, όπως αυτό ορίζεται από την εκφώνηση της εργασίας. Σε περίπτωση λάθους εκτυπώνει κατάλληλο μήνυμα και τερματίζει.
- *MergeFiles* : δέχεται ως ορίσματα τρεις συμβολοσειρές που οι πρώτες δυο αντιστοιχούν σε δύο υπάρχοντες ιεραρχίες φακέλων και η τρίτη στον φάκελο που πρόκειται να δημιουργηθεί μέσω *Merging* των πρώτων δύο φακέλων όπως περιγράφεται πάλι στην εκφώνηση. Αν ο τρίτος φάκελος υπάρχει ήδη, απλά διαγράφονται τα περιεχόμενα του. Σε περίπτωση λάθους εκτυπώνεται κατάλληλο μήνυμα λάθους.

Οι παραπάνω συναρτήσεις δηλώνονται σε ένα *header file*(*Compare.h*) το οποίο θα γίνει έπειτα *include* από την *main*(*cmpcats.cpp*). Συνεχίζουμε με μια περιγραφή των βοηθητικών συναρτήσεων εντός του αρχείου *Compare.cpp*, η σειρά είναι η λογική σειρά που χρησιμοποιούνται στην εκτέλεση ενός προγράμματος και όχι η σειρά που ορίζονται στο αρχείοα :

- *void Compare*($char* \ path1, char* \ path2$) : Αυτή είναι η συνάρτηση που καλείται από την *main* για την σύγκριση δύο φακέλων. Δέχεται σαν όρισμα τα *paths* στους δύο φακέλους και αφού ελέγξει ότι πρόκειται για φακέλους αρχικοποιεί μια μεταβλητή τύπου *DataReturned* που είναι μια *struct* που έχει οριστεί στην αρχή του αρχείου. Αυτή η δομή χρησιμοποιείται για να αποθηκεύσει τα μονοπάτια των αρχείων που είναι μοναδικά στους δύο φακέλους σε έναν πίνακα. Όπως καταλαβαίνουμε, το πλήθος των αρχείων αυτών δεν είναι σταθερό, οπότε χρειαζόμαστε μια δυναμική δομή, για να αποφύγουμε την χρήση της *realloc* σε κάθε προσθήκη νέου μονοπατιού, κάτι που θα ήταν ιδιαίτερα χρονοβόρο, ξεκινάμε με *fixed* μέγεθος πίνακα (16) και κάθε φορά που χρειάζεται ο πίνακας μεγαλύτερο χώρο, αυξάνουμε το μέγεθος του κατά 2. Έπειτα, με κατάλληλη χρήση της συνάρτησης *CheckDirectories*, η δομή αυτή ανανεώνεται έχοντας αποθηκευμένα όλα τα μονοπάτια. Τέλος με δύο κλήσεις της συνάρτησης *printStringsStartingWith* εκτυπώνονται όπως ζητάει η εκφώνηση τα μοναδικά αρχεία.
- *void CheckDirectories*($char* \ path1, char* \ path2, DataReturned* \ ToReturn$) : Σκοπός αυτής της συνάρτησης είναι, αφού η συνάρτηση επιστρέψει, ο δείκτης στην δομή *ToReturn* να αποθηκεύει τα μοναδικά αρχεία των δύο ιεραρχιών. Για να γίνει αυτό αρχικά

