Εισαγωγή

Στόχος αυτής της εργασίας είναι να εξοικειωθείτε με τη δημιουργία διεργασιών (processes) χρησιμοποιώντας τα system calls fork/exec, την επικοινωνία διεργασιών μέσω pipes, τη χρήση low-level I/O και τη δημιουργία bash scripts.

Στα πλαίσια αυτής της εργασίας θα υλοποιήσετε ένα κατανεμημένο εργαλείο `travelMonitor` που θα δέχεται αιτήματα από πολίτες που θέλουν να ταξιδέψουν σε άλλες χώρες, θα ελέγχει αν έχουν κάνει τον κατάλληλο εμβολιασμό, και θα εγκρίνει αν επιτρέπεται η είσοδος σε μια χώρα από έναν ταξιδιώτη. Συγκεκριμένα, θα υλοποιήσετε την εφαρμογή `travelMonitor` η οποία θα δημιουργεί μια σειρά από monitor διεργασίες που, μαζί με την εφαρμογή, θα απαντούν σε ερωτήματα του χρήστη.

Α) Η εφαρμογή travelMonitor (90%)

Η εφαρμογή `travelMonitor` θα χρησιμοποιείται ως εξής:

```
./travelMonitor –m numMonitors -b bufferSize -s sizeOfBloom -i input_dir
```

όπου:

- Η παράμετρος `numMonitors` είναι ο αριθμός Monitor διεργασιών που θα δημιουργήσει η εφαρμογή.
- Η παράμετρος `bufferSize:` είναι το μέγεθος του buffer για διάβασμα πάνω από τα pipes.
- Η παράμετρος `sizeOfBloom` καθορίζει το μέγεθος των bloom filter σε *bytes*. Ενδεικτικό μέγεθος του bloom filter για τα δεδομένα της άσκησης θα είναι της τάξης των 100Kbytes. Η παράμετρος αυτή είναι της ίδιας λογικής ακριβώς με την αντίστοιχη παράμετρο της πρώτης εργασίας.
- Η παράμετρος `input_dir`: είναι ένα directory το οποίο περιέχει subdirectories με τα αρχεία που θα επεξεργάζονται οι Monitor processes. Κάθε subdirectory θα έχει το όνομα μιας χώρας και θα περιέχει ένα ή περισσότερα αρχεία. Για παράδειγμα, το `input_dir` θα μπορούσε να περιέχει subdirectories `China/` `Italy/` και `Germany/` τα οποία έχουν τα εξής αρχεία:
-

```
/input_dir/China/China-1.txt
/input_dir/China/China-2.txt
/input_dir/China/China-3.txt
…
/input_dir/Italy/Italy-1.txt
/input_dir/Italy/Italy-2.txt
…
/input_dir/France/France-1.txt
/input_dir/France/France-2.txt
/input_dir/France/France-3.txt
…
```

- Κάθε αρχείο περιέχει μια σειρά από εγγραφές πολίτων όπου κάθε γραμμή θα περιγράφει την κατάσταση εμβολιασμού ενός πολίτη για κάποια συγκεκριμένη ίωση. Για παράδειγμα, αν τα περιεχόμενα στο αρχείο /input_dir/France/France-1.txt .txt είναι:

```
889 John Papadopoulos France 52 COVID-19 YES 27-12-2020
889 John Papadopoulos France 52 H1N1 NO
776 Maria Tortellini  France 36 SARS-1 NO
125 Jon Dupont France 76 H1N1 YES 30-10-2020
```

σημαίνει πως στη Γαλλία έχουμε έναν πολίτη (John Papadopoulos) που έχει εμβολιαστεί για COVID-19 στις 27-12-2020 αλλά όχι για H1N1, την Maria Tortellini που δεν έχει εμβολιαστεί ακόμα για το SARS-1, και τον Jon Dupont ο οποίος εμβολιάστηκε για το H1N1. Αντίστοιχα με την πρώτη άσκηση αν υπάρχουν γραμμές που είναι contradicting με προηγούμενες, μπορείτε να πετάτε τις εγγραφές που δημιουργούν inconsistency με τον ίδιο τρόπο όπως και στην 1η άσκηση.

