ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
Τμήμα Πληροφορικής και Τηλεπικοινωνιών
3η Εργασία - Τμήμα: Περιττών Αριθμών Μητρώου
K22: Λειτουργικά Συστήματα – Χειμερινό Εξάμηνο '24
Ημερομηνία Ανακοίνωσης: Σάββατο 30 Νοεμβρίου 2024
Ημερομηνία Υποβολής: Δευτέρα 23 Δεκεμβρίου 2024 Ώρα 23:55

Εισαγωγή στην Εργασία:

Ο στόχος αυτής της εργασίας είναι να δημιουργήσετε ανεξάρτητα προγράμματα τα οποία απο κοινού λύνουν το πρόβλημα γνωστό σαν το «*Μπαρ στο Μονοπάτι για την Νεμέα*». Το πρόβλημα αυτό έχει να κάνει με την δυνατότητα που έχουν περιπατητές στο μονοπάτι από το *Πετρί* στην *Νεμέα*, να σταματήσουν σε ένα υπαίθριο μπαρ ώστε να πιουν ή και να φάνε κάτι πριν συνεχίσουν στην πορεία τους.

Ο κάθε πελάτης που φτάνει στο μπαρ πρέπει πρώτα να πάρει μια διαθέσιμη καρέκλα από ένα τραπέζι και μετά να παραγγείλει. Το υπαίθριο μπαρ διαθέτει 3 τραπέζια με 4 καρέκλες το καθένα. Ο χώρος διευθύνεται από έναν υπεύθυνο υποδοχής (ή απλά 'υπεύθυνο') ο οποίος παίρνει παραγγελίες από πελάτες που καταφθάνουν και τους/τις εξυπηρετεί με την σειρά που οι πελάτες εμφανίζονται στο τραπέζι σερβιρίσματος.

Ο κάθε πελάτης μπορεί να παραγγείλει κρασί ή νερό καθώς επίσης τυρί ή/και σαλάτα. Αφού ο κάθε πελάτης παραλάβει αυτά που επιθυμεί κάθεται στο τραπέζι και αρχίζει και 'συνομιλεί' με τους υπόλοιπους στο τραπέζι αν υπάρχουν, και να τρώει ή να πίνει. Όταν ένα τραπέζι γεμίσει, δεν μπορεί να δεχτεί κάποιον επιπλέον πελάτη ακόμα και αν ένας ή πιο πολλοί πελάτες αποχωρήσουν από το συγκεκριμένο τραπέζι. Το τραπέζι για να 'εξυπηρετήσει' νέους πελάτες θα πρέπει να εκκενωθεί «πλήρως από όλους τους προηγούμενους που έτρωγαν, έπιναν και συζητούσαν μέχρι τώρα». Αν δεν υπάρχει καρέκλα διαθέσιμη σε τραπέζι, ο επισκέπτης περιμένει για το *πρώτο τραπέζι* που θα αδειάσει πλήρως και θα γίνει διαθέσιμο.

Οι επισκέπτες και ο υπεύθυνος συγχρονίζονται χρησιμοποιώντας σήματα P() και V() και καταγράφουν πληροφορίες που κρίνεται απαραίτητο σε ένα shared memory segment όπου διατηρούνται επιπλέον και τα στατιστικά στοιχεία για το σέρβις που προσφέρει το *«Μπαρ στο Μονοπάτι για Νεμέα»*.

