ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Διαδικτυακός και Ταυτόχρονος Προγραμματισμός

http://eclass.uth.gr/eclass/courses/INFS212/

Εαρινό εξάμηνο, ακαδημαϊκή περίοδος 2018-2019
Άσκηση 2, Τμήματα Παρασκευής 13:00-16:00
Νήματα, συγχρονισμός και ενδο-διεργασιακή επικοινωνία μέσω TCP sockets

Σκοπός: Στις εργασίες σας θα υλοποιήσετε ένα σύστημα αγοράς θέσεων σε θεατρικές παραστάσεις με χρήση του πακέτου νημάτων POSIX threads (pthreads). Στο σύστημα αυτό κάθε πελάτης αρχικά κλείνει τις θέσεις που επιθυμεί, στη συνέχεια τις πληρώνει με πιστωτική κάρτα και, τέλος, τα χρήματα μεταφέρονται στο λογαριασμό της εταιρείας. Στα συστήματα αυτά έχουμε έναν μεγάλο αριθμό πελατών οι οποίο εξυπηρετούνται από έναν περιορισμένο αριθμό σημείων εξυπηρέτησης, συνεπώς το πρόγραμμά σας πρέπει να υλοποιεί αμοιβαίο αποκλεισμό (με mutexes) και συγχρονισμό (με μεταβλητές συνθήκης). Η επικοινωνία μεταξύ των πελατών και του εξυπηρετητή θα γίνεται με την χρήση TCP sockets.

Προσοχή: το σύστημα σε αυτή την εργασία δεν είναι τόσο απλουστευμένο όσο στην πρώτη εργασία! Διαβάστε προσεκτικά το αντικείμενο και κάντε τις απαραίτητες αλλαγές στην λογική.

Η εργασία είναι ομαδική και είναι σχεδιασμένη για ομάδες δύο (2) ατόμων, μπορείτε όμως να την υλοποιήσετε και σε μικρότερες ομάδες.

Αντικείμενο: Το σύστημα αγοράς θέσεων διαθέτει έναν λογαριασμό στην τράπεζα, έναν μετρητή συναλλαγών, ένα τηλεφωνικό κέντρο με N_{tel} τηλεφωνητές οι οποίοι εξυπηρετούν τους πελάτες (βρίσκουν θέσεις) και N_{cash} ταμίες οι οποίοι εκτελούν τις πληρωμές (χρεώνουν τις πιστωτικές κάρτες). Οι θέσεις του θεάτρου έχουν ορθογώνια διάταξη, με κάθε σειρά να έχει N_{seat} θέσεις και οι σειρές να χωρίζονται σε τρεις κατηγορίες με διαφορετικό κόστος ανά θέση: οι πρώτες N_{zoneA} σειρές κοστίζουν C_{zoneA} ευρώ ανά θέση, οι επόμενες N_{zoneB} σειρές κοστίζουν C_{zoneB} ευρώ ανά θέση, και οι τελευταίες N_{zoneC} σειρές κοστίζουν C_{zoneC} ευρώ ανά θέση. Όταν όλοι οι τηλεφωνητές είναι απασχολημένοι, οι πελάτες περιμένουν τον επόμενο διαθέσιμο τηλεφωνητή. Όταν ένας πελάτης συνδεθεί με κάποιον τηλεφωνητή, επιλέγει τυχαία μία ζώνη με πιθανότητα P_{zoneA}, P_{zoneB}, P_{zoneC}, κι έναν τυχαίο ακέραιο αριθμό εισιτηρίων στο διάστημα $[N_{seatlow}, N_{seathigh}]$ και ο τηλεφωνητής χρειάζεται ένα τυχαίο ακέραιο πλήθος δευτερολέπτων στο διάστημα $[t_{seatlow},t_{seathigh}]$ για να εξετάσει αν υπάρχουν αρκετές συνεχόμενες θέσεις σε κάποια σειρά στη συγκεκριμένη ζώνη. Αν δεν υπάρχουν συνεχόμενες θέσεις, ο πελάτης λαμβάνει ένα μήνυμα λάθους και ολοκληρώνει την κλήση. Αν υπάρχουν συνεχόμενες θέσεις, δεσμεύονται στο πλάνο του θεάτρου, υπολογίζεται το συνολικό τους κόστος, και προχωράμε στην πληρωμή με πιστωτική κάρτα. Όταν όλοι οι ταμίες είναι απασχολημένοι, οι πελάτες περιμένουν τον επόμενο διαθέσιμο ταμία. Όταν ο πελάτης συνδεθεί με τον ταμία, ο ταμίας χρειάζεται ένα τυχαίο ακέραιο πλήθος δευτερολέπτων στο διάστημα $[t_{cashlow},t_{cashhigh}]$ για να δοκιμάσει

