Σκοπός

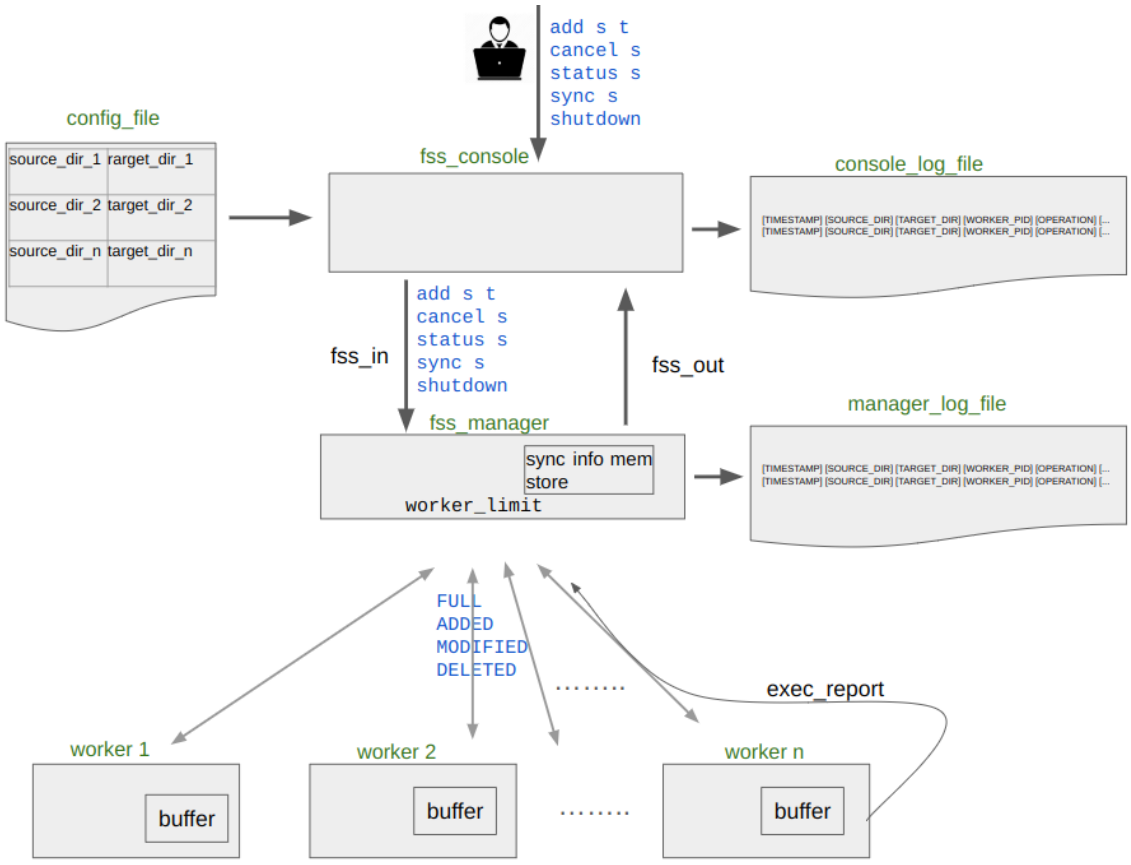
Στόχος αυτής της εργασίας είναι η εξοικείωσή σας με τις κλήσεις συστήματος της οικογένειας exec*(), της fork(), όπως επίσης και με κλήσεις συστήματος χαμηλού επιπέδου για τη διαχείριση αρχείων και το bash scripting.

Θα υλοποιήσετε ένα **Σύστημα Συγχρονισμού Αρχείων (FileSync System - FSS)** που θα διαχειρίζεται αυτόματα το συγχρονισμό αρχείων μεταξύ καταλόγων πηγής και των αντίστοιχων καταλόγων προορισμού. Το σύστημα θα διασφαλίζει να δημιουργούνται αντίγραφα αρχείων σε πραγματικό χρόνο, με δυνατότητα παρακολούθησης αλλαγών, διαχείρισης ερωτημάτων χρηστών και παροχής ενός μηχανισμού αναφοράς.

Η υλοποίησή σας πρέπει να περιλαμβάνει:

1. **fss_manager**: Μια εφαρμογή διαχειριστή υπεύθυνο για την οργάνωση της διαδικασίας συγχρονισμού,
2. **fss_console**: Μία διεπαφή χρήστη για τη διαχείριση και την υποβολή ερωτημάτων στο σύστημα, και
3. **fss_script.sh :** ενα βοηθητικό πρόγραμμα (bash script) για τη δημιουργία αναφορών και τον καθαρισμό παλαιών αντιγράφων αρχείων.