Συγκεκριμένα, μια εγγραφή είναι μια γραμμή ASCII κειμένου που αποτελείται από τα εξής στοιχεία με αυτή τη σειρά:

- `citizenID`: μια συμβολοσειρά (μπορεί να έχει και ένα μόνο ψηφίο) που καθορίζει την κάθε τέτοια εγγραφή.
- `firstName`: μια συμβολοσειρά που αποτελείται από γράμματα χωρίς κενά.
- `lastName`: μια συμβολοσειρά που αποτελείται από γράμματα χωρίς κενά.
- `age`: ένας θετικός (>0), ακέραιος <= 120.
- `virusName`: μια συμβολοσειρά που αποτελείται από γράμματα, αριθμούς, και ενδεχομένως και μια παύλα "-" αλλά χωρίς κενά.
- `YES` ή `NO`: υποδεικνύει αν ο πολίτης έχει κάνει το εμβόλιο κατά τον συγκεκριμένο ιό.
- `dateVaccinated`: ημερομηνία που εμβολιάστηκε ο πολίτης. Αν το προηγούμενο πεδίο είναι `NO` , δεν υπάρχει πεδίο `dateVaccinated` στην εγγραφή.

Ξεκινώντας, η εφαρμογή `travelMonitor` θα πρέπει να δημιουργεί μια σειρά από named pipes για επικοινωνία με `numMonitors` child processes που θα δημιουργήσει. Στη συνέχεια, θα πρέπει να κάνει fork `numMonitors` child processes. Κάθε child process, θα καλεί την exec με εκτελέσιμο αρχείο ένα πρόγραμμα που ονομάζεται `Monitor` που θα γράψετε και που θα παίρνει ως ορίσματα τα μονοπάτια των named pipes που θα χρησιμοποιούνται για επικοινωνία ανάμεσα στο αρχικό parent process `travelMonitor` και κάθε exec'd Monitor process. Ύστερα, το parent process θα ενημερώνει το κάθε Monitor μέσω named pipe για τα subdirectories που θα αναλάβει το Monitor. Τα `travelMonitor` θα μοιράζει ομοιόμορφα (με round-robin alphabetically) τα subdirectories με τις χώρες που βρίσκονται στο input_dir στα Monitor processes. Μπορείτε να υποθέσετε πως τα subdirectories θα είναι flat, δηλαδή, θα περιέχουν μόνο αρχεία, όχι υποκαταλόγους.

Όταν η εφαρμογή (το parent process) τελειώσει τις ενέργειες αρχικοποίησης, θα περιμένει μια σειρά από bloom filters από τα Monitor processes (δείτε παρακάτω) και όταν λάβει όλες τις πληροφορίες, θα είναι έτοιμη να δεχθεί είσοδο (εντολές) από τον χρήστη από το πληκτρολόγιο (πιο κάτω για τις εντολές).

