ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
Τμήμα Πληροφορικής και Τηλεπικοινωνιών
2η Εργασία - Τμήμα: Αρτίων Αριθμών Μητρώου
K22: Λειτουργικά Συστήματα – Χειμερινό Εξάμηνο '23
Ημερομηνία Ανακοίνωσης: Τρίτη 05 Δεκεμβρίου 2023
Ημερομηνία Υποβολής: Πεμπτη 28 Δεκεμβρίου 2023 Ώρα 23:55

Εισαγωγή στην Εργασία:

Ο στόχος αυτής της εργασίας είναι να δημιουργήσετε ανεξάρτητα προγράμματα τα οποία μπορούν να τρέχουν *ταυτόχρονα* και να *διαβάζουν/γράφουν-ενημερώνουν* εγγραφές που βρίσκονται σε ένα αρχείο πελατών. Αυτό είναι το πρόβλημα των *αναγνωστών/συγγραφέων* το οποίο συζητήσαμε στην τάξη και για το οποίο επιθυμούμε μια λύση που είναι ελευθέρα–λιμού. Τα προγράμματα αυτά θα λειτουργούν χρησιμοποιώντας P() και V() σήματα συγχρονισμού καθώς και ένα shared memory segment όπου διατηρείται η κατάσταση των εν εξελίξει διεργασιών και η δημιουργία στατιστικών στοιχείων.

Το πρωτόκολλο που πρέπει να ακολουθηθεί είναι απλό: μπορεί να υπάρχουν ταυτόχρονα ένας ή και πιο πολλοί αναγνώστες που διαβάζουν το περιεχόμενο πολλαπλών εγγραφών από το αρχείο. Κάθε φορά που ένας συγγραφέας εμφανίζεται και επιθυμεί να ενημερώσει μια εγγραφή πελάτη, θα πρέπει να περιμένει ώστε όλοι οι προηγούμενοι αναγνώστες στην εν λόγω εγγραφή να ολοκληρώσουν το διάβασμά τους.

Οι αναγνώστες και οι συγγραφείς ξεκινούν την εργασία τους σε διαφορετικές χρονικές στιγμές και διαρκούν για τυχαίο χρονικό διάστημα. Για παράδειγμα, ένας αναγνώστης μπορεί να 'μείνει' διαβάζοντας εγγραφές πελατών για μέχρι και 100 δευτερόλεπτα ενώ ένας συγγραφέας μπορεί να θέλει να 'μείνει' ενημερώνοντας μια εγγραφή την φορά για μέχρι και 10 δευτερόλεπτα. Τέτοιες 'καθυστερήσεις' είναι επιλογές των αναγνωστών/συγγραφέων και αναφέρονται στην γραμμής κλήσης των προγραμμάτων.

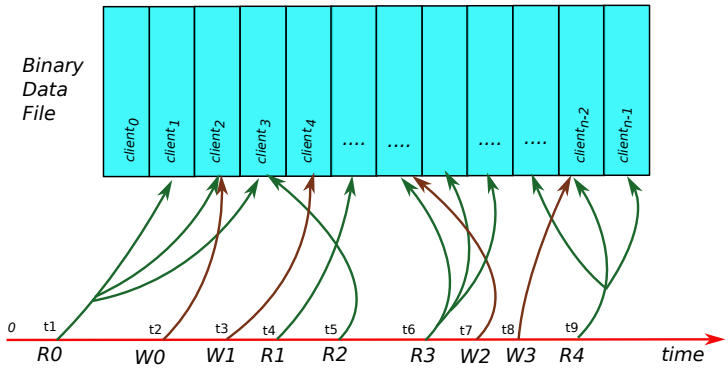
Όπως αναφέραμε και στην τάξη, λύσεις για το πρόβλημα μπορούν να επιφέρουν λιμό για τους συγγραφείς. Ο στόχος της εργασίας είναι να προταθεί και να υλοποιηθεί ένα πρωτόκολλο που δεν εμπλέκει λιμό είτε για τους συγγραφείς είτε για τους αναγνώστες. Συνολικά, σε αυτή την άσκηση θα πρέπει να:

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 τους,
3. παρέχετε τη δυνατότητα τα προγράμματά σας να παραμείνουν 'εργαζόμενα' για περιόδους χρόνου που καθορίζονται από τους χρήστες,
4. έχετε συγγραφείς να μεταβάλουν το περιεχόμενο των εγγραφών αλλάζοντάς το 'τρέχον υπόλοιπο' των πελατών προσθαφαιρώντας σχετικό ποσό ώστε οι αναγνώστες που επέρχονται να μπορούν να ανακτήσουν τη νέα πληροφορία.

