



Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχ. και Μηχανικών Υπολογιστών
Εργαστήριο Υπολογιστικών Συστημάτων

3^η Εργαστηριακή Άσκηση:

Συγχρονισμός

Λειτουργικά Συστήματα Υπολογιστών
6ο Εξάμηνο, 2022-2023

Σύνοψη

- ◆ Δύο προβλήματα συγχρονισμού
- ◆ Χρήση νημάτων: Υλοποιήσεις με POSIX Threads
- ◆ Μηχανισμοί συγχρονισμού:
 - ➔ Συγχρονισμός Διεργασιών/Νημάτων (Process Synchronization)
 - POSIX Mutexes και Spinlocks
 - POSIX Semaphores
 - POSIX Condition Variables
 - ➔ Συγχρονισμός σε Κοινά Δεδομένα (Data Synchronization)
 - GCC atomic operations
 - ➔ Ο συγχρονισμός διεργασιών υλοποιείται βασίζεται στο συγχρονισμό σε κοινά δεδομένα και συχνά περιλαμβάνει τη συνδρομή του Λειτουργικού Συστήματος.

Σύνοψη

- ◆ Z1: Συγχρονισμός σε υπάρχοντα κώδικα (κρίσιμο τμήμα)
 - ➔ `simplesync.c`
 - ➔ Με POSIX mutexes (ή spinlocks) και GCC atomic ops
- ◆ Z2: Παραλληλοποίηση υπάρχοντα κώδικα (ανάγκη σειριοποίησης)
 - ➔ Συγχρονισμός νημάτων για παράλληλο υπολογισμό
 - ➔ Με POSIX semaphores και conditional vars

Σύνοψη



- ◆ Δύο προβλήματα συγχρονισμού
- ◆ Χρήση νημάτων: Υλοποιήσεις με POSIX Threads
- ◆ Μηχανισμοί συγχρονισμού:
 - ➔ Συγχρονισμός Διεργασιών/Νημάτων (Process Synchronization)
 - POSIX Mutexes και Spinlocks
 - POSIX Semaphores
 - POSIX Condition Variables
 - ➔ Συγχρονισμός σε Κοινά Δεδομένα (Data Synchronization)
 - GCC atomic operations

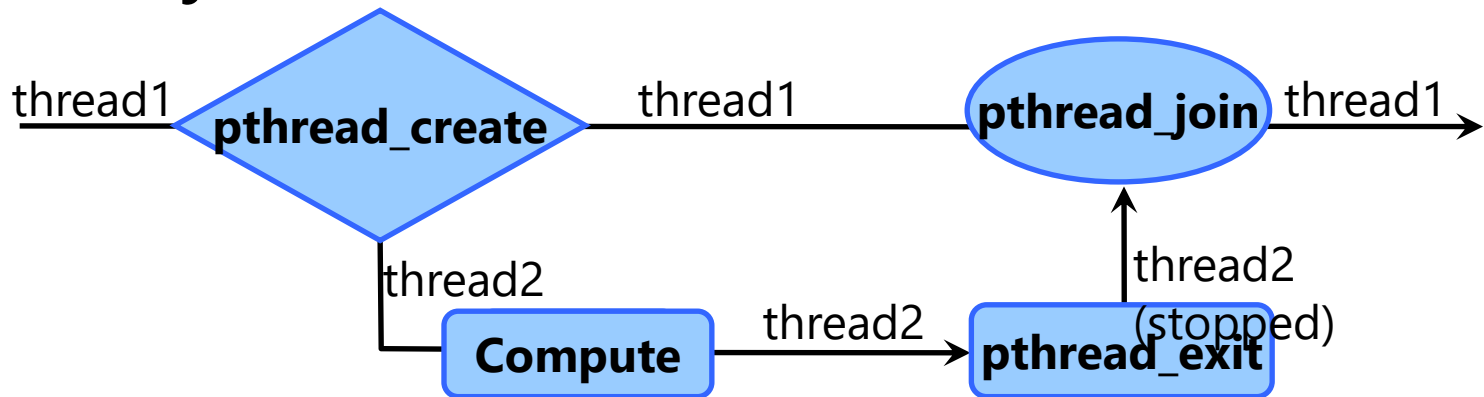
Δημιουργία νημάτων στα POSIX Threads

◆ Δημιουργία με **pthread_create()**

➔ `int pthread_create(pthread_t * thread, pthread_attr_t * attr, void * (*start_routine)(void *), void * arg);`