Κάθε Monitor process, για κάθε κατάλογο που του έχει ανατεθεί, θα διαβάζει όλα του τα αρχεία και θα γεμίζει μια σειρά από δομές δεδομένων που θα χρησιμοποιεί για να απαντάει σε ερωτήματα που του προωθεί το parent process. Μια δομή θα είναι το bloom filter που, όπως και στην πρώτη εργασία, θα χρησιμοποιείται για γρήγορο έλεγχο για το αν έχει εμβολιαστεί ένας πολίτης (με αναγνωριστικό `citizenID`) για τη συγκεκριμένη ίωση. Το κάθε Monitor process, αφού έχει τελειώσει την ανάγνωση των input files, θα στέλνει μέσω named pipe στο parent process ένα bloom filter για κάθε ίωση που θα αναπαριστά το σύνολο των εμβολιασμένων πολιτών των χωρών που διαχειρίζεται το Monitor process. Η επιλογή του αριθμού των named pipes ανάμεσα στο parent process και τα Monitor processes, καθώς επίσης και άλλων δομών δεδομένων είναι δική σας σχεδιαστική επιλογή. Αν κατά τη διάρκεια ανάγνωσης αρχείων, ένα Monitor process εντοπίσει προβληματική εγγραφή (π.χ. στη σύνταξη ή μια inconsistent εγγραφή), θα αγνοεί την προβληματική εγγραφή.
Όταν το Monitor process τελειώσει την ανάγνωση των αρχείων του και έχει στείλει όλα τα Bloom filters στο parent, ειδοποιεί το parent process μέσω named pipe πως είναι έτοιμο το Monitor να δεχτεί αιτήματα.

Αν ένα Monitor process λάβει ένα signal SIGINT ή SIGQUIT τότε τυπώνει σε ένα αρχείο με ονομασία log_file.xxx (όπου το xxx είναι το process ID του) το όνομα των χωρών (των subdirectories) που διαχειρίζεται, το συνολικό αριθμό αιτημάτων που δέχθηκε για είσοδο στις χώρες που διαχειρίζεται, και το συνολικό αριθμό αιτημάτων που εγκρίθηκαν και απορρίφθηκαν.

Logfile format:

```
Italy
China
Germany
TOTAL TRAVEL REQUESTS 29150
ACCEPTED 25663
REJECTED 3487
```

Αν ένα Monitor process λάβει ένα SIGUSR1 σήμα, αυτό σημαίνει πως έχουν τοποθετηθεί 1 ή περισσότερα νέα αρχεία σε κάποια από τα subdirectories που του έχουν ανατεθεί. Υποθέτουμε πως δεν δημιουργούνται καινούργια directories με χώρες και πως τα υπάρχοντα αρχεία δεν τροποποιούνται. Θα υπάρχουν δηλαδή μόνο καινούργια αρχεία. Το monitor process, θα ελέγχει τα subdirectories για να βρει τα νέα αρχεία, θα τα διαβάζει και θα ενημερώνει τις δομές δεδομένων που κρατάει στη μνήμη. Αφού ολοκληρώσει την ανάγνωση των νέων αρχείων, θα στέλνει τα ενημερωμένα bloom filters στο parent process.

Αν ένα Monitor process τερματίσει ξαφνικά, θα πρέπει το parent process να κάνει fork νέο Monitor process που θα το αντικαταστήσει. Ως εκ τούτου, το parent process θα πρέπει να χειρίζεται το SIGCHLD σήμα, όπως επίσης και το SIGINT και SIGQUIT.

Αν το parent process λάβει SIGINT ή SIGQUIT, θα πρέπει πρώτα να τελειώσει την επεξεργασία της τρέχουσας εντολής από το χρήστη και αφού έχει απαντήσει στο χρήστη, θα στέλνει ένα SIGKILL σήμα στους Monitors, θα τους περιμένει να τερματίσουν, και στο τέλος θα τυπώνει σε ένα αρχείο με ονομασία log_file.xxx όπου το xxx είναι το process ID του, το όνομα όλων των χωρών (των subdirectories) που συμμετείχαν στην εφαρμογή με δεδομένα, το συνολικό αριθμό αιτημάτων που δέχθηκε για είσοδο στις χώρες, και το συνολικό αριθμό αιτημάτων που εγκρίθηκαν και απορρίφθηκαν.