Διάγραμμα Γενικής Επισκόπησης Συστήματος FSS



Εφαρμογή fss_manager και worker processes

Επισκόπηση του fss_manager

Ο διαχειριστής **fss_manager,** ξεκινά αρχικοποιώντας το σύστημα, δημιουργώντας τα απαραίτητα κανάλια επικοινωνίας (named-pipes), πραγματοποιώντας τον αρχικο συγχρονισμό των καταλόγων και προετοιμάζοντας την εν συνεχεία παρακολούθηση τους για αλλαγές.

Παρακολούθηση Καταλόγων: Ο διαχειριστής παρακολουθεί συγκεκριμένους καταλόγους πηγής για αλλαγές, όπως δημιουργία, τροποποίηση ή διαγραφή αρχείων. Οι αλλαγές υποβάλλονται σε επεξεργασία από διεργασίες εργαζομένων (**worker processes**) που εκτελούν τις εργασίες συγχρονισμού.

Υποδοχή νέων αιτημάτων: Ο διαχειριστής υποδέχεται νέα αιτήματα για παρακολούθηση καταλόγων μεσω των καναλιών επικοινωνίας που έχει δημιουργήσει.

Αποθηκευση πληροφοριών σε εσωτερική δομή: Ο fss-manager χρησιμοποιεί δομές δεδομένων sync_info_mem_store (δικής σας επιλογής) για την αποδοτική αποθήκευση και αναζήτηση πληροφοριών σχετικά με όλους τους παρακολουθούμενους καταλόγους . Η sync_info_mem_store (π.χ. hash table ή linked list) αποθηκεύει πληροφορίες ανά παρακολουθούμενο κατάλογο πχ (source_dir, target_dir, status, last_sync_time, active, error_count).

Διαχείριση Ορίου Εργαζομένων: Εάν έχει ήδη επιτευχθεί ο μέγιστος επιτρεπτός αριθμός διεργασιών εργαζομένων, ο διαχειριστής τοποθετεί επιπλέον εργασίες συγχρονισμού σε ουρά και τις εκτελεί όταν ελευθερωθούν εργαζόμενοι.

Εκτέλεση fss_manager

Η εφαρμογή fss_manager θα χρησιμοποιείται ως εξής:

```
./fss_manager -l <manager_logfile> -c <config_file> -n <worker_limit>
```

όπου:

- Η παράμετρος `logfile` είναι το αρχείο καταγραφής του συστήματος.
- Η παράμετρος `config_file` είναι ένα αρχείο που περιέχει ζεύγη καταλόγων. Συγκεκριμένα, κάθε γραμμή περιέχει ένα ζεύγος, (`source_dir`, `target_dir`). Το `source_dir` υποδεικνύει τον κατάλογο του οποίου τα περιεχόμενα θα αντιγραφούν στον κατάλογο `target_dir`. Το αρχείο config_file περιέχει ζεύγη source και target καταλόγων, διαχωρισμένα με ένα κενό. Υποθέτουμε πως τα source/target dirs δεν είναι αλληλοκαλυπτόμενα δηλαδή δεν θα υπάρχουν διαφορετικές γραμμές που να είναι sub-directories άλλων γραμμών.
- Η παράμετρος `worker_limit` είναι ο μέγιστος αριθμός διεργασιών εργαζομένων (worker processes) που μπορούν να εκτελούνται ταυτόχρονα (default τιμή 5).

Λειτουργικότητα κατά την εκκίνηση

Ο fss_manager στην αρχή δημιουργεί δύο named-pipes (`fss_in` και `fss_out`) για την επικοινωνία με το fss_console.

Στη συνέχεια, προετοιμάζει την παρακολούθηση καταλόγων που καθορίζονται στο `config_file` και ξεκινάει τον συγχρονισμό τους. Συγκεκριμένα, για κάθε ζεύγος (`source_dir, target_dir`) στο `config_file` ξεκινάει έναν νέο worker process που θα αναλάβει τον συγχρονισμό των δυο καταλόγων και το καταγράφει στο manager-log-file:

```
Τυπώνει στην οθόνη και γράφει στο manager-log-file
[2025-02-10 10:00:01] Added directory: /home/user/docs -> /backup/docs
[2025-02-10 10:00:01] Monitoring started for /home/user/docs
```

Για λόγους απλότητας θεωρήστε ότι οι κατάλογοι ειναι flat και περιέχουν μόνο αρχεία και όχι υποκαταλόγους.
Εάν έχει ήδη επιτευχθεί ο μέγιστος επιτρεπτός αριθμός (`worker_limit`) των worker processes, ο fss_manager τοποθετεί επιπλέον εργασίες συγχρονισμού σε ουρά (queue) και τις εκτελεί όταν ολοκληρώσει την εργασία του και τερματίσει κάποιος worker. Ο fss_manager επικοινωνεί με τους worker processes μέσω pipes (περιγράφεται παρακάτω).
Ο διαχειριστής (`fss_manager`) **πρέπει** να χρησιμοποιεί τις **συναρτήσεις συστήματος inotify** (`inotify_init(), inotify_add_watch(), read() inotify_rm_watch()`) για να παρακολουθεί καταλόγους **χωρίς να δημιουργεί επιπλέον διεργασίες**. Οι αλλαγές που θα πρέπει να παρακολουθούνται είναι μόνο η δημιουργία, τροποποίηση, και διαγραφή αρχείων (όχι μετακίνηση, όχι μετονομασία).

