# Λειτουργικά Συστήματα (ΗΥ-345)
# Χειμερινό Εξάμηνο 2020
# Άσκηση 2

Φροντιστήριο: 23/10/2020
Παράδοση: 06/11/2020

Σε αυτήν την άσκηση θα εξοικειωθείτε με τη δημιουργία και το χειρισμό threads, καθώς και semaphores για τον συγχρονισμό των διαφορετικών threads.

Συγκεκριμένα, θα υλοποιήσετε το πρόβλημα της παραγγελίας take away σε ένα σουβλατζίδικο, διότι λόγω περιορισμών COVID-19 μόνο ένας πελάτης μπορεί να βρίσκεται εντός του καταστήματος ανά πάσα στιγμή. Θεωρούμε ότι κατα τη διάρκεια που κάποιος βρίσκεται στο κατάστημα και περιμένει να πάρει την παραγγελία του μπορεί να υπάρχουν από 0 έως Ν πελάτες εκτός καταστήματος που περιμένουν να εξυπηρετηθούν.

Η διαδικασία που ακολουθεί ο Σεφ για την εξυπηρέτηση ενός πελάτη είναι η εξής:

- Αν υπάρχει ένας πελάτης που πρέπει να εξυπηρετηθεί, ο Σεφ καλεί τον πελάτη στο κατάστημα και τον εξυπηρετεί. Ο πελάτης αφού πάρει την παραγγελία του φεύγει από το κατάστημα και το συγκεκριμένο customer thread τερματίζει.

- Εάν δεν υπάρχει κάποιος που εξυπηρετείται και υπάρχουν πελάτες που περιμένουν, ο Σεφ δέχεται έναν από αυτούς στο κατάστημα και τον εξυπηρετεί.

- Εάν δεν υπάρχουν πελάτες που περιμένουν, ο Σεφ ασχολείται με το κινητό του τηλέφωνο στο Facebook μέχρι να ειδοποιηθεί από τον επόμενο πελάτη.

Ο συγχρονισμός των threads πρέπει να γίνει μέσω semaphores και δεν υπάρχει κάποιος περιορισμός στον αριθμό των semaphores που μπορείτε να χρησιμοποιήσετε.

Για την υλοποίηση πρέπει αναγκαστικά να χρησιμοποιήσετε 1 + N pthreads (POSIX Threads). Ένα thread για τη δημιουργία του Σεφ και N threads για τους πελάτες. Τα threads που θα δημιουργήσετε πρέπει να κάλουν τις συναρτήσεις void *chef()  και void *customer() αντίστοιχα.

Τα threads θα πρέπει αναγκαστικά να δημιουργούνται **ΜΟΝΟ** στη main συνάρτηση του προγράμματος σας. Οι συναρτήσεις customer() και chef() **δεν** πρέπει να δημιουργούν άλλα threads.

Για την επαλήθευση της σωστής λειτουργίας του προγράμματός σας, μπορείτε να ορίσετε μια μικρή καθυστέρηση κατα την δημιουργία των thread των πελατών, έτσι ώστε να μην υπάρχουν πάντα πελάτες που περιμένουν να εξυπηρετηθούν.
Παράδειγμα:
Για N = 9 customer threads, αφού έχει γίνει η δημιουργία του thread του Σεφ, η main συνάρτηση θα δημιουργεί 3 customer threads κάθε φορά και μετά θα περιμένει για ένα χρονικό διάστημα X δευτερολέπτων με την χρήση της συνάρτησης sleep(). Με αυτό τον τρόπο ο Σεφ θα εξυπηρετήσει 3 ομάδες των τριών πελατών και στο ενδιάμεσο θα ασχολείται με τον κινητό του στο Facebook.

Το thread του Σεφ μπορεί να εκτελείται επ αόριστον ή να τερματίζει όταν έχουν εξυπηρετηθεί όλα τα N threads των πελατών.

Η λύση στο συγκεκριμένο πρόβλημα της άσκησης **δεν** πρέπει να καταλήγει σε deadlock (π.χ ο Σεφ να μην μπορεί να εξυπηρετήσει κάποιο πελάτη).

Η μεταγλώττιση πρέπει να γίνει με την παράμετρο -lpthread ώστε ο linker να μπορεί να βρει τα σύμβολα στην pthread βιβλιοθήκη.

Μερικές απο τις βιβλιοθήκες που θα χρησιμοποιήσετε είναι η pthread.h και η semaphore.h

## Χρήση των Man Pages

Ένα man page περιγράφει τον τρόπο λειτουργίας ενός προγράμματος, ενός system call ή μιας library function. Η εμφάνιση ενός man page γίνεται με τη χρήση της εντολής man. Για να δείτε στο Linux το man page που αναφέρεται στη συνάρτηση foo εκτελείται: man foo .

Ο συμβολισμός foo(N) αναφέρεται στο man page που περιγράφει τη foo στη κατηγορία (section) 'N'. Για να δείτε στο Linux το man page που αναφέρεται στη συνάρτηση foo στο section N 3 εκτελείται: man -S N foo. Για παράδειγμα μπορείτε να δείτε το man που αναφέρεται στην open(2) ως εξής: man -S 2 open.

Σας παραθέτουμε man pages με συναρτήσεις που μπορεί να χρειαστείτε για την υλοποίηση της άσκησης. Η παρακάτω λίστα **δεν** είναι δεσμευτική. Μπορείτε να χρησιμοποιήσετε και εναλλακτικούς τρόπους.

sem_init(3), sem_wait(3), sem_post(3),sem_destroy(3), pthread_create(3), pthread_exit(3), pthread_join(3), pthread_exit(3), sleep(1).

## Παρατηρήσεις:

1. Η άσκηση είναι ατομική. Τυχόν αντιγραφές μπορούν να ανιχνευθούν εύκολα από κατάλληλο πρόγραμμα και θα μηδενιστούν. Συμπεριλάβετε το όνομα σας και το

λογαριασμό σας (account) σε όλα τα αρχεία.

2. Γράψτε ένα αρχείο README, το πολύ 30 γραμμών, με επεξηγήσεις για τον τρόπο υλοποίησης της άσκησης.

3. Κατασκευάστε ένα αρχείο Makefile, έτσι ώστε πληκτρολογώντας make all να γίνεται η μεταγλώττιση (compilation) των αρχείων. Επίσης πληκτρολογώντας make clean να καθαρίζονται όλα τα περιττά αρχεία.

4. Τοποθετήστε σε ένα κατάλογο όλα τα αρχεία προς παράδοση για την άσκηση 2 (Όλα τα απαραίτητα .c και .h αρχεία, το Makefile και το README). Παραδώστε τα παραπάνω αρχεία χρησιμοποιώντας το πρόγραμμα turnin (πληκτρολογήστε turnin assignment_2@hy345 directory_name από τον κατάλογο που περιέχει τον κατάλογο directory_name με τα αρχεία της άσκησης).

5. Σε πολλές περιπτώσεις τα ονόματα των αρχείων είναι ενδεικτικά. Μπορείτε να χρησιμοποιήσετε όποια σας βολεύουν.

6. Χρησιμοποιήστε την mailing list του μαθήματος για απορίες. Αποφύγετε τα προσωπικά μηνύματα προς τους βοηθούς.

7. Μην στέλνετε κομμάτια της υλοποίησης σας στην mailing list του μαθήματος.