Logfile format:

```
Italy
China
Germany
TOTAL TRAVEL REQUESTS 883832
ACCEPTED 881818
REJECTED 2014
```

Ο χρήστης θα μπορεί να δίνει τις ακόλουθες εντολές στην εφαρμογή:

- `/travelRequest citizenID date countryFrom countryTo virusName`

  Η εφαρμογή πρώτα θα ελέγχει το bloom filter που του έχει στείλει το Monitor process που διαχειρίζεται τη χώρα `countryFrom`. Αν το bloom filter υποδεικνύει πως ο πολίτης `citizenID` δεν έχει εμβολιαστεί κατά του `virusName` τυπώνει

  ```
  REQUEST REJECTED – YOU ARE NOT VACCINATED
  ```

  Αν το bloom filter υποδεικνύει πως ο πολίτης `citizenID` ίσως έχει εμβολιαστεί κατά του `virusName`, η εφαρμογη ζητάει μέσω named pipe από το Monitor process που διαχειρίζεται τη χώρα `countryFrom` αν όντως έχει εμβολιαστεί ο `citizenID`. Το Monitor process απαντάει μέσω named pipe YES/NO όπου στη περίπτωση του YES, στέλνει και την ημερομηνία εμβολιασμού. Για την απάντηση το Monitor process μπορεί

(αλλά δεν είναι απαραίτητο) να χρησιμοποιήσει τις δομές που είχατε στην 1η εργασία (π.χ. τη skip list). Η εφαρμογή ελέγχει αν έχει εμβολιαστεί ο πολίτης λιγότερο από 6 μήνες πριν την επιθυμητή ημερομηνία ταξιδιού `date` και τυπώνει ένα από τα ακόλουθα μηνύματα

```
REQUEST REJECTED – YOU ARE NOT VACCINATED
REQUEST REJECTED – YOU WILL NEED ANOTHER VACCINATION BEFORE TRAVEL DATE
REQUEST ACCEPTED – HAPPY TRAVELS
```

- `/travelStats virusName date1 date2 [country]`

Αν δεν δοθεί `country` όρισμα, η εφαρμογή θα τυπώνει τον αριθμό πολιτών που έχουν ζητήσει έγκριση να ταξιδέψουν μεσα στο διαστημα `[date1...date2]` σε χώρες που ελέγχουν για εμβολιασμό κατά του `virusName`, και τον αριθμο πολιτών που εγκρίθηκαν και που απορρίφθηκαν. Αν δοθεί `country` όρισμα, η εφαρμογή θα τυπώνει την ίδια πληροφορία αλλά μόνο για τη συγκεκριμένη χώρα `country`. Τα `date1 date2` ορίσματα θα έχουν μορφή DD-MM-YYYY.

Output format: Παράδειγμα:

```
TOTAL REQUESTS 29150
ACCEPTED 25663
REJECTED 3487
```

- `/addVaccinationRecords country`

Με αυτό το αίτημα ο χρήστης έχει τοποθετήσει στο `input_dir/country` ένα ή περισσότερα αρχεία για επεξεργασία από το Monitor process. Το parent process στέλνει ειδοποίηση μέσω SIGUSR1 σήμα στο Monitor process που διαχειρίζεται τη χώρα `country` ότι υπάρχουν input files για ανάγνωση στον κατάλογο. Το Monitor process διαβάζει ότι νέο αρχείο βρει, ενημερώνει τις δομές δεδομένων και στέλνει πίσω στο parent process, μέσω named pipe, τα ενημερωμένα του bloom filters που αναπαριστούν το σύνολο πολιτών που έχουν εμβολιαστεί

- `/searchVaccinationStatus citizenID`

Το parent process προωθεί σε όλους τα Monitor processes το αίτημα μέσω named pipes. Το Monitor process που διαχειρίζεται τον πολίτη με αναγνωριστικό `citizenID` στέλνει μέσω named pipe ό,τι πληροφορίες έχει για τους εμβολιασμούς που έχει κάνει/δεν έχει κάνει ο συγκεκριμένος πολίτης. Όταν λάβει τις πληροφορίες, το parent τις τυπώνει στο stdout.

