ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
Τμήμα Πληροφορικής και Τηλεπικοινωνιών
4η Εργασία - Τμήμα: Αρτίων Αριθμών Μητρώου
K22: Λειτουργικά Συστήματα – Χειμερινό Εξάμηνο '19
Ημερομηνία Ανακοίνωσης: Τρίτη 24/12
Ημερομηνία Υποβολής: Πέμπτη, 23/01 και Ώρα 23:59

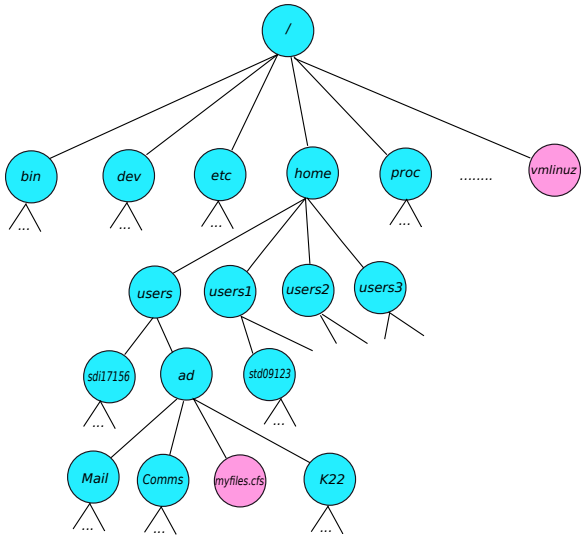
Εισαγωγή στην Εργασία:

Ο στόχος αυτής της εργασίας είναι να δημιουργήσετε ένα περιορισμένης εμβέλειας αλλά πλήρως λειτουργικό σύστημα αρχείων στο χώρο του χρήστη. Το σύστημα που ονομάζεται `cfs (container file-system)` παρέχει όλες του τις οντότητες (αρχεία, καταλόγους και συνδέσμους) καθώς και όλες του τις λειτουργίες μέσα σε *ένα κανονικό αρχείο* του Linux. Αρχεία που φιλοξενούν `cfs`-περιεχόμενο, έχουν το επίθεμα '`.cfs`' για να διαχωρίζονται από τα κανονικά αρχεία Linux.

Το `cfs` παρέχει ουσιαστικά εν γένει την δενδρική λογική δομή του Linux FS πάνω στα βασικά του στοιχεία που είναι αρχεία, κατάλογοι και σύνδεσμοι. Επίσης, ένας αριθμός από λειτουργίες θα πρέπει να υλοποιηθούν στο `cfs` και αυτές περιλαμβάνουν δημιουργία, διαγραφή, εμφάνιση και τροποποίηση αρχείων, καταλόγων και δεσμών. Η πλήρης περιγραφή των εντολών δίνεται παρακάτω. Οι εντολές αυτές θα πρέπει να είναι εκτελέσιμες από το κέλυφος (δηλ. συνηθισμένη γραμμή εντολής).

Σε φυσικό επίπεδο, όλα τα αρχεία ενός `cfs` αποθηκεύονται σε ένα μοναδικό κανονικό Linux αρχείο το οποίο θα τοποθετείται σε μια καθορισμένη θέση στο δίσκο, π.χ. `/home/users/ad/myfile.cfs`. Πέρα από τις βασικές πράξεις, θα πρέπει να υποστηρίζετε λειτουργικότητα εισαγωγής και εξαγωγής αρχείων και καταλόγων από το `cfs` στο σύστημα αρχείων Linux και αντίστροφα.

Το Σχήμα 1 ένα διάγραμμα οργάνωσης αρχείων/καταλόγων για τον χρήστη `ad`. Στο αρχείο `myfiles.cfs` εμπεριέχεται μια λογική ιεραρχία αρχείων/δεσμών και καταλόγων που 'ζουν' μέσα σε ένα αρχείο Linux στο home directory του χρήστη `ad`.



Σχήμα 1: Διάγραμμα τοποθεσίας `cfs` ιεραρχίας στο αρχείο `myfiles.cfs` του χρήστη `ad`.

Θα πρέπει να *επιδείξετε την ορθότητά* της λειτουργίας του `cfs`.

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 Παναγιωτόπουλος `cs2190012+AT-di`, ο κ. Μάριος Παπαμιχαλόπουλος `cs3190006+AT-di`, και ο κ. Γιάννης Καλύβας `cs2190016+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/fall2019/k22/home`
- Προφανώς η εργασία σας θα αποτελείται από πολλαπλά προγράμματα τα οποία θα πρέπει να συμβολομεταφραστούν με χρήση separate compilation.