Λήψη εντολών από fss_console

Αφού ξεκινήσει την παρακολούθηση και συγχρονισμό καταλόγων που αναφέρονται στο `config_file`, ο fss_manager περιμένει να δεχθεί εντολές μέσω του named pipe `fss_in` από το fss_console.

Για κάθε εντολή που δέχεται ο fss_manager την επεξεργάζεται και την έξοδο:

1. τη στέλνει πίσω στο fss_console μέσω του `fss_out` named pipe, και

2. την αποθηκεύει στο αρχείο καταγραφής (manager-log-file). Το αρχείο αυτό είναι διαφορετικό από το console-log-file της κονσόλας που θα αναφερθεί παρακάτω.

Εντολές που μπορούν να δεχθεί ο fss_manager

`add <source> <target>`

Καταχωρεί τον κατάλογο `<source>` για παρακολούθηση και ξεκινά άμεσα τον συγχρονισμό των περιεχομένων του στον κατάλογο `<target>`. Μετέπειτα αλλαγές στο `<source>` συγχρονίζονται σε πραγματικό χρόνο:

```
Τυπώνει στην οθόνη και γράφει στο manager-log-file
[2025-02-10 10:00:01] Added directory: /home/user/docs -> /backup/docs
[2025-02-10 10:00:01] Monitoring started for /home/user/docs
```

[Σημείωση: αν είναι ήδη καταχωρημένο το ζευγος `<source, target>` για παρακολούθηση και συγχρονισμό, δεν ξεκινάει συγχρονισμό.]

```
Τυπώνει στην οθόνη μόνο
```

```
[2025-02-10 10:00:01] Already in queue: /home/user/docs
```

cancel <source dir>

Ακυρώνει την παρακολούθηση του καταλόγου <source dir>:

```
Τυπώνει στην οθόνη και γράφει στο manager-log-file
[2025-02-10 10:23:01] Monitoring stopped for /home/user/docs
```

Αν ο κατάλογος δεν παρακολουθείται τότε:

```
Τυπώνει στην οθόνη μόνο
[2025-02-10 10:00:01] Directory not monitored: /home/user/docs
```

status <source dir>

Επιστρέφει πληροφορίες για την κατάσταση συγχρονισμού του καταλόγου <source dir> όπως ο χρόνος τελευταίου συγχρονισμού μαζί με όποια λάθη προέκυψαν, και αν παρακολουθείται ενεργά ή όχι ο κατάλογος <source_dir> την παρούσα στιγμή. Χρησιμοποιεί πληροφορίες από την sync_info_mem_store. Τυπώνει στην οθόνη μόνο:

```
[2025-04-10 10:00:01] Status requested for /home/user/docs
Directory: /home/user/docs
Target: /backup/docs
Last Sync: 2025-03-30 14:25:30
Errors: 0
Status: Active
```

Αν ο κατάλογος source δεν είναι ενεργός τυπώνει στην οθόνη μόνο

```
[2025-04-10 10:00:01] Directory not monitored: /home/user/docs
```

sync <source dir>

Εκκινεί έναν συγχρονισμό του καταλόγου <source_dir> ανεξάρτητα από το αν έχουν εντοπιστεί αλλαγές. Ο διαχειριστής αναζητά τον κατάλογο προορισμού στη δομή δεδομένων (sync_info_mem_store) στη μνήμη του.

```
Τυπώνει στην οθόνη και γράφει στο manager-log-file
[2025-02-10 10:23:01] Syncing directory: /home/user/docs -> /backup/docs
[2025-03-30 10:24:00] Sync completed /home/user/docs -> /backup/docs Errors:0
```

[Σημείωση: αν ήδη εκτελείται μια εργασία συγχρονισμού από κάποιο worker για τον κατάλογο source_dir, δεν ξεκινάει άλλη εργασία συγχρονισμού ο fss_manager και απλώς περιμένει να τελειώσει η υπάρχουσα που τρέχει.]

```
Τυπώνει στην οθόνη
[2025-03-30 10:24:00] Sync already in progress /home/user/docs
```

shutdown

Σταματά την παρακολούθηση καταλόγων, περιμένει τις ενεργές worker διεργασίες να τερματίσουν, επεξεργάζεται όσες εργασίες συγχρονισμού έχουν μείνει στην ουρά αναμονής και τερματίζει ομαλά:

```
Τυπώνει στην οθόνη μόνο
[2025-02-10 10:23:02] Shutting down manager...
[2025-02-10 10:23:02] Waiting for all active workers to finish.
[2025-02-10 10:23:03] Processing remaining queued tasks.
[2025-02-10 10:24:01] Manager shutdown complete.
```

