Κοινή μνήμη στο POSIX

Αρκετοί μηχανισμοί διαδιεργασιακής επικοινωνίας (IPC) είναι διαθέσιμοι για τα συστήματα POSIX, συμπεριλαμβανομένων της κοινής μνήμης και του περάσματος μηνυμάτων. Εδώ, θα διερευνήσουμε το POSIX API για κοινή μνήμη.

Μία διεργασία πρέπει πρώτα να δημιουργήσει ένα τμήμα κοινής μνήμης χρησιμοποιώντας την κλήση συστήματος shmget() (η shmget() προέρχεται από το SHared Memory GET). Το παράδειγμα που ακολουθεί δείχνει την χρήση της shmget():

```
segment_id= shmget(IPC_PRIVATE,size,S_IRUSR|S_IWUSR);
```

Η πρώτη παράμετρος καθορίζει το κλειδί (ή τον αναγνωριστή) του τμήματος (segment) της κοινής μνήμης. Αν έχει την τιμή IPC_PRIVATE, τότε δημιουργείται ένα νέο τμήμα κοινής μνήμης. Η δεύτερη παράμετρος καθορίζει το μέγεθος (σε bytes) του τμήματος της κοινής μνήμης. Τέλος, η τρίτη παράμετρος ορίζει τον τρόπο που θα χρησιμοποιηθεί το τμήμα της κοινής μνήμης – δηλαδή για ανάγνωση, εγγραφή ή και τα δύο. Θέτοντας τον τρόπο πρόσβασης σε S_IRUSR | S_IWUSR, υποδεικνύουμε ότι ο ιδιοκτήτης μπορεί να διαβάσει ή να γράψει στο τμήμα της κοινής μνήμης. Μία επιτυχημένη κλήση της shmget() επιστρέφει έναν ακέραιο αναγνωριστή για το τμήμα της κοινής μνήμης. Άλλες διεργασίες που θέλουν να χρησιμοποιήσουν αυτήν την περιοχή της κοινής μνήμης πρέπει να προσδιορίσουν αυτόν τον αναγνωριστή.

Οι διεργασίες που επιθυμούν να έχουν πρόσβαση στο τμήμα της κοινής μνήμης πρέπει να το επισυνάψουν στο χώρο διευθύνσεών τους, χρησιμοποιώντας την κλήση συστήματος shmat() (SHared Memory ATtach). Η κλήση της shmat() αναμένει επίσης τρεις παραμέτρους. Η πρώτη είναι ο ακέραιος αναγνωριστής του τμήματος της κοινής μνήμης το οποίο θα επισυναφθεί. Η δεύτερη είναι ένας δείκτης σε μία θέση στη μνήμη που υποδεικνύει που θα επισυναφθεί η κοινή μνήμη. Αν περάσουμε μία τιμή NULL, το λειτουργικό σύστημα επιλέγει την τοποθεσία για λογαριασμό του χρήστη. Η τρίτη παράμετρος καθορίζει μέσω μίας ένδειξης (flag) αν η περιοχή της κοινής μνήμης βρίσκεται σε κατάσταση μόνο-ανάγνωσης ή εγγραφής-ανάγνωσης· αν της δώσουμε την τιμή 0, επιτρέπουμε και ανάγνωση και εγγραφή στην κοινή περιοχή.

Η τρίτη παράμετρος ορίζει μία ένδειξη τρόπου πρόσβασης. Αν της θέσουμε την τιμή 1, επιτρέπει στην περιοχή κοινής μνήμης να επισυναφθεί με τρόπο πρόσβασης μόνο-ανάγνωσης· αν της τεθεί η τιμή 0, επιτρέπει και ανάγνωση και εγγραφή στην κοινή περιοχή. Επισυνάπτουμε μία περιοχή κοινής μνήμης με χρήση της shmat() ως εξής:

```
shared_memory=(char *) shmat (id, NULL, 0);
```

Μία επιτυχημένη κλήση της shmat() επιστρέφει ένα δείκτη στην αρχική θέση της μνήμης στην οποία έχει επισυναφθεί η περιοχή κοινής μνήμης.

Μόλις μία περιοχή κοινής μνήμης επισυναφθεί στο χώρο διευθύνσεων μίας διεργασίας, η διεργασία μπορεί να προσπελάσει την κοινή μνήμη σαν μία συνηθισμένη προσπέλαση μνήμης, χρησιμοποιώντας το δείκτη που επιστράφηκε από την shmat(). Σε αυτό το παράδειγμα, η shmat() επιστρέφει ένα δείκτη προς μία συμβολοσειρά χαρακτήρων. Συνεπώς, θα μπορούσαμε να γράψουμε στην κοινή περιοχή μνήμης όπως παρακάτω:

```
sprintf(shared_memory, “Writing to shared memory”);
```

Οι ενημερώσεις στο τμήμα της κοινής μνήμης είναι ορατές από όλες τις διεργασίες που μοιράζονται αυτό το τμήμα.