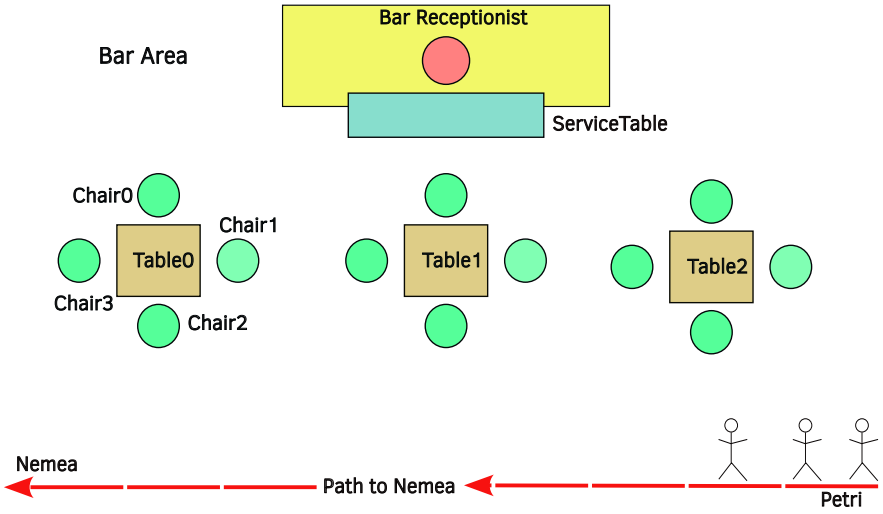
Οι κανόνες που πρέπει να ακολουθούν οι διεργασίες αναλύονται σε λεπτομέρεια πιο κάτω. Γενικά, αναμένεται οι επισκέπτες να μην εμπλέκονται σε κατάσταση λιμού. Συνολικά, σε αυτή την άσκηση θα πρέπει να:

1. χρησιμοποιήσετε *POSIX Semaphores* για να υλοποιήσετε τη λύση σας που θα εμπεριέχει σχετικές κλήσεις P() και V(),
2. δημιουργήσετε ένα *shared memory segment* προσπελάσιμο από διεργασίες ώστε να γίνεται σωστή καταγραφή των λειτουργιών σε εξέλιξη και των στατιστικών,
3. παρέχετε τη δυνατότητα τα προγράμματά σας να παραμείνουν 'εργαζόμενα' για περιόδους χρόνου που καθορίζονται από τους χρήστες,
4. έχετε προγράμματα-επισκέπτες που μεταβάλλουν, με την λήξη τους, τα στατιστικά που διατηρούνται από το μπαρ και τα οποία στο τέλος της λειτουργίας του υπεύθυνου παρουσιάζονται στο σχετικό `tty`,
5. μπορεί ένα ανεξάρτητο πρόγραμμα που ονομάζεται monitor σε οποιαδήποτε χρονική στιγμή να παρουσιάζει την επικρατούσα κατάσταση στο μπαρ σε σχέση με τις διαθέσιμες καρέκλες/τραπέζια αλλά και την κατανάλωση προϊόντων μέχρι στιγμής.

Τέλος θα πρέπει να επιδείξετε την ορθότητα της λύσης σας με τα διάφορα προγράμματα πιθανόν να εκτελούνται από διαφορετικά ttys και υιοθετώντας ένα μηχανισμό ιστορικού–των–γεγονότων (logging).

Διαδικαστικά:

- Το πρόγραμμα σας θα πρέπει να γραφτεί σε `C` (ή `C++` αν θέλετε αλλά χωρίς τη χρήση `STL/Templates`) και να τρέχει στα Linux *workstations* του τμήματος.
- Ο πηγαίος κώδικας σας (source code) πρέπει να αποτελείται από ***τουλάχιστον δυο*** (και κατά προτίμηση πιο πολλά) διαφορετικά αρχεία και θα πρέπει *απαραιτήτως να γίνεται χρήση* separate compilation .



Σχήμα 1: Πολλαπλοί επισκέπτες που καταφθάνουν στο μπαρ και εξυπηρετούνται από τον υπεύθυνο αφού πρώτα έχουν εξασφαλίσει την καρέκλα τους.

- Παρακολουθείτε την ιστοσελίδα του μαθήματος `https://www.alexdelis.eu/k22/` για επιπρόσθετες ανακοινώσεις.
- Υπεύθυνοι για την άσκηση αυτή είναι ο Δρ. Σαράντης Πασκαλής `paskalis+AT-di`, ο κ. Αναστάσης Αντωνιάδης `anantoni+AT-di`, και ο κ. Παναγιώτης Χατζημίχος `sdi2000211+AT-di`.