Output format: Παράδειγμα:

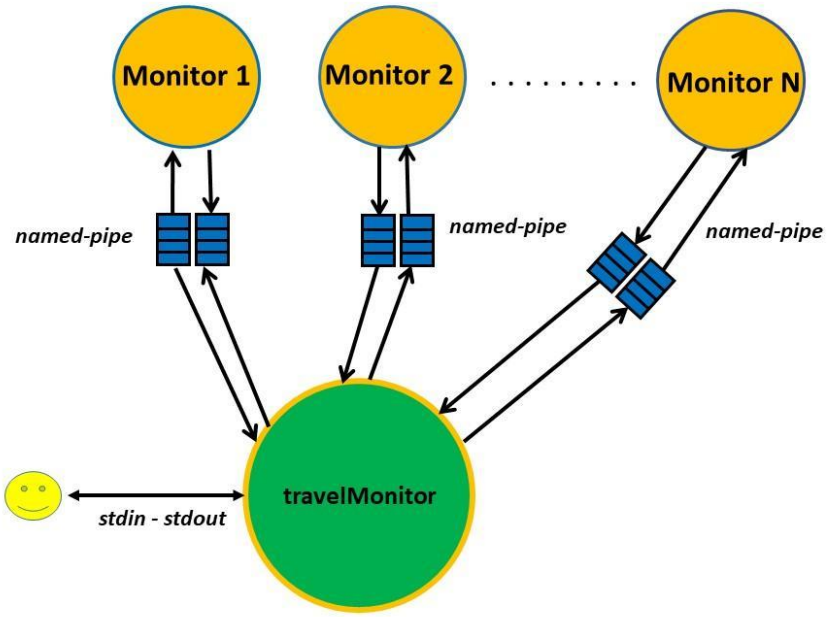
```
889 JOHN PAPADOPOULOS GREECE
AGE 52
COVID-19 VACCINATED ON 27-12-2020
H1N1 NOT YET VACCINATED
```

- `/exit`

Έξοδος από την εφαρμογή. Το parent process στέλνει ένα SIGKILL σήμα στους Monitors, τους περιμένει να τερματίσουν, και τυπώνει σε ένα αρχείο με ονομασία log_file.xxx όπου το xxx είναι το process ID του, το όνομα όλων των χωρών (των subdirectories) που συμμετείχαν στην εφαρμογή με δεδομένα, το συνολικό αριθμό αιτημάτων που δέχθηκε για είσοδο στις χώρες, και το συνολικό αριθμό αιτημάτων που εγκρίθηκαν και απορρίφθηκαν. Πριν τερματίσει, θα απελευθερώνει σωστά όλη τη δεσμευμένη μνήμη.

Log file format (όπως και πιο πάνω):

```
Italy
China
Germany
TOTAL TRAVEL REQUESTS 883832
ACCEPTED 881818
REJECTED 2014
```



Η επικοινωνία ανάμεσα στο parent process και κάθε Monitor λαμβάνει χώρα μέσω named pipes (δείτε εικόνα).

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 κλήσης 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)`. Είστε ελεύθεροι να διαλέξετε οποία μέθοδο λειτουργίας των σωληνώσεων θέλετε.

Συμβουλές: Το πιο δύσκολο κομμάτι της εργασίας είναι η διαχείριση των named pipes. Καλό θα ήταν καθώς σχεδιάζετε την εργασία να σκεφτείτε ζητήματα όπως:

1) Τι γίνεται όταν η παράμετρος `-b bufferSize` είναι αρκετά μικρότερη από τα δεδομένα που θέλει μια διεργασία να στείλει πάνω από ένα named pipe;
2) Τι γίνεται όταν ο Monitor λαμβάνει ένα SIGUSR1 σήμα ενώ βρίσκεται στη μέση αποστολής δεδομένων προς στο parent process;
3) Πώς θα σχεδιάσετε το parent process ώστε να μην μπλοκάρει περιμένοντας κάποιον αργό Monitor ενώ υπάρχει άλλος Monitor που έχει στείλει δεδομένα και περιμένουν να διαβαστούν στο named pipe; (Ίσως σας φανεί χρήσιμη η select() call.)
4) Πώς θα ενημερώνεται ένας Monitor ότι σκοπεύει το parent process να του μεταβιβάσει κάποια εντολή μέσω του named pipe έτσι ώστε ο δεύτερος να πάει να τη διαβάσει; (Αυτό, για παράδειγμα, θα μπορούσε να

