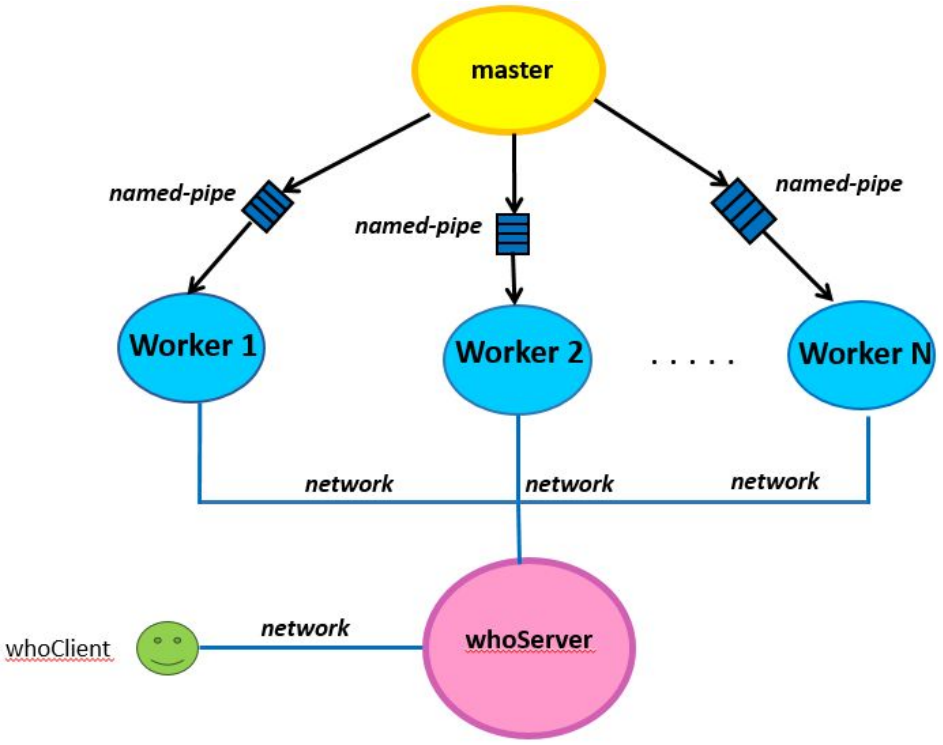
Εισαγωγή

Στόχος αυτής της εργασίας είναι να εξοικειωθείτε με τον προγραμματισμό νημάτων και τη δικτυακή επικοινωνία. Στα πλαίσια αυτής της εργασίας θα υλοποιήσετε ένα κατανεμημένο σύστημα διεργασιών που θα παρέχει την ίδια λειτουργικότητα με την εφαρμογή diseaseAggregator της δεύτερης εργασίας. Συγκεκριμένα, θα υλοποιήσετε τρία προγράμματα: 1) ένα master πρόγραμμα που θα δημιουργεί μια σειρά από Worker διεργασίες (όπως έκανε το parent process στη δεύτερη εργασία), 2) έναν multi-threaded whoServer που θα συλλέγει μέσω δικτύου summary statistics από τα Worker processes και ερωτήματα από πελάτες, και 3) ένα multithreaded client πρόγραμμα whoClient που θα δημιουργεί πολλά νήματα, όπου το κάθε νήμα παίζει το ρόλο ενός πελάτη που θα στέλνει ερωτήματα στον whoServer.



A) Το πρόγραμμα master (10%)

