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 επεξεργασίας πληροφοριών που θα δέχεται, θα επεξεργάζεται, θα καταγράφει και θα απαντάει ερωτήματα για κρούσματα ιώσεων. Συγκεκριμένα, θα υλοποιήσετε την εφαρμογή diseaseAggregator η οποία θα δημιουργεί μια σειρά από Worker διεργασίες που, μαζί με την εφαρμογή, θα απαντούν σε ερωτήματα του χρήστη.

Α) Η εφαρμογή diseaseAggregator (75%)

Η εφαρμογή `diseaseAggregator` θα χρησιμοποιείται ως εξής:

```
./diseaseAggregator –w numWorkers –b bufferSize –i input_dir
```

όπου:

- Η παράμετρος `numWorkers` είναι ο αριθμός Worker διεργασιών που θα δημιουργήσει η εφαρμογή.
- Η παράμετρος `bufferSize:` είναι το μέγεθος του buffer για διάβασμα πάνω από τα pipes.
- Η παράμετρος `input_dir:` είναι ένα directory το οποίο περιέχει subdirectories με τα αρχεία που θα επεξεργάζονται οι Workers. Κάθε subdirectory θα έχει το όνομα μιας χώρας και θα περιέχει αρχεία με ονόματα που είναι ημερομηνίες της μορφής DD-MM-YYYY. Για παράδειγμα το `input_dir` θα μπορούσε να περιέχει subdirectories `China/` `Italy/` και `Germany/` τα οποία έχουν τα εξής αρχεία:
-

```
/input_dir/China/21-12-2019
/input_dir/China/22-12-2019
/input_dir/China/23-12-2019
/input_dir/China/31-12-2019
…
/input_dir/Italy/31-01-2020
/input_dir/Italy/01-02-2020
…
/input_dir/Germany/31-01-2020
/input_dir/Germany/20-2-2020
/input_dir/Germany/02-3-2020
…
```

- Κάθε αρχείο DD-MM-YYYY περιέχει μια σειρά από εγγραφές ασθενών όπου κάθε γραμμή θα περιγράφει έναν ασθενή που εισήχθη/πήρε εξιτήριο σε/από νοσοκομείο εκείνη την ημέρα και περιέχει το recordID, το όνομα του, την ίωση, και την ηλικία του. Για παράδειγμα αν τα περιεχόμενα στο αρχείο /input_dir/Germany/02-03-2020 του αρχείου είναι:

```
889 ENTER Mary Smith COVID-2019 23
776 ENTER Larry Jones SARS-1 87
125 EXIT Jon Dupont H1N1 62
```

σημαίνει πως στη Γερμανία, στις 02-03-2020 μπήκαν στο νοσοκομείο δύο ασθενείς (Mary Smith και Larry Jones) με κρούσματα COVID-2019 και SARS-1 ενώ ο Jon Dupont με H1N1 πήρε εξιτήριο από νοσοκομείο. Για μια εγγραφή EXIT θα πρέπει ήδη να υπάρχει στο σύστημα μια εγγραφή ENTER με το ίδιο recordID και

προγενέστερη ημερομηνία εισαγωγής. Σε διαφορετική περίπτωση τυπώνεται η λέξη ERROR σε μία κενή γραμμή (δείτε και πιο κάτω την περιγραφή των Workers).

Συγκεκριμένα, μια εγγραφή είναι μια γραμμή ASCII κειμένου που αποτελείται από τα εξής στοιχεία:

1. `recordID`: μια συμβολοσειρά (μπορεί να έχει και ένα μόνο ψηφίο ή γράμμα) που με μοναδικό τρόπο καθορίζει την κάθε τέτοια εγγραφή.
2. Η συμβολοσειρά `ENTER` ή `EXIT`: υποδεικνύει εισαγωγή στο ή εξιτήριο από νοσοκομείο.
3. `patientFirstName`: μια συμβολοσειρά που αποτελείται από γράμματα χωρίς κενά.
4. `patientLastName`: μια συμβολοσειρά που αποτελείται από γράμματα χωρίς κενά.
5. `disease`: μια συμβολοσειρά που αποτελείται από γράμματα, αριθμούς, και ενδεχομένως και μια παύλα "-" αλλά χωρίς κενά.
6. `age`: ένας θετικός (>0), ακέραιος <= 120.

