ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
Τμήμα Πληροφορικής και Τηλεπικοινωνιών
4η Εργασία - Τμήμα: Περιττών Αριθμών Μητρώου
K22: Λειτουργικά Συστήματα – Χειμερινό Εξάμηνο '18
Ημερομηνία Ανακοίνωσης: Τετάρτη, 19/12
Ημερομηνία Υποβολής: Πέμπτη, 17/01 και Ώρα 23:59

Εισαγωγή στην Εργασία:

Στο πλαίσιο αυτής της εργασίας θα πρέπει να αναπτύξετε την εφαρμογή `mirr` η οποία θα παρακολουθεί *δυναμικά* μια ιεραρχία αρχείων και καταλόγων και όταν υπάρξουν αλλαγές το πρόγραμμά σας ενημερώνει ένα πλήρες αντίγραφό της ιεραρχίας που βρίσκεται σε έναν άλλο κατάλογο. Για αυτό το σκοπό, θα χρησιμοποιήσετε το *inotify* system call interface του Linux [1, 2].

Η κλήση του `mirr` μπορεί να γίνει ως εξής:
`prompt >> ./mirr source backup`

Ένα τέτοιο πρόγραμμα, βοηθά όταν εργαζόμαστε με μια σειρά αρχείων και επιθυμούμε να κρατάμε αντίγραφα τους σε κάποιο άλλο φάκελο για λόγους ασφάλειας ή και απλά επειδή βολεύει (όπως π. χ. κάνει το `Dropbox`). Αν δεν υπάρχει αυτοματισμός, αλλαγές που έχουμε κάνει στο κατάλογο `source` μπορούν να ξεχαστούν με τις προφανείς συνέπειες για την ορθότητα του περιεχομένου του αντιγράφου στο `backup`. Αν και μπορεί να δημιουργήσει λίγο παραπάνω φόρτο (στο σύστημα), η πλέον ενδεδειγμένη στιγμή για να βαστήξουμε ένα αντίγραφο συνεπές είναι να υπάρξει άμεση αντίδραση όταν μια τροποποίηση/αλλαγή διαπιστωθεί.

Γενικά, το πρόγραμμα σας θα πρέπει να κάνει τα παρακάτω δύο πράγματα:

1. Αρχικά, όταν το `backup` δεν υφίσταται, θα πρέπει να αντιγράφει τα περιεχόμενα του `source` ώστε και τα δύο να έχουν την ίδια πληροφορία (δομή και δεδομένα).
2. Εν συνεχεία, το πρόγραμμα σας θα πρέπει να παρακολουθεί το `source` και όλους τους υποκείμενους καταλόγους και όταν μια αλλαγή συμβεί, το `mirr` θα πρέπει να την 'αντιληφθεί' άμεσα και ενεργήσει ώστε και η δομή αλλά και το περιεχόμενο του στόχου να ενημερωθούν ανάλογα.

Διαδικαστικά:

Το πρόγραμμά σας θα πρέπει να γραφτεί σε `C` (ή `C++` αν θέλετε αλλά χωρίς την χρήση STL/Templates) και να τρέχει στις μηχανές Linux workstations του τμήματος.

- Υπεύθυνοι για την άσκηση αυτή (ερωτήσεις, αξιολόγηση, βαθμολόγηση κλπ.) είναι ο κ. Γιώργος Αποστολοπουλος `cs2180003+AT-di`, και ο κ. Δημήτρης Σπυρόπουλος `jimsp+AT-di`.
- Παρακολουθείτε την ιστοσελίδα του μαθήματος `http://www.di.uoa.gr/~ad/k22/` για επιπρόσθετες ανακοινώσεις αλλά και την ηλεκτρονική-λίστα (η-λίστα) του μαθήματος στο URL `https://piazza.com/uoa.gr/fall2018/k22/home`
- Το πρόγραμμα σας (source code) πρέπει να αποτελείται από ***τουλάχιστον τρία*** (και κατά προτίμηση πιο πολλά) διαφορετικά αρχεία. Το πρόγραμμά σας θα πρέπει *απαραιτήτως να κάνει χρήση* separate compilation .

