

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 στην εφαρμογή κατάλληλη διεπαφή προγραμματισμού, π.χ.:

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 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 με (δικά σας) νήματα, όπου η εναλλαγή γίνεται με αυτόματο τρόπο. Η υλοποίηση σας πρέπει να παρέχει κατάλληλη διεπαφή, π.χ.:

int mythreads init();	Αρχικοποίηση περιβάλλοντος.
int mythreads create(thr t *thr, void (body)(void *), void *arg);	Δημιουργία νέου νήματος.
int mythreads yield();	Εθελοντική εναλλαγή.
int mythreads join(thr t *thr);	Αναμονή για τερματισμό νήματος.
int mythreads destroy(thr t *thr);	Καταστροφή νήματος.
int mythreads sem create(sem t *s, int val);	Δημιουργία και αρχικοποίηση σηματοφόρου
int mythreads sem down(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 τον συγχρονισμό μεταξύ των νημάτων, υλοποιήστε τις βασικές λειτουργίες δυναμικών 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/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 υλοποίηση σας θα πρέπει να κατανέμει την εκτέλεση των νημάτων εφαρμογής ανάμεσα στα νήματα του συστήματος (είτε «στατικά» με βάση μια αρχική ανάθεση ενός νήματος εφαρμογής όταν αυτό δημιουργείται σε ένα νήμα συστήματος, είτε «δυναμικά» με στόχο ανά πάσα στιγμή να υπάρχει μια ισορροπημένη κατανομή των νημάτων εφαρμογής στα νήματα συστήματος).

Παράδοση: Σάββατο 27 Ιανουαρίου 2024, 23:59