Ξεκινώντας, η εφαρμογή diseaseAggregator θα πρέπει να ξεκινάει `numWorkers` Workers child processes και να μοιράζει ομοιόμορφα τα subdirectories με τις χώρες που βρίσκονται στο `input_dir` στους Workers. Θα ξεκινάει τους Workers και θα πρέπει να ενημερώνει τον κάθε Worker μέσω named pipe για τα subdirectories που θα αναλάβει ο Worker. Μπορείτε να υποθέσετε πως τα subdirectories θα είναι flat, δηλαδή, θα περιέχουν μόνο αρχεία, όχι υποκαταλόγους.

Όταν η εφαρμογή (το parent process) τελειώσει τη δημιουργία των Worker processes, θα περιμένει είσοδο (εντολές) από τον χρήστη από το πληκτρολόγιο (δείτε πιο κάτω τις εντολές).

Κάθε Worker process, για κάθε κατάλογο που του έχει ανατεθεί, θα διαβάζει όλα του τα αρχεία με χρονολογική σειρά βάσει των ονομάτων των αρχείων και θα γεμίζει μια σειρά από δομές δεδομένων που θα χρησιμοποιεί για να απαντάει σε ερωτήματα που του προωθεί το parent process. Η επιλογή του αριθμού των named pipes καθώς επίσης και των δομών δεδομένων είναι δική σας σχεδιαστική επιλογή. (Ίσως σας φανούν χρήσιμες κάποιες δομές από την πρώτη σας εργασία). Αν κατά τη διάρκεια ανάγνωσης αρχείων, ο Worker εντοπίσει προβληματική εγγραφή (π.χ. στη σύνταξη ή μια EXIT εγγραφή που δεν υπάρχει προγενέστερη ENTER εγγραφή, κλπ) τότε θα τυπώνει ERROR σε μια νέα γραμμή. Όταν τελειώσει την ανάγνωση ενός αρχείου, ο Worker θα στέλνει, μέσω named pipe, στο parent process, summary statistics του αρχείου που περιέχουν τις ακόλουθες πληροφορίες: για κάθε ίωση, στη συγκεκριμένη χώρα, θα στέλνει τον αριθμό κρουσμάτων ανά ηλικιακή κατηγορία. Για παράδειγμα, για ένα αρχείο /input_dir/China/22-12-2020 που περιέχει εγγραφές για SARS και COVID-19 θα στέλνει summary statistics όπως:

```
22-12-2020
China
SARS
Age range 0-20 years: 10 cases
Age range 21-40 years: 23 cases
Age range 41-60 years: 34 cases
Age range 60+ years: 45 cases

COVID-19
Age range 0-20 years: 20 cases
Age range 21-40 years: 230 cases
Age range 41-60 years: 340 cases
Age range 60+ years: 450 cases
```

Αυτά τα statistics θα πρέπει να σταλούν για κάθε αρχείο εισόδου (δηλαδή για κάθε αρχείο ημερομηνίας) από κάποιον Worker. Όταν ο Worker τελειώσει την ανάγνωση των αρχείων του και έχει στείλει όλα τα summary statistics, ειδοποιεί το parent process μέσω named pipe πως είναι έτοιμος ο Worker να δεχτεί αιτήματα.

Αν ένας Worker 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 από το parent process και το συνολικό αριθμό αιτημάτων που απάντησε με επιτυχία (δηλαδή δεν υπήρξε κάποιο error στο ερώτημα).

Logfile format:

```
Italy
China
Germany
TOTAL 29150
SUCCESS 25663
FAIL 3487
```

Αν ένας Worker λάβει ένα SIGUSR1 σήμα, αυτό σημαίνει πως έχουν τοποθετηθεί 1 ή περισσότερα νέα αρχεία σε κάποια από τα subdirectories που του έχουν ανατεθεί. Θα ελέγχει τα subdirectories για να βρει τα νέα αρχεία, θα τα διαβάζει και θα ενημερώνει τις δομές δεδομένων που κρατάει στη μνήμη. Για κάθε νέο αρχείο, θα στέλνει summary statistcs στο parent process.

Αν ένας Worker process τερματίσει ξαφνικά, θα πρέπει το parent process να κάνει fork νέο Worker 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 Workers, 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, τον συνολικό αριθμό αιτημάτων που δέχθηκε από τον χρήστη και απάντησε με επιτυχία και τον συνολικό αριθμό αιτημάτων όπου προέκυψε κάποιο σφάλμα ή/και δεν ολοκληρώθηκαν.

Logfile format:

```
Italy
China
Germany
TOTAL 29150
SUCCESS 25663
FAIL 3487
```

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

- `/listCountries`
  Η εφαρμογή θα τυπώνει κάθε χώρα μαζί με το process ID του Worker process που διαχειρίζεται τα αρχεία της. Αυτή η εντολή χρησιμεύει όταν ο χρήστης θέλει να προσθέσει νέα αρχεία μιας χώρας για επεξεργασία και πρέπει να μάθει ποιο είναι το Worker process για να του στείλει ειδοποίηση μεσω SIGUSR1 σήματος.