Τέλος θα πρέπει να επιδείξετε την ορθότητα της λύσης σας (σκελετοί προγραμμάτων αναγνωστών/συγγραφέων) πιθανόν εκτελώντας προγράμματα από διαφορετικά ttys ή/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 .
- 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`, η κ. Άννα Καββαδά `ankavvada+AT-di`, και ο κ. Άγγελος Πουλής `sdi1900230+AT-di`.



Σχήμα 1: Πολλαπλοί αναγνώστες/συγγραφείς που δουλεύουν ταυτόχρονα με εγγραφές πελατών.

Σύνθεση Ανεξαρτήτων Προγραμμάτων Readers/Writers:

Το Σχήμα 1 δείχνει ένα ένα αριθμό από εγγραφές πελατών που είναι μέρος ενός δυαδικού αρχείου και που οι αναγνώστες/συγγραφείς χρησιμοποιούν στην εργασία τους. Κάθε αναγνώστης μπορεί να διαβάζει 1 ή και πιο πολλές συνεχόμενες εγγραφές για να παράγει το μέσο όρο των υπολοίπων όλων των εγγραφών, ενώ κάθε συγγραφέας μπορεί να προσθέτει ή να αφαιρεί ένα ποσό από το υπόλοιπο που βρίσκεται σε μια εγγραφή την φορά.

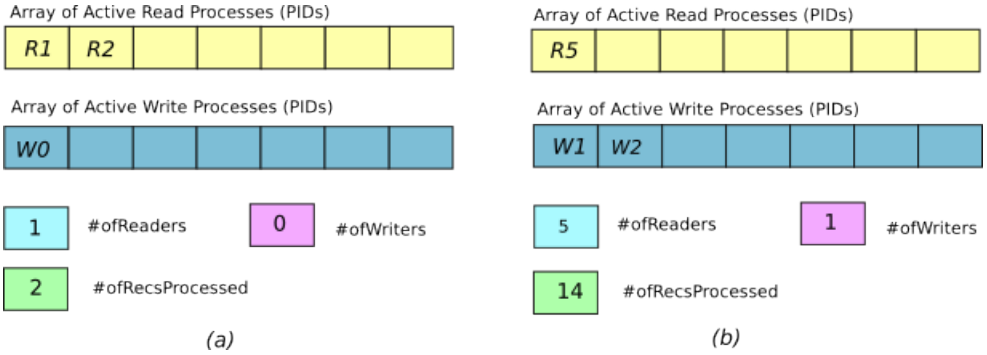
Ο χρόνος άφιξης των αναγνωστών/συγγραφέων είναι τυχαίος και αφού τους επιτραπεί, διαβάζουν/γράφουν τις σχετικές εντολές από το δυαδικό αρχείο δεδομένων και να 'δουλεύουν' με τα παραπάνω στοιχεία για μια τυχαία χρονική περίοδο. Η τιμή της εν λόγου περιόδου προσδιορίζεται στην κλήση του αντίστοιχου προγράμματος.

Για παράδειγμα, το Σχήμα 1 δείχνει ένα γκρουπ από εγγραφές στο κοινά προσπελάσιμο αρχείο . Οι αναγνώστες/συγγραφείς καταφθάνουν σε *διακριτές χρονικές στιγμές* ο καθένας και έτσι έχουμε την παρακάτω χρονολογική σειρά: *R0, W0, W1, R1, R2, R3, W2, W3* και *R4*. Ο *R0* διαβάζει τις εγγραφές 1, 2 και 3 ενώ ο *W0* ενημερώνει την εγγραφή 2, ο *W1* ενημερώνει την εγγραφή, κλπ.