➔ π.χ. `pthread_create(&tid, &attr, thread_fn, arg)`

◆ Αναμονή για τερματισμό (**pthread_exit()**) με **pthread_join()**



Σύνοψη

- ◆ Δύο προβλήματα συγχρονισμού
- ◆ Χρήση νημάτων: Υλοποιήσεις με POSIX Threads
- ◆ Μηχανισμοί συγχρονισμού:
 - ➔ Συγχρονισμός Διεργασιών/Νημάτων (Process Synchronization)
 - POSIX Mutexes και Spinlocks
 - POSIX Semaphores
 - POSIX Condition Variables
 - ➔ Συγχρονισμός σε Κοινά Δεδομένα (Data Synchronization)
 - GCC atomic operations

Μηχανισμοί (POSIX)

- ◆ POSIX Threads <pthread.h>
 - ➔ pthread_create(), pthread_join(), pthread_exit()
- ◆ POSIX Mutexes <pthread.h>
 - ➔ pthread_mutex_init(), pthread_mutex_lock(), pthread_mutex_unlock()
- ◆ POSIX Spinlocks <pthread.h>
 - ➔ pthread_spin_init(), pthread_spin_lock(), pthread_spin_unlock()
- ◆ POSIX (unnamed) Semaphores <semaphore.h>
 - ➔ sem_overview(), sem_init(), sem_post(), sem_wait(), Manpages: Sections 7, 3.
- ◆ POSIX condition variables:
 - ➔ pthread_cond_init(), pthread_cond_wait(), pthread_cond_signal(), pthread_cond_broadcast()
- ◆ **Εγκαταστήστε** τα πακέτα manpages-posix, manpages-posix-dev δίνοντας
(sudo) apt-get install:

man -a sem post

Μηχανισμοί (GCC atomic operations)

- ◆ GCC atomic operations
 - ➔ <http://gcc.gnu.org/onlinedocs/gcc-4.1.2/gcc/Atomic-Builtins.html>
- ◆ Ειδικές εντολές (builtins) / συναρτήσεις για **ατομική εκτέλεση** σύνθετων εντολών
- ◆ **`__sync_add_and_fetch()`, `__sync_sub_and_fetch()`, ...**