Η Διεπαφή του `cfs` Μέσω των Βασικών Λειτουργιών του:

Το `cfs` χειρίζεται αρχεία, καταλόγους και συνδέσμους, μέσω ενός μοναδικού αρχείου το οποίο βρίσκεται στο σύστημα αρχείων του Linux. Σε λογικό επίπεδο ο χρήστης έχει την ψευδαίσθηση ότι εργάζεται κανονικά σε ιεραρχικό σύστημα αρχείων. Οι βασικές πράξεις/λειτουργίες του `cfs` που θα πρέπει να υποστηρίζονται και να εκτελούνται από το κέλυφος είναι οι ακόλουθες:

- Δυνατότητα ορισμού αρχείου `.cfs` τύπου πάνω στο οποίο οι εντολές που ακολουθούν εκτελούνται.
- Δημιουργία αρχείων.
- Παράθεση (listing) αρχείων.
- Διαγραφή αρχείων.
- Αντιγραφή αρχείων.
- Συγχώνευση αρχείων.
- Εισαγωγή/Εξαγωγή αρχείων από/προς το `cfs`.

Σημειώνουμε ότι τα αρχεία περιλαμβάνουν κανονικά αρχεία, καταλόγους και συνδέσμους (links). Οι αντίστοιχες εντολές είναι οι εξής:

1. `cfs_workwith <FILE>`
   οι εντολές που θα ακολουθήσουν, θα εκτελεστούν στο αρχείο τύπου `cfs` που δίνεται στη παράμετρο <FILE>.

2. `cfs_mkdir <DIRECTORIES>`
   Δημιουργία νέου καταλόγου με το (τα) όνομα (ονόματα) `<DIRECTORIES>`.

3. `cfs_touch <OPTIONS> <FILES>`
   δημιουργία αρχείου/-ων ή αν τα παρατιθέμενα αρχεία υπάρχουν τροποποίηση των χρονοσήμων των αρχείων `<FILES>`. Στην περίπτωση τροποποίησης των χρονοσήμων τα `<OPTIONS>` είναι:
   `-a`: Ανανέωση μόνο του χρονόσημου πρόσβασης.
   `-m`: Ανανέωση μόνο του χρονόσημου τροποποίησης.

4. `cfs_pwd`
   Εμφάνιση τρέχοντος καταλόγου ξεκινώντας απο την 'κορυφή' του `cfs`.

5. `cfs_cd <PATH>`
   Αλλαγή τρέχοντος καταλόγου στο `<PATH>`. Η επιλογή του `<PATH>` περιλαμβάνει και τον τρέχοντα κατάλογο '.' ή και τον γονικό κατάλογο '..'

6. `cfs_ls <OPTIONS> <FILES>`
   Εκτύπωση περιεχομένων αρχείων όπου <OPTIONS> είναι:
   `-a:` Προβολή όλων των αρχείων συμπεριλαμβανομένων και των κρυφών (αρχεία που το όνομά τους ξεκινά από το χαρακτήρα *'.'*)
   `-r:` Αναδρομική εκτύπωση αρχείων με βάση τον τρέχοντα κατάλογο
   `-l:` Προβολή όλων των χαρακτηριστικών των αρχείων. Τα χαρακτηριστικά που πρέπει οπωσδήποτε να περιλαμβάνονται είναι χρόνος δημιουργίας, χρόνος τελευταίας πρόσβασης, χρόνος τελευταίας τροποποίησης και μέγεθος.
   `-u:` Προβολή των αρχείων χωρίς ταξινόμηση, αλλά με τη σειρά που αποθηκεύονται στον κατάλογο.
   `-d:` Προβολή μόνο των καταλόγων.
   `-h:` Προβολή μόνο των συνδέσμων.
   και <FILES> είναι:
   μία η πιο πολλές οντότητες που βρίσκονται στο συγκεκριμένο σημείο της ιεραχίας του `cfs`.

7. `cfs_cp <OPTIONS> <SOURCE> <DESTINATION> | <OPTIONS> <SOURCES> ... <DIRECTORY>`
   Αντιγραφή αρχείων και καταλόγων από <SOURCE> στο <DESTINATION> ή αντιγραφή των (πιθανώς πολλών) <SOURCES> στο <DIRECTORY>. Οι επιλογές σημαιών <OPTIONS> είναι:
   `-R`: Αντιγραφή του περιεχομένου καταλόγου <SOURCE> σε βάθος-1 στο <DESTINATION>.
   `-i`: Αντιγραφή κατόπιν αποδοχής του χρήστη, ύστερα από ερώτηση του συστήματος.
   `-r`: Αναδρομική αντιγραφή καταλόγων.