να κάνει την πληρωμή. Με πιθανότητα $P_{cardsucces}$ η πληρωμή γίνεται αποδεκτή, ο πελάτης χρεώνεται το κατάλληλο κόστος, τα χρήματα μεταφέρονται στο λογαριασμό της εταιρείας και ο πελάτης ενημερώνεται για το συνολικό κόστος, τις θέσεις του στην επιλεγμένη ζώνη και τον αριθμό συναλλαγής. Αν η πληρωμή αποτύχει, οι θέσεις επιστρέφονται στο πλάνο του θεάτρου από τον ταμία και ο πελάτης λαμβάνει ένα μήνυμα λάθους. Όταν επεξεργαστεί N_{cust} πελάτες, το σύστημα τυπώνει τα αποτελέσματά του.

Είσοδος και δεδομένα: Το αρχικό ποσό στο λογαριασμό της εταιρείας είναι 0 ευρώ και ο αρχικός αριθμός συναλλαγής είναι 0. Οι ακόλουθες σταθερές θα ορίζονται σε ένα αρχείο δηλώσεων:

- N_{tel}=8 τηλεφωνητές, N_{cash}=4 ταμίες
- N_{seat}=10 θέσεις
- N_{zoneA}=5 σειρές, N_{zoneB}=10 σειρές, N_{zoneC}=10 σειρές
- P_{zoneA}=20%, P_{zoneB}=40%, P_{zoneC}=40%,
- C_{zoneA}=30 ευρώ, C_{zoneB}=25 ευρώ, C_{zoneC}=20 ευρώ,
- $N_{seatlow}$=1 θέση, $N_{seathigh}$=5 θέσεις
- $t_{seatlow}$=5 sec, $t_{seathigh}$=10 sec
- $t_{cashlow}$=2 sec, $t_{cashhigh}$=4 sec
- $P_{cardsucces}$=90%

Το πρόγραμμά σας θα δέχεται δύο (ακριβώς) παραμέτρους με το πλήθος των πελατών προς εξυπηρέτηση, N_{cust}, και τον τυχαίο σπόρο για τη γεννήτρια των τυχαίων αριθμών.

Έξοδος εργασίας: Τα μηνύματα εξόδου ανά κράτηση είναι τα ίδια με αυτά της πρώτης εργασίας, η μόνη διαφορά είναι ότι μια κράτηση ματαιώνεται όχι όταν δεν υπάρχουν γενικά αρκετές θέσεις, αλλά όταν δεν υπάρχουν αρκετές συνεχόμενες θέσεις στη συγκεκριμένη ζώνη. Η σειρά των γραμμών που θα τυπώνονται ανά κράτηση θα είναι τυχαία, αλλά οι γραμμές δεν πρέπει να μπλέκονται μεταξύ τους. Στο τέλος της εκτέλεσης, το σύστημα θα τυπώνει τα ακόλουθα:

- Το πλάνο των θέσεων, π.χ. Ζώνη Α / Θέση 1 / Πελάτης 3, Ζώνη 2 / Θέση 2 / Πελάτης 4, κ.λπ.
- Τα συνολικά έσοδα από τις πωλήσεις.
- Το ποσοστό των συναλλαγών που ολοκληρώνεται με κάθε έναν από τους παραπάνω τέσσερις τρόπους.
- Το μέσο χρόνο αναμονής των πελατών (από τη στιγμή που εμφανίζεται ο πελάτης, μέχρι να μιλήσει με τον τηλεφωνητή και από τη στιγμή που τελειώνει ο τηλεφωνητής μέχρι τη στιγμή που αναλαμβάνει ο ταμίας, αν η συναλλαγή δεν ματαιωθεί νωρίτερα).
- Το μέσο χρόνο εξυπηρέτησης των πελατών (από τη στιγμή που εμφανίζεται ο πελάτης, μέχρι να ολοκληρωθεί η κράτηση, επιτυχώς ή ανεπιτυχώς).

Δομή κώδικα και υποδείξεις: Πέρα από αυτά που χρειάζονται και στην πρώτη εργασία, θα χρειαστείτε τουλάχιστον τα ακόλουθα:

- Μία ακέραιη μεταβλητή και ένα mutex για να μετράτε το πλήθος των διαθέσιμων ταμιών και μία μεταβλητή συνθήκης για να συγχρονίσετε τους

ταμίες με τους πελάτες, έτσι ώστε όταν δεν υπάρχουν διαθέσιμοι ταμίες να μπλοκάρονται οι πελάτες.
- Έναν πίνακα με το πλάνο των θέσεων ανά ζώνη κι ένα mutex για να τον κλειδώνετε (μην ξεχνάτε ότι χρειάζεται κλείδωμα και στη δέσμευση και στην αποδέσμευση θέσεων). Προσοχή στο ότι οι θέσεις κάθε κράτησης πρέπει να είναι συνεχόμενες και στην ίδια σειρά. Σκεφτείτε πώς μπορείτε να επιταχύνετε τη διαδικασία.

