Παράλληλη Επεξεργασία

Φροντιστήριο:

- *Νήματα & Συγχρονισμός*
- *Μεταβλητές κλειδιά – Φράγματα*

Εργαστήριο Πληροφοριακών Συστημάτων Υψηλής Επίδοσης

Parallel and Distributed Systems Group

Συγχρονισμός μεταξύ νημάτων

Πολύ συχνά, στις διάφορες παράλληλες εφαρμογές δημιουργείται η ανάγκη για συγχρονισμό μεταξύ των διαφόρων νημάτων

- Για την προσπέλαση από τα διάφορα νήματα κάποιας κοινής μεταβλητής
- Για να ειδοποιήσει κάποιο νήμα κάποιο άλλο όταν συμβεί κάποιο γεγονός
  - Νήματα που υλοποιούν ένα pipeline
  - Νήματα που δουλεύουν με το μοντέλο producer –consumer
- Όταν τα νήματα πρέπει να λειτουργήσουν σαν μια ομάδα
  - Όταν όλα τα νήματα πρέπει να ξεκινήσουν ταυτόχρονα να εκτελέσουν μια εργασία (π.χ. παράλληλο βρόχο)

Προσπέλαση κοινής μεταβλητής χωρίς προστασία

	Νήμα 1	Νήμα 2	A	B
T0	Διάβασε B=10		A=100	B=10
T1		Διάβασε B=10	A=100	B=10
T2		Διάβασε A=100	A=100	B=10
T3	Διάβασε A=100		A=100	B=10
T4	Πρόσθεσε A=A+B		A=110	B=10
T5		Πρόσθεσε A=A+B	A=110	B=10

Αποτέλεσμα A=110

Συμπέρασμα: Απαιτείται συγχρονισμός για την προσπέλαση στην κοινή μεταβλητή!

Αμοιβαίος Αποκλεισμός

- Όταν δύο νήματα ανανεώνουν την ίδια μεταβλητή θα πρέπει να συγχρονίζονται κατά την διαδικασία εγγραφής
- Αυτό ονομάζεται αμοιβαίος αποκλεισμός
- Ο αμοιβαίος αποκλεισμός επιτυγχάνεται με την χρήση κλειδιών
- Ένα κλειδί μπορεί να είναι κλειδωμένο ή ελεύθερο
- Ένα νήμα πριν ανανεώσει μια κοινή μεταβλητή θα πρέπει να κλειδώσει ένα κλειδί, το οποίο θα ελευθερώσει αμέσως μετά την εγγραφή

Προσπέλαση κοινής μεταβλητής με χρήση κλειδιού

	Νήμα 1	Νήμα 2	A	B
T0	Διάβασε B=10		A=100	B=10
T1		Διάβασε B=10	A=100	B=10
T2		Πάρε το κλειδί	A=100	B=10
T3	Πάρε το κλειδί	Διάβασε A=100	A=100	B=10
T4	Περίμενε το κλειδί	Πρόσθεσε A=A+B	A=110	B=10
T5	Περίμενε το κλειδί	Άσε το κλειδί	A=110	B=10
T6	Διάβασε A=110		A=110	B=10
T7	Πρόσθεσε A=A+B		A=120	B=10
T8	Άσε το κλειδί		A=120	B=10

Με αμοιβαίο αποκλεισμό μέσω κλειδιού επιτυγχάνεται συγχρονισμός μεταξύ των νημάτων και η κοινή μεταβλητή A έχει την σωστή τιμή A=120

Μεταβλητές κλειδιά – Δημιουργία

- Το πρότυπο POSIX παρέχει τον τύπο *pthread_mutex_t*, οποίος ορίζει μια μεταβλητή τύπου κλειδί
- Οι μεταβλητές αυτού του τύπου θα πρέπει να αρχικοποιηθούν πριν χρησιμοποιηθούν:
  - Στατική αρχικοποίηση:
    pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
  - Δυναμική αρχικοποίηση χρησιμοποιώντας την κλήση:
    int pthread_mutex_init(&mutex, NULL);
  - Ένα αρχικοποιημένο κλειδί αρχικά είναι στην κατάσταση ελεύθερο.

Μεταβλητές κλειδιά – Κλείδωμα

- Ένα νήμα ελέγχου μπορεί να κλειδώσει μια μεταβλητή κλειδί χρησιμοποιώντας την κλήση:

  pthread pthread_mutex_lock (&mutex);

- Μόλις η συνάρτηση αυτή επιστρέψει το κλειδί θα είναι σε κατάσταση “κλειδωμένο”

- Αν το κλειδί είναι ήδη κλειδωμένο τότε το νήμα που καλεί την παραπάνω κλήση μπλοκάρετε μέχρι το κλειδί να ξαναγίνει “ελεύθερο”

- Όταν πολλά νήματα καλούν ταυτόχρονα την κλήση αυτή, τότε μόνο ένα από αυτά θα καταφέρει να κλειδώσει το κλειδί

Μεταβλητές κλειδιά-Απελευθέρωση

- Ένα νήμα ελέγχου μπορεί να απελευθερώσει μια μεταβλητή κλειδί χρησιμοποιώντας την κλήση:

  pthread_mutex_unlock (mutex);

- Μόλις η συνάρτηση αυτή επιστρέψει το κλειδί θα είναι σε κατάσταση "ελεύθερο"