8. `cfs_cat <SOURCE_FILES> -o <OUTPUT_FILE>`
   Συγχώνευση των <SOURCE_FILES> στα <OUTPUT_FILE>. Η λίστα <INPUT_FILES> εννοείται σαν λίστα κανονικών αρχείων. Το <OUTPUT_FILE> είναι ένα κανονικό αρχείο που (επανα-)δημιουργείται με την εντολή.

9. `cfs_ln <SOURCE_FILE> <OUTPUT_FILE>`
   Δημιουργείστε ένα νέο αρχείο <OUTPUT_FILE> που είναι σκληρός σύνδεσμος στο αρχείο <INPUT_FILE>. Σύνδεσμοι αυτής της μορφής σε καταλόγους δεν επιτρέπονται.

10. `cfs_mv <OPTIONS> <SOURCE> <DESTINATION> | <OPTIONS> <SOURCES> ... <DIRECTORY>`
    Μετονομασία του <SOURCE> στο <DESTINATION> ή μετακίνηση των (πιθανώς πολλών) <SOURCES> στο < DIRECTORY>. Οι επιλογές σημαιών <OPTIONS> είναι
    `-i`: Μετακίνηση κατόπιν αποδοχής του χρήστη, ύστερα από ερώτηση του συστήματος.

11. `cfs_rm <OPTIONS> <DESTINATIONS>`
    Διαγραφή των αρχείων που βρίσκονται στο κατάτολογο(-ους) <DESTINATIONS> σε-βάθος-1 χωρίς την διαγραφή εμπεριεχομένων καταλόγων που πιθανόν υπάρχουν στο/α <DESTINATIONS>. Οι επιλογές σημαιών <OPTIONS> είναι
    `-i`: Διαγραφή στοιχείων συστήματος αρχείου ύστερα από ερώτηση του συστήματος και κατόπιν αποδοχής του χρήστη.
    `-r`: Αναδρομική διαγραφή όλων καταλόγων που εμπεριέχονται και συμπεριλαμβάνουν τους καταλόγους που περιγράφονται στο <DESTINATIONS>.

12. `cfs_import <SOURCES> ... <DIRECTORY>`
    Εισαγωγή των αρχείων/καταλόγων <SOURCES> του συστήματος αρχείων Linux στο <DIRECTORY> του `cfs`.

13. `cfs_export <SOURCES> ... <DIRECTORY>`
    Εξαγωγή των αρχείων/καταλόγων <SOURCES> του `cfs` στο <DIRECTORY> του συστήματος αρχείων Linux.

14. `cfs_create <OPTIONS> <FILE>`. Δημιουργία ενός `cfs` στο αρχείο <FILE>.
    Οι πιθανές επιλογές σημαίας <OPTIONS> είναι:
    `-bs <BLOCK_SIZE>`: Καθορισμός μεγέθους μπλοκ δεδομένων σε Bytes.
    `-fns <FILENAME_SIZE>`: Καθορισμός μεγέθους ονόματος αρχείων σε Bytes.
    `-cfs <MAX_FILE_SIZE>`: Καθορισμός μέγιστου μεγέθους αρχείων σε Bytes.
    `-mdfn <MAX_DIRECTORY_FILE_NUMBER>`: Καθορισμός μέγιστου αριθμού αρχείων ανά κατάλογο.

Σχεδιασμός των Προγραμμάτων σας:

Σε φυσικό επίπεδο όλα τα στοιχεία του `cfs` τοποθετούνται κάτω από ένα ***μοναδικό αρχείο*** με επίθεμα `.cfs`.

Όπως συμβαίνει στα περισσότερα συστήματα αρχείων, έτσι και στο `cfs`, για κάθε οντότητα υπάρχει και η αντίστοιχη καταχώρηση μεταδεδομένων. Στο `ext2/ext3/ext4` η δομή αυτή ονομάζεται `inode`-structure.

Στο `cfs` η δομή θα πρέπει οπωσδήποτε να περιλαμβάνει όνομα αρχείου/καταλόγου/συνδέσμου, χαρακτηριστικό αριθμό αρχείου που είναι αντίστοιχο του `inode`-number και μπορεί να χρησιμοποιηθεί για την δεικτοδότηση των οντοτήτων του `cfs`, μέγεθος, τύπο, χρονόσημο δημιουργίας, τελευταίας πρόσβασης και τελευταίας τροποποίησης, γονικό κατάλογο (αν πρόκειται για κατάλογο) και μια λίστα μπλοκ δεδομένων. Όποιες άλλες προσθήκες είναι θετικές αλλά όχι υποχρεωτικές.

