OS Project 4

Βησσαρίων Μουτάφης sdi1800119@di.uoa.gr
1115201800119

Notes για τους διορθωτές:

- Γίνεται χρήση της *lstat* **μόνο**, για να μπορώ να εντοπίζω και τα symlinks.
- Μην διαγράψετε τα directories μέσα στο φάκελο της εργασίας, χρειάζονται για τα test-runs (δες στο επόμενο section).
- Αν έχουμε το path src/phony και dest/phony και το ένα ειναι dir/file και το δεύτερο file/dir, διαγράφουμε το τελευταίο και κάνουμε κανονικά copy το πρώτο.
- Τα sym links δεν λαμβάνονται υπ όψιν όταν δεν έχω το flag "-l".
- Αν το path src/hardlink ειναι ένα hardlink σε ένα path και τρέξουμε quic **χωρίς** το "-l", τότε το αντιμετωπίζουμε σαν απλό αρχείο και αγνοούμε το link behaviour του. Επίσης, εφόσον υπάρχει ήδη,την επόμενη φορά που θα τρέξουμε quic και **θα βάλουμε το "-l"**, **θα μετατραπεί σε hard link.**
- Αν το dest dir **υπάρχει ήδη**, τότε δεν το μετράμε στα copied items οπότε αν έχω λόγου χάρη 18 στοιχεία στο src hierarchy μαζί με το src dir, τότε τα statistics θα δείξουν 17/18 copied/detected items.
- Στα στατιστικά τα total items in hierarchy ειναι πάντα αυτά του src. Λαμβάνω υπ όψιν και τα directories.
- **Διαφορετικά στοιχεία κατά Link Extension**: Χρησιμοποιώ την ευρετική της εκφώνησης, προσθέτωντας και ως παράγοντα ομοιότητας τον αριθμό των hard links στο συγκεκριμένο i-node.
- Τα symlinks δεν τα λαμβάνω υπ όψιν αν δεν συμπεριλάβω το flag "-l" σε αντίθεση με τα hard links. Σε κάθε περίπτωση τα symlinks γίνονται copy βάση του path που δείχνουν. Αν είναι relative path και δείχνει στο src τότε το αντίστοιχο θα δείχνει μέσα στο dest directory. Αν είναι absolute, επειδή είναι αρκετά συγκεκριμένα ορισμένο από τον χρήστη, δεν θα πειράξουμε το path καθόλου, έχοντας μία πιθανότητα αν αλλάξουμε file system το link να ειναι dangling. Το πρόγραμμα δεν θα φταίει για αυτό καθώς αν ο χρήστης θέλει να έχει μία πλήρη εικόνα του src dir και τον ενδιαφέρουν τα links, πρέπει να προσέχει μόνος του το άν θα αντιγραφούν σωστά ή όχι.

Compilation and Run

Παρέχω ένα Makefile που φροντίζει για το separate compilation των modules που χρησιμοποιεί το πρόγραμμα. Για να τρέξετε το πρόγραμμα τρέξτε τις παρακάτω εντολές.

```
~$  make # compile the program and copy an instance of the executable at root directory.
~$ make run # compile the program and run a test input (do not delete any of the directories in the root dir)
~$ make clean # uninstall the program
~$ make clean-dest # clean the dest dir of the test run
```

Listing 1: Compilation and Run Commands

Για να τρέξετε το executable (./quic)

```
~$ ./quic "src dir path" "trg dir path" [-l] [-v] [-d]

#or

~$ [-l] [-v] [-d] ./quic "src dir path" "trg dir path"
```

Listing 2: Compilation and Run Commands

Τα flags "-l, -v, -d" μπορούν να τοποθετηθούν είτε στην αρχή ή στο τέλος όλα μαζί και ποτέ ανάμεσα στα src dir path και trg dir path. Παρ' όλα αυτά μπορούμε να έχουμε ένα run του τύπου *$ ./quic -l src-path dest-path -v -d*.

Abstract

Η εφρμογή quic, έχει υλοποιηθεί έτσι ώστε να αντιγράφει αναδρομικά ένα directory από το δοσμένο path. Διαθέτει 3 flags τα οποία ελέγχουν το verbosity, την διαγραφή των φακέλων που δεν υπάρχουν στο src dir και την διαχείριση των links (soft & hard).

Συγκεκριμένα αν το dest-dir είναι άδειο και το "-l" **δεν έχει τεθεί** τότε πολύ απλά ακολουθούμε όλα τα hard links και φτιάχνουμε τους φακέλους. Αν όμως έχουμε όλα τα links έτοιμα (από προηγούμενο run) τότε δεν πειράζουμε τιποτα καθώς το τελικό αποτέλεσμα θα είναι ίδιο στον end user. Επίσης, σε κάθε περίπτωση, **αν θέσουμε** το "-l", τότε, *αν το αντίστοιχο link δεν υπάρχει*, το δημιουργούμε, είτε υπάρχει ο αντίστοιχος φάκελος έιτε όχι, οπότε στα hard links θα έχω διαγραφή του ξεχωριστού i-node και point στο proper inode. Τέλος, ειναι υπεύθυνος ο χρήστης αν θα εκτελέσει σωστά το πρόγραμμα και δεν θα καταλήξει με dangling symlinks που να δείχνουν σε φακέλους εκτός του src dir hierarchy.