Output format: μια γραμμή για κάθε χώρα και process ID. Παράδειγμα:

```
Canada 123
Italy 5769
China 5770
```

- `/diseaseFrequency virusName date1 date2 [country]`

Αν δεν δοθεί `country` όρισμα, η εφαρμογή θα τυπώνει για την ασθένεια `virusName` τον αριθμό κρουσμάτων που έχουν καταγραφεί στο σύστημα μέσα στο διάστημα `[date1...date2]`. Αν δοθεί `country` όρισμα, η εφαρμογή θα τυπώνει για την ασθένεια `virusName`, τον αριθμό κρουσμάτων στην χώρα `country` που έχουν καταγραφεί μέσα στο διάστημα `[date1...date2]`. Τα `date1 date2` ορίσματα θα έχουν μορφή DD-MM-YYYY.

Output format: μια γραμμή με τον αριθμό κρουσμάτων. Παράδειγμα:
153

- `/topk-AgeRanges k country disease date1 date2`

Η εφαρμογή θα τυπώνει, για την χώρα `country` και την ίωση `disease` τις top k ηλικιακές κατηγορίες που έχουν εμφανίσει κρούσματα της συγκεκριμένης ίωσης στη συγκεκριμένη χώρα και το ποσοστό κρουσμάτων τους. Τα `date1 date2` ορίσματα θα έχουν μορφή DD-MM-YYYY.

Output format: μια γραμμή για κάθε ηλικιακή κατηγορία. Παράδειγμα με k=2:
20-30: 40%
60+: 30%

- `/searchPatientRecord recordID`

Το parent process προωθεί σε όλους τους Workers το αίτημα μέσω named pipes και περιμένει απάντηση από το Worker με το record `recordID`. Όταν το λάβει, το τυπώνει.

Παράδειγμα για κάποιον που μπήκε στο νοσοκομείο 23-02-2020 και βγήκε 28-02-2020:

```
776 Larry Jones SARS-1 87 23-2-2020 28-2-2020
```

Παράδειγμα για κάποιον που μπήκε στο νοσοκομείο 23-02-2020 και δεν έχει βγει ακόμα:

```
776 Larry Jones SARS-1 87 23-2-2020 --
```

- `/numPatientAdmissions disease date1 date2 [country]`

Αν δοθεί `country` όρισμα, η εφαρμογή θα τυπώσει, σε μια νέα γραμμή, το συνολικό αριθμό ασθενών που μπήκαν στο νοσοκομείο με την ασθένεια `disease` στη συγκεκριμένη χώρα μέσα στο διάστημα [`date1 date2`]. Αν δε δοθεί `country` όρισμα, η εφαρμογή θα τυπώσει, για κάθε χώρα, τον αριθμό ασθενών με την ασθένεια `disease` που μπήκαν στο νοσοκομείο στο διάστημα [`date1 date2`]. Τα `date1 date2` ορίσματα θα έχουν μορφή DD-MM-YYYY.

Output format:

```
Italy 153
Spain 769
France 570
```

- `/numPatientDischarges disease date1 date2 [country]`

Αν δοθεί το όρισμα `country`, η εφαρμογή θα τυπώσει σε μια νέα γραμμή, τον συνολικό αριθμό ασθενών με την ασθένεια `disease` που έχουν βγει από το νοσοκομείο στη συγκεκριμένη χώρα μέσα στο διάστημα [`date1 date2`]. Αν δεν δοθεί όρισμα `country`, η εφαρμογή θα τυπώσει, για κάθε χώρα, τον αριθμό ασθενών με την ασθένεια `disease` που έχουν βγει από το νοσοκομείο στο διάστημα [`date1 date2`]. Τα `date1 date2` ορίσματα θα έχουν μορφή DD-MM-YYYY.