Σύνοψη

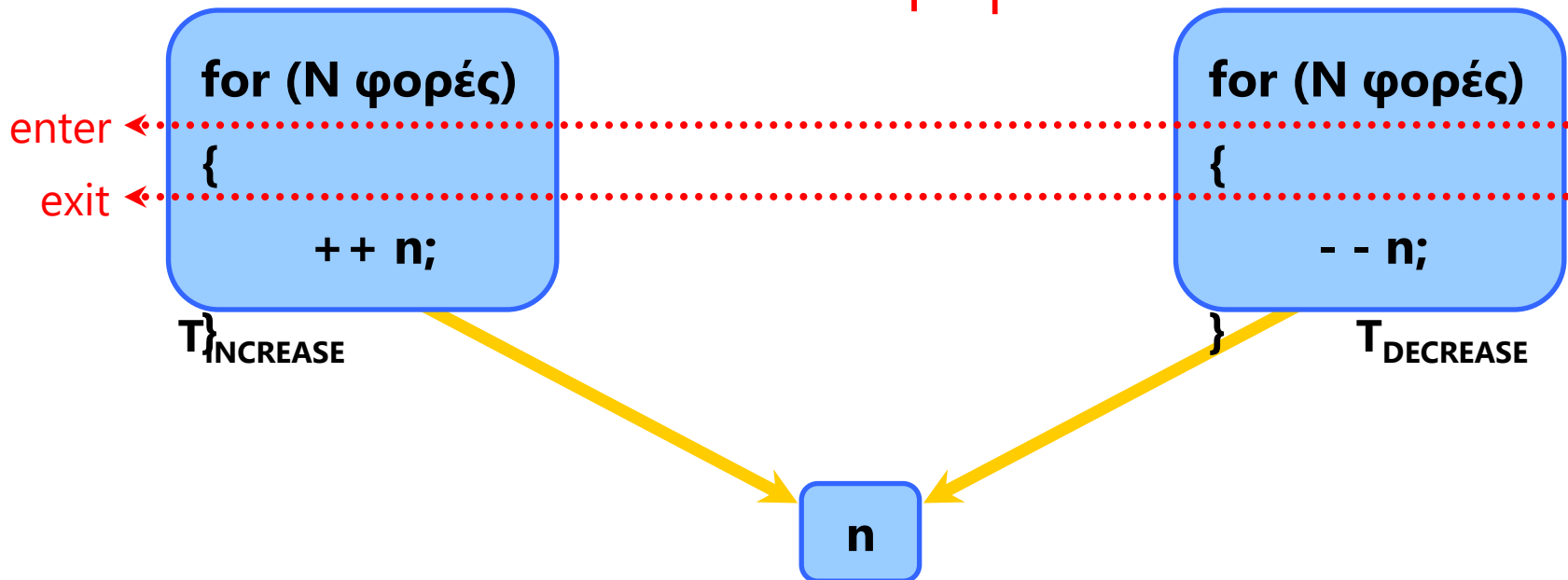


- ◆ Z1: Συγχρονισμός σε υπάρχοντα κώδικα (κρίσιμο τμήμα)
 - ➔ `simplesync.c`
 - ➔ Με POSIX mutexes και GCC atomic ops
- ◆ Z2: Παραλληλοποίηση υπάρχοντα κώδικα (ανάγκη σειριοποίησης)
 - ➔ Συγχρονισμός νημάτων για παράλληλο υπολογισμό

Z1: Συγχρονισμός σε υπάρχοντα κώδικα

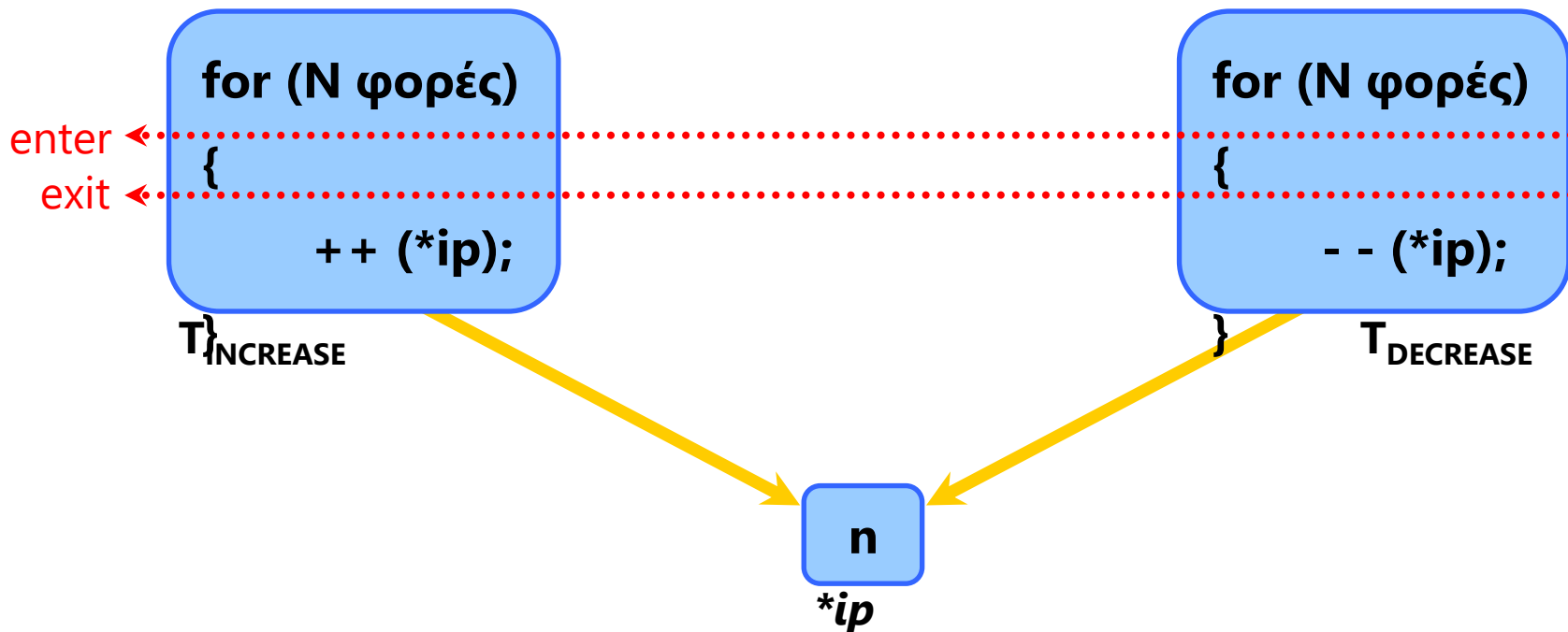
- ◆ Δύο νήματα: T_{INCREASE} , T_{DECREASE}
- ◆ Αυξάνουν/μειώνουν το **κοινό** n , N φορές, αντίστοιχα
- ◆ Αρχική τιμή $n = 0$. Σχήμα συγχρονισμού ώστε