Τυπικά, μία διεργασία που χρησιμοποιεί ένα υπάρχον τμήμα κοινής μνήμης πρώτα επισυνάπτει την περιοχή κοινής μνήμης στο χώρο διευθύνσεών της και στη συνέχεια προσπελαύνει (και πιθανόν ενημερώνει) την περιοχή κοινής μνήμης. Όταν μια διεργασία δεν χρειάζεται πια πρόσβαση στο τμήμα της κοινής μνήμης, αποσπά το τμήμα αυτό από το χώρο διευθύνσεών της. Για να αποσπάσει μία περιοχή κοινής μνήμης, η διεργασία μπορεί να περάσει το δείκτη της περιοχής κοινής μνήμης στην κλήση συστήματος shmdt(), όπως παρακάτω:

```
shmdt(shared_memory);
```

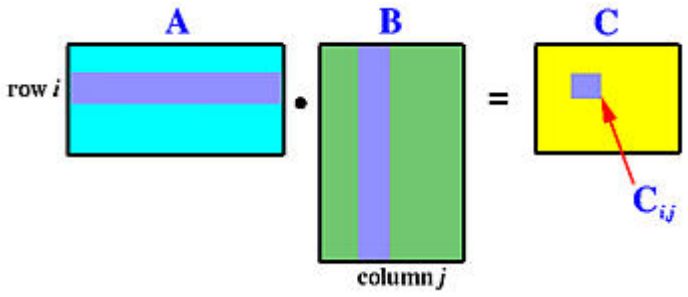
Τέλος, ένα τμήμα κοινής μνήμης μπορεί να αφαιρεθεί από το σύστημα με την κλήση συστήματος shmctl(), στην οποία περνιέται ο αναγνωριστής του κοινού τμήματος μαζί με τη σημαία IPC_RMID.

Άσκηση

Η άσκηση αυτή σας ζητά να υλοποιήσετε ένα διαμοιραζόμενο C πρόγραμμα με κατάλληλη χρήση των IPC shared memory calls. Οι κλήσεις αυτές χρησιμοποιούνται για να επιτρέπουν την κοινή χρήση τμημάτων μνήμης από διαφορετικές διεργασίες. Οι διεργασίες θα δημιουργηθούν με χρήση της fork.

Πολλαπλασιασμός πινάκων με Unix shared memory

Υποθέστε **A** και **B** είναι λ X μ και μ X ν πίνακες. Το προϊόν τους, **C =A *B** είναι ένας λ X ν πίνακας. Η πράξη που θα πρέπει να κάνετε φαίνεται καλύτερα με ένα σχήμα.



Μιας και κάθε C [i , j] υπολογίζεται ανεξάρτητα, μπορούμε να χρησιμοποιήσουμε λ X ν διεργασίες, κάθε μια από τις οποίες υπολογίζει μια θέση του πίνακα **C** .

Γράψτε δύο προγράμματα **main** και **compute**. Το πρώτο δεν παίρνει κανένα argument και το δεύτερο παίρνει δύο command line arguments, Row και Column. Το πρόγραμμα **main** πρέπει να κάνει τα εξής.

1. Να διαβάζει δύο πίνακες **A** και **B** σε μια κοινή μνήμη. Έστω λ Χ μ και μ Χ ν οι διαστάσεις των δύο πινάκων.
2. Να εκτυπώνει τους δύο πίνακες.
3. Κατόπιν, η **main** δημιουργεί *λ* * *ν* διεργασίες, μια για κάθε είσοδο του προϊόντος του πίνακα **C** = **A** * **B**. Κάθε μία από τις διεργασίες αυτές τρέχει το πρόγραμμα **compute** με τα κατάλληλα ορίσματα χρησιμοποιώντας την execvp.
4. Η **main** περιμένει να τελειώσουν όλες οι άλλες διεργασίες και μετά εκτυπώνει τον πίνακα **C**.

Το πρόγραμμα **compute** πρέπει να κάνει τα εξής.

1. Όταν το πρόγραμμα **compute** τρέχει, λαμβάνει έναν αριθμό γραμμής **Row** και ένας αριθμός στήλης **Column** από την γραμμή εντολών.
2. Κατόπιν, εκτελεί τον πολλαπλασιασμό της γραμμής με την στήλη και γράφει το αποτέλεσμα στην κατάλληλη θέση του πίνακα **C**.
3. Μετά από αυτό, πρόγραμμα **compute** τερματίζει.

Το πρόγραμμα main θα λαμβάνει ως είσοδο ένα αρχείο με την εξής μορφή.

l m <---------- # of rows and columns of matrix A

a11 a12 a13 ... a1*m* <--------- row 1 of A

a21 a22 a23 ... a2*m* <--------- row 2 of A

.....

a*l*1 a*l*2 a*l*3 ... a*lm* <--------- row *l* of A

u v <---------- # of rows and columns of matrix B

b11 b12 b13 ... b1*v* <--------- row 1 of B

b21 b22 b23 ... b2*v* <--------- row 2 of B

.....

b*u*1 b*u*2 b*u*3 ... b*uv* <--------- row *u* of B