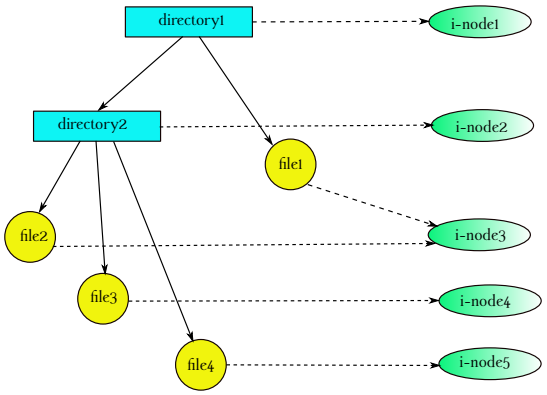
Διατύπωση του Προβλήματος:

Τα αρχεία στο λειτουργικό σύστημα Linux καταγράφονται σε δύο διαφορετικά επίπεδα, των ονομάτων και των κόμβων-δεικτών (i-nodes). Κάθε `i-node` συνδέεται με τα δεδομένα ενός και μόνο αρχείου και τηρεί όλες τις πληροφορίες εκτός από το όνομα του αρχείου (π.χ. αριθμός `i-node`, ημερομηνία τελευταίας αλλαγής, μέγεθος). Η σχέση είναι αμφιμονοσήμαντη, καθώς κάθε αρχείο δεικτοδοτείται από ένα και μόνο `i-node`. Όμως ένα i-node μπορεί να είναι συνδεδεμένο με πολλαπλά ονόματα αρχείων. Αυτό επιτυγχάνουν οι σύνδεσμοι, τα λεγόμενα

hard links. Γενικά, λογική δομή του συστήματος αρχείου μπορεί να σκιαγραφηθεί με ένα κατευθυνόμενο χωρίς κύκλους γράφο (ή και δένδρο) καθώς:

- Κατάλογοι (που έχουν λογικά ονόματα) μπορούν να εμπεριέχουν αρχεία καθώς επίσης και άλλους καταλόγους με αναδρομικό τρόπο.
- Δεν υπάρχουν hard-links όσον αφορά στους καταλόγους. Αυτό βοηθά να αποφεύγουμε κύκλους στο σύστημα αρχείων.

Το Σχήμα 1 παρουσιάζει την λογική δομή ενός καταλόγου (*directory1*) ο οποίος διαθέτει ένα εμφωλιασμένο κατάλογο (*directory2*) και 4 αρχεία. Κάθε μια από αυτές τις οντότητες αναπαριστάται από ένα `i-node` αλλά τα αρχεία *file1* και *file2* αποτελούν ουσιαστικά το ίδιο αρχείο καθώς δείχνουν στο ίδιο `i-node`.



Σχήμα 1: Λογική οργάνωση και η `i-node` λίστα των στοιχείων στο *directory1*

Ένας τρόπος να δούμε το `i-node` ενός αρχείου είναι η εντολή `ls -i` (σε επίπεδο προγράμματος συστήματος) όπως και η `stat()` (σε επίπεδο κλήσης συστήματος). Ειδικά στο Linux υπάρχει και το πρόγραμμα συστήματος με το ίδιο όνομα stat που κάνει την ίδια δουλειά σε επίπεδο κελύφους. Η δημιουργία συνδέσμου σε υπάρχον αρχείο γίνεται με την εντολή: `ln <existing name> <new name>`

Για την κλήση συστήματος `inotify`, δείτε `man inotify`. Εκεί υπάρχει μια αρκετά εκτενής περιγραφή του υποσυστήματος αυτού. Επίσης δείτε το πρόγραμμα υπόδειγμα `inotify.c` για τον τρόπο χρήσης του. Παρατηρήστε ότι η δομή που περιγράφει ένα γεγονός (`struct inotify_event`) είναι μεταβλητού μεγέθους (το πεδίο `len` δηλώνει πόσοι χαρακτήρες ακολουθούν για να ολοκληρωθεί το όνομα), άρα χρειάζεται κάποια προσοχή στην ανάγνωσή της (το υπόδειγμα το διαχειρίζεται ήδη).

Κατευθύνσεις υλοποίησης:

Κατά την διάρκεια του αρχικού συγχρονισμού (Βήμα 1), θα πρέπει να δημιουργήσετε δομές οι οποίες να αντικατοπτρίζουν την ιεραρχία ονομάτων και `i-nodes`, όπως την συντηρεί και το υποσύστημα διαχείρισης αρχείων του λειτουργικού συστήματος. Πιο συγκεκριμένα, θα πρέπει να καταγράψετε τα `i-nodes` που περιλαμβάνονται στην δομή σας και τα ονόματα που δείχνουν σε αυτά. Στο τέλος αυτού του βήματος θα πρέπει να έχετε ενημερωμένες και συγχρονισμένες δομές τόσο για τον κατάλογο-πηγή, όσο και για τον κατάλογο-προορισμό.