Σύνθεση Ανεξαρτήτων Προγραμμάτων: visitor, receptionist & monitor

Το Σχήμα 1 δείχνει πως τα διάφορα στοιχεία του προβλήματος συνεργάζονται για να επιτύχουν μια συνεργατική λειτουργία για όλες τις εμπλεκόμενες διεργασίες: πελάτες ή επισκέπτες (visitors) και την διαδικασία υπευθύνου (receptionist).

Ο χρόνος άφιξης των επισκεπτών είναι τυχαίος. Καθώς ένας επισκέπτης φτάνει στο μπαρ, πρώτα θα επιχειρήσει να 'πάρει' μια καρέκλα. Η πολιτική του μπαρ είναι ότι μέχρι 4 το πολύ επισκέπτες που φτάνουν ο ένας μετά τον άλλον, καταλαμβάνουν καρέκλες στο ίδιο τραπέζι. Οι επόμενοι 4 στο επόμενο τραπέζι και ούτω καθεξής.

Όταν ένας επισκέπτης πάρει μια καρέκλα, θα 'προχωρήσει' μέχρι το τραπέζι εξυπηρέτησής του μπαρ και θα αγοράσει *(1)* νερό/κρασί (ένα από τα δύο υποχρεωτικά! – μπορεί και τα δύο), *(2)* 100 γρμ. τοπικό τυρί (προαιρετικό) και *(3)* μια σαλάτα (προαιρετική). Το τι τελικά αγοράζεται από τον κάθε επισκέπτη απο τις διαθέσιμες επιλογές στο μενού, είναι τυχαία επιλογή του καθενός. Ο υπεύθυνος μπορεί να εξυπηρετήσει *1-επισκέπτη την φορά* και η εν λόγω διαδικασία παραγγελίας διαρκεί μέχρι ένα μέγιστο αριθμό από δευτερόλεπτα που ορίζονται στην γραμμή εντολής του προγράμματος receptionist.

Όταν ο επισκέπτης πάρει την παραγγελία του, κάθεται στο τραπέζι του και αρχίζει να τρώει/πίνει και να 'συνομιλεί' με τους άλλους το πολύ 3 συνδαιτυμόνες που βρίσκονται στο συγκεκριμένο τραπέζι. Μετά από ένα τυχαίο διάστημα χρόνου με μέγιστο αριθμό από δευτερόλεπτα που ορίζονται στην γραμμή του προγράμματος visitor, ο επισκέπτης αναχωρεί. Όταν μια καρέκλα εκκενωθεί από ένα τραπέζι που έχει ήδη δεχτεί 4 επισκέπτες, ο επόμενος επισκέπτης που περιμένει στην είσοδο του μπαρ δεν μπορεί να 'κάτσει'. Θα πρέπει όλοι οι 4 της παρέας να αναχωρήσουν πριν το τραπέζι γίνει και πάλι διαθέσιμο στην πελατεία.

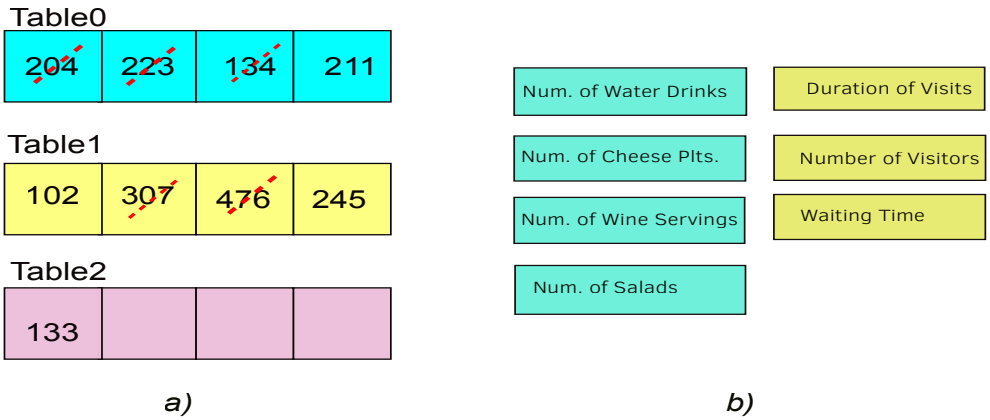
Αν όλα τα τραπέζια είναι γεμάτα με πελάτες, ένας επισκέπτης που φτάνει στο μπαρ, αναμένει μέχρι *ένα* από τα 3 τραπέζια να γίνει διαθέσιμο αφού αναχωρήσουν και οι 4 συνδαιτυμόνες που υπήρχαν εκεί.