Διεργασίες Εργαζομένων (Worker Processes):

Κάθε worker εκτελεί μια μεμονωμένη εργασία συγχρονισμού, καθορίζει μια κατάσταση εξόδου για επιτυχία ή αποτυχία και στη συνέχεια τερματίζεται. Μια εργασία συγχρονισμού προκύπτει σε δύο περιπτώσεις.

Όταν ο fss_manager :

1. διαβάζει ένα νέο ζεύγος στο config-file ή
2. εντοπίζει μια αλλαγή σε έναν κατάλογο που ήδη παρακολουθεί μέσω inotify.

Για να ξεκινήσει μια εργασία συγχρονισμού, ο fss_manager δημιουργεί ένα pipe() και καλεί τη fork(), δημιουργώντας μια νέα διεργασία.

Η διεργασία-παιδί ανακατευθύνει το stdout προς το pipe και στη συνέχεια καλεί exec() για να τρέξει το `worker` binary που παίρνει ως είσοδο 4 παραμέτρους:

- `source_directory`: ο κατάλογος πηγής από τον οποίο συγχρονίζονται τα αρχεία.
- `target_directory`: ο κατάλογος προορισμού στον οποίο συγχρονίζονται τα αρχεία.
- `filename`: το όνομα του αρχείου που τροποποιήθηκε (χρησιμοποιείται μόνο στην περίπτωση εντόπισης αλλαγής μέσω inotify). Σε περίπτωση πλήρους συγχρονισμού (initial/full sync), μπορεί να είναι `ALL`.
- `Operation`: ο τύπος λειτουργίας:
  - `FULL` (για πλήρη συγχρονισμό)
  - `ADDED`(για δημιουργία νέου αρχείου),
  - `MODIFIED`(για τροποποίηση ενός υπάρχοντος αρχείου)
  - `DELETED` (για διαγραφή ενός υπάρχοντος αρχείου)

Αν δηλαδή ο worker λάβει filename = ALL και operation = FULL, εκτελεί πλήρη αντιγραφή όλων των αρχείων από source σε target.

Υλοποιηση εργασίας συγχρονισμού

Στην υλοποίησή σας, καθε worker πρέπει να χρησιμοποιεί χαμηλού επιπέδου I/O syscalls (π.χ., open, read, write, unlink, close) για την εργασία συγχρονισμού που επεξεργάζεται.

(Σημείωση: **Απαγορεύεται** η χρήση κλήσεων όπως (`system("cp -r source dest"` ή `exec("rsync",...))`. Ο worker πρέπει να ελέγχει την επιτυχία ή αποτυχία κάθε λειτουργίας που εκτελεί κατά τη διάρκεια του συγχρονισμού αρχείων. Αυτό γίνεται με δύο βασικούς τρόπους: 1) Έλεγχος των συναρτήσεων συστήματος που χρησιμοποιούνται για την αντιγραφή/μεταφορά αρχείων και 2) Χρήση strerror() για περιγραφή σφάλματος. Αν οποιαδήποτε από τις συναρτήσεις συστήματος επιστρέψει -1, τότε η strerror(errno) τυπώνει τον ακριβή λόγο της αποτυχίας**. Ο worker καταγράφει όλα τα σφάλματα σε ένα buffer στη μνήμη .** Όταν ο worker τελειώσει την εργασία συγχρονισμού, πριν καλέσει την exit() syscall, στέλνει **ένα τελικό συνοπτικό μήνυμα (exec_report) μαζί με όλα τα σφάλματα** (αν έχουν προκύψει) μέσω του pipe στο fss_manager.

Παράδειγμα exec_report

EXEC_REPORT_START
STATUS: PARTIAL
DETAILS: 7 files copied, 2 skipped
ERRORS:
- File X: Permission denied
- File Y: Read error
EXEC_REPORT_END

Όταν ο worker τελειώσει, το fss_manager λαμβάνει το SIGCHLD σήμα και καλεί wait() ή waitpid() για να συλλέξει την κατάσταση εξόδου του worker, να ενημερώσει τις δομές δεδομένων (sync_info_mem_store) του και να εκκινήσει νέους εργαζομένους εάν το επιτρέπει το `worker_limit`.