Ένα *παράδειγμα της δομής μεταδεδομένων* έχει ώς εξής:

```
typedef struct{
        unsigned int nodeid;
        char *filename;
        unsigned int size;
        unsigned int type;
        unsigned int parent-nodeid;
        time_t creation_time;
        time_t access_time;
        time_t modification_time;
        Datastream data;
        } MDS;

typedef struct{
        unsigned int datablocks[DATABLOCK_NUM];
        } Datastream;
```

Το πεδίο `nodeid` είναι ο χαρακτηριστικός αριθμός αρχείου που είναι μοναδικός στο `cfs`, το `size` είναι το μέγεθος του αρχείου και το `type` είναι ο τύπος του αρχείου. Οι βασικοί τύποι περιλαμβάνουν τους εξής: απλό αρχείο, κατάλογος, σύνδεσμος. Οποιαδήποτε άλλη προσθήκη είναι ευπρόσδεκτη.

Αν η οντότητα συστήματος αρχείου `cfs` είναι κατάλογος, τότε το πεδίο `parent-nodeid` συμβολίζει τον χαρακτηριστικό αριθμό του γονικού καταλόγου (νούμερο). Τα πεδία `creation_time,access_time` και `modification_time` είναι ο χρόνος δημιουργίας, τελευταίας πρόσβασης και τελευταίας τροποποίησης, αντίστοιχα. Τέλος, το πεδίο `data` συγκρατεί το πλήθος των μπλοκ που καταλαμβάνει η εν λόγω οντότητα.

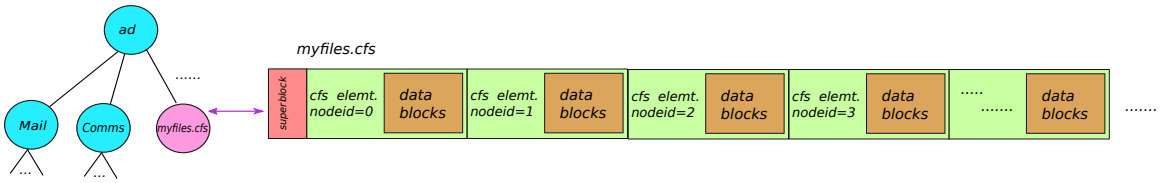
Στην παραπάνω δομή, απεικονίζεται η πιο απλή μορφή όσον αφορά στα `data` που είναι ένας πίνακας από αριθμούς μπλοκ, μεγίστου πλήθους `DATABLOCK_NUM`. Αυτό δεν σημαίνει ότι θα πρέπει να χρησιμοποιούνται και όλα τα μπλοκ.

Εναλλακτικά θα μπορούσατε να υιοθετήσετε το κλασσικό μοντέλο (δηλ. του `inode`) σύμφωνα με το οποίο υπάρχουν άμεσα και έμμεσα (μέχρι τρίτου επιπέδου) μπλοκ δεικτοδότησης για δεδομένα. Τα άμεσα δείχνουν απευθείας σε μπλοκ δεδομένων, ενώ τα έμμεσα σε άλλα μπλοκ δεικτοδότησης (άμεσα ή έμμεσα) για δεδομένα. Η `struct` που προτείνουμε είναι απλά πιο εύκολα διαχειρίσιμη και φυσικά είναι κοινή σε για αρχεία, καταλόγους και συνδέσμους.

Τα μπλοκ δεδομένων των αρχείων σε ένα `cfs` δεν ακολουθούν κάποια συγκεκριμένη μορφή. Μπορείτε να επιλέξετε εσείς την εν λόγω μορφή οργάνωσης/διαχείρισης. Απεναντίας, τα μπλοκ δεδομένων των καταλόγων θα περιλαμβάνουν -κατ΄ ελάχιστον- ζεύγη χαρακτηριστικού αριθμού αρχείων - ονομάτων αρχείων, τα οποία ανήκουν στον εκάστοτε κατάλογο. Τα ζεύγη μπορεί να είναι ταξινομημένα σε μια λίστα ή οποιαδήποτε άλλη δομή της επιλογής σας.

