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 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, 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_tuple_out(char *fmt, …);	Προσθήκη πλειάδας.
int mythreads_tuple_in(char *fmt, …);	Απομάκρυνση πλειάδας.

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 ενός χώρου πλειάδων (tuple space), tuple_out() και tuple_in() για προσθήκη και απομάκρυνση μιας πλειάδας, αντίστοιχα (οι παράμετροι είναι στο πνεύμα της printf() και scanf(), αντίστοιχα). Οι λειτουργίες θα πρέπει να είναι ασφαλείς υπό ταυτόχρονη εκτέλεση, ελέγχοντας κατάλληλα την εναλλαγή (χειρισμό του alarm/timer) έτσι ώστε να αποφεύγονται συνθήκες ανταγωνισμού. Η αρχικοποίηση του tuple space γίνεται ως μέρος της γενικής διαδικασίας αρχικοποίησης mythreads_init().

4.3 Στενή γέφυρα (με tuple space)

Δοκιμάστε την παραπάνω υλοποίηση, αναπτύσσοντας μια λύση στο πρόβλημα της ρύθμισης κυκλοφορίας πάνω από μια στενή γέφυρα, που λύσατε στις εργασίες 2.3 και 3.2 (με σηματοφόρους και ελεγκτή, αντίστοιχα) χρησιμοποιώντας αυτή τη φορά τα «δικά σας» νήματα και τις λειτουργίες tuple space που υποστηρίζει η υλοποίηση σας.

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 σταθερό όρισμα του περιβάλλοντος εκτέλεσης). Η υλοποίηση σας θα πρέπει ιδανικά να κατανέμει τα νήματα της εφαρμογής που υφίστανται ανά πάσα στιγμή ανάμεσα στα N νήματα του συστήματος έτσι ώστε κάθε νήμα συστήματος να είναι υπεύθυνο για παρόμοιο αριθμό των νημάτων εφαρμογής.

Παράδοση: Σάββατο 21 Ιανουαρίου 2022, 23:59