Έστω λοιπόν ένας πίνακας με i-nodes, με περιεχόμενα:

- Ημερομηνία τελευταίας αλλαγής

- Μέγεθος αρχείου
- Λίστα με ονόματα που δείχνουν εδώ
- Αριθμός ονομάτων που δείχνουν εδώ (μπορεί να υπολογίζεται δυναμικά)
- Ειδικά για την πηγή, ένας δείκτης στην αντίστοιχη δομή του προορισμού (στο αντίγραφό του δηλαδή)

Και ένα δέντρο με ονόματα, με περιεχόμενα (σε κάθε κόμβο):

- Όνομα
- Δείκτης σε i-node

Ένας αλγόριθμος που συγχρονίζει την ιεραρχία-πηγή με την ιεραρχία-προορισμό είναι ο εξής:

1. Δημιούργησε τις παραπάνω δομές και για τις δύο ιεραρχίες, αντλώντας πληροφορίες για κάθε όνομα. Ταξινόμησε αλφαβητικά τους κόμβους-αδελφούς στο δέντρο των ονομάτων.
2. Διάτρεξε παράλληλα τα δύο δέντρα κατά βάθος, και:
   (α′) Εάν ένα όνομα-κατάλογος υπάρχει στην πηγή αλλά όχι στον προορισμό, δημιούργησέ το και στον κατάλογο-προορισμό (πιθανόν αποσυνδέοντας πρώτα κάποιο όνομα-αρχείο, ίδιο όνομα δηλαδή που υπάρχει στον προορισμό αλλά δείχνει σε κόμβο-αρχείο και όχι σε κόμβο-κατάλογο)
   (β′) Εάν ένα όνομα-κατάλογος δεν υπάρχει στην πηγή αλλά υπάρχει στον προορισμό, διέγραψέ τον από τον προορισμό
   (γ′) Εάν ένα όνομα-αρχείο υπάρχει στην πηγή αλλά όχι στον προορισμό, πήγαινε στο i-node του και δες αν έχει ήδη αντίγραφο στον προορισμό.
      - Αν ναι, συνέδεσε το όνομα στο σωστό i-node του προορισμού.
      - Αν όχι, δημιούργησε ένα νέο αρχείο στον προορισμό και σύνδεσε το i-node της πηγής με αυτό του προορισμού.
   (δ′) Εάν ένα όνομα-αρχείο δεν υπάρχει στην πηγή αλλά υπάρχει στον προορισμό, αποσύνδεσε το
   (ε′) Εάν ένα όνομα-αρχείο υπάρχει και στους δύο, σύγκρινε τα στοιχεία του i-node. Εάν είναι ίδια (ημερομηνία τελευταίας αλλαγής και μέγεθος), άφησε το όπως είναι και ενημέρωσε τις δομές. Αν όχι, αποσύνδεσε το από τον προορισμό και κάνε ότι στο βήμα 2γ′.

Στα παραπάνω, ισχύει η εξής αντιστοιχία όρων και κλήσεων συστήματος:

Αντλώ πληροφορίες αρχείου = `stat` system call
Δημιουργώ αρχείο = `creat` or `open` system call
Αποσυνδέω αρχείο = `unlink` system call
Δημιουργώ κατάλογο = `mkdir` system call
Διαγράφω κατάλογο = `rmdir` system call (αλλά πρέπει να σβήσω όλα τα περιεχόμενά του πρώτα)
Συνδέω όνομα σε υπάρχων i-node = `link` system call

Σημειώστε ότι το πεδίο `st_nlink` της δομής `stat` (δείτε `man 2 stat`) δεν επιτρέπει την εξαγωγή συμπερασμάτων για την συγκεκριμένη εφαρμογή, καθώς μπορεί να περιλαμβάνει ονόματα εκτός της ιεραρχίας-πηγής. Για αυτό το λόγο, θα πρέπει να συντηρηθεί αυτόνομα ο 'Αριθμός ονομάτων που δείχνουν εδώ' από το πρόγραμμά σας.

Στο δεύτερο βήμα, θα πρέπει να αρχίσει η 'παρακολούθηση' των καταλόγων που περιλαμβάνονται στον πηγαίο κατάλογο. Η διαδικασία, συνοπτικά έχει ως εξής:

1. Δημιουργία ουράς (queue) γεγονότων με `inotify_init() ->fd`
2. Προσθήκη των αντικειμένων προς παρακολούθηση με `inotify_add_watch() ->wd (watch descriptor)`. Στην συγκεκριμένη περίπτωση πρόκειται για όλους τους καταλόγους στην ιεραρχία κάτω από τον πηγαίο κατάλογο. Συντήρηση ενός πίνακα συσχέτισης του `wd` που επιστρέφεται με το όνομα που αντιστοιχεί.