η συνάρτηση ανοίγει τους φακέλους που αντιστοιχούν στα *path*1 και *path*2. Διαδοχικά θα εντοπίσει τα κοινά αρχεία πρώτα στον φάκελο 1 και μετά στο φάκελο ακολουθώντας την εξής διαδικασία : έχουμε μια εξωτερική λούπα που διατρέχει όλα τα αρχεία του φακέλου 1 και μια εσωτερική που διατρέχει όλα τα αρχεία του φακέλου 2 όπου στην εσωτερική λούπα, ελέγχεται αν το εκάστοτε αρχείο που έχει γίνει $fixed$ από την εξωτερική λούπα, είναι ίδιο με κάποιο από τα αρχεία του φακέλου 2. Για να επιτευχθεί αυτό πρέπει με κατάλληλο τρόπο να κατασκευαστούν τα σχετικά μονοπάτια καθώς και να κληθούν οι συναρτήσεις *LinkSame* και *FileSame* που ελέγχουν αν δύο *links* ή *Regular Files* είναι ίδια, όπως αυτό το ορίζει η εκφώνηση. Πρέπει να σημειώσουμε ότι αυτή η συνάρτηση είναι δυνητικά αναδρομική, αφού σε περίπτωση ελέγχου δύο φακέλων με το ίδιο όνομα, καλείται ξανά. Στο τέλος αυτής της εξωτερικής λούπας, με την βοήθεια της *bool flag* ανανεώνεται κατάλληλα η δομή *ToReturned* σε περίπτωση που το αρχείο είναι μοναδικό. Αντίστοιχη διαδικασία, αλλά με ανεστραμμένες λούπες ακολουθείται για την εύρεση των μοναδικών αρχείων και στο *path*2.

- *bool LinksSame*(*char* $*$ *path*1, *char* $*$ *path*2) : απλή συνάρτηση που δέχεται σαν όρισμα δυο μονοπάτια σε δύο αρχεία (έχει ήδη ελεγχθεί ότι τα σχετικά ονόματα είναι ίδια στην *CheckDirectories*) και αφού ανοίξει τα *links* ελέγχει αν δείχνουν σε ίδιο αρχείο. Για να το κάνει αυτό θα χρειαστεί να διαβάσει το όνομα του αρχείου στο οποίο δείχνει μέσω της *readlink* και έπειτα να κατασκευάσει το πλήρες μονοπάτι. Έπειτα, ανάλογα με τον τύπο αρχείων, καλεί τις κατάλληλες συναρτήσεις, σε περίπτωση *link* σε *link* η συνάρτηση καλείται αναδρομικά. Σημειώνουμε εδώ ότι η συνάρτηση αυτή δεν είναι υπεύθυνη για τα *hard links*, σε τέτοια περίπτωση το πρόγραμμα μας τα θεωρεί ως απλά αρχεία και καλεί την συνάρτηση ελέγχου αρχείων που περιγράφεται ευθύς αμέσως.

- *bool FilesSame*(*char* $*$ *path*1, *char* $*$ *path*2) : μια απλή συνάρτηση που ελέγχει *byte per byte* αν δύο αρχεία είναι ίδια.

- *void TraverseDirectory*(*const char* $*$ *path*, *DataReturned* $*$ *ToReturn*) : μια ακόμη βοηθητική συνάρτηση που χρησιμοποιείται από την *CheckDirectories* και παίρνει σαν όρισμα ένα μονοπάτι σε κάποιον φάκελο και την προς επιστροφή δομή. Η ιδέα πίσω αυτή την συνάρτηση είναι ότι όταν βρεθούν δύο φάκελοι που είναι μοναδικοί, τότε πρέπει να αποθηκευτούν όλα τα μονοπάτια εντός του φακέλου αυτού ή υποφακέλων του. Αυτό είναι που υλοποιεί η συνάρτηση αυτή, αφού τερματίσει, τα μονοπάτια αυτά έχουν αποθηκευτεί με κατάλληλο τρόπο στην μεταβλητή *ToReturn*. Φυσικά η συνάρτηση αυτή μπορεί να κληθεί αναδρομικά.