Καταγραφή σε manager-log-file εργασίας συγχρονισμού

Ο fss_manager γράφει στο manager-log-file μια εγγραφή που περιγράφει την εργασία συγχρονισμού που ολοκληρώθηκε από τον worker. Το format της εγγραφής θα είναι ως εξής:

`[TIMESTAMP] [SOURCE_DIR] [TARGET_DIR] [WORKER_PID] [OPERATION] [RESULT] [DETAILS]`

όπου:

- `TIMESTAMP` είναι η ημερομηνία και ώρα του γεγονότος
- `SOURCE_DIR` είναι ο κατάλογος προέλευσης (`source`).
- `TARGET_DIR` είναι ο κατάλογος προορισμού (`target`)
- `WORKER_PID` είναι το pid της διεργασίας worker.
- `OPERATION` είναι ο τύπος συγχρονισμού: `FULL`, `ADDED`, `MODIFIED`, `DELETED`.
- `RESULT` είναι το αποτέλεσμα της εργασίας συγχρονισμού: `SUCCESS`, `ERROR`, ή `PARTIAL`.
- `DETAILS` είναι λεπτομέρειες ή περιγραφή του αποτελέσματος

Αρχικός Πλήρης Συγχρονισμός (Full Sync) από το `config.txt`:

```
[2025-02-10 10:00:01] [/home/user/docs] [/backup/docs] [1234] [FULL]
[SUCCESS] [10 files copied]

[2025-02-10 10:00:02] [/home/user/photos] [/backup/photos] [1235] [FULL]
[PARTIAL] [5 files copied, 2 skipped]
```

Συγχρονισμός Μεμονωμένης Αλλαγής από `inotify` (προσθήκη αρχείου)

```
[2025-02-10 10:15:10] [/home/user/docs] [/backup/docs] [1237] [ADDED]
[SUCCESS] [File: report.pdf]
```

Συγχρονισμός Μεμονωμένης Αλλαγής (τροποποίηση αρχείου):

```
[2025-02-10 10:20:15] [/home/user/docs] [/backup/docs] [1240] [MODIFIED]
[SUCCESS] [File: notes.txt]
```

Συγχρονισμός Μεμονωμένης Αλλαγής (διαγραφή αρχείου):

```
[2025-02-10 10:25:20] [/home/user/docs] [/backup/docs] [1245] [DELETED]
[SUCCESS] [File: old_draft.docx]
```

Παράδειγμα Αποτυχίας σε Συγχρονισμό (Error Case):

```
[2025-02-10 10:30:10] [/home/user/docs] [/backup/docs] [1249] [MODIFIED]
[ERROR] [File: budget.xlsx - Permission Denied]
```

Εμφάνιση Σφάλματος με Μερική Επιτυχία (Partial Success):

```
[2025-02-10 10:32:40] [/home/user/photos] [/backup/photos] [1250] [FULL]
[PARTIAL] [7 files copied, 2 skipped]
```

Παράδειγμα Ολοκληρωμένου `fss_manager.log`:

```
[2025-02-10 10:00:01] [/home/user/docs] [/backup/docs] [1234] [FULL]
[SUCCESS] [10 files copied]
[2025-02-10 10:00:02] [/home/user/photos] [/backup/photos] [1235] [FULL]
[PARTIAL] [5 files copied, 2 skipped]
[2025-02-10 10:15:10] [/home/user/docs] [/backup/docs] [1237] [ADDED]
[SUCCESS] [File: report.pdf]
[2025-02-10 10:20:15] [/home/user/docs] [/backup/docs] [1240] [MODIFIED]
[SUCCESS] [File: notes.txt]
[2025-02-10 10:25:20] [/home/user/docs] [/backup/docs] [1245] [DELETED]
[SUCCESS] [File: old_draft.docx]
[2025-02-10 10:30:10] [/home/user/docs] [/backup/docs] [1249] [MODIFIED]
[ERROR] [File: budget.xlsx - Permission Denied]
[2025-02-10 10:32:40] [/home/user/photos] [/backup/photos] [1250] [FULL]
[PARTIAL] [7 files copied, 2 skipped]
```

Διαχείριση Σημάτων (Signal Handling)

Το σύστημα πρέπει να υποστηρίζει και να διαχειρίζεται τα παρακάτω σήματα:

- **SIGCHLD**: Παράγεται όταν ένας worker process τερματίζεται. Ο fss_manager το χειρίζεται με τον τρόπο που περιγράφεται παραπάνω.

Εφαρμογή fss_console

Οι χρήστες αλληλεπιδρούν με το σύστημα μέσω του **fss_console**, το οποίο επικοινωνεί με τον διαχειριστή για την προσθήκη και διαγραφή καταλόγων, την αποστολή ερωτημάτων κατάστασης και την ενεργοποίηση συγχρονισμού καταλόγων.