Το n να παραμείνει **0**



Z1: Συγχρονισμός στο **simplesync.c**

- ◆ **Δύο** υλοποιήσεις
- ◆ **Z1α.** POSIX mutexes
- ◆ **Z1β.** GCC atomic operations: `__sync_*()`



Z1: Συγχρονισμός σε υπάρχοντα κώδικα

◆ Z1α. POSIX mutexes/semaphores

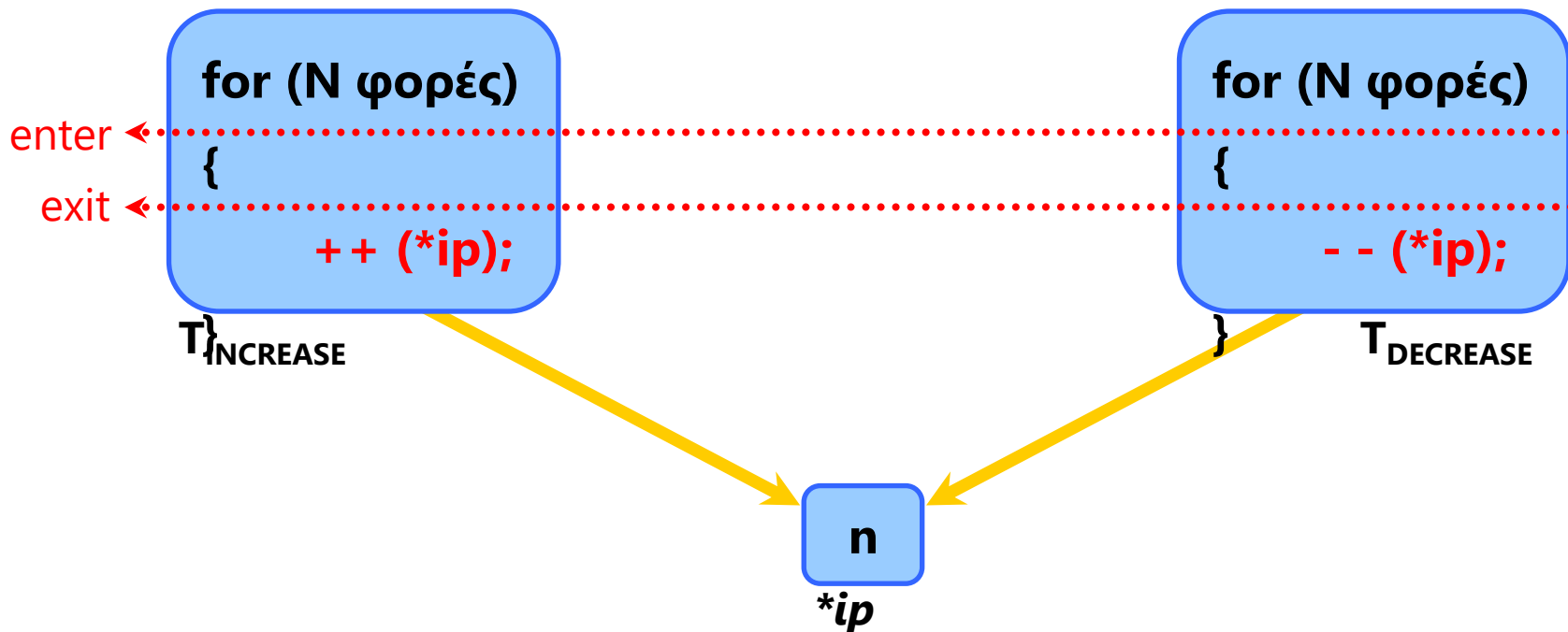
- ➔ Κώδικας **μόνο** στα σημεία "enter", "exit"
- ➔ Κατάλληλα αρχικοποιημένα mutexes ή σηματοφόροι
- ➔ wait(), signal() σε αυτούς
- ➔ Χωρίς αλλαγή του κώδικα που πειράζει τη μεταβλητή

◆ Z1β. GCC atomic operations

- ➔ **Αλλαγή** του τρόπου πρόσβασης στη μεταβλητή
- ➔ Απαιτείται πλέον κώδικας στα "enter", "exit";