Στα πρώτα Bytes του αρχείου στο οποίο θα χτιστεί το `cfs` θα βρίσκεται η δομή μεταδεδομένων του αρχικού καταλόγου (root directory - /). Πέραν τούτου, το αρχείο θα περιλαμβάνει δομές μεταδεδομένων (π.χ. λίστα με όλα τα `nodeid`) καθώς επίσης και τα μπλοκ αρχείων/καταλόγων με σειρά που θα επιλέξετε. Το στοιχείο που απλοποιεί την κατάσταση είναι ότι η δομή μεταδεδομένων και το μπλοκ δεδομένων έχουν σταθερά μεγέθη. Η πρώτη είναι ίση με το μέγεθος της δομής, ενώ το δεύτερο έχει προκαθορισμένο μέγεθος, που αποφασίζεται κατά την αρχικοποίηση του `cfs`, π.χ. 1K.

Μια εναλλακτική προσέγγιση είναι να υπάρχει ένα superblock στο οποίο θα αποθηκεύονται κρίσιμες πληροφορίες σχετικές με το `cfs`, όπως το μέγεθος του μπλοκ δεδομένων, το μέγεθος της δομής μεταδεδομένων, τη θέση του δομής μεταδεδομένων του αρχικού καταλόγου, κ.ά.. Το superblock θα τοποθετείται στο πρώτο byte του αρχείου του `cfs`. Ένας τέτοιος σχεδιασμός για το `cfs` δίνεται στο Σχήμα 2. Όταν υπάρχουν διαγραφές αρχείων



Σχήμα 2: Υλοποίηση για το `myfiles.cfs` του χρήστη `ad`.

η/και καταλόγων οι αλλαγές αυτές θα πρέπει να αντικατοπτριστούν στο αρχείο `cfs` και είναι πιθανόν ότι το εν λόγω αρχείο δεν θα είναι πια συμπαγές (δηλ. θα έχει τρύπες).

Σχεδιασμός των Προγραμμάτων σας:

Αυτό που ζητείται στην παρούσα εργασία είναι η δημιουργία των εντολών που προαναφέρθηκαν. Η αρχιτεκτονική του `cfs`, εκτός των βασικών σημείων, επαφίεται στον καθένα. Πιθανή προσθήκη στην εργασία είναι η δημιουργία συστήματος δικαιωμάτων μέσω δημιουργίας ομάδων χρηστών και χρήσης bits προστασίας (`rwx`).

Επιπλέον, θα διευκόλυνε να υλοποιήσετε ένα εύκολα κατανοητό τρόπο που να δίνει την συνολική δομή του συστήματος αρχείου σας. Κάτι τέτοιο θα σας βοηθήσει στην αποσφαλμάτωση του προγράμματος σας.

Η σειρά με την οποία εμφανίζονται οι σημαίες δεν είναι προκαθορισμένη. Προφανώς μπορείτε να χρησιμοποιήσετε πιο πολλές σημαίες στα παραπάνω προγράμματα ώστε να διευκολυνθείτε ή και να σχεδιάσετε τα προγράμματά σας *ριζικά διαφορετικά*.

Τέλος καλό θα ήταν –και για λόγους επιβεβαίωσης και φυσικά μόνο για το σκοπό της άσκησης– να υπάρχει ένα μηχανισμός logging με την μορφή ενός append-only αρχείου. Εδώ, αποτυπώνεται όλη η δραστηριότητά μέχρι

στιγμής ώστε να έχετε ένα εύκολο τρόπο να βλέπετε τι γίνεται *όσον αφορά στην ταυτόχρονη εκτέλεση* των παραπάνω προγραμμάτων.

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 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. Η εργασία είναι είτε *ατομική* ή μπορεί να γίνει σε ομάδες των *δύο ατόμων*. Αν επιλέξετε έναν/μία συνάδελφο, καλό είναι να κάνετε μια καλή επιλογή ώστε να έχετε τον ίδιο βαθμό ενδιαφέροντος επιτυχούς ολοκλήρωσης της άσκησης.
2. Τα πρόγραμματα σας θα πρέπει να τρέχουν στα `Linux` συστήματα του τμήματος αλλιώς **δεν μπορούν να βαθμολογηθούν**.
3. Αν και αναμένεται να συζητήσετε με φίλους και συνεργάτες το πως θα επιχειρήσετε να δώσετε λύση στο πρόβλημα, αντιγραφή κώδικα (οποιαδήποτε μορφής) μεταξύ ομάδων είναι κάτι που **δεν επιτρέπεται** και δεν πρέπει να γίνει. Οποιοσδήποτε ομάδα βρεθεί αναμεμειγμένη σε αντιγραφή κώδικά **απλά παίρνει μηδέν** στο μάθημα. Αυτό ισχύει για **όσους εμπλέκονται ανεξάρτητα** από το ποια ομάδα 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 επιτρεπτή* πλατφόρμα για την υλοποίηση ή παρουσίαση αυτής της άσκησης.