Προσοχή στο ότι οι χρόνοι αναμονής μπορεί να περιλαμβάνουν δύο στάδια αναμονής.

Δομή κώδικα και υποδείξεις: Πέρα από αυτά που χρειάζονται και στην πρώτη εργασία, θα χρειαστείτε τουλάχιστον τα ακόλουθα:

- Μία ακέραιη μεταβλητή και ένα mutex για να μετράτε το πλήθος των διαθέσιμων ταμιών και μία μεταβλητή συνθήκης για να συγχρονίσετε τους ταμίες με τους πελάτες, έτσι ώστε όταν δεν υπάρχουν διαθέσιμοι ταμίες να μπλοκάρονται οι πελάτες.
- Έναν πίνακα με το πλάνο των θέσεων ανά ζώνη κι ένα mutex για να τον κλειδώνετε (μην ξεχνάτε ότι χρειάζεται κλείδωμα και στη δέσμευση και στην αποδέσμευση θέσεων). Προσοχή στο ότι οι θέσεις κάθε κράτησης πρέπει να είναι συνεχόμενες και στην ίδια σειρά. Σκεφτείτε πώς μπορείτε να επιταχύνετε τη διαδικασία.
- Προσοχή στο ότι οι χρόνοι αναμονής μπορεί να περιλαμβάνουν δύο στάδια αναμονής.

Παραδοτέα: Ο κώδικάς σας πρέπει να αποτελείται από ένα αρχείο με δηλώσεις (συμπεριλαμβανομένων των σταθερών) και **δύο αρχεία** κώδικα C για το πρόγραμμα. Τα αρχεία αυτά πρέπει να έχουν ονόματα της μορφής `AM1-AM2-proj2.h` για τις δηλώσεις, `AM1-AM2-proj2_server.c` και `AM1-AM2-proj2_client.c` για τον κώδικα C , όπου και `AM1-AM2` είναι οι αριθμοί μητρώου σας, και τα server/client αρχεία αντίστοιχα ο κώδικας του εξυπηρετητή και ο κώδικας του πελάτη αντίστοιχα. Εκτός από τον κώδικα, θα πρέπει να γράψετε μία αναφορά η οποία να περιγράφει τη δομή του κώδικά σας και να αναφέρει τυχόν περιορισμούς ή πρόσθετα χαρακτηριστικά που έχετε υλοποιήσει. Η αναφορά πρέπει να είναι ένα αρχείο σε μορφή PDF με όνομα της μορφής `AM1-AM2-proj2.pdf`. Τέλος, θα πρέπει να συμπεριλάβετε ένα αρχείο με όνομα `test-proj2.sh` το οποίο θα είναι εκτελέσιμο και θα μεταγλωττίζει και θα εκτελεί το πρόγραμμά σας με παραμέτρους 100 πελάτες και αρχικό σπόρο 1000. Η σειρά με την οποία θα εκτελείτε τα αρχεία που μεταγλωττίσατε είναι : πρώτα εκτελείται ο server και κατόπιν εκτελείται 100 φορές ο client. Για να έχετε non-blocking εκτέλεση του server και των clients χρησιμοποιήστε στο τέλος της γραμμής εκτέλεσης τον χαρακτήρα & (δείτε εδώ[1] παραδείγματα).

[1] https://stackoverflow.com/questions/6665604/how-to-execute-programs-in-non-blocking-way-from-scripts

Επίσης, για να εκτελέσετε ένα αρχείο 100 φορές μέσω του test-proj2.sh δείτε εδώ[2] παραδείγματα.

Αυτά τα πέντε αρχεία (και τίποτα άλλο) θα πρέπει να συμπιεστούν σε ένα αρχείο σε μορφή 7zip με όνομα της μορφής `AM1-AM2-proj1.7z` και να υποβληθούν από ένα μόνο μέλος της ομάδας μέσω της υποβολής εργασιών του eclass.

Βοηθητικοί σύνδεσμοι

Για την ανάπτυξη της άσκησης, μπορείτε να συμβουλευτείτε τις παρουσιάσεις των φροντιστηρίων που είναι αναρτημένες στο site του μαθήματος http://eclass.uth.gr/eclass/courses/INFS212/ καθώς και τους παρακάτω συνδέσμους:

Εγχειρίδια

http://man7.org/linux/man-pages/man2/fork.2.html

Δείγματα κώδικα

https://computing.llnl.gov/tutorials/pthreads/

https://www.gnu.org/software/libc/manual/html_node/Setting-an-Alarm.html

[2] https://stackoverflow.com/questions/3737740/is-there-a-better-way-to-run-a-command-n-times-in-bash