


Παράλληλα Συστήματα

Χειμερινό εξάμηνο 2024–2025

#2

Διαταγές υποστήριξης συγχρονισμού

- ▶ Αμοιβαίος αποκλεισμός (mutual exclusion)
 - ▶ Συγχρονισμός φράγματος (barrier)
 - ~~▶ Υπό συνθήκη αναμονή (condition synchronization)~~
 - ▶ Μηχανισμοί κλειδώματος
- 

Διαταγή 'critical'

- ▶ Βασική διαταγή αμοιβαίου αποκλεισμού
- ▶ `#pragma omp critical [(name)] new-line structured-block`

Εκτέλεση $x = x + 1$

Νήμα 1

load x

add 1

store x

Νήμα 2

load x

add 1

store x

Παράδειγμα omp6.c

- Παράλληλο πρόγραμμα σε OpenMP στο οποίο εκτελείται η πράξη $x = x + 1$ επαναληπτικά
 - Αποτέλεσμα;
 - Προστασία;

Παράδειγμα omp7.c

- Προστατευμένος ο κώδικας αύξησης της κοινής μεταβλητής
- Πως;
 - Μέσω διαταγής critical

Δήλωση 'critical' (2)

- ▶ Στη δήλωση critical, μπορούμε προαιρετικά να εισάγουμε ένα όνομα το οποίο θα αντιστοιχεί στην κρίσιμη περιοχή την οποία προστατεύουμε.
 - Δηλώσεις critical με το ίδιο όνομα αντιμετωπίζονται από το σύστημα ως η ίδια κρίσιμη περιοχή
 - Δηλώσεις με διαφορετικό όνομα αντιμετωπίζονται ως διαφορετικές.
- ▶ Όλες οι δηλώσεις critical στις οποίες δεν έχει δοθεί όνομα αντιμετωπίζονται ως μία (η ίδια) κρίσιμη περιοχή.

Διαταγή 'atomic'

- ▶ Πιο απλός και γρήγορος μηχανισμός προστασίας κρίσιμων περιοχών
- ▶ `#pragma omp atomic new-line expression-stmt`
- ▶ Το `expression-stmt` είναι μία έκφραση η οποία μπορεί να έχει μία από τις ακόλουθες μορφές:
 - `x <op> = expr` , `x ++` , `++ x` , `x --` , `-- x`
 - όπου x μία κοινή μεταβλητή και <op> ένας από τους ακόλουθους τελεστές: `+`, `*`, `-`, `/`, `&`, `^`, `|`, `<<`, `>>`

Παράδειγμα omp8.c

- Ίδιο πρόβλημα με omp6.c και omp7.c
- Λύση μέσω `atomic` αντί για `critical`

Μηχανισμοί κλειδώματος

- ▶ Το OpenMP παρέχει έναν γενικότερο μηχανισμό κλειδώματος
- ▶ Βασικές συναρτήσεις:
 - `void omp_init_lock(omp_lock_t *lock);`
 - `void omp_set_lock(omp_lock_t *lock);`
 - `void omp_unset_lock(omp_lock_t *lock);`

Παράδειγμα omp9.c

- Ίδιο πρόβλημα με omp6.c, omp7.c και omp8.c
- omp_set_lock και omp_set_unlock
 - Αντί για critical και atomic

Μηχανισμοί κλειδώματος (2)

- ▶ Επιπρόσθετα κλειδώματα;
- ▶ Επιπλέον συναρτήσεις:
 - `void omp_destroy_lock(omp_lock_t *lock);`
 - `int omp_test_lock(omp_lock_t *lock);`

Διαταγή 'barrier'

- ▶ Αυτοματοποιημένη υλοποίηση καθολικής αναμονής τύπου φράγματος του συνόλου των νημάτων που έχουν αρχικοποιηθεί σε μία παράλληλη περιοχή
- ▶ `#pragma omp barrier new-line`

Παραδείγματα 'barrier'

```
#pragma omp parallel
{
    Do_Job1();
    #pragma omp barrier
    #pragma omp for schedule(static)
    {
        Do_Job2();
    }
    Do_Job3();
}
```

▶ bar1.c

▶ bar2.c

Διαταγή 'master'

- ▶ `#pragma omp master new-line
structured-block`
- ▶ `tid = omp_get_thread_num();
if (tid==0)
structured-block`
- ▶ Μοιάζει με την διαταγή `single`
- ▶ Διαφορές από τη `single`;

Παράδειγμα master.c

```
int main(int argc, char *argv[])
{
    #pragma omp parallel
    {
        #pragma omp master
        {
            printf("Hello from %d\n", omp_get_thread_num());
        }
        #pragma omp single
        {
            printf("This is %d\n", omp_get_thread_num());
        }
        printf("Test %d\n", omp_get_thread_num());
    }
    return 0;
}
```


Διαταγή 'flush'

- ▶ Συνεπής εικόνα συγκεκριμένων κοινών αντικειμένων στη μνήμη
- ▶ `#pragma omp flush [list ...] new-line`
- ▶ Η 'flush' υπονοείται σε ορισμένες περιπτώσεις


Παράδειγμα (!!!)

- ▶ Thread 0

```
a = test()  
flag = 1;
```

- ▶ Thread 1

```
while (!flag);  
b = a;
```



Παράδειγμα

- ▶ Thread 0

a=test();

#pragma omp flush

flag=1;

#pragma omp flush

- ▶ Thread 1


#pragma omp flush

while (!flag) { pragma omp flush }

#pragma omp flush

b = a;

Ορισμός διακριτών εργασιών (tasks)

- ▶ Ελλείψεις σε εκδόσεις OpenMP ≤ 2.5
 - ▶ Αυστηρά δομημένη διαδικασία παραλληλοποίησης
 - ▶ Δυνατότητα δυναμικού ορισμού διακριτών εργασιών (tasks)
- 

Ορισμός διακριτών εργασιών (tasks)

- ▶ `#pragma omp task [clause ...] new-line
structured-block`
 - ▶ Στη θέση του clause ορισμένες παράμετροι, όπως και σε άλλες διαταγές
- 