Σειρά Εργασιών 4

4.1 Κορουτίνες

Υλοποιήστε υποστήριξη για ταυτόχρονη εκτέλεση κώδικα μέσα από κορουτίνες (coroutines) όπου η εναλλαγή ανάμεσα στα νήματα εκτέλεσης γίνεται με ρητό τρόπο. Δίνεται μια ενδεικτική διεπαφή προγραμματισμού:

int mycoroutines_init(co_t *main);	Αρχικοποίηση κύριας κορουτίνας.
int mycoroutines_create(co_t *co, void (body)(void *), void *arg);	Δημιουργία νέας κορουτίνας.
int mycoroutines_switchto(co_t *co);	Εναλλαγή σε άλλη κορουτίνα.
int mycoroutines_destroy(co_t *co);	Καταστροφή κορουτίνας.

Βασίστε την υλοποίηση σας στις λειτουργίες getcontext(), makecontext(), setcontext(), swapcontext() που παρέχει το λειτουργικό για τη δημιουργία και την εναλλαγή ανάμεσα σε ξεχωριστά λεγόμενα «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 των δεδομένων, ο έλεγχος πρέπει να επιστρέφει στο κυρίως πρόγραμμα, που καταστρέφει τις κορουτίνες, ελέγχει ότι η μεταφορά δεδομένων έγινε επιτυχώς (diff μεταξύ του αρχείου και του αντιγράφου που δημιουργήθηκε μέσω της παραπάνω μεταφοράς), και τερματίζει.

4.2 Νήματα με αυτόματη εναλλαγή

Επεκτείνετε/αλλάξτε την παραπάνω υλοποίηση, για να υποστηρίξετε την ταυτόχρονη εκτέλεση κώδικα μέσα από δικά σας νήματα (user-level threads), όπου η εναλλαγή γίνεται με αυτόματο τρόπο. Ενδεικτική διεπαφή προγραμματισμού:

int mythreads_init();	Αρχικοποίηση περιβάλλοντος.
int mythreads_create(mythr_t *thr, void (body)(void *), void *arg);	Δημιουργία νέου νήματος.
int mythreads_yield();	Εθελοντική εναλλαγή.
int mythreads_sleep(int secs);	Αναμονή για πάροδο χρονικού διαστήματος.
int mythreads_join(mythr_t *thr);	Αναμονή για τερματισμό νήματος.
int mythreads_destroy(mythr_t *thr);	Καταστροφή νήματος.
int mythreads_sem_create(mysem_t *s, int val);	Δημιουργία και αρχικοποίηση σηματοφόρου
int mythreads_sem_down(mysem_t *s);	Μείωση σηματοφόρου
int mythreads_sem_up(mysem_t *s);	Αύξηση σηματοφόρου
int mythreads_sem_destroy(mysem_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 τον συγχρονισμό μεταξύ των νημάτων, υλοποιήστε τις βασικές λειτουργίες δυαδικών σηματοφόρων. Όλες οι λειτουργίες (πλην της init) θα πρέπει να είναι ασφαλείς υπό ταυτόχρονη εκτέλεση, ελέγχοντας κατάλληλα την εναλλαγή (χειρισμό του alarm/timer) έτσι ώστε να αποφεύγονται συνθήκες ανταγωνισμού.

4.3 Αναγνώστες/εγγραφείς

Δοκιμάστε την παραπάνω υλοποίηση αναπτύσσοντας μια λύση για το πρόβλημα των αναγνωστών/εγγραφέων (readers/writers) χρησιμοποιώντας τα δικά σας νήματα και δυαδικούς σηματοφόρους, μαζί με ένα κατάλληλο πρόγραμμα προσομοίωσης στο πνεύμα των εργασιών 2.3/2.4.

Προαιρετικά: Επεκτείνετε την υλοποίηση σας έτσι ώστε η εκτέλεση των δικών σας νημάτων να γίνεται πάνω από N νήματα συστήματος pthreads (το N δίνεται ως όρισμα του περιβάλλοντος εκτέλεσης). Η πολυπλεξία μπορεί να γίνεται στατικά με βάση μια αρχική ανάθεση ενός νήματος εφαρμογής (κατά την δημιουργία του) σε ένα νήμα συστήματος ή δυναμικά με στόχο να υπάρχει μια ισορροπημένη κατανομή των νημάτων εφαρμογής στα νήματα συστήματος.

Παράδοση: Σάββατο 21 Δεκεμβρίου 2024, 23:59