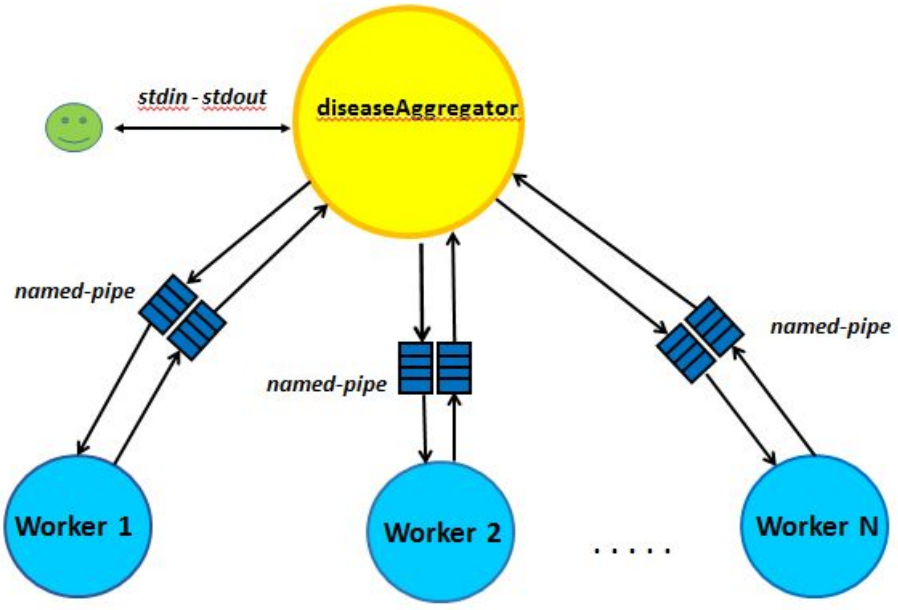
Παράδειγμα:

```
Italy 153
Spain 769
France 570
```

- `/exit`

Έξοδος από την εφαρμογή. Το parent process στέλνει ένα SIGKILL σήμα στους Workers, τους περιμένει να τερματίσουν, και τυπώνει σε ένα αρχείο με ονομασία log_file.xxx όπου το xxx είναι το process ID του, το όνομα των χωρών (των subdirectories) που “συμμετείχαν” στην εφαρμογή, το συνολικό αριθμό αιτημάτων που δέχθηκε από τον χρήστη και απάντησε με επιτυχία και το συνολικό αριθμό αιτημάτων όπου προέκυψε κάποιο σφάλμα ή/και δεν ολοκληρώθηκαν. Πριν τερματίσει, θα απελευθερώνει σωστά όλη τη δεσμευμένη μνήμη.
Παράδειγμα:

```
Italy
China
Germany
TOTAL 29150
SUCCESS 25663
FAIL 3487
```



Η επικοινωνία ανάμεσα στο parent process και κάθε Worker 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 Worker 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 Worker ενώ υπάρχει άλλος Worker που έχει στείλει δεδομένα και περιμένουν να διαβαστούν στο named pipe; (Ίσως σας φανεί χρήσιμη η select() call.)
4) Πώς θα ενημερώνεται ο Worker 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) To script create_infiles.sh (25%)

Θα γράψετε ένα bash script το οποίο δημιουργεί test subdirectories και input files που θα χρησιμοποιήσετε για να κάνετε 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 diseasesFile countriesFile input_dir numFilesPerDirectory numRecordsPerFile
```

- `diseaseFile`: ένα αρχείο με ονόματα ιώσεων (ένα ανά γραμμή)
- `countriesFile`: ένα αρχείο με ονόματα χωρών (ένα ανά γραμμή)
- `input_dir`: το όνομα ενός κατάλογου όπου θα τοποθετηθούν τα subdirectories και input files

Το script κάνει τα εξής:

1. Κάνει ελέγχους για νούμερα εισόδου
2. Διαβάζει τα αρχεία `diseasesFile` και `countriesFile`
3. Δημιουργεί κατάλογο με όνομα που δίνεται στο τρίτο όρισμα `input_dir`
4. Μέσα στον κατάλογο δημιουργεί subdirectories, ένα για κάθε όνομα χώρας μέσα στο `countriesFile`
5. Σε κάθε subdirectory, δημιουργεί `numFilesPerDirectory` αρχεία με όνομα μιας ημερομηνίας DD-MM-YYYY. Σε κάθε αρχείο, τοποθετεί `numRecordsPerFile` ακολουθώντας τη μορφή των input files που περιγράφεται πιο πάνω. Για diseases, θα επιλέγει τυχαία το script κάποιο από

το `diseasesFile`. Για το όνομα και επώνυμο μπορείτε να δημιουργήσετε τυχαίες συμβολοσειρές με τυχαίο μήκος από 3 έως 12 χαρακτήρες.

Simplifying info/hints:
α) Μπορείτε να υποθέσετε πως όλοι οι μήνες έχουν 30 μέρες.
β) Ίσως σας φανεί χρήσιμη η $RANDOM Bash function. (Δείτε, π.χ. http://tldp.org/LDP/abs/html/randomvar.html)

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.
- 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, github, 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/spring2020/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.
- 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.

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.
- 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.