επιτευχθεί με την αποστολή κάποιου προσυμφωνημένου σήματος (signal-software interrupt) ή μέσω χρήσης select() call ή μέσω χρήσης του named_pipe που θα σχεδιάσετε).

5) Πώς θα ξέρει ένα process πόσα bytes αποτελούν ένα μήνυμα από ένα άλλο process; Δηλαδή, πως θα ερμηνεύει τα bytes που του στέλνει το άλλο process; (Θα χρειαστεί να σχεδιάσετε ένα "πρωτόκολλο επικοινωνίας" ανάμεσα στα processes).

Όποιες σχεδιαστικές επιλογές κάνετε, θα πρέπει να τις περιγράψετε σε ένα README αρχείο που θα υποβάλλετε μαζί με τον κώδικά σας.

B) Το script create_infiles.sh (10%)

Θα γράψετε ένα bash script το οποίο δημιουργεί test subdirectories και input files που θα χρησιμοποιήσετε για debugging του προγράμματός σας. Φυσικά, κατά τη διάρκεια της ανάπτυξης του προγράμματός σας μπορείτε να χρησιμοποιήσετε λίγα και μικρά αρχεία για να κάνετε debug. Το script create_infiles.sh δουλεύει ως εξής:

```
./create_infiles.sh inputFile input_dir numFilesPerDirectory
```

- `inputFile`: ένα αρχείο το οποίο έχει το ίδιο format με το `citizenRecordsFile` input file της πρώτης εργασίας. Θυμίζουμε το sample παράδειγμα που είχαμε δώσει:

```
889 John Papadopoulos Greece 52 COVID-19 YES 27-12-2020
889 John Papadopoulos Greece 52 H1N1 NO
776 Maria Tortellini Italy 36 SARS-1 NO
125 Jon Dupont USA 76 H1N1 YES 30-10-2020
```

- `input_dir`: το όνομα ενός καταλόγου που θα τοποθετηθούν τα subdirectories και input files
- `numFilesPerDirectory`: ο αριθμός αρχείων που θα φτιαχτούν σε κάθε subdirectory

Το script κάνει τα εξής:

1. Κάνει ελέγχους για νούμερα εισόδου
2. Δημιουργεί κατάλογο με όνομα που δίνεται στο δεύτερο όρισμα `input_dir` . Αν ο κατάλογος υπάρχει τυπώνεται μήνυμα λάθους και τερματίζει.
3. Διαβάζει το αρχείο `inputFile`
4. Μέσα στον κατάλογο `input_dir` δημιουργεί subdirectories, ένα για κάθε όνομα χώρας που εντοπίζει μέσα στο `inputFile`
5. Σε κάθε subdirectory, δημιουργεί `numFilesPerDirectory` αρχεία με όνομα `country-n.txt` οπου το `n` είναι ένας θετικός ακέραιος. Καθώς επεξεργάζεται το `inputFile` θα εντοπίζει όλες τις γραμμές που αντιστοιχούν σε μια χώρα και θα κατανέμει τις γραμμές round robin στα αρχεία με όνομα `country-n.txt`

Παρατηρήσεις

- Η συγκεκριμένη εργασία απαιτεί αρκετή σκέψη και καλό σχεδιασμό όσον αφορά κατανεμημένους πόρους, forks, blocking/non-blocking I/O κλπ. Η άσκηση δεν περιγράφει όλες τις λεπτομέρειες και τις δομές για τον απλό λόγο πως οι σχεδιαστικές επιλογές είναι αποκλειστικά δικές σας (βεβαιωθείτε φυσικά πως τις περιγράφετε αναλυτικά στο README). Αν έχετε διάφορες επιλογές για κάποιο σημείο της άσκησης σκεφτείτε τα υπέρ και τα κατά, τεκμηριώστε τα στο README, επιλέξτε αυτό που θεωρείτε σωστό και λογικό και περιγράψτε γιατί το επιλέξατε στο README.