Z1: Συγχρονισμός στο **simplesync.c**

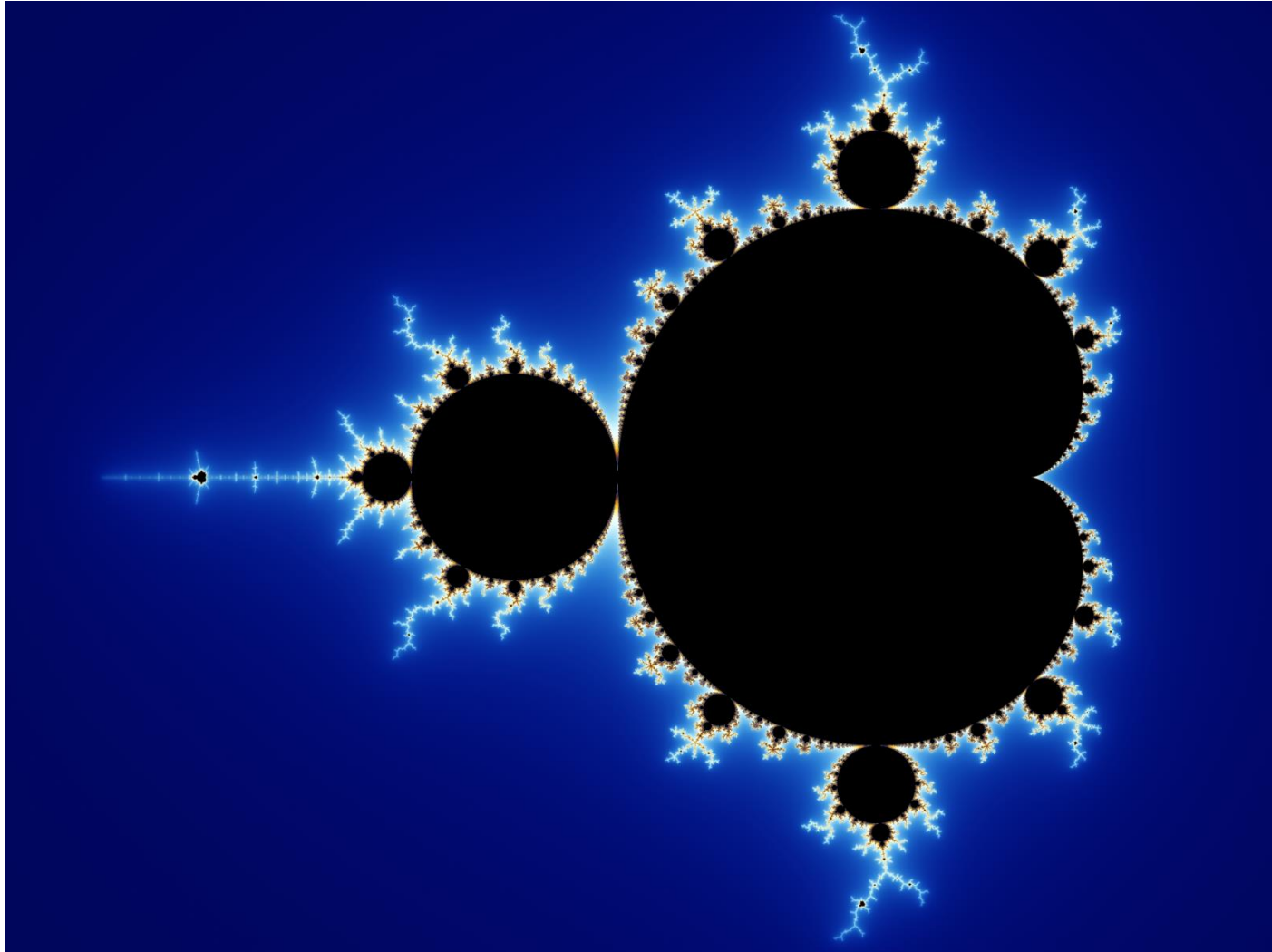
- ◆ **Δύο** υλοποιήσεις
- ◆ **Z1α.** POSIX mutexes
- ◆ **Z1β.** GCC atomic operations: `__sync_*()`



Σύνοψη

- ◆ Z1: Συγχρονισμός σε υπάρχοντα κώδικα (κρίσιμο τμήμα)
 - ➔ `simplesync.c`
 - ➔ Με POSIX mutexes και GCC atomic ops
- ◆ Z2: Παραλληλοποίηση υπάρχοντα κώδικα (ανάγκη σειριοποίησης)
 - ➔ Συγχρονισμός νημάτων για παράλληλο υπολογισμό
 - ➔ Με POSIX semaphores και conditional vars