Το πρόγραμμα `master` θα χρησιμοποιείται ως εξής:

```
./master -w numWorkers -b bufferSize -s serverIP  -p serverPort -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 `serverIP:` είναι η IP διεύθυνση του whoServer στον οποίο θα συνδεθούν οι Worker διεργασίες για να του στείλουν summary statistics.
- Η παράμετρος `serverPort:` είναι ο αριθμός θύρας όπου ακούει ο whoServer.
- Η παράμετρος `input_dir:` είναι ένα directory το οποίο περιέχει subdirectories με τα αρχεία που θα επεξεργάζονται οι Workers. Όπως και στη δεύτερη εργασία, κάθε subdirectory θα έχει το όνομα μιας χώρας και θα περιέχει αρχεία με ονόματα που είναι ημερομηνίες της μορφής DD-MM-YYYY. Κάθε αρχείο DD-MM-YYYY θα έχει ακριβώς το ίδιο format που είχε στη δεύτερη εργασία και θα 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.

Ξεκινώντας, το πρόγραμμα master θα πρέπει να ξεκινάει `numWorkers` Workers child processes και να μοιράζει ομοιόμορφα τα subdirectories με τις χώρες που βρίσκονται στο `input_dir` στους Workers (όπως στη 2η εργασία). 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. Το parent process θα στέλνει μέσω named pipe και την IP address και το port number του whoServer. Όταν τελειώσει τη δημιουργία των Worker processes, το parent process θα παραμείνει (δηλαδή δε θα τερματίζει) για να κάνει fork νέο Worker process σε περίπτωση που τερματίσει ξαφνικά κάποιο υπάρχον Worker.

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 θα του προωθεί ο whoServer. Θα συνδέεται στο whoServer και θα του στέλνει τις ακόλουθες πληροφορίες: 1) ένα port number όπου θα ακούει ο Worker process για ερωτήματα που θα του προωθεί ο whoServer, και 2) τα summary statistics (ίδια με της δεύτερης εργασίας). Όταν το Worker process τελειώσει τη μεταφορά πληροφοριών στον whoServer, θα ακούει στο port number που έχει επιλέξει και θα περιμένει συνδέσεις από τον whoServer για αιτήματα που αφορούν τις χώρες που διαχειρίζεται. Το signal handling παραμένει ίδιο με τη δεύτερη εργασία.

Σημείωση: θα πρέπει να σκεφτείτε κάποιον τρόπο για να ανατίθεται ένα port number σε κάθε Worker process. Ο πιο απλός τρόπος είναι να χρησιμοποιήσετε το port 0 ως παράμετρο στις αντίστοιχες συναρτήσεις και να βρείτε το port που ανατέθηκε επίσης χρησιμοποιώντας τις αντίστοιχες συναρτήσεις.

Β) Το πρόγραμμα whoServer (60%)

Το πρόγραμμα `whoServer` θα χρησιμοποιείται ως εξής:

```
./whoServer –q queryPortNum -s statisticsPortNum –w numThreads –b bufferSize
```

όπου:

- Η παράμετρος `queryPortNum` είναι ένας αριθμός θύρας όπου θα ακούει ο whoServer για συνδέσεις με ερωτήματα από τον πελάτη whoClient
- Η παράμετρος `statisticsPortNum` είναι ένας αριθμός θύρας όπου θα ακούει ο whoServer για συνδέσεις με summary statistics από Worker processes
- Η παράμετρος `numThreads:` είναι ο αριθμός των νημάτων που θα δημιουργήσει ο whoServer για να εξυπηρετούν εισερχόμενες συνδέσεις από το δίκτυο. Τα threads θα πρέπει δημιουργηθούν μια φορά στην αρχή όταν θα ξεκινάει ο whoServer.
- Η παράμετρος `bufferSize` είναι το μέγεθος ενός **κυκλικού** buffer που θα μοιράζεται ανάμεσα στα νήματα που δημιουργούνται από το whoServer process. Το `bufferSize` αναπαριστά τον αριθμό των file/socket descriptors που μπορούν να αποθηκευτούν σε αυτόν (π.χ. 10, σημαίνει 10 descriptors).

  Στην υλοποίηση σας, όταν ξεκινά ο whoServer θα πρέπει το αρχικό νήμα να δημιουργεί `numThreads` νήματα. Το αρχικό (main process) thread θα ακούει στα ports `queryPortNum` και `statisticsPortNum`, θα δέχεται συνδέσεις με την accept() κλήση συστήματος και θα τοποθετεί τους περιγραφείς αρχείων (file/socket descriptors) που αντιστοιχούν στις συνδέσεις σε έναν κυκλικό buffer μεγέθους που ορίζεται από το `bufferSize`. Το αρχικό νήμα ΔΕΝ θα διαβάζει από τις συνδέσεις που δέχεται. Απλώς, όποτε δέχεται κάποια σύνδεση θα τοποθετεί τον περιγραφέα αρχείου που επιστρέφει η accept() στον buffer και θα συνεχίζει να δέχεται επόμενες συνδέσεις. Η δουλειά των `numThreads` νημάτων είναι να εξυπηρετούν τις συνδέσεις των οποίων οι αντίστοιχοι περιγραφείς αρχείων έχουν τοποθετηθεί στον buffer. Κάθε ένα από τα `numThreads` νήματα ξυπνάει όταν υπάρχει τουλάχιστον ένας περιγραφέας στον buffer.

  Πιο συγκεκριμένα, το αρχικό thread θα ακούει στο `statisticsPortNum` για συνδέσεις από Worker processes για να λάβει τα summary statistics και το port number όπου ακούει το κάθε Worker process, και θα ακούει στο `queryPortNum` για συνδέσεις από τον whoClient για να λάβει ερωτήματα για κρούσματα που έχουν καταγραφεί στο κατανεμημένο σύστημα διεργασιών.

Ο whoServer θα δέχεται και θα εξυπηρετεί τα εξής αιτήματα τα οποία θα έρχονται από τον whoClient (παρόμοια με την εργασία 2):

- `/diseaseFrequency virusName date1 date2 [country]`
  Αν δεν δοθεί `country` όρισμα, ο whoServer θα βρίσκει για την ασθένεια `virusName` τον αριθμό κρουσμάτων που έχουν καταγραφεί στο σύστημα μέσα στο διάστημα `[date1...date2].` Αν δοθεί `country` όρισμα, ο whoServer θα βρίσκει 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. Στο πρωτόκολλο επικοινωνίας ανάμεσα στον πελάτη και τον whoServer θα πρέπει να διαχειρίζεται με κάποιο τρόπο το γεγονός πως το `[country]` είναι προαιρετικό στο ερώτημα αυτό.
- `/topk-AgeRanges k country disease date1 date2`
  Ο whoServer θα βρίσκει, για τη 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.
- `/searchPatientRecord recordID`
  Ο whoServer προωθεί σε όλους τους Workers το αίτημα και περιμένει απάντηση από το Worker με το record `recordID`.
- `/numPatientAdmissions disease date1 date2 [country]`
  Αν δοθεί `country` όρισμα, ο whoServer θα πρέπει να προωθήσει το αίτημα στους workers έτσι ώστε να βρει το συνολικό αριθμό ασθενών που μπήκαν στο νοσοκομείο με την ασθένεια `disease` στη συγκεκριμένη χώρα μέσα στο διάστημα `[date1 date2].` Αν δεν δοθεί `country` όρισμα, θα βρει 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.
- `/numPatientDischarges disease date1 date2 [country]`
  Αν δοθεί το όρισμα `country`, ο whoServer θα βρίσκει 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`, ο whoServer θα βρίσκει 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.

  Όταν ο whoServer δεχτεί ένα ερώτημα, το προωθεί στα αντίστοιχα worker processes μέσω ενός socket και αναμένει την απάντηση από τα workers. Το ερώτημα που προώθησε σε ένα Worker process μαζί με τις απαντήσεις που λαμβάνει ο whoServer από αυτό το Worker, τις τυπώνει στο stdout. Επίσης ο whoServer προωθεί την απάντηση στο αντίστοιχο νήμα του whoClient που έκανε το ερώτημα.

  Σημείωση: ανάλογα την υλοποίησή σας πιθανώς θα πρέπει να χειριστείτε με κάποιον τρόπο το πρόβλημα πως θα γράφουν πολλά threads ταυτόχρονα στο stdout και θα είναι μπερδεμένο το output.

C) Το πρόγραμμα whoClient (30%)

Το πρόγραμμα `whoClient` θα χρησιμοποιείται ως εξής:

```
./whoClient –q queryFile –w numThreads –sp servPort –sip servIP
```

- Η παράμετρος `queryFile` είναι το αρχείο που περιέχει τα ερωτήματα που πρέπει να σταλούν στον whoServer.
- Η παράμετρος `numThreads:` είναι ο αριθμός των νημάτων που θα δημιουργήσει ο whoClient για την αποστολή ερωτημάτων στον whoServer
- Η παράμετρος `servPort` είναι ο αριθμός θύρας όπου ακούει ο whoServer στον οποίο θα συνδεθεί ο whoClient.
- Η παράμετρος `servIP` είναι η IP διεύθυνση του whoServer στον οποίο θα συνδεθεί ο whoClient.

Η λειτουργία του multithreaded whoClient είναι η εξής. Θα ξεκινάει και θα ανοίγει το αρχείο queryFile το οποίο και θα διαβάζει γραμμή γραμμή. Σε κάθε γραμμή θα υπάρχει μία εντολή από αυτές που μπορεί να δεχτεί ο whoServer. Για κάθε εντολή θα δημιουργεί ένα νήμα το οποίο θα αναλαβει να στείλει μία εντολή (δηλαδή μία γραμμή) στον whoServer. Το νήμα θα δημιουργείται αλλά δε θα συνδέεται αμέσως στον whoServer. Όταν δημιουργηθούν όλα τα νήματα, δηλαδή έχουμε ένα

νημα για κάθε εντολή του αρχείου, τότε **θα πρέπει τα νήματα να ξεκινήσουν όλα μαζί** να προσπαθήσουν να συνδεθούν στον whoServer και να στείλουν την εντολή τους. Όταν αποσταλεί η εντολή, κάθε νήμα θα τυπώνει την απάντηση που έλαβε απο το whoServer στο stdout και μπορεί να τερματίσει. Όταν τερματίσουν όλα τα νήματα τερματίζει και ο whoClient.

Παρατηρήσεις

Στα προγράμματά σας, κοινές μεταβλητές που μοιράζονται ανάμεσα σε πολλαπλά νήματα θα πρέπει να προστατεύονται με τη χρήση mutexes. Τονίζεται πως το busy-waiting δεν είναι αποδεκτή λύση για τη παραμονή πρόσβασης σε κάποιον κοινό buffer ανάμεσα στα νήματα των προγραμμάτων σας. 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 OnomaEponymoProject3.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.