Παραδοτέα

- Μια σύντομη και περιεκτική εξήγηση για τις επιλογές που έχετε κάνει στο σχεδιασμό του προγράμματός σας. 1-2 σελίδες ASCII κειμένου είναι αρκετές. Συμπεριλάβετε την εξήγηση και τις οδηγίες για το compilation και την εκτέλεση του προγράμματός σας σε ένα αρχείο README μαζί με τον κώδικα που θα υποβάλετε.
- Ο κώδικας που θα υποβάλετε θα πρέπει να είναι δικός σας. Απαγορεύεται η χρήση κώδικα που δεν έχει γραφεί από εσάς (**αυτό συμπεριλαμβάνει και κώδικα από το Διαδίκτυο!)** .
- Όλη η δουλειά σας (πηγαίος κώδικας, Makefile και README) σε ένα tar.gz file με ονομασία OnomaEponymoProject2.tar.gz. Προσοχή να υποβάλετε μόνο κώδικα, Makefile, README και όχι τα binaries.
- Καλό θα είναι να έχετε ένα backup .tar της άσκησής σας όπως ακριβώς αυτή υποβλήθηκε σε κάποιο εύκολα προσπελάσιμο μηχάνημα (server του τμήματος, private github repository, private cloud).
- Η σωστή υποβολή ενός σωστού tar.gz που περιέχει τον κώδικα της άσκησής σας και ό,τι αρχεία χρειάζονται είναι αποκλειστικά ευθύνη σας. **Άδεια tar/tar.gz ή tar/tar.gz που έχουν λάθος και δεν γίνονται extract δεν βαθμολογούνται**.

Διαδικαστικά

- Για επιπρόσθετες ανακοινώσεις, παρακολουθείτε το forum του μαθήματος στο piazza.com. Η πλήρης διεύθυνση είναι https://piazza.com/uoa.gr/spring2021/k24/home. Η παρακολούθηση του φόρουμ στο Piazza είναι υποχρεωτική.
- Το πρόγραμμά σας θα πρέπει να γραφεί σε C (ή C++). Στην περίπτωσή που χρησιμοποιήσετε C++ δεν μπορείτε να χρησιμοποιήσετε τις έτοιμες δομές της Standard Template Library (STL). Σε κάθε περίπτωση το πρόγραμμά σας θα πρέπει να τρέχει στα Linux workstations του Τμήματος.
- Το πρόγραμμά σας θα πρέπει να κάνει compile στο εκτελέσιμο (`travelMonitor`) και (`Monitor`) να έχει τα ίδια ονόματα για τις παραμέτρους (`-m, -b, -s, -i`) όπως ακριβώς περιγράφεται στην εκφώνηση.
- 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.
- Η υποβολή θα γίνει μέσω eclass.

Άλλες σημαντικές παρατηρήσεις

- Οι εργασίες είναι ατομικές.
- Όποιος υποβάλει / παρουσιάσει κώδικα που δεν έχει γραφτεί από την ίδια/τον ίδιο μηδενίζεται στο μάθημα.
- 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.)
- Οι ασκήσεις προγραμματισμού μπορούν να δοθούν με καθυστέρηση το πολύ 3 ημερών και με ποινή 5% για κάθε μέρα αργοπορίας. Πέραν των 3 αυτών ημερών, δεν μπορούν να κατατεθούν ασκήσεις.