Z2: Παραλληλοποίηση: the Mandelbrot Set

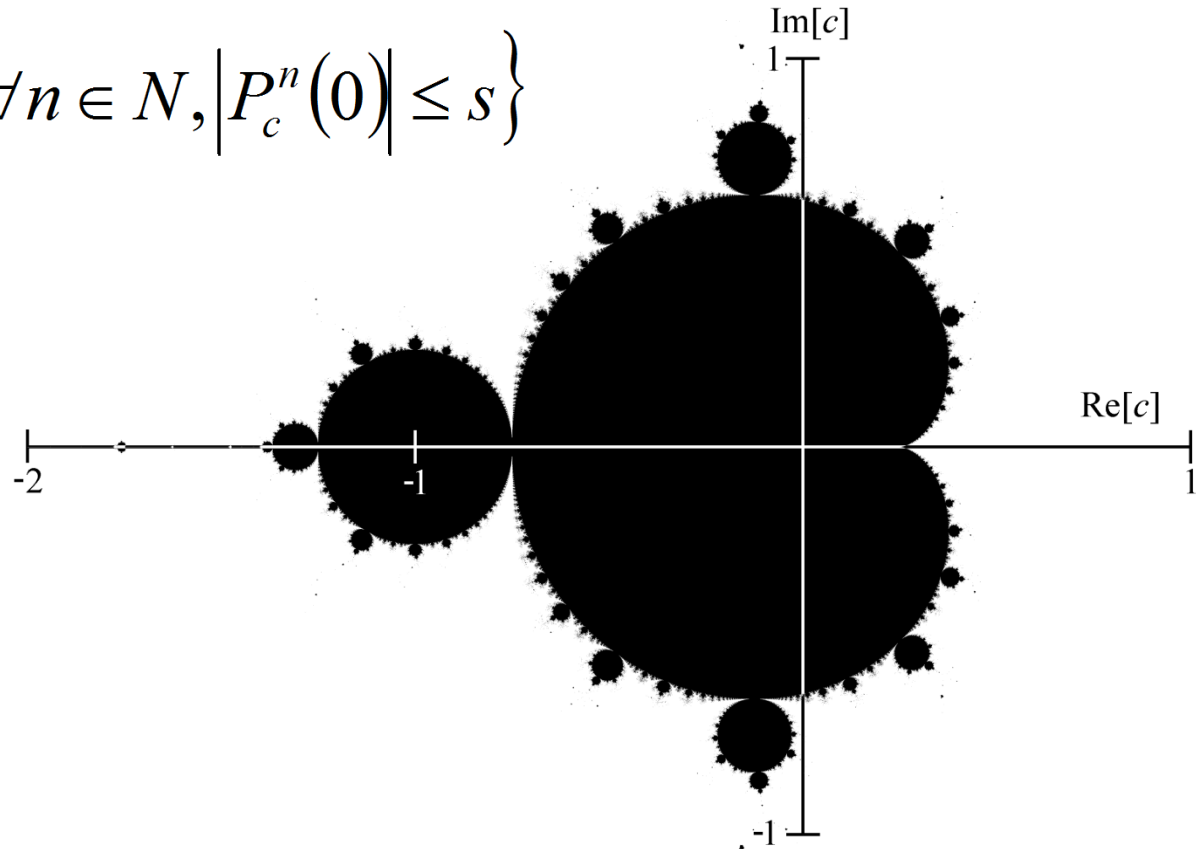


The Mandelbrot Set: Ορισμός

$$P_c : C \rightarrow C$$

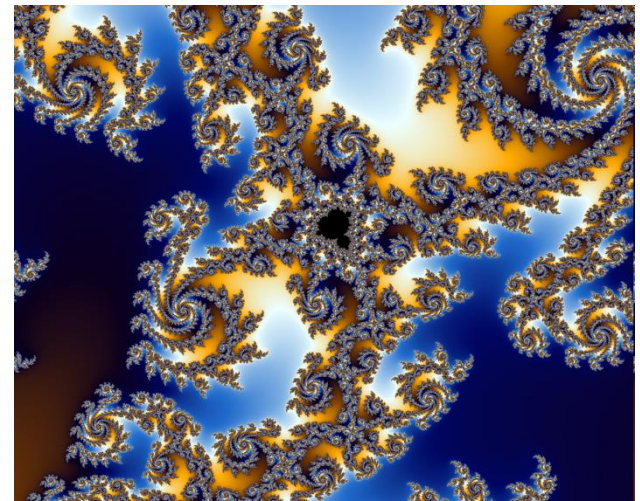
$$P_c : z \rightarrow z^2 + c$$

$$M = \left\{ c \in C : \exists s \in R, \forall n \in N, |P_c^n(0)| \leq s \right\}$$