Η εφαρμογή fss_console παίρνει μοναδικό όρισμα το όνομα του console-log αρχείου και θα χρησιμοποιείται ως εξής:

```
./fss_console -l <console-logfile>
```

Μετά την εκκίνηση της fss_console οι χρήστες μπορούν να εκτελούν τις παρακάτω εντολές (case sensitive):

- `add <source> <target>`: Προσθέτει ένα νέο ζεύγος καταλόγων προς συγχρονισμό.
- `status <directory>`: Επιστρέφει πληροφορίες για την κατάσταση συγχρονισμού ενός καταλόγου.
- `cancel <source>` : Ακυρώνει την παρακολούθηση του καταλόγου source. (Σημείωση: Αν υπάρχει εργασία συγχρονισμού στην ουρά που αφορά τον κατάλογο `<source>`, την αφήνουμε στην ουρά.)
- `sync <directory>`: Εκκινεί άμεσα τον συγχρονισμό για έναν συγκεκριμένο κατάλογο.
- `shutdown`: Σταματά τη λειτουργία του διαχειριστή και των εργαζομένων.

Κατ΄ αντιστοιχία με τον fss_manager το fss_console χρησιμοποιεί ένα δικό του console-log-file διαφορετικό από αυτό του fss-manager για να καταγράψει με απλό ομως τρόπο τις εντολές που εισάγονται από τον χρήστη πχ

```
[2025-02-10 10:00:01] Command add /home/user/docs -> /backup/docs
[2025-02-10 10:00:01] Command status /home/user/docs
[2025-02-10 10:23:01] Command cancel /home/user/docs
[2025-02-10 10:23:01] Command sync /home/user/docs
[2025-02-10 10:23:01] Command shutdown /home/user/docs
```

Το fss_console στέλνει τις εντολές στον fs_manager μέσω του fss_in named pipe. Ο fss_manager απαντά με τα αντίστοιχα μηνύματα κατάστασης, τα αποτελέσματα συγχρονισμού ή τυχόν σφάλματα, γράφοντάς τα στο fss_out named pipe, από το οποίο το fss_console τα διαβάζει και στη συνέχεια

το fss_console (προσοχή όχι ο fss_manager)

- εμφανίζει στον χρήστη τα αποτελέσματα ή τυχόν σφάλματα και
- τα καταγράφει στο console-log-file του.

Η εμφάνιση των αποτελεσμάτων και η καταγραφή τους στο console-log-file είναι πανομοιότυπη τη μορφοποίηση που χρησιμοποιήθηκε για την εγγραφή στο manager-log-file.

Το fss_console τερματίζεται όταν ο χρήστης την κλείσει ή όταν εκτελεστεί η εντολή shutdown.

Παράδειγμα χρήσης:

```
$ ./fss_console -l console-logfile
> add /home/user/docs /backup/docs
[2025-02-10 10:00:01] Added directory: /home/user/docs -> /backup/docs
[2025-02-10 10:00:01] Monitoring started for /home/user/docs

> status /home/user/docs
[2025-02-10 10:00:01] Status requested for /home/user/docs
Directory: /home/user/docs
Target: /backup/docs
Last Sync: 2025-03-30 14:25:30
Errors: 0
Status: Active

> shutdown
[2025-02-10 10:23:02] Shutting down manager...
[2025-02-10 10:23:02] Waiting for all active workers to finish.
[2025-02-10 10:23:03] Processing remaining queued tasks.
[2025-02-10 10:24:01] Manager shutdown complete.
```

Bash fss_script.sh

Το fss_script.sh δημιουργεί αναφορές σχετικά με συγχρονισμένους καταλόγους και μπορεί να διαγράψει παλιά αντίγραφα καταλόγων όταν είναι απαραίτητο. Είναι ένα Bash script που χρησιμοποιείται ως εξής:

```
./fss_script.sh -p <path> -c <command>
```

όπου:

- Η παράμετρος `path` είναι ένα αρχείο καταγραφής (log-file) ή ένας target κατάλογος
- Η παράμετρος `command` είναι ένα από τα ακόλουθα:
    - `listAll`: Αναζητά στο log-file `(path)` και εμφανίζει όλους τους καταλόγους, την ημερομηνία και ώρα τελευταίου συγχρονισμού, μαζί με status (Success, Error, Partial)
    - `listMonitored`: Εμφανίζει όλους τους καταλόγους που παρακολουθούνται ενεργά.
    - `listStopped`: Εμφανίζει όλους τους καταλόγους που δεν παρακολουθούνται πλέον.

- `purge`: Διαγράφει το αρχείο/κατάλογο που βρίσκεται στο `path`. Η εντολή purge διαγράφει μόνο τον target κατάλογο ή το logfile, όχι τα source dirs.

