Σειρά Εργασιών 4

4.1 Κορουτίνες

Υλοποιήστε υποστήριξη για ταυτόχρονη εκτέλεση κώδικα με κορουτίνες (coroutines) όπου η εναλλαγή γίνεται με ρητό τρόπο. Η υλοποίηση σας πρέπει να παρέχει κατάλληλη διεπαφή προγαμματισμού, π.χ.:

int mycoroutines_init(co_t *main);	Αρχικοποίηση κορουτίνας για το κυρίως πρόγραμμα.
int mycoroutines_create(co_t *co, void (body)(void *), void *arg);	Δημιουργία νέας κορουτίνας για την εκτέλεση της συνάρτησης body με παράμετρο arg.
int mycoroutines_switchto(co_t *co);	Εναλλαγή από την τρέχουσα κορουτίνα, στην κορουτίνα που δίνεται ως παράμετρος.
int mycoroutines_destroy(co_t *co);	Καταστροφή κορουτίνας.

Βασίστε την υλοποίηση σας στις λειτουργίες getcontext(), makecontext(), setcontext(), swapcontext() που παρέχει το λειτουργικό για την δημιουργία και την εναλλαγή ανάμεσα σε ξεχωριστά πλαίσια εκτέλεσης.

Δοκιμάστε την υλοποίηση σας μέσω ενός προγράμματος που δημιουργεί δύο κορουτίνες που μεταφέρουν τα δεδομένα ενός αρχείου μέσω ενδιάμεσης αποθήκης (εργασία 1.1). Η εναλλαγή από την κορουτίνα του παραγωγού προς την κορουτίνα του καταναλωτή πρέπει να γίνεται κάθε φορά που η αποθήκη γεμίζει πλήρως, και η εναλλαγή από την κορουτίνα του καταναλωτή προς την κορουτίνα του παραγωγού πρέπει να γίνεται κάθε φορά που η αποθήκη αδειάζει εντελώς. Όταν ολοκληρωθεί η μεταφορά, ο έλεγχος πρέπει να επιστρέφει στο κυρίως πρόγραμμα, που καταστρέφει τις κορουτίνες, ελέγχει ότι η μεταφορά δεδομένων έγινε επιτυχώς (diff μεταξύ του αρχείου που διάβασε ο παραγωγός και του αρχείου που δημιούργησε ο καταναλωτής και όπου αποθήκευσε τα δεδομένα που έλαβε από τον παραγωγό), και τερματίζει.

4.2 Νήματα με αυτόματη εναλλαγή

Επεκτείνετε την εργασία 4.1, για να υποστηρίξετε την ταυτόχρονη εκτέλεση κώδικα με νήματα όπου η εναλλαγή γίνεται με αυτόματο τρόπο. Η υλοποίηση σας πρέπει να παρέχει κατάλληλη διεπαφή προγραμματισμού, π.χ..:

int mythreads_init();	Αρχικοποίηση περιβάλλοντος εκτέλεσης.
int mythreads_create(thr_t *thr, void (body)(void *), void *arg);	Δημιουργία και επιστροφή νέου νήματος για την εκτέλεση της συνάρτησης body με παράμετρο arg
int mythreads_yield();	Εθελοντική εναλλαγή / παραχώρηση του επεξεργαστή.
int mythreads_join(thr_t *thr);	Αναμονή για τερματισμό νήματος.
int mythreads_destroy(thr_t *thr);	Καταστροφή νήματος.
int mythreads_sem_init(sem_t *s, int val);	Αρχικοποίηση σηματοφόρου.
int mythreads_sem_downt(sem_t *s);	Μείωση σηματοφόρου.
int mythreads_sem_up(sem_t *s);	Αύξηση σηματοφόρου.
int mythreads_sem_destroy(sem_t *s);	Καταστροφή σηματοφόρου.

Η αυτόματη εναλλαγή πρέπει να υλοποιηθεί μέσω ενός περιοδικού alarm/timer με κατάλληλο χειρισμό του αντίστοιχου σήματος. Η επιλογή του επόμενου νήματος προς εκτέλεση πρέπει να γίνεται με την πολιτική round-robin (για όσα νήματα δεν έχουν τερματίσει). Επίσης, υλοποιήστε γενικούς σηματοφόρους, ελέγχοντας κατάλληλα την εναλλαγή έτσι ώστε να αποφεύγονται συνθήκες ανταγωνισμού κατά την εκτέλεση των λειτουργιών down/up.

Δοκιμάστε την υλοποίηση σας μέσω του προγράμματος που αναπτύξατε στην εργασία 2.2, χρησιμοποιώντας τα «δικά σας» νήματα και τους «δικούς σας» σηματοφόρους.

Προεραιτικό: επεκτείνετε την υλοποίηση σας έτσι ώστε η εκτέλεση των «δικών σας» νημάτων να γίνεται πάνω από N νήματα pthreads (το N δίνεται ως όρισμα του περιβάλλοντος εκτέλεσης και δεν εξαρτάται από τον αριθμό των «δικών σας» νημάτων που θα δημιουργήσει η εφαρμογή μέσα του παραπάνω λογισμικού).