The Mandelbrot Set: σχεδίαση

- ◆ Για κάθε σημείο c μιας περιοχής του μιγαδικού επιπέδου
 - ➔ Επαναληπτικός υπολογισμός του $\mathbf{z}_{n+1} = \mathbf{z}_n^2 + \mathbf{c}$, $\mathbf{z}_0 = \mathbf{0}$, μέχρι να ξεφύγει το $|\mathbf{z}_n| > 2$
 - ➔ Κάθε pixel χρωματίζεται ανάλογα με τον αριθμό των επαναλήψεων που χρειάστηκαν, ή \mathbf{n}_{\max}
- ◆ Υπάρχουν κι άλλοι αλγόριθμοι



The Mandelbrot Set: κώδικας



- ◆ Σας δίνεται κώδικας (mandel.c) που ζωγραφίζει εικόνες από το σύνολο Mandelbrot
 - ➔ Στο τερματικό, με χρωματιστούς χαρακτήρες
 - ➔ Κάθε εικόνα είναι πλάτους **x_chars**, ύψους **y_chars**
- ◆ Η σχεδίαση γίνεται επαναληπτικά, για κάθε γραμμή
- ◆ Συναρτήσεις
 - ➔ compute_and_output_mandel_line(fd, line)
 - ➔ mandel_iterations_at_point(x, y, MAX)
 - ➔ set_xterm_color(fd, color)

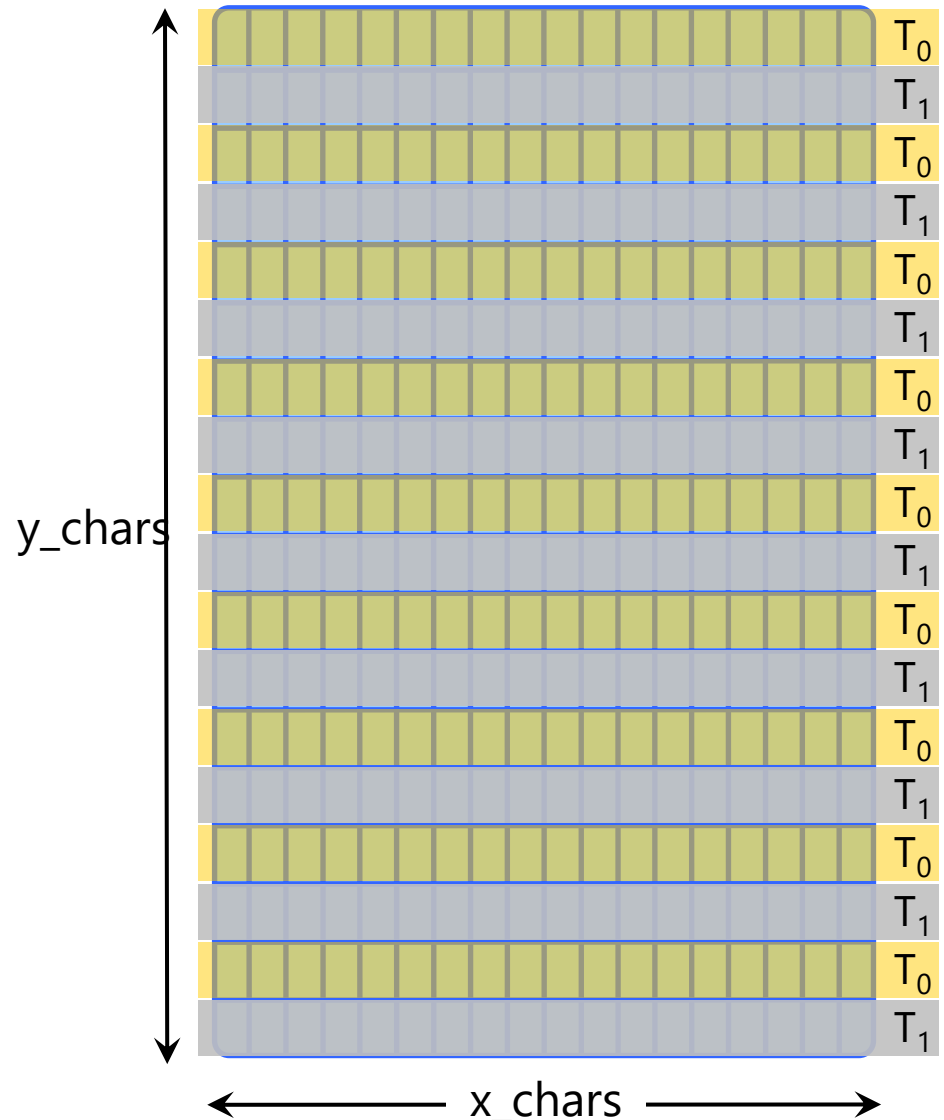
The Mandelbrot Set: Παραλληλοποίηση

- ◆ Κατανομή του φορτίου ανά γραμμές
- ◆ Ξεκινώντας από το πρώτο νήμα, ανάθεση γραμμών με κυκλική επαναφορά

- ◆ Νήμα i από N :

$i, i + N, i + 2*N, i + 3*N$
κλπ

Συγχρονισμός;



Μηχανισμοί (POSIX)

- ◆ POSIX Threads <pthread.h>
 - ➔ pthread_create(), pthread_join(), pthread_exit()
- ◆ POSIX Mutexes <pthread.h>
 - ➔ pthread_mutex_init(), pthread_mutex_lock(), pthread_mutex_unlock()
- ◆ POSIX Spinlocks <pthread.h>
 - ➔ pthread_spin_init(), pthread_spin_lock(), pthread_spin_unlock()
- ◆ POSIX (unnamed) Semaphores <semaphore.h>
 - ➔ sem_overview(), sem_init(), sem_post(), sem_wait()
- ◆ POSIX condition variables:
 - ➔ pthread_cond_init(), pthread_cond_wait(), pthread_cond_signal(), pthread_cond_broadcast()
- ◆ **Εγκαταστήστε** τα πακέτα manpages-posix, manpages-posix-dev δίνοντας
(sudo) apt-get install:

man -a sem_post

Condition Variables



```
pthread_mutex_t Lock;  
pthread_cond_t cond;  
int counter = 0;
```

```
/* Thread A */  
pthread_mutex_lock(&Lock);  
while (counter < 10)  
    pthread_cond_wait(&cond, &Lock);  
...  
pthread_mutex_unlock(&Lock);
```

```
/* Thread B */  
pthread_mutex_lock(&Lock);  
counter++;  
pthread_cond_signal(&cond);  
pthread_mutex_unlock(&Lock);
```

Σωστό!

... αλλά γιατί να κάνω signal σε κάθε αύξηση του counter;

Condition Variables



```
pthread_mutex_t Lock;  
pthread_cond_t cond;  
int counter = 0;
```

```
/* Thread A */  
pthread_mutex_lock(&Lock);  
while (counter < 10)  
    pthread_cond_wait(&cond, &Lock);  
...  
pthread_mutex_unlock(&Lock);
```

```
/* Thread B */  
pthread_mutex_lock(&Lock);  
counter++;  
if (counter == 10)  
    pthread_cond_signal(&cond);  
pthread_mutex_unlock(&Lock);
```

Σωστό MONO για 2 νήματα

Condition Variables



```
pthread_mutex_t Lock;  
pthread_cond_t cond;  
int counter = 0;
```

```
/* Threads A,C,D,E */  
pthread_mutex_lock(&Lock);  
while (counter < 10)  
    pthread_cond_wait(&cond, &Lock);  
...  
pthread_mutex_unlock(&Lock);
```

```
/* Thread B */  
pthread_mutex_lock(&Lock);  
counter++;  
if (counter == 10)  
    pthread_cond_broadcast(&cond);  
pthread_mutex_unlock(&Lock);
```

Χρήσιμα Links

- ◆ Δημιουργία νημάτων
 - ➔ <https://www.cs.cmu.edu/afs/cs/academic/class/15492-f07/www/pthreads.html>
- ◆ Συγχρονισμός Διεργασιών/Νημάτων
 - ➔ <https://www.embhack.com/difference-between-spinlock-and-mutex/>
- ◆ Συγχρονισμός σε Κοινά Δεδομένα
 - ➔ <https://gcc.gnu.org/onlinedocs/gcc-4.1.1/gcc/Atomic-Builtins.html>
- ◆ The lost-wakeup problem
 - ➔ <https://askldjd.com/2010/04/24/the-lost-wakeup-problem/>

Ερωτήσεις;