- Μόλις το νήμα που έχει κλειδώσει το κλειδί καλέσει την παραπάνω συνάρτηση τότε ένα από τα υπόλοιπα νήματα που έχουν μπλοκάρει (περιμένοντας το κλειδί) ξεμπλοκάρεται

- Και πάλι ένα μόνο από τα μπλοκαρισμένα νήματα θα καταφέρει να ξανακλειδώσει το κλειδί

Μεταβλητές κλειδιά- Κλείδωμα με έλεγχο

- Ένα νήμα ελέγχου μπορεί να δοκιμάσει να κλειδώσει μια μεταβλητή κλειδιού χρησιμοποιώντας την κλήση:

pthread_mutex_trylock (&mutex);

- Αν το κλειδί είναι σε κατάσταση ελεύθερο τότε το νήμα που έκανε την κλήση θα κλειδώσει το κλειδί

- Αν το κλειδί είναι σε κατάσταση κλειδωμένο τότε η κλήση αυτή επιστρέφει έναν ειδικό κωδικό που ενημερώνει τον χρήστη για την κατάσταση

Μεταβλητές κλειδιά – Καταστροφή

- Για κάθε κλήση *pthread_mutex_lock* που υπάρχει στο πρόγραμμα μας, θα πρέπει να υπάρχει και η αντίστοιχη κλήση *pthread_mutex_unlock* για την ίδια μεταβλητή κλειδί
- Αλλιώς το πρόγραμμα μας μπορεί να ανασταλεί μόνιμα (deadlock)
- Όταν μια μεταβλητή κλειδί δεν χρησιμοποιείται πια, τότε αυτή μπορεί να καταστραφεί με την κλήση:

pthread_mutex_destroy(&mutex);

Μεταβλητές κλειδιά - Παράδειγμα

```
...
pthread_mutex_t mutex =
   PTHREAD_MUTEX_INITIALIZER;
int Global;
...

void work_function(void * arg);
{
        ...

        pthread_mutex_lock(&mutex);
        ...
        // Read/Write Access Global
        // ("Critical Section")
        ...
        pthread_mutex_unlock(&mutex);

        ...
}
```

Συγχρονισμός με χρήση Φράγματος

- Συγχρονισμός (κυριολεκτικά) των νημάτων σε συγκεκριμένα σημεία του προγράμματος
- Κατά τον ορισμό ενός φράγματος ορίζεται ο αριθμός των νημάτων που πρόκειται να το χρησιμοποιήσουν
- Όποτε κάποιο νήμα συναντήσει ένα φράγμα αναστέλλει την εκτέλεση του έως ότου να συναντήσουν το συγκεκριμένο φράγμα όλα τα υπόλοιπα νήματα που έχει οριστεί να το χρησιμοποιήσουν (πλήθος νημάτων)
- Όταν το τελευταίο νήμα συναντήσει το φράγμα, ειδοποιούνται και όλα τα υπόλοιπα να συνεχίσουν την εκτέλεση τους.
- Με αυτό τον τρόπο όλα τα νήματα συγχρονίζονται με το “πιο αργό” νήμα
- Απλός και σε αρκετούς αλγορίθμους απαραίτητος τρόπος συγχρονισμού
- Η επίδραση του είναι μικρή όταν το φορτίο είναι ισοκατανεμημένο (the load is balanced)

Προσπέλαση κοινής μεταβλητής με χρήση κλειδιού

	Νήμα 1	Νήμα 2	Γύρος
T0	Εκτέλεση Υπολογισμού	Εκτέλεση Υπολογισμού	N
T1	Συνάντηση φράγματος	Εκτέλεση Υπολογισμού	N
T2	Αναμονή στο φράγμα	Εκτέλεση Υπολογισμού	N
T3	Αναμονή στο φράγμα	Εκτέλεση Υπολογισμού	N
T4	Αναμονή στο φράγμα	Συνάντηση φράγματος	-
T5	Έξοδος από το φράγμα	Έξοδος από το φράγμα	-
T6	Εκτέλεση Υπολογισμού	Εκτέλεση Υπολογισμού	N+1
T7			

Το ταχύτερο σε αυτή τη συγκυρία νήμα 1 συγχρονίζεται με το αργότερο νήμα 2.

Παράδειγμα χρήσης: όταν δεδομένα που παράγονται στο γύρο N είναι απαραίτητα για τον υπολογισμό κατά τον γύρο N+1 κ.ο.κ.

Υλοποίηση Φραγμάτων στα PThreads

- Αρχικά δεν προβλεπόταν η υποστήριξη τους
- Ορίστηκαν σε μεταγενέστερη έκδοση του προτύπου
- Η υλοποίηση τους στα διάφορα λειτουργικά συστήματα αυτή τη στιγμή είναι προαιρετική
- Ωστόσο στο linux διατίθεται πλέον υλοποίηση τους.
- Οι κυριότερες συναρτήσεις για χρήση φραγμάτων είναι:

Τύπος: pthread_barrier_t bar;
int pthread_barrier_init(&bar, NULL, numOfThreads);
int pthread_barrier_wait&bar);
int pthread_barrier_destroy(&bar);

- *Απορίες και συζήτηση μέσω του forum του μαθήματος στο My.CEID*
- *e-mail επικοινωνίας kik@hpclab.ceid.upatras.gr*
- *Ανακοίνωση ασκήσεων και υλικό στην ιστοσελίδα του μαθήματος http://parallel.hpclab.ceid.upatras.gr/*