Listing all directories:

```
/home/user/docs -> /backup/docs [Last Sync: 2025-03-30 14:25:30] [SUCCESS]
/home/user/pics -> /backup/pics [Last Sync: 2025-02-28 12:37:32] [PARTIAL]
/home/user/photos -> /backup/photos [Last Sync: 2025-03-31 14:25:30] [ERROR]
```

Listing monitored directories:

```
/home/user/docs -> /backup/docs [Last Sync: 2025-03-30 14:25:30]
/home/user/pics -> /backup/pics [Last Sync: 2025-02-28 12:37:32]
```

Listing stopped directories:

```
/home/user/photos -> /backup/photos [Last Sync: 2025-03-31 14:25:30]
```

Purging backup directory:

```
Deleting /backup/pics...
Purge complete.
```

Purging a log-file:

```
Deleting manager-logfile...
Purge complete.
```

Helpful tips

- Μπορείτε να υποθέσετε πως οι κατάλογοι θα είναι όλοι flat και θα περιέχουν μόνο αρχεία, όχι subdirectories.

- Για κάθε source επιτρέπεται ένα μοναδικό ζεύγος (source, target). Εάν δοθεί νέο target για ήδη γνωστό source, η εντολή απορρίπτεται.

- Φροντίστε να καθαρίζετε τα logfiles και τα pipes κατά την εκκίνηση των προγραμματων σας απο πιθανές προηγούμενες εκτελέσεις ώστε να μη δημιουργούνται προβλήματα λειτουργίας. Ο fss_manager οφείλει κατά την έναρξη να ελέγχει την ύπαρξη των named pipes και, αν υπάρχουν, να τα διαγράφει.

- Η εύρεση των αρχείων του καταλόγου γίνεται στον fss_manager. Το fss_console στέλνει μόνο το source_dir.

- Θα πρέπει να σκεφτείτε πως θα διαχειρίζεται ο fss_manager τα πολλαπλά file descriptors που έχει ανοιχτά (pipes, named pipes, inotify fd). Όταν δυο διεργασίες επικοινωνούν μέσω named pipes (κλήση συστήματος mkfifo()), η προκαθορισμένη συμπεριφορά των named pipes είναι να μπαίνει σε κατάσταση αναμονής η διεργασία που ανοίγει το ένα άκρο μέχρι να ανοιχτεί η σωλήνωση και από το άλλο άκρο. Βέβαια, μπορούμε να αποφύγουμε την παραπάνω συμπεριφορά αν θέσουμε το O_NONBLOCK flag στο δεύτερο όρισμα της κλήσεως συστήματος open(). Για παράδειγμα, αν θέλουμε να ανοίξουμε ένα named pipe για ανάγνωση χωρίς να τεθούμε σε αναμονή, κάνουμε την κλήση open(pipe_name, O_RDONLY | O_NONBLOCK). Προσοχή, αν σε αυτήν την περίπτωση ανοιχτεί το named pipe για γράψιμο (από το ένα άκρο) και δεν είναι ανοιχτό για διάβασμα (από το άλλο άκρο), τότε θα επιστραφεί σφάλμα. Είστε ελεύθεροι να διαλέξετε όποια μέθοδο λειτουργίας των σωληνώσεων θέλετε.

- Μπορείτε να θεωρήσετε ότι σε περίπτωση που υπάρχει μια εκδοση ενός αρχείου στο target dir μπορείτε απλώς να το κάνετε overwrite χωρίς να ελέγξετε χρονοσημάνσεις.

- Όλες οι εγγραφές ημερομηνίας/ώρας πρέπει να παραχθούν με μορφή

  ("%Y-%m-%d %H:%M:%S").

Διαδικαστικά

- Το πρόγραμμά σας θα πρέπει να γραφεί σε C (ή C++) και σας θα πρέπει να τρέχει στα Linux workstations του Τμήματος. Κώδικας που δε μεταγλωττίζεται εκεί, θεωρείται ότι δεν μεταγλωττίζεται. Δε θα γίνει αποδεκτή η εξέταση της εργασίας σε άλλον υπολογιστή.
- Για επιπρόσθετες ανακοινώσεις, παρακολουθείτε το forum του μαθήματος στο piazza.com.

Η πλήρης διεύθυνση είναι https://piazza.com/uoa.gr/spring2025/k24/home. Η παρακολούθηση του φόρουμ στο Piazza είναι υποχρεωτική.