Όταν ένας επισκέπτης ετοιμάζεται να αναχωρήσει, θα πρέπει να καταγράψει το χρόνο της επίσκεψής του αλλά και τα προϊόντα που κατανάλωσε ώστε στο τέλος της ημέρας να μπορεί το υπεύθυνος να δημιουργήσει στατιστικά κατανάλωσης και μέσο χρονικό όρο επίσκεψής πελατών.

Καθώς οι πελάτες επισκέπτονται το μπαρ θα πρέπει να συμμορφώνονται με τις παρακάτω συνθήκες συγχρονισμού:

1. Κανένας πελάτης δεν πρέπει να γνωρίσει λιμό.
2. Ο υπεύθυνος εξυπηρετεί ένα πελάτη την φορά.
3. Η εξυπηρέτηση ενός πελάτη από τον υπεύθυνο μπορεί να διαρκέσει για τυχαίο αριθμό από δευτερόλεπτα στο εύρος $[0.50 * t_{order} \; .. \; t_{order}]$.
4. Η διάρκεια που ένας πελάτης μένει στο τραπέζι του είναι μεταξύ $[0.70 * t_{rest} \; .. \; t_{rest}]$ δευτερόλεπτα.
5. Η χρονική διάρκεια που μια διεργασία αναμένεται να 'περιμένει' ή να 'συνδιαλέγεται' με άλλες, υλοποιείται κάνοντας `sleep()`.

Στην υλοποίησή σας θα πρέπει να εισαγάγετε και ένα shared memory segment όπου θα πρέπει να διατηρούνται δομές για την κατάσταση που επικρατεί στα τραπέζια σε οποιαδήποτε χρονική στιγμή αλλά και στατιστικά για την χρήση των προϊόντων που καταναλώνονται, το μέσο όρο που οι επισκέπτες παραμένουν στο μπαρ, αλλά και τον αριθμό των επισκεπτών που υπάρχουν μέχρι στιγμής. Για παράδειγμα, στην εν λόγω περιοχή θα μπορούσε να έχει πίνακες από pids που να δείχνουν τα process-ids όλων των εν ενεργεία επισκεπτών. Εδώ επίσης θα μπορούσαν να υπάρχουν αντικείμενα που βοηθούν στην παραγωγή στατιστικών χρήσης προϊόντων και κίνησης επισκεπτών. Το Σχήμα 2(a) παρουσιάζει την κατάσταση που υπάρχει σε μια χρονική στιγμή οπού το τραπέζι 0 έχει δεχτεί 4 επισκέπτες 3 εκ των οποίων ήδη έχουν αναχωρήσει, το τραπέζι 1 έχει δεχτεί 4 επισκέπτες εκ των οποίων δύο έχουν αποχωρήσει και τέλος το τραπέζι 2 έχει δεχτεί μόνο ένα επισκέπτη με procID: 133. Το



Σχήμα 2: Παράδειγμα shared-memory resident objects που βοηθούν στο να απεικονίσουν την κατάσταση των επισκεπτών στα τραπέζια.

Σχήμα 2(b) παρουσιάζει αντικείμενα που μπορούν να χρησιμοποιηθούν για την παρουσίαση της χρήσης του μπαρ είτε από τον receptionist ή το ανεξάρτητο monitor πρόγραμμα.

Σχεδιασμός Προγραμμάτων:

Έχετε ελευθερία να επιλέξετε οποιαδήποτε δομή επιθυμείτε για τα προγράμματα επισκεπτών/υπεύθυνου/monitor. Θα πρέπει να υιοθετήσετε ένα περιορισμένο αριθμό από σηματοφόρους, βοηθητικές δομές και ίσως άλλων προγραμμάτων αν το κρίνετε απαραίτητο ή χρήσιμο. Για παράδειγμα, θα ήταν καλή ιδέα να αναπτύξετε ένα πρόγραμμα

που δημιουργεί το shared memory segment, αρχικοποιεί δομές και καταστάσεις, και τέλος γνωστοποιεί το ID του κοινού τμήματος στους αναγνώστες/συγγραφείς. Το εν λόγω πρόγραμμα θα μπορούσε να αρχικοποιήσει και τους σηματοφόρους που είναι απαραίτητοι για την λύση χωρίς λιμό που επιθυμούμε.