Καθώς οι αναγνώστες/συγγραφείς δουλεύουν με τις εγγραφές πελατών θα πρέπει να συμμορφώνονται με τις παρακάτω συνθήκες συγχρονισμού:

1. Κανένας αναγνώστης ή συγγραφέας δεν πρέπει να γνωρίσει λιμό.
2. Ένας ή περισσότεροι αναγνώστες μπορούν ταυτόχρονα να διαβάζουν τις ίδιες εγγραφές (κανένας συγγραφέας δεν είναι παρών!)
3. Μόνο ένας συγγραφέας τη φορά μπορεί να κάνει αλλαγές στο περιεχόμενο μιας εγγραφής.
4. Μόλις ένας συγγραφέας αρχίσει να δουλεύει με μια εγγραφή, κανένας άλλος δε γίνεται δεκτός. Άλλες διεργασίες θα πρέπει να περιμένουν μέχρι ο παραπάνω συγγραφέας πρώτα να ολοκληρώσει τη συγγραφή του.
5. Ο κάθε συγγραφέας ενημερώνει το 'υπόλοιπο' της εγγραφής που επιλέγει στο αρχείο δεδομένων και αφού 'δουλέψει' πάνω στην εν λόγω εγγραφή για ένα τυχαίο αριθμό από δευτερόλεπτα κάνοντας sleep, τυπώνει την αλλαγμένη εγγραφή στο standard output, ενημερώνει τα στατιστικά, και τερματίζει.
6. Κάθε αναγνώστης διαβάζει το περιεχόμενο των εγγραφών που επιθυμεί και τις τυπώνει στο standard output. Ο αναγνώστης 'εργάζεται' κάνοντας sleep για τυχαίο αριθμό από δευτερόλεπτα, τυπώνει στο standard output το μέσο όρο των υπολοίπων των εγγραφών που έχει διαβάσει, ενημερώνει σχετικά τα στατιστικά της εκτέλεσης, και τερματίζει.

Στην υλοποίησή σας θα πρέπει να εισαγάγετε και ένα shared memory segment όπου θα πρέπει να διατηρούνται δομές και στατιστικά για την χρήση των εγγραφών του αρχείου που κάνουν τα ταυτόχρονα προγράμματα σας. Για παράδειγμα, στην εν λόγω περιοχή θα μπορούσατε να έχετε πίνακες από pids που να δείχνουν τα process-ids όλων των εν ενεργεία αναγνωστών/συγγραφέων σε οποιαδήποτε χρονική στιγμή είναι οι διαδικασίες αυτές ενεργές. Εδώ επίσης θα μπορούσαμε να αποθηκεύσουμε αντικείμενα που εν τέλει μας βοηθούν στην παραγωγή στατιστικών χρήσης των προγραμμάτων μας. Το Σχήμα 2(a) παρουσιάζει όλα τα στοιχεία που αποδίδουν την κατάσταση εργασίας αυτή την στιγμή με τις R1, R2 και W0 υπό εκτέλεση, ένα αναγνώστη να έχει ήδη ολοκληρώσει, και 2 εγγραφές να έχουν χρησιμοποιηθεί. Παρομοίως σε μετέπειτα χρονική στιγμή το Σχήμα 2(b)



Σχήμα 2: Παράδειγμα shared-memory resident objects που βοηθούν στο να απεικονίσουν την κατάσταση εκτέλεσης των ταυτόχρονων αναγνωστών/συγγραφέων

δείχνει ότι υπάρχει 1 αναγνώστης και 2 συγγραφείς σε δράση, με 5 αναγνώστες και 1 συγγραφέα να έχουν ολοκληρώσει την δουλειά τους, και συνολικά 14 εγγραφές να έχουν διαβαστεί/γραφτεί μέχρι στιγμής.

Σχεδιασμός Προγραμμάτων:

Έχετε ελευθερία να επιλέξετε οποιαδήποτε δομή επιθυμείτε για τα προγράμματα αναγνωστών/συγγραφέων. Θα πρέπει να υιοθετήσετε ένα περιορισμένο αριθμό από σηματοφόρους, βοηθητικές δομές και ίσως προγράμματα. 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 αναγνωστών/συγγραφέων. 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 αναγνωστών που δούλεψαν με το αρχείο εγγραφών.
2. Μέσο χρονικό διάστημα στο οποίο οι αναγνώστες δραστηριοποιήθηκαν.
3. Αριθμός των συγγραφέων που ενημέρωσαν εγγραφές στο αρχείο.
4. Μέσος χρονικός όρος δραστηριότητας συγγραφέων.
5. Μέγιστη καθυστέρηση που παρατηρήθηκε για την έναρξη εργασίας είτε από αναγνώστη είτε από συγγραφέα.
6. Συνολικός αριθμός από εγγραφές που είτε διαβάστηκαν είτε ενημερώθηκαν στην διάρκεια εκτέλεσης των πολλαπλών αναγνωστών/συγγραφέων.