- Ο κώδικάς σας θα πρέπει να αποτελείται από τουλάχιστον δύο (και κατά προτίμηση περισσότερα) διαφορετικά αρχεία. Η χρήση του separate compilation είναι επιτακτική και ο κώδικάς σας θα πρέπει να έχει ένα Makefile.
- Βεβαιωθείτε πως ακολουθείτε καλές πρακτικές software engineering κατά την υλοποίηση της άσκησης. Η οργάνωση, η αναγνωσιμότητα και η ύπαρξη σχολίων στον κώδικα αποτελούν κομμάτι της βαθμολογίας σας.
- Ο κώδικάς σας θα πρέπει να κάνει compile στα εκτελέσιμα fss_manager, fss_console, και worker όπως **ακριβώς** ορίζει η άσκηση τα οποία θα λειτουργούν με τις παραμέτρους ακριβώς όπως προδιαγράφονται. **(penalty -10%).**

Τι πρέπει να παραδοθεί

Όλη η δουλειά σας θα παραδοθεί στο github repository που θα πρεπει να δημιουργήσετε ακολουθώντας τις οδηγίες στο κείμενο
https://docs.google.com/document/d/12_a_Uuw1FtGdgHK0T9LQKFiKp6m6rGhEylBv4HsydGs/edit?usp=sharing

Να υποβάλετε μόνο κώδικα, Makefile, README και **όχι τα binaries**. Η άσκησή σας θα γίνει compile από την αρχή πριν βαθμολογηθεί.

- Όποιες σχεδιαστικές επιλογές κάνετε, θα πρέπει να τις περιγράψετε σε ένα README (απλό text αρχείο) που θα υποβάλλετε μαζί με τον κώδικά σας. Το README χρειάζεται να περιέχει μια σύντομη και περιεκτική εξήγηση για τις επιλογές που έχετε κάνει στον σχεδιασμό του προγράμματός σας σε 1-2 σελίδες ASCII κειμένου. Συμπεριλάβετε την εξήγηση και τις οδηγίες για το compilation και την εκτέλεση του προγράμματός σας.

- Ο κώδικας που θα υποβάλετε θα πρέπει να είναι δικός σας. Απαγορεύεται η χρήση κώδικα που δεν έχει γραφεί από εσάς ή κώδικας που έχει γραφτεί με τη βοήθεια μηχανών τύπου chatGPT. Θα χρησιμοποιηθούν AI code detector για την αναγνώριση τέτοιου είδους προσπαθειών.

- Θα παρθεί snapshot των εργασιών από το github την ημέρα και ώρα λήξης της προθεσμίας. Κανένα commit μετα από αυτό δε θα γίνει δεκτό.

Υποχρεούστε να διατηρήσετε τον κώδικά σας ιδιωτικό στο github

Τι θα βαθμολογηθεί

● Η συμμόρφωση του κώδικά σας με τις προδιαγραφές της άσκησης.
● Η οργάνωση και η αναγνωσιμότητα (μαζί με την ύπαρξη σχολίων) του κώδικα.
● Η χρήση Makefile και το separate compilation.

Άλλες σημαντικές παρατηρήσεις

● Οι εργασίες είναι ατομικές.
● Όποιος υποβάλλει / δείχνει κώδικα που δεν έχει γραφτεί από την ίδια/τον ίδιο **μηδενίζεται** στο μάθημα.
● Ακολουθήστε αυστήρα το format του log file και του config file που διαβάζετε **(penalty -20%).**
● Ακολουθήστε αυστηρά οτι ζητείται να εκτυπώνεται στην έξοδο. Τίποτα περισσότερο τίποτα λιγότερο. Οχι debug μηνύματα **(penalty -20%).**
● Αν και αναμένεται να συζητήσετε με φίλους και συνεργάτες το πώς θα επιχειρήσετε να δώσετε λύση στο πρόβλημα, αντιγραφή κώδικα (οποιασδήποτε μορφής) είναι κάτι που δεν επιτρέπεται. **Οποιοσδήποτε** βρεθεί αναμεμειγμένος σε αντιγραφή κώδικα απλά παίρνει μηδέν στο μάθημα. Αυτό ισχύει για όσους εμπλέκονται ανεξάρτητα από το ποιος έδωσε/πήρε κλπ. Τονίζουμε πως θα πρέπει να λάβετε τα κατάλληλα μέτρα ώστε να είναι προστατευμένος ο κώδικάς σας και να μην αποθηκεύεται κάπου που να έχει πρόσβαση άλλος χρήστης (π.χ., η δικαιολογία «Το είχα βάλει σε ένα github repo και μάλλον μου το πήρε από εκεί», δεν είναι δεκτή.)
● Το πρόγραμμά σας θα πρέπει να γραφτεί σε C ή C++. Μπορείτε να χρησιμοποιήσετε μόνο εντολές οι οποίες είναι διαθέσιμες στα μηχανήματα Linux του τμήματος. Πρόγραμμα που πιθανόν μεταγλωτίζεται ή εκτελείται στο προσωπικό σας υπολογιστή αλλά όχι στα μηχανήματα Linux του τμήματος, θεωρείται ότι δε μεταγλωτίζεται ή εκτελείται αντίστοιχα.