Μπορείτε να επικαλεστείτε τα προγράμματα από διαφορετικά `ttys` ή αν θέλετε να χρησιμοποιήσετε μια ιεραρχία από `forks()/exec*()` που να δημιουργούν το απαραίτητο συνολικό αποτέλεσμα εκτελέσεως πολλαπλών επισκεπτών. Τα προγράμματά σας θα πρέπει να δημιουργούν εξόδους που εύκολα μπορούν να δείξουν την ορθότητα αλλά και το ταυτόχρονο της εκτέλεσής τους.

Πριν ολοκληρωθεί η εκτέλεση των παραπάνω προγραμμάτων, θα πρέπει να υπολογιστούν και να τυπωθούν τα παρακάτω στατιστικά:

1. Αριθμός των επισκεπτών του μπαρ.
2. Μέσο χρονικό διάστημα στο οποίο οι επισκέπτες παρέμειναν στο μπαρ.
3. Μέσο χρόνο αναμονής που χρειάστηκαν οι επισκέπτες πριν τελικά καταλάβουν μια καρέκλα.
4. Αριθμός των παραγγελιών που έγιναν όσον αφορά στην κατανάλωση ποτών, μερίδων τυριών και σαλάτας από όλους τους επισκέπτες.

Στο τέλος της εκτέλεσης όλων των αναγνωστών/συγγραφέων, επιβάλλεται ένα πρόγραμμα –που μπορεί να είναι ο receptionist– να *καθαρίσει* και να *διαγράψει* το κοινό τμήμα μνήμης και τους σηματοφόρους που χρησιμοποιήθηκαν. Η απελευθέρωση (purging) τέτοιων πόρων είναι επιτακτική. Σε διαφορετική περίπτωση υπάρχει κίνδυνος ο πυρήνας να μην μπορεί να εξυπηρετήσει μέλλουσες ανάγκες σε κοινή μνήμη και σηματοφόρους.

Γραμμή Κλήσης της Εφαρμογής:

Η εφαρμογή μπορεί να κληθεί με τον παρακάτω τρόπο στη γραμμή εντολής (`tty`):

```
./receptionist -d ordertime -s shmid
```

όπου `./receptionist` είναι το εκτελέσιμο του υπευθύνου του μπαρ και μπορεί να έχει τουλάχιστον τις παρακάτω σημαίες:

- `-d ordertime` παρέχει την μέγιστη δυνατή περίοδο στην οποία ο υπεύθυνος χρειάζεται για να εξυπηρετήσει ένα επισκέπτη με την παραγγελία του.
- `-s shmid` δίνει το κλειδί που το κοινό τμήμα μνήμης έχει (και όπου υπάρχουν αντικείμενα κοινού ενδιαφέροντος).

Παρομοίως, οι επισκέπτες μπορούν να κληθούν ως εξής:

```
./visitor -d resttime -s shmid
```

όπου `./visitor` είναι το εκτελέσιμο και οι σχετικές σημαίες είναι:

- `-d resttime` παρέχει την μέγιστη δυνατή περίοδο στην οποία ο επισκέπτης παραμένει στο τραπέζι αφότου έχει εξυπηρετηθεί από τον υπεύθυνο (σε κατάσταση sleep).
- `-s shmid` δίνει το κλειδί που το κοινό τμήμα μνήμης έχει (και όπου υπάρχουν αντικείμενα κοινού ενδιαφέροντος).

Ο `monitor` μπορει να προσπελάσει την κοινή μνήμη και παρέχει το status quo του μπαρ ως εξής:

```
./monitor -s shmid
```

όπου `./monitor` είναι το εκτελέσιμο και ή σχετική σημαία είναι:

- `-s shmid` δίνει το κλειδί που το κοινό τμήμα μνήμης έχει (και όπου υπάρχουν αντικείμενα κοινού ενδιαφέροντος).