Στο τέλος της εκτέλεσης όλων των αναγνωστών/συγγραφέων, επιβάλλεται ένα πρόγραμμα 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 αυστηρό τρόπο στη γραμμή εντολής (`tty`):

```
./reader -f filename -l recid[,recid] -d time -s shmid
```

όπου `./reader` είναι το εκτελέσιμο του αναγνώστη και μπορεί να έχει τουλάχιστον τις παρακάτω σημαίες:

- `-f` δείχνει το όνομα του δυαδικού αρχείου εγγραφών `filename`.
- `-l recid[,recid]` υποδηλώνει είτε μία συγκεκριμένη εγγραφή η έναν αριθμό από συνεχόμενες εγγραφές στο αρχείο δεδομένων `filename` που ο αναγνώστης θέλει να επεξεργαστεί.
- `-d time` παρέχει την μέγιστη δυνατή περίοδο στην οποία ο αναγνώστης θα παραμείνει εργαζόμενος (σε κατάσταση sleep) με τα δεδομένα ανάγνωσης.
- `-s shmid` δίνει το κλειδί που το κοινό τμήμα μνήμης έχει (και όπου υπάρχει την δομή των εγγραφών).

Παρομοίως, οι συγγραφείς μπορούν να κληθούν ως εξής:

```
./write -f filename -l recid -v value -d time -s shmid
```

όπου `./writer` ειναι το εκτελέσιμο των συγγραφέων και οι σχετικές σημαίες είναι:

- `-f` δείχνει το όνομα του δυαδικού αρχείου εγγραφών `filename`.
- `-l recid` παρέχει το ID της εγγραφής που θα ενημερωθεί.
- `-v value` δείχνει το ποσό (θετικό ή αρνητικό) με το οποίο θα πρέπει να μεταβληθεί η υπόλοιπο της εγγραφής στο αρχείων δεδομένων.
- `-d time` παρέχει την μέγιστη δυνατή περίοδο στην οποία ο συγγραφέας θα παραμείνει εργαζόμενος (σε κατάσταση sleep) με την εγγραφή που μεταβάλει.
- `-s shmid` δίνει το κλειδί που το κοινό τμήμα μνήμης έχει (και όπου υπάρχει την δομή των εγγραφών).

Οι παραπάνω in-line παράμετροι (και αντίστοιχες σημαίες) είναι υποχρεωτικές όσον αφορά την κλήση τους στην γραμμή εντολής. Οι σημαίες μπορούν να εμφανίζονται με *οποιαδήποτε σειρά*. Η ακριβής αναμενόμενη *μορφή εξόδου* για το πρόγραμμα δίνεται με *λεπτομέρεια* στη σελίδα του μαθήματος.

Μορφοποίηση Δεδομένων στο Αρχείο Εγγραφών & Δικαιολόγηση Επιλογών:

1. Το δυαδικό αρχείο αποτελείται από εγγραφές που έχουν την εξής μορφοποίηση:
   - Αριθμός Μητρώου Πελάτη `int`–για ευκολία πάντα ξεκιναει απο το 1.
   - Επίθετο Πελάτη `char[20]`.
   - Όνομα Πελάτη `char[20]`.
   - Υπόλοιπο Λογαριασμού `int`.
2. Οποιαδήποτε σχεδιαστική επιλογή που θα υιοθετήσετε, θα πρέπει να τη δικαιολογήσετε στην αναφορά σας.

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 γίνει 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 αλλιώς **δεν μπορεί να βαθμολογηθεί**.
3. Αν και αναμένεται να συζητήσετε με φίλους και συνεργάτες το πως θα επιχειρήσετε να δώσετε λύση στο πρόβλημα, 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 ποιος έδωσε/πήρε κλπ.
4. Το παραπάνω επίσης ισχύει αν διαπιστωθεί ***έστω και μερική άγνοια*** του κώδικα που έχετε υποβάλει ή ***άπλα 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 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.