3. Επανέλαβε:
   (α′) Ανάγνωση του επόμενου γεγονότος από την ουρά
   (β′) Ανάλογα με τον τύπο του γεγονότος, ενημέρωσε με τις αλλαγές τον κατάλογο προορισμό

Τα γεγονότα που ενδιαφέρουν και ο συνιστώμενος τρόπος διαχείρισής τους είναι τα εξής:

Table for Events and Actions	
EVENT	Action
`IN_CREATE`	if it is a file find the `i-node` of the source If there exists already a copy link it otherwise, create a new one otherwise, if it is a catalog create a catalog at the destination add the new catalog in the objects under monitoring
`IN_ATTRIB`	If it is a file and the last date of modification has changed update the replica at destination
`IN_MODIFY`	(this event is received with *every* modification) If it is a file, mark it as modified
`IN_CLOSE_WRITE`	If it has been marked as modified, copy it
`IN_DELETE`	If it is a file, unlink it from the target
`IN_DELETE_SELF`	(This concerns only a catalog as we monitor only directories) Delete it from the target Take it off the list of monitored objects using `inotify_rm_watch()`
`IN_MOVED_FROM`	(A name has been moved outside of the monitored catalog) Make a note of the field `cookie` of the event. No action is taken until the next event. If the next event is `IN_MOVED_TO` follow up what the next down description says Otherwise, unlink the name (as it has moved outside the source hierarchy)
`IN_MOVED_TO`	(A name has been moved/introduced within the monitored hierarchy) If the field `cookie` is the same with that recorded just above (the movement is within the monitored hierarchy of catalogs) move the name respectively to the target catalog Otherwise, act as in `IN_CREATE` moreover, copy the data.

Άλλες Υποθέσεις Εργασίας:

◇ Ο κατάλογος-προορισμός **δεν** περιλαμβάνεται άμεσα ή έμμεσα στον κατάλογο πηγή
◇ Η ιεραρχία-προορισμός δεν επιτρέπεται να τροποποιηθεί από άλλες διεργασίες
◇ Η ιεραρχία-πηγή δεν τροποποείται κατά τη διάρκεια του αρχικού συγχρονισμού (Βήμα 1)
◇ Οι συμβολικοί σύνδεσμοι (symbolic links) δεν χρειάζεται να υποστηρίζονται από την εφαρμογή σας

Τι πρέπει να Παραδοθεί:

1. Μια σύντομη και περιεκτική εξήγηση για τις επιλογές που έχετε κάνει στο σχεδιασμό του προγράμματος σας (2-3 σελίδες σε ASCII κειμένου είναι αρκετές).

2. Οπωσδήποτε ένα `Makefile` (που να μπορεί να χρησιμοποιηθεί για να γίνει αυτόματα το compile του προγράμματος σας). Πιο πολλές λεπτομέρειες για το (`Makefile`) και πως αυτό δημιουργείται δίνονται στην ιστοσελίδα του μαθήματος.
3. Ένα `tar-file` με όλη σας την δουλειά σε έναν κατάλογο που πιθανώς να φέρει το όνομα (ή τα ονοματά) σας και θα περιέχει όλη σας την δουλειά δηλ. `source files, header files, output files` (αν υπάρχουν) και οτιδήποτε άλλο χρειάζεται.

Άλλες Σημαντικές Παρατηρήσεις:

1. Οι εργασίες είναι είτε **ατομικές** ή μπορείτε να δουλέψετε με μια/ενα συνεργάτη (σε γρουπ των δύο). Στην τελευταία περίπτωση θα πρέπει να εξηγήσετε τους διακριτους ρόλους που ανέλαβε και εφερε σε περας ο καθε ένας στην ομάδα (στην γραπτή αναφορά που θα υποβάλετε).
2. Το πρόγραμμα σας θα πρέπει να τρέχει στα Linux συστήματα του τμήματος αλλιώς **δεν μπορεί να βαθμολογηθεί**.
3. Αν και αναμένεται να συζητήσετε με φίλους και συνεργάτες το πως θα επιχειρήσετε να δώσετε λύση στο πρόβλημα, αντιγραφή κώδικα (οποιαδήποτε μορφής) είναι κάτι που **δεν επιτρέπεται** και δεν πρέπει να γίνει. Οποιοσδήποτε βρεθεί αναμεμειγμένος σε αντιγραφή κώδικά **απλά παίρνει μηδέν** στο μάθημα. Αυτό ισχύει για **όσους εμπλέκονται ανεξάρτητα** από το ποιος έδωσε/πήρε κλπ.
4. Το παραπάνω ισχύει αν διαπιστωθεί *έστω και μερική άγνοια* του κώδικα που έχετε υποβάλει ή *άπλα υπάρχει υποψία* ότι ο κώδικας είναι προϊόν συναλλαγής με τρίτο/-α άτομο/α.
5. Προγράμματα που **δεν χρησιμοποιούν** separate compilation χάνουν αυτόματα 5% του βαθμού.
6. Σε καμιά περίπτωση τα Windows *δεν είναι επιλέξιμη* πλατφόρμα για την παρουσίαση αυτής της άσκησης.

Αναφορές

[1] Michael Kerrisk, *The Linux Programming Interface*, No Starch Paper, San Fracisco, CA June 2010.

[2] Ian Shields, *Monitor Linux file system events with inotify*, `https://www.ibm.com/developerworks/library/l-inotify/` April 2006.