- *void printStringsStartingWith*(*char* $*$ *array*[], *int size*, *char* $*$ $prefix$) Αφού τερματίσει η *CheckDirectories*, έχουμε στην μεταβλητή *Returned* με κατάλληλο τρόπο αποθηκεύσει τα μονοπάτια στα αρχεία τα οποία είναι μοναδικά και πρέπει να τα τυπώσουμε όπως περιγράφεται στην εκφώνηση. Για να γίνει αυτό έχουμε υλοποιήσει την συνάρτηση *printStringsStartingWith* η οποία παίρνει σαν όρισμα έναν πίνακα από συμβολοσειρές (στην περίπτωση μας τον πίνακα με τα μονοπάτια), το μέγεθος του πίνακα και μια ακόμη συμβολοσειρά που λειτουργεί ως φίλτρο (στην περίπτωση μας είναι τα ονόματα των προς

σύγκριση φακέλων). Αυτό που γίνεται είναι ότι εκτυπώνονται όλες οι συμβολοσειρές του πίνακα $array$ που ξεκινούν με την συμβολοσειρά $prefix$.

Στο σημείο αυτό καλύψαμε όλες τις συναρτήσεις που χρησιμοποιούνται για την περίπτωση σύγκρισης αρχείων. Συνεχίζουμε με τις συναρτήσεις που αφορούν το *Merging*, πάλι σε λογική σειρά :

- $void\ MergeFiles(char * path1, char * path2, char * Destination)$: αυτή είναι η βασική συνάρτηση που καλείται από την *main* και παίρνει σαν όρισμα τα μονοπάτια στους δύο ήδη υπάρχοντες φακέλους και το μονοπάτι του νέου φακέλου. Η υλοποίηση βασίζεται στην εξής απλή σκέψη : τα αρχεία που θα περιέχονται στον φάκελο *Destination* είναι τα όλα τα αρχεία που βρίσκονται στον πρώτο φάκελο ∪ τα μοναδικά αρχεία χωρίς να συμπεριληφθούν δεύτερη φορά τα μοναδικά αρχεία του πρώτου φακέλου. Οπότε η διαδικασία σε υψηλό επίπεδο αφαίρεσης είναι : αποθήκευση των μοναδικών αρχείων, διαγραφή του φακέλου *Destination* (αν υπάρχει), δημιουργία του νέου φακέλου, εντοπισμός των *hardlinks* στους δυο αρχικούς φακέλους, αντιγραφή των περιεχομένων του πρώτου φακέλου στον τελικό φάκελο, ενσωμάτωση των μοναδικών αρχείων του δεύτερου φακέλου και προσθήκη των *hardlinks*. Σημειώνουμε ότι η προσθήκη των *hardlinks* πρέπει να γίνει αναγκαστικά στο τέλος γιατί δεν μπορούμε να έχουμε κάτι σαν *dangling hardlink*. Για την ευκολότερη κατανόηση του προγράμματος έχουν υλοποιηθεί βοηθητικές συναρτήσεις όπως περιγράφονται παρακάτω.

- $void\ DeleteDirectory(const\ char * path)$: μια απλή συνάρτηση που παίρνει σαν όρισμα ένα μονοπάτι σε φάκελο και διαγράφει όλα τα περιεχόμενα του. Για να το κάνει αυτό διατρέχει όλα τα αρχεία που περιέχονται στον φάκελο και ανάλογα με τον τύπο τους τα διαγράφει κατάλληλα, σε περίπτωση υποφακέλων καλείται αναδρομικά.

- $void\ CopyDirectory(constchar * source, constchar * destination, char * * * hardLinks, int\ noofhardlinks)$: συνάρτηση που παίρνει σαν όρισμα ένα μονοπάτι σε έναν ήδη υπάρχοντα φάκελο και ένα μονοπάτι σε έναν φάκελο που πρόκειται να δημιουργηθεί. Αφού δημιουργήσει τον φάκελο, διατρέχει όλα τα αρχεία του πρώτο φακέλου και τα αντιγράφει στον νέο φάκελο αφού πρώτα κατασκευάσει με κατάλληλο τρόπο τα σχετικά μονοπάτια. Καλούνται κατάλληλες συναρτήσεις όπως περιγράφονται παρακάτω σύμφωνα με τον τύπου του εκάστοτε αρχείου, φυσικά και αυτή η συνάρτηση μπορεί να κληθεί αναδρομικά.