Οι παραπάνω in-line παράμετροι (και αντίστοιχες σημαίες) είναι υποχρεωτικές όσον αφορά την κλήση τους στην γραμμή εντολής. Αν κρίνετε απαραίτητο, μπορείτε να εισαγάγετε και επιπλέον σημαίες. Όπως πάντα, οι σημαίες μπορούν να εμφανίζονται με *οποιαδήποτε σειρά*.

Τι πρέπει να Παραδοθεί:

1. Μια σύντομη και περιεκτική εξήγηση για τις επιλογές που έχετε κάνει στο σχεδιασμό του προγράμματος σας (2-3 σελίδες σε ASCII κείμενο είναι αρκετές).
2. Οπωσδήποτε ένα `Makefile` που να μπορεί να χρησιμοποιηθεί για να γίνουν αυτόματα compile τα πρόγραμμα σας.
3. Ένα `zip/7z` αρχείο με όλη σας τη δουλειά σε έναν κατάλογο που πιθανώς να φέρει το όνομα σας και θα περιέχει όλη σας τη δουλειά δηλ. `source files, header files, output files` (αν υπάρχουν) και οτιδήποτε άλλο χρειάζεται.

Βαθμολόγηση Προγραμματιστικής Άσκησης:

Σημεία Αξιολόγησης Άσκησης	*Ποσοστό Βαθμού (0–100)*
Ποιότητα στην Οργάνωση Κώδικα & Modularity	15%
Σωστή Αντιμετώπιση Συνθηκών Συγχρονισμού	20%
Χρήση Μηχανισμού Παρατήρησης & Εκτέλεσης Ταυτόχρονων Διεργασιών	25%
Παραγωγή Εξόδου και Στατιστικών	20%
Χρήση Makefile & Separate Compilation	06%
Ορθή Χρήση Σημαιών σε Γραμμές Εντολών	04%
Καθαρισμός του shared memory segement & semaphores	05%
Επαρκής/Κατανοητός Σχολιασμός Κώδικα	05%

Άλλες Σημαντικές Παρατηρήσεις:

1. Η εργασία είναι **ατομική**.
2. Το πρόγραμμα σας θα πρέπει να τρέχει στα `Linux` συστήματα του τμήματος χρησιμοποιώντας το `bash prompt` αλλιώς **δεν μπορεί να βαθμολογηθεί**.
3. Αν και αναμένεται να συζητήσετε με φίλους και συνεργάτες το πως θα επιχειρήσετε να δώσετε λύση στο πρόβλημα, η αντιγραφή κώδικα (οποιαδήποτε μορφής), είναι κάτι που **δεν επιτρέπεται** και δεν πρέπει να γίνει. Οποιοσδήποτε βρεθεί αναμεμειγμένος σε αντιγραφή κώδικά **απλά παίρνει μηδέν** στο μάθημα. Αυτό ισχύει για **όσους εμπλέκονται ανεξάρτητα** από το ποιός έδωσε/πήρε κλπ.
4. Το παραπάνω επίσης ισχύει αν διαπιστωθεί ***έστω και μερική άγνοια*** του κώδικα που έχετε υποβάλει ή **απλά *υπάρχει υποψία*** ότι ο κώδικας είναι προϊόν συναλλαγής με τρίτο/-α άτομο/α ή με συστήματα αυτόματης παραγωγής λογισμικού ή με αποθηκευτήρια (repositories) οποιασδήποτε μορφής.
5. Αναμένουμε ότι όποια/ος υποβάλει την εν λόγω άσκηση θα πρέπει να έχει πλήρη γνώση και δυνατότητα εξήγησης του κώδικα. Αδυναμία σε αυτό το σημείο οδηγεί σε μηδενισμό στην άσκηση.
6. Σε καμιά περίπτωση τα Windows *δεν είναι επιλέξιμη* πλατφόρμα για την υλοποίηση αυτής της άσκησης.