Utilities

Τα *utilities* αποτελούνται από μικρές συναρτήσεις, οι οποίες βοηθούν στον σωστό έλεγχο και το *manipulation φακέλων* (copy, creation, delete, etc) καθώς και στην ανίχνευση διάφορων edge cases (πχ detect cycle in the hierarchy, check for deleted files/directories in the src directory, if needed, etc). Επίσης μέσα σε αυτή έχει τοποθετηθεί και μία συνάρτηση για τον έλεγχο της ομοιότητας 2 αρχείων, καθώς και μικρές συναρτήσεις που δημιουργούν τα links.

Συγκεκριμένα, υπάρχει μια απλή συνάρτηση που δημιουργεί symlinks και μία η οποία δημιουργεί hard links.

Για τα **hard links** διαθέτουμε ένα map με entries $<$ *src st_ino*, path to the repsective dest inode $>$, οπότε την πρώτη φορά που συναντούμε ένα src entry με inode number = n, τότε θα το **εισάγουμε στο map**. Την επόμενη φορά απλά θα μπορούμε να τσεκάρουμε αν το current inode number ειναι στο map και αφού το βρούμε, θα μπορούμε να το ενώσουμε με ένα σχετικό path στο dest dir.

Παράλληλα έχουμε υλοποιήσει και την δικιά μας έκδοση της εντολής rm -rf του unix καθώς **διαγράφουμε αναδρομικά τα directories** αναγνωρίζοντας αν το path που δίνεται είναι directory/file και δρώντας ανάλογα σε κάθε βήμα. Για τον έλεγχο των διαγεγραμμένων από το src dir, αρχείων ή και directories κάνουμε χρήση της *check_deleted*, την οποία τρέχουμε σε κάθε level στο dest hierarchy όπως κάνουμε το copy για να είναι αποδοτικός ο έλεγχος και να μην κάνουμε περιττά system calls σε όλο το dest-hierarchy.

File Manipulation Routines

Για το file manipulation εχουμε μια γενική συναρτηση που δίνεις ένα in και ένα out path και κάνει το copy. Η ρουτίνα αυτή ελέγχει αν τα 2 elements διαφέρουν ως προς τους τύπους (δες και στα notes 3ο bullet) και αναγνωρίζει αν το src path είναι **file,** οπότε καλεί την κατάλληλη ρουτίνα που το **αντιγράφει κανονικά** (διαφορετική συμπεριφορά μέσα στην copy file αν εκτελώ με το link extension), ή αν είναι **directory**, οπου το **αντιγράφει αναδρομικά**, με την κατάλληλη συνάρτηση.

Η συνάρτηση για αντιγραφή φακέλων υλοποιεί και το link extension, όταν το flag *manage_links* είναι true, οπότε θα εντοπίσει και αντιγράψει τα hard/sym links. Για τα hard links, συγκεκριμένα παρέχω και ένα map ώστε να μπορώ να έχω επίγνωση των φακέλων που αντιγράφτηκαν κάθε στιγμή εκτέλεσης, ώστε να τα ενώνω σωστά και αν ψάχνω αποδοτικά για την πληροφορία που χρειάζομαι (υπάρχει το inode ωστε να το ενώσω ή πρέπει να το φτιάξω;). Επίσης, για να είμαστε σίγουροι πως ο χρήστης θα έχει πλήρης εικόνα του src directory πρέπει να αντιγράψουμε τουλάχιστον ένα instance του src inode, οπότε αν ο κατάλογος έχει μέσα ένα link σε ένα αρχείο εκτός καταλόγου αυτό, υπό κανονικές συνθήκες θα πρέπει να περαστεί στο dest directory. Παρ' αυτά η quic θα προσπαθεί κάθε φορά να δημιουργήσει ένα hard link στο αντίστοιχο αρχείο καθώς θα 2 σχετικά inodes ποτέ δεν θα είναι ίδια, εφόσσον θα διαφέρουν στο πόσα links έχουν (δες στα Notes για τα **διαφορετικα αρχεία** κατα το Link Extension).

Για directory copy παρέχουμε μια αναδρομική συνάρτηση όπου ουσιαστικά κάνει χρήση της γενικής copy routine (1η παράγραφος), για να παρέχει generic copy behaviour. Επίσης μέσα σε αυτή γίνεται και ο έλεγχος για διαγεγραμένα από το src, στοιχεία τα οποία φροντίζει αν διαγράψει με τον τρόπο που πρέπει (αντίστοιχα για directories και για files/links).

Hash Table with Double Probing

Το hash table που παρέχεται για την διαχείριση των hard links, είναι υλοποιημένο με ώς ένα array από void pointers. Ο χρήστης παρέχει key-item creation, deletion, comparison και hashing functions έτσι ώστε να μπορεί να λειτουργήσει και να απελευθερώσει την μνήμη σω-

στά. Το double probing γίνεται με την χρήση της συνάρτησης $probing\_hash(key, ht\_size)$, η οποία κατανέμει ομοιόμορφα τα probes ωστε να πετύχουμε καλύτερη κατανομή στο hash table. Παρέχονται όλες οι βασικές λειτουργίες $create, insert, delete, contains, destroy$. Επίσης παρέχεται και μια συνάρτηση για την εκτύπωση των κλειδιών (debbuging purposes).

References

- Διαφάνειες του μαθήματος.
- Διάφορα posts του stack overflow που εξηγούσαν γιατί πρέπει να χρησιμοποιήσω lstat για να παίζω με links.
- Manpages για τις διάφορες system calls για την κατανόηση των error codes που επιστρέφουν και για να καταλάβω πως λειτουργούν εξ ολοκλήρου.