- $void\ CopyFile(constchar * source, constchar * destination)$: μια απλή συνάρτηση που *byte per byte* αντιγράφει τα περιεχόμενα του αρχείου *source* στο αρχείο *destination* το οποίο κιόλας δημιουργεί.

- $void\ CopyLink(constchar * source, constchar * destination)$: συνάρτηση που δημιουργεί έναν νέο σύνδεσμο, απλή υλοποίηση μέσω *readlink* και *symlink*. Σημειώνουμε εδώ, ότι δεν χρειάζεται να κατασκευάσουμε το πλήρες μονοπάτι του αρχείου, καθώς η *symlink* θα ορίσει κατάλληλα την ιεραρχία.

- *void copyFiles(char * paths[], const char * path1, const char * destFolder, int numFiles, constchar * path2, char * * * hardLinks, int noofhardlinks)* : αυτή η συνάρτηση αντιγράφει τα μοναδικά αρχεία προσπερνώντας αυτά που ανήκουν στον πρώτο φάκελο. Με την σειρά παίρνει τα εξής ορίσματα : πίνακα από τα μονοπάτια στα μοναδικά αρχεία, το μονοπάτι στον πρώτο φάκελο, το μονοπάτι στον νέο φάκελο, το πλήθος των μοναδικών αρχείων, το μονοπάτι στον δεύτερο φάκελο. Η συνάρτηση διατρέχει τα μοναδικά αρχεία και προσπερνά αυτά που ξεκινούν με την συμβολοσειρά *path2*. Για τα υπόλοιπα αφού πρώτα κατασκευάσει το σχετικό μονοπάτι (μονοπάτι στον φάκελο 2 του αρχικού αρχείου) και το νέο σχετικό μονοπάτι (στον φάκελο προορισμού), ανάλογα με το περί τίνος αρχείου πρόκειται καλεί την κατάλληλη συνάρτηση για την αντιγραφή. Στο σημείο αυτό πρέπει να διαχειριστούμε και την περίπτωση ύπαρξης αρχείων με κοινό όνομα αλλά διαφορετικό περιεχόμενο (δηλαδή δεν είναι ίδια βάσει της εκφώνησης), Τότε θα πρέπει να κρατήσουμε αυτό που τροποποιήθηκε πιο πρόσφατα. Για να το πετύχουμε αυτό στην συνάρτηση μας, αφού κατασκευάσουμε το σχετικό μονοπάτι στον νέο φάκελο, ελέγχουμε αν υπάρχει ήδη αυτό το αρχείο. Αν υπάρχει, τότε θα είχε δημιουργηθεί κατά την αντιγραφή του πρώτου φακέλου. Θα πρέπει τότε να ελέγξουμε πιο τροποποιήθηκε τελευταίο και αν είναι το ήδη υπάρχον στον νέο φάκελο να μην κάνουμε τίποτα, αλλιώς να το διαγράψουμε και να αντιγράψουμε το προς συζήτηση αρχείο. Τέλος, σε περίπτωση που ένα αρχείο είναι *hardlink* (εντοπίζεται μέσω διάσχισης του πίνακα με τα *hardlinks*, περισσότερες πληροφορίες παρακάτω), τότε απλά το προσπερνάμε.

- *void HardlinksFromTo(char * path, Inode * *inodes, int * NumberOfInodes, char * * * *ToReturn, int * NumberOfHardlinks)* : στόχος αυτής της συνάρτησης είναι να επιστρέψει μέσω του ορίσματος *ToReturn* έναν πίνακα $Nx2$ συμβολοσειριών, όπου N το πλήθος των *hardlinks* στον φάκελο *path* και η πρώτη συμβολοσειρά αντιστοιχεί στο *path* στο αρχείο που θεωρείται ως *hardlink* και η δεύτερη στο *path* στο οποίο το *hardlink* 'δείχνει'. Έτσι αν θέλουμε να πάρουμε από το $i-$οστό *hardlink* το μονοπάτι του, αυτό γίνεται μέσω της $ToReturn[i][0]$ ενώ για το μονοπάτι του αρχείο στο οποίο δείχνει χρησιμοποιείται η $ToReturn[i][1]$. Για ευκολότερη δεικτοδότηση, έχουν οριστεί τα εξής με $define$:

  - $\#define\ FROM\ 0$
  - $\#define\ TO\ 1$

  Η λογική είναι ότι μέσω δύο κλήσεων των συναρτήσεων με κατάλληλο τρόπο θα πάρουμε τους πίνακες με τα *hardlinks* και αφού πάρουμε την συνένωση τους θα τους χρησιμοποιήσουμε στις συναρτήσεις που πραγματοποιούν την αντιγραφή των αρχείων, δηλαδή στις *CopyDirectory* και *copyFiles*. Η ιδέα είναι ότι όταν πάει να αντιγραφεί ένα αρχείο που βρίσκεται εντός του πίνακα με τα *hardlinks*, αντί για την *CopyFile* θα δημιουργήσουμε το *link*, για να γίνει αυτό, στις *CopyDirectory* και *copyFiles* προσπερνάμε τα *hardlinks* και έχει δημιουργηθεί κατάλληλη συνάρτηση για την αντιγραφή των *hardlinks*. Φυσικά πρέπει πρώτα να κατασκευάσουμε το μονοπάτι στον νέο φάκελο που δημιουργήθηκε. Σημειώνουμε εδώ ότι δεν χρειάζεται το *hardlink* να είναι το ΄πραγματικό΄ *hardlink*, αφού μάλιστα,

από τη στιγμή που δημιουργηθεί κάποιο *hardlink* είναι αδύνατο να ξεχωρίσεις το πρωτότυπο από το αντίγραφο, απλά έχεις δύο μονοπάτια με κοινό *inode*. Πριν προχωρήσουμε σε μια σύντομη περιγραφή της συνάρτησης, σημειώνουμε ότι έχουμε δημιουργήσει επίσης μια δομή που αναπαριστά το *Inode* η οποία κρατά το *inode* σαν *integer* και μια συμβολοσειρά. Αυτή η συμβολοσειρά αντιστοιχεί στο μονοπάτι που βρέθηκε πρώτο το οποίο αντιστοιχεί στο εκάστοτε *inode*. Τα *inodes* που έχουν βρεθεί μέχρι στιγμής περινιούνται με κατάλληλη τρόπο σαν όρισμα στην συνάρτηση. Ακόμη η συνάρτηση δέχεται σαν όρισμα το μέχρι στιγμής πλήθος των *inodes* καθώς και των *hardlinks*. Οπότε η λογική σειρά της συνάρτησης είναι η εξής : διατρέχει όλα τα αρχεία του φακέλου και υποφακέλων, αν εντοπίσει καινούριο *inode* το αποθηκεύει στον κατάλληλο πίνακα, αν εντοπίσει αρχείο όπου το *inode* του έχει εμφανιστεί ήδη το αποθηκεύει στον πίνακα με τα *hardlinks*.

- *void CopyHardlinks(char * destination, char *** ToReturn, int NumberOf Hardlinks, char * FolderA, char * FolderB)* : η συνάρτηση αυτό χρησιμοποιείται για την δημιουργία των *hardlinks* στον φάκελο προορισμού. Για να το κάνει αυτό διατρέχει τον πίνακα με τα *hardlinks* και λειτουργεί ως εξής : ελέγχει αν το μονοπάτι που θέλει να δημιουργήσει το *harlink* αντιστοιχεί σε άλλο αρχείο. Αν συμβαίνει αυτό πρέπει να κατασκευάσει το μονοπάτι στον αρχικό του φάκελο (λεπτομέρειες για το πως γίνεται αυτό στα σχόλια του προγράμματος) και να το διαγράψει αν το *hardlink* έχει τροποποιηθεί νωρίτερα. Αν το ήδη υπάρχον αρχείο έχει τροποποιηθεί μεταγενέστερα, δεν γίνεται τίποτα. Αν δεν υπάρχει αρχείο, ή το αρχείο που υπήρχε διαγράφηκε με τον τρόπο που περιγράφηκε παραπάνω, τότε παίρνουμε από τον πίνακα με τα *hardlinks* τον μονοπάτι προορισμού και με κατάλληλο τρόπο δημιουργούμε το μονοπάτι στον νέο φάκελο και τελικά δημιουργούμε το *link* με μια απλή κλήση της *link*.

Στο σημείο αυτό τελειώσαμε την περιγραφή των συναρτήσεων
Όσον αφορά την *main* είναι ιδιαίτερα απλή και δεν χρήζει σχολιασμού.

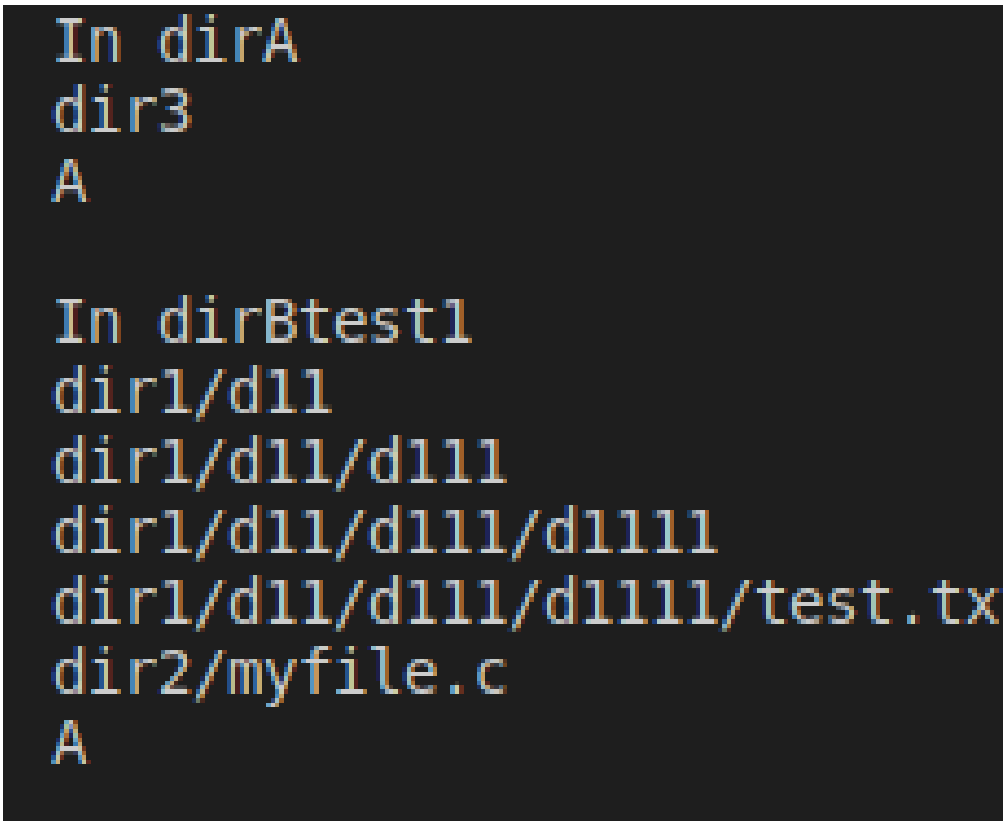
Παραδείγματα Εκτελέσεων

Μερικά παραδείγματα εκτελέσεων είναι :

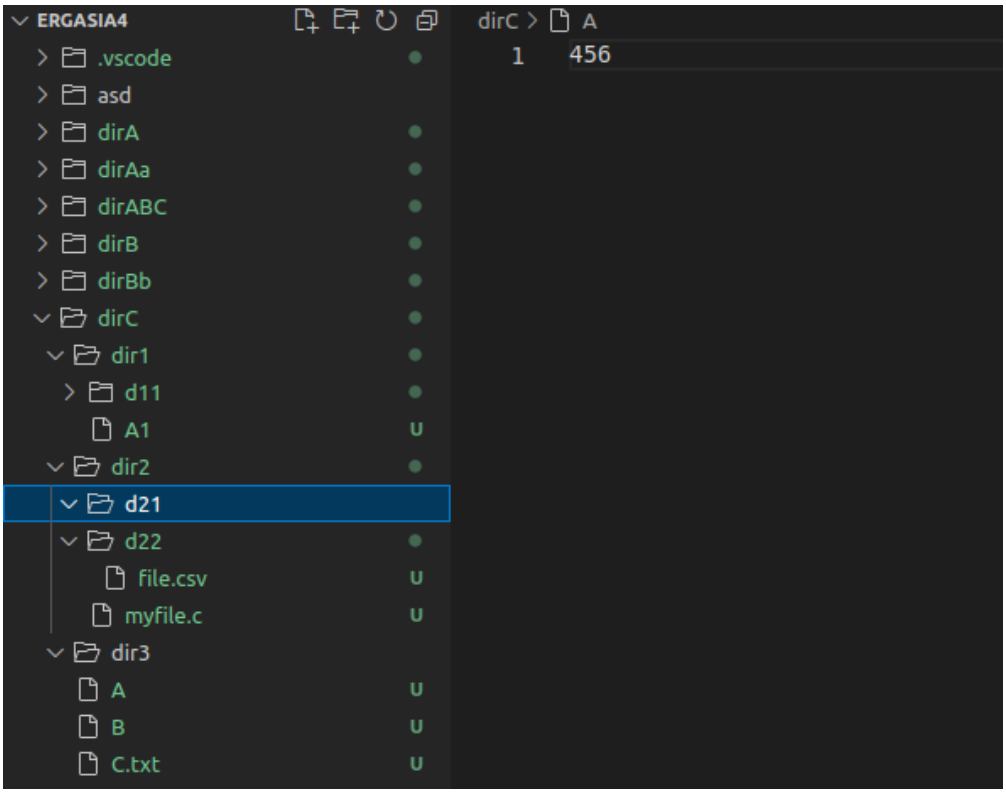
```
In dirA
dir3
A

In dirB
dir1/d11
dir1/d11/d111
dir1/d11/d111/d1111
dir1/d11/d111/d1111/test.txt
dir2/myfile.c
```

Ενώ για την συγχώνευση δημιουργείται ο φάκελος :

Δηλαδή έχουμε ίδια αποτελέσματα με αυτά της εκφώνησης. Για να δοκιμάσουμε ότι το πρόγραμμα μας λειτουργεί σωστά για αρχεία ίδιου τύπου αλλά με διαφορετικό περιεχόμενο, προσθέτουμε στον *dirB* (στο παραδοτέο μας αυτός είναι ο φάκελος *dirBtest*1) ένα αρχείο και το ονομάζουμε A και σαν περιεχόμενο του αρχείου A του *dirA* ορίζουμε την συμβολοσειρά 123 ενώ για τον φάκελο B την συμβολοσειρά 456, όπου η τροποποίηση στον φάκελο B είναι μεταγενέστερη. Αυτό που θέλουμε είναι για την σύγκριση τα δύο αρχεία να εμφανίζονται ως διαφορετικά, πράγματι :



Ενώ για την συγχώνευση θέλουμε το να κρατείται το περιεχόμενο του αρχείο στον φάκελο B αφού εκεί έγινε η πιο πρόσφατη τροποποίηση, αυτός είναι ο φάκελος και το περιεχόμενο του A :



Για να μην κουράζουμε, θα περιορίσουμε στο σημείο αυτό τα *screenshots*, συνεχίζουμε με απλό σχολιασμό των πειραμάτων μας. Δημιουργώντας έναν νέο φάκελο (*drBtest*2) όπου τώρα το αρχείο A είναι ένας φάκελος, πάλι θεωρούνται διαφορετικά τα αρχεία, και στην συγχώνευση κρατείται αυτό που τροποποιήθηκε τελευταία.
Στο επόμενο παράδειγμα μας, έχουμε δημιουργήσει δύο φακέλους *dirAa*, *dirBb* όπου υπάρχουν κάποια απλά αρχεία για επαλήθευση της ορθότητας του προγράμματος καθώς και κάποια *links*, μπορείτε να δοκιμάσετε να τα ελέγξετε, θεωρούμε τετριμμένα τα αποτελέσματα.
Συνεχίζουμε με πειράματα με *links*, έχοντας δημιουργήσει τους φακέλους *dirD* και *dirL*, όπου εντός τους, βρίσκονται κάποια *links* με ίδιο όνομα που δείχνουν σε αρχεία με διαφορετικό περιεχόμενο και όνομα, οπότε θεωρούνται διαφορετικά. Στην σύγκριση, σωστά εμφανίζονται ως διαφορετικά ενώ στην συγχώνευση, δημιουργούνται τα αρχεία των δύο φακέλων, και τα *links* δείχνουν στο αρχείο που έχει τροποποιηθεί μεταγενέστερα. Τέλος, δημιουργούμε δύο φακέλους *dirL* και *dirL*2 οι οποίοι είναι αντιγραφή του *dirE* με την μόνη διαφορά ότι στον *dirL* έχει ίδιο περιεχόμενο το αρχείο στο οποίο δείχνουν τα *links* (*twotext*) ενώ στον *dirL*2 το περιεχόμενο είναι διαφορετικό. Όπως αναμενόταν, στην πρώτη περίπτωση η φάκελοι είναι ίδιου, αφού η σύγκριση δεν εμφανίζει τίποτα, ενώ στην δεύτερη, το αρχείο και τα *links* θεωρούνται διαφορετικά, και ας έχουν κοινό όνομα. Τέλος, για να πειραματιστούμε και με κάποια *hardlinks* έχουμε δημιουργήσει δύο φακέλους *hardlinks* και *hardlinks*2. Στον φάκελο *hardlinks* τα αρχεία A και *hardlink* είναι απλά αρχεία, ενώ στον φάκελο *hardlinks*2 το αρχείο *hardlink* είναι *hardlink* στο A. Τα αποτελέσματα από τις συγκρίσεις και τις συγχωνεύσεις είναι τα αναμενόμενα για όλα τα πειράματα.

Μεταγλώττιση και Εκτέλεση Προγράμματος

Για την μεταγλώττιση και την παραγωγή εκτελέσιμου έχει δημιουργηθεί $Makefile$. Αρκεί η εντολή *make* στο *tty* για την παραγωγή του εκτελέσιμου *cmpcat* και τα ορίσματα εκτέλεσης είναι αυτά που περιγράφονται στην εκφώνηση. Για την διαγραφή των εκτελέσιμων και των $object\ files$ μπορεί να χρησιμοποιηθεί η εντολή *make clean* στο *tty*.
Εφόσον το αρχείο που υποβάλλεται είναι *tar.gz* για να το ανοίξετε θα χρειαστεί να εκτελέσετε σε $linux\ tty$ την εντολή $tar\ -xzvf\ ApostolosKoukouvinis + NataliaBekakou$ αυτή θα δημιουργήσει έναν φάκελο που ονομάζεται *ergasia*4, εντός του βρίσκονται όλα τα αρχεία που χρειάζεται η εργασία.

Ονοματεπώνυμα - ΑΜ

Απόστολος Κουκουβίνης - 1115202000098
Ναταλία Μπεκάκου - 1115201900124