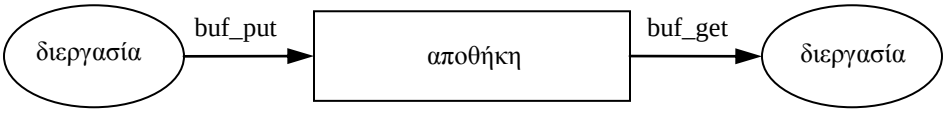
Εργασία 4 – Ενδιάμεση αποθήκη μέσω κοινόχρηστης μνήμης

Υλοποιήστε την επικοινωνία μεταξύ δύο διεργασιών μέσω μιας ενδιάμεσης αποθήκης που παίζει τον ρόλο ενός FIFO (first-in-first-out) καναλιού επικοινωνίας, στο πνεύμα του παρακάτω σχήματος.



Η διαχείριση/πρόσβαση στην αποθήκη θα πρέπει να γίνει μέσα από κατάλληλες συναρτήσεις, στο πνεύμα της παρακάτω διασύνδεσης προγραμματισμού (programming interface):

`int buf_init();`	Αν η αποθήκη δεν υπάρχει, δημιουργείται και επιστρέφεται 1. Αν υπάρχει ήδη, απλά επιστρέφεται 1. Αν προκύψει κάποιο πρόβλημα επιστρέφεται κωδικός λάθους.
`int buf_destroy();`	Καταστροφή της αποθήκης. Για επιτυχία επιστρέφεται 1. Αν προκύψει κάποιο πρόβλημα επιστρέφεται κωδικός λάθους.
`int buf_put(char c);`	Τοποθέτηση στην αποθήκη ενός byte. Για επιτυχία επιστρέφεται 1. Αν η αποθήκη δεν έχει ελεύθερο χώρο επιστρέφεται 0. Αν προκύψει κάποιο άλλο πρόβλημα επιστρέφεται κωδικός λάθους.
`int buf_get(char *c);`	Απομάκρυνση/διάβασμα από την αποθήκη ενός byte. Για επιτυχία επιστρέφεται 1. Αν δεν υπάρχουν διαθέσιμα δεδομένα επιστρέφεται 0. Αν προκύψει κάποιο άλλο πρόβλημα επιστρέφεται κωδικός λάθους

Η αποθήκη πρέπει να υλοποιηθεί χρησιμοποιώντας ένα τμήμα κοινόχρηστης μνήμης το οποίο θα δημιουργείται και θα προσπελαύνεται μέσω των παραπάνω συναρτήσεων. Πιο συγκεκριμένα:

- Η `buf_init` δημιουργεί/εντοπίζει ένα κοινόχρηστο τμήμα μνήμης με βάση μια προσυμφωνημένη τιμή κλειδιού (εσείς αποφασίζετε ποια τιμή θα χρησιμοποιεί η υλοποίηση σας), και το προσαρτά στην διεργασία.
- Η `buf_destroy` αποσυνδέει το κοινόχρηστο τμήμα μνήμης από την διεργασία, και το αποδεσμεύει.
- Η `buf_put` τοποθετεί ένα byte στην πρώτη ελεύθερη θέση της κοινόχρηστης μνήμης.
- Η `buf_get` επιστρέφει το περιεχόμενο της πρώτης κατειλημμένης θέσης της κοινόχρηστης μνήμης, και ελευθερώνει την θέση ώστε να μπορεί να χρησιμοποιηθεί για την αποθήκευση άλλων δεδομένων.

Η λειτουργικότητα FIFO πρέπει να επιτευχθεί με την τεχνική της «κυκλικής» αποθήκης, χρησιμοποιώντας έναν πίνακα από N bytes (την τιμή του N την αποφασίζετε εσείς) και δύο ακέραιες μεταβλητές, έστω in και out, όπου η in «δείχνει» στην πρώτη ελεύθερη θέση και η out «δείχνει» στη πρώτη κατειλημμένη θέση του πίνακα. Αρχικά, in=0 και out=0. Η αποθήκευση γίνεται στην θέση in που αυξάνεται «κυκλικά»: in = (in+1)%N. Αντίστοιχα, η ανάγνωση γίνεται από την θέση out που αυξάνεται «κυκλικά»: out = (out+1)%N. Ο πίνακας είναι άδειος (δεν υπάρχουν δεδομένα για διάβασμα) όταν in = = out, και αντίστοιχα ο πίνακας είναι γεμάτος όταν out = = (in+1)%N. Με άλλα λόγια, ο πίνακας θεωρείται γεμάτος όταν περιέχει N-1 στοιχεία. **Προσοχή: Τόσο ο πίνακας όσο και οι μεταβλητές in και out πρέπει να βρίσκονται στην κοινόχρηστη μνήμη. Για απλότητα, η πρόσβαση στην κοινή μνήμη μπορεί να γίνει μέσω ενός δείκτη σε μια δομή (struct) που περιέχει τις δύο μεταβλητές και τον πίνακα.**

Δοκιμάστε την υλοποίησή σας χρησιμοποιώντας δύο απλά προγράμματα, όπου το ένα τοποθετεί στην αποθήκη τους χαρακτήρες που διαβάζει από την είσοδο του, και το άλλο εκτυπώνει στην έξοδο του τους χαρακτήρες που απομακρύνει από την αποθήκη. Αν η `buf_put`/`buf_get` επιστρέψει 0, η προσπάθεια τοποθέτησης/ανάγνωσης δεδομένων πρέπει να επαναλαμβάνεται. Η εκτέλεση θα πρέπει να τερματίζεται όταν διαβαστεί ο χαρακτήρας 'q'. Δεν επιτρέπεται η άσκοπη χρήση καθολικών μεταβλητών.

Σκεφτείτε κατά πόσο η προτεινόμενη λύση είναι πραγματικά «ασφαλής» υπό ταυτόχρονη εκτέλεση (με τυχαίες εναλλαγές ανάμεσα στην διεργασία που τοποθετεί και την διεργασία που απομακρύνει δεδομένα από την αποθήκη).

Παράδοση: **Παρασκευή 31 Μαΐου 2013, 22:00**
Οδηγίες Παράδοσης: Στην ιστοσελίδα του μαθήματος