ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ
ΤΟΜΕΑΣ ΛΟΓΙΚΟΥ ΤΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Λειτουργικά Συστήματα Ι

http://netcins.ceid.upatras.gr/OpSys-I/

Χειμερινό εξάμηνο, ακαδημαϊκή περίοδος 2013-2014

Άσκηση 2:
Νήματα και συγχρονισμός.

1 Εισαγωγή

1.1 Στόχος

Ο στόχος της άσκησης είναι η εξοικείωση με θεμελιώδεις έννοιες και μηχανισμούς που προσφέρει το λειτουργικό σύστημα , όπως η δημιουργία και η διαχείριση διεργασιών (`fork, exit`), η επικοινωνία διεργασιών (`sockets, shared memory, signals`) και ο συγχρονισμός (`semaphores`).

2 Περιγραφή

Στόχος της άσκησης είναι η δημιουργία ενός πληροφοριακού συστήματος διαχείρισης εισιτηρίων για θεατρικές παραστάσεις με τηλεφωνική κράτηση και αγορά με πιστωτική κάρτα. Το πληροφοριακό αυτό σύστημα θα το χρησιμοποιήσει μια εταιρία κρατήσεων, όπως πχ η tickethour.com. Η εταιρία διαθέτει:

- Ένα τηλεφωνικό κέντρο με $N_{company}$ τηλεφωνητές, οι οποίοι διεκπεραιώνουν τις κλήσεις των πελατών (βρίσκουν διαθέσιμες θέσεις και χρεώνουν τις πιστωτικές κάρτες). Ο κάθε τηλεφωνητής χρειάζεται ένα συγκεκριμένο χρονικό διάστημα $t_{seatfind}$ για να βρει διαθέσιμες θέσεις. Όταν όλοι οι τηλεφωνητές είναι απασχολημένοι, ο πελάτης μπαίνει σε μια ουρά και περιμένει τον επόμενο διαθέσιμο τηλεφωνητή.
- Σύνδεση με το πληροφοριακό σύστημα της τράπεζας για τον έλεγχο της πιστωτικής κάρτας του πελάτη. Το σύστημα της τράπεζας αποτελείται από N_{bank} συσκευές ελέγχου (τερματικά) και ο έλεγχος μιας πιστωτικής κάρτας διαρκεί $t_{cardcheck}$ χρόνο. Το σύστημα της τράπεζας μπορεί να εξυπηρετεί ταυτόχρονα το πολύ N_{bank} αιτήσεις ελέγχου: εάν είναι παραπάνω οι αιτήσεις, μπαίνουν σε μια ουρά και περιμένουν να εξυπηρετηθούν όταν ελευθερωθεί μια συσκευή. Με το πληροφοριακό σύστημα της τράπεζας συνδέονται μόνο οι $N_{company}$ τηλεφωνητές της εταιρίας.
- Ένα σύστημα χρέωσης το οποίο συγκεντρώνει τα ποσά από την αγορά εισιτηρίων στον λογαριασμό της εταιρίας κρατήσεων (company_account). Το σύστημα επίσης

είναι υπεύθυνο να μεταφέρει τις εισπράξεις στον τραπεζικό λογαριασμό του θεάτρου (theater_account).

- Το «πλάνο» θέσεων του συγκεκριμένου θεάτρου. Το πλάνο αποτελείται από αριθμημένες θέσεις διάφορων κατηγοριών (ζώνες). Η τιμή ενός εισιτηρίου είναι διαφορετική, ανάλογα με την κατηγορία στην οποία ανήκει. Ένα πιθανό πλάνο είναι το εξής:
  Ζώνη Α: [Θ1,Θ2,….$Θ_A$]
  Ζώνη Β: [Θ1,Θ2,….$Θ_B$]
  κ.ο.κ.
  Όπου $Θ_A$ και $Θ_B$ ο αριθμός διαθέσιμων θέσεων των κατηγοριών Α και Β αντίστοιχα.

Οι πελάτες κάνουν μια κλήση στο τηλεφωνικό κέντρο, και εάν υπάρχει διαθέσιμος τηλεφωνητής, συνδέονται μαζί του για να κάνουν την παραγγελία, αλλιώς μπαίνουν στην αναμονή. Κάθε t_{wait} δευτερόλεπτα στην αναμονή, ο πελάτης ακούει ένα ηχογραφημένο μήνυμα «συγγνώμης» για την καθυστέρηση. Ο κάθε πελάτης διαλέγει τον αριθμό και την κατηγορία των εισιτηρίων που θέλει να αγοράσει.

Οι τηλεφωνητές παίρνουν την παραγγελία του χρήστη και κάνουν τις εξής **ταυτόχρονες** κινήσεις:

1. Ελέγχουν εάν ο χρήστης έχει δώσει έγκυρη πιστωτική κάρτα. Ο έλεγχος γίνεται με την σύνδεση στο πληροφοριακό σύστημα της τράπεζας (αφού καταφέρουν να συνδεθούν με το τερματικό, ο έλεγχος απαιτεί $t_{cardcheck}$ χρόνο).
2. Ψάχνουν στις αριθμημένες θέσεις του θεάτρου για να βρουν ελεύθερες θέσεις. Η αναζήτηση διαρκεί $t_{seatfind}$ χρόνο.

Εάν επιτύχει ο έλεγχος και βρεθούν ελεύθερες θέσεις, γίνονται οι απαραίτητες κρατήσεις χρεώνεται η πιστωτική κάρτα και ενημερώνεται ο πελάτης για το ποσό και τις θέσεις που έχουν κρατηθεί.

Τα έσοδα από τα εισιτήρια συγκεντρώνονται αρχικά στον τραπεζικό λογαριασμό της εταιρίας κρατήσεων (company_account). Σε τακτά χρονικά διαστήματα ($t_{transfer}$), τα χρήματα από αυτόν τον λογαριασμό μεταφέρονται με μια τραπεζική εντολή στον λογαριασμό του θεάτρου (theater_account).

2.1 Δεδομένα άσκησης

Για την εκτέλεση της άσκησης, θεωρούμε τα εξής:

Το θέατρο διαθέτει 4 ζώνες, τις Α, Β, Γ και Δ. Στον παρακάτω πίνακα βλέπουμε την τιμή εισιτηρίου για κάθε ζώνη, καθώς και τον αριθμό θέσεων για κάθε ζώνη.

Ζώνη	Τιμή εισιτηρίου	Αριθμός θέσεων
Α	50€	100
Β	40€	130
Γ	35€	180
Δ	30€	230

Ο κάθε πελάτης διαλέγει τα εξής:

- **Αριθμό εισιτηρίων.** Με τυχαίο τρόπο (με την χρήση της συνάρτησης rand()), ο πελάτης θα διαλέγει από 1 μέχρι και 4 εισιτήρια μιας συγκεκριμένης ζώνης χρέωσης.

- **Ζώνη χρέωσης.** Πάλι με τυχαίο τρόπο, ο κάθε πελάτης επιλέγει με ποσοστό 10% εισιτήρια Α ζώνης, με ποσοστό 20% εισιτήρια Β ζώνης, με ποσοστό 30% εισιτήρια Γ ζώνης και με ποσοστό 40% εισιτήρια Δ ζώνης (όλα τα εισιτήρια θα ανήκουν σε μια συγκεκριμένη ζώνη χρέωσης, δεν μπορεί πχ ένας πελάτης να ζητήσει 1 εισιτήριο Α ζώνης και 1 Β ζώνης).

- **Πιστωτική κάρτα.** Η πιστωτική κάρτα του πελάτη με ποσοστό 90% θα είναι έγκυρη και θα μπορεί να γίνει η συναλλαγή. Με ποσοστό 10% δεν θα είναι έγκυρη (πχ, είτε από λάθος στοιχεία που έδωσε ο πελάτης, είτε επειδή δεν έχει διαθέσιμο υπόλοιπο).

Εάν υπάρχουν διαθέσιμες θέσεις και η πιστωτική κάρτα του πελάτη είναι έγκυρη, η παραγγελία εκτελείται και θα ενημερώνεται για τις θέσεις που κρατήθηκαν, αλλιώς επιστρέφεται μήνυμα λάθους στον πελάτη. Ο πελάτης θα παίρνει τέσσερα πιθανά διαφορετικά μηνύματα:

Α) Η κράτηση ολοκληρώθηκε επιτυχώς. Το αναγνωριστικό της κράτησης είναι <pelatis_id>, οι θέσεις σας είναι οι <Α1, Α2, κλπ> και το κόστος της συναλλαγής είναι <Χ> ευρώ.
Β) Η πιστωτική κάρτα δεν είναι έγκυρη.
Γ) Δεν υπάρχουν διαθέσιμες θέσεις για την συγκεκριμένη ζώνη.
Δ) Δεν υπάρχουν **καθόλου** διαθέσιμες θέσεις (το θέατρο έχει γεμίσει).

Το σύστημα δέχεται παραγγελίες **μέχρι να γεμίσουν όλες οι θέσεις** του θεάτρου. Μόλις γεμίσουν όλες οι θέσεις ανεξαρτήτως κατηγορίας, το σύστημα επιστρέφει το μήνυμα Δ).

Θεωρήστε τις παρακάτω τιμές για τις μεταβλητές του συστήματος:

N_{thl}=10	N_{bank}=4	$t_{seatfind}$ =6sec	$t_{cardcheck}$=2sec	t_{wait}=10sec	$t_{transfer}$=30sec

3 Λεπτομέρειες υλοποίησης

Ο server θα είναι μια διεργασία που θα εκτελείται στο παρασκήνιο (background). Ο server θα δεσμεύει ένα τμήμα μνήμης, στο οποίο θα αποθηκεύονται όλες οι απαραίτητες πληροφορίες για την λειτουργία του συστήματος (πλάνο θέσεων, υπόλοιπο λογαριασμού εταιρίας κρατήσεων, υπόλοιπο λογαριασμού θεάτρου, κλπ).

- Για κάθε αίτηση που θα λαμβάνει ο server θα δημιουργεί νέα νήματα, τα οποία θα εξυπηρετούν την αίτηση. Ο server αμέσως θα μπαίνει πάλι σε κατάσταση αναμονής, περιμένοντας την επόμενη αίτηση.
- Αντί για fork(), σε αυτή την άσκηση θα χρησιμοποιήσετε την εντολή δημιουργίας νημάτων pthread_create().

- Αντί για sem_wait() και sem_post() θα χρησιμοποιήσετε τις εντολές pthread_mutex_lock(), pthread_mutex_trylock() και pthread_mutex_unlock() για **βραχυπρόθεσμο** κλείδωμα τμήματος κώδικα (με σκοπό τον αμοιβαίο αποκλεισμό νημάτων κατά την πρόσβαση σε ένα κρίσιμο τμήμα του κώδικα).
- Αντί για sem_wait() και sem_post() για πιο **μακροπρόθεσμο** κλείδωμα και συγχρονισμό μεταξύ νημάτων θα χρησιμοποιήσετε τις λειτουργίες των condition variables pthread_cond_init(), pthread_cond_wait(), pthread_cond_signal().
- Για να συγχρονίσετε τα νήματα έτσι ώστε πχ ένα νήμα να περιμένει τον τερματισμό ενός άλλου, χρησιμοποιήστε την συνάρτηση pthread_join().
- Πρέπει να δώσετε προσοχή στο γεγονός ότι τα νήματα έχουν όλα πρόσβαση στην ίδια μνήμη της διεργασίας που τα δημιούργησε.
- Τέλος, τα νήματα που εξυπηρετούν τις αιτήσεις θα πρέπει να τερματίζονται κατάλληλα. Γενικότερα, δώστε προσοχή να μην αφήνετε ανοιχτούς πόρους που δεν χρειάζονται πλέον οι διεργασίες.

3.1 Έλεγχος ορθότητας

Για να βεβαιωθείτε ότι οι λειτουργίες του server έχουν υλοποιηθεί σωστά (και ιδιαίτερα ο συγχρονισμός), δημιουργήστε clients οι οποίοι δεν θα περιμένουν είσοδο από τον χρήστη, αλλά θα στέλνουν προκαθορισμένες παραγγελίες, έως ότου γεμίσουν οι θέσεις του θεάτρου. Με ένα script ξεκινήστε 10 clients μαζί οι οποίοι στέλνουν συνέχεια παραγγελίες, μέχρι να γεμίσουν όλες οι θέσεις του θεάτρου. Αν εκτελέσετε το πείραμα πολλές φορές θα πρέπει το σύστημά σας να λειτουργεί σωστά. Ο κώδικας που θα παραδώσετε θα πρέπει να δέχεται στοιχεία και από τον χρήστη.

Στο τέλος της εκτέλεσης, τυπώστε στην οθόνη τα παρακάτω (τα οποία θα συμπεριλάβετε και στην γραπτή αναφορά):

- Το τελικό πλάνο των κρατήσεων. Το πλάνο θα έχει την μορφή:
  Ζώνη Α: [Π1,Π1,Π2,Π3….Π100]
  Ζώνη Β: [Π6,Π6,….Π70]
  Όπου οι Π1, Π2, κλπ, είναι οι πελάτες που αγοράσανε τα εισιτήρια.
- Το ποσοστό αποτυχημένων παραγγελιών σε σχέση με τις συνολικές παραγγελίες.
- Τον μέσο χρόνο αναμονής των παραγγελιών. Ο χρόνος αναμονής είναι ο χρόνος που ο πελάτης περιμένει μέχρι να συνδεθεί με εκπρόσωπο.
- Τον μέσο χρόνο εξυπηρέτησης των παραγγελιών. Ο μέσος χρόνος εξυπηρέτησης είναι ο χρόνος που ο πελάτης μιλάει με τον εκπρόσωπο.
- Έναν πίνακα με τις μεταφορές των ποσών που έγιναν από τον λογαριασμό της εταιρίας στον λογαριασμό του θεάτρου.
- Το τελικό ποσό που συγκεντρώθηκε από την πώληση των εισιτηρίων.

3.2 Βοηθητικοί σύνδεσμοι

Για την ανάπτυξη της άσκησης, μπορείτε να συμβουλευτείτε τις παρουσιάσεις των φροντιστηρίων που είναι αναρτημένες στο site του μαθήματος http://netcins.ceid.upatras.gr/OpSys-I/ καθώς και τους παρακάτω συνδέσμους:

Εγχειρίδια
http://netcins.ceid.upatras.gr/OpSys-I/project/Socket.htm
http://netcins.ceid.upatras.gr/OpSys-I/project/server_client.htm
http://netcins.ceid.upatras.gr/OpSys-I/project/Signals.htm
http://netcins.ceid.upatras.gr/OpSys-I/project/POSIXThreads.htm
http://netcins.ceid.upatras.gr/OpSys-I/project/pthread.htm
http://www.gnu.org/software/make/manual/make.html#Simple-Makefile
http://man7.org/linux/man-pages/man7/pthreads.7.html
http://man7.org/linux/man-pages/man7/signal.7.html

Δείγματα κώδικα
http://netcins.ceid.upatras.gr/OpSys-I/project/threads/thread_condition.c
http://netcins.ceid.upatras.gr/OpSys-I/project/threads/thread_create.c
http://netcins.ceid.upatras.gr/OpSys-I/project/threads/thread_join.c
http://netcins.ceid.upatras.gr/OpSys-I/project/threads/thread_mutex.c
http://netcins.ceid.upatras.gr/OpSys-I/project/threads/thread_timed_wait.c

4 Διαδικαστικά

- Η άσκηση θα υλοποιηθεί είτε ατομικά, είτε σε ομάδες των 2 ατόμων. Οι ίδιες ομάδες θα παραμείνουν και για τις υπόλοιπες ασκήσεις.
- Η ημερομηνία παράδοσης της άσκησης είναι η **ημερομηνία εξέτασης του μαθήματος**.
- Ο κώδικας που θα παραδώσετε θα πρέπει να είναι καλά δομημένος.
- Ο κώδικας που θα παραδώσετε θα πρέπει να είναι καλά σχολιασμένος.
- • Ο κώδικας θα πρέπει να χρησιμοποιεί τα εργαλεία που αναφέρθηκαν νωρίτερα (pthreads, exit, sockets, mutexes και condition variables).
- Ο έλεγχος του κώδικα θα γίνει σε 2 φάσεις:
    - Στην πρώτη φάση θα εκτελεστεί η εντολή **`make`** στον κατάλογο που υπάρχει το πρόγραμμα. Εάν το πρόγραμμα μεταγλωττιστεί κανονικά, περνάμε στην επόμενη φάση.
    - Η επόμενη φάση ελέγχει την σωστή λειτουργία του κώδικα. Θα φροντίσετε να υπάρχει ένα εκτελέσιμο script με το όνομα **`run_tests.sh`** Η εκτέλεση του script αυτού θα κάνει τον έλεγχο ορθότητας, όπως τον περιγράψαμε παραπάνω. Ο κώδικάς σας θα πρέπει να παίρνει και input από τον χρήστη.
- Μαζί με τον κώδικα, θα δώσετε και μια **σύντομη αναφορά** όπου θα περιγράφετε την μεθοδολογία που ακολουθήσατε (όχι κώδικας εδώ). Θα περιγράφετε τις

δομές/σήματα που χρησιμοποιήσατε και τις συναρτήσεις/αρχεία με τα οποία αλληλεπιδράτε με αυτές. Η αναφορά θα αποθηκευτεί στο αρχείο **report.pdf**.

- Σε περίπτωση καθυστέρησης στην παράδοση της άσκησης θα υπάρχει μείωση 10% για κάθε 24 ώρες παράδοσης.
- Σε περίπτωση που εντοπιστεί αντιγραφή, ο βαθμός θα μηδενίζεται για όλους όσους εμπλέκονται (και αυτούς που έλαβαν την άσκηση και αυτούς που την έδωσαν).

5 Τρόπος Αποστολής

Η αποστολή της άσκησης θα γίνει μέσω ηλεκτρονικού ταχυδρομείου στην διεύθυνση: opsys-i-2013-ceid@googlegroups.com

Το e-mail με το οποίο θα στείλετε την άσκηση, θα πρέπει να είναι της μορφής user@ceid.upatras.gr και όχι προσωπικό e-mail τύπου gmail ή yahoo.

Η ώρα παράδοσης της άσκησης θεωρείται η ώρα αποστολής του email.

Το email θα είναι δομημένο ως εξής:

- **Θέμα:** Το θέμα του email θα είναι της μορφής **AM1_AM2_project2** όπου AM1 και AM2 οι Αριθμοί μητρώου των συνεργατών. Σε περίπτωση που η εργασία γίνει μόνο από ένα άτομο, τότε θα έχει την μορφή AM1_project2.
- **Κυρίως κείμενο (body):** Κενό.
- **Επισυναπτόμενο αρχείο (attachment):** Το email θα περιέχει 1 μόνο attachment σε μορφή .zip, το οποίο θα περιέχει τα αρχεία της άσκησης. Το όνομα του attachment θα είναι το ίδιο με το θέμα του email, δηλαδή θα έχει την μορφή **AM1_AM2_project2.zip**
- **Περιεχόμενα επισυναπτόμενου αρχείου .zip:** Το αρχείο .zip θα περιέχει τα αρχεία του πηγαίου κώδικα σε c, το αρχείο **Makefile**, το αρχείο εκτέλεσης ελέγχου **run_tests.sh** και ένα αρχείο με την αναφορά (σε μορφή pdf) με τον τίτλο **report.pdf.** Τα παραπάνω αρχεία δεν θα περιέχονται σε κανέναν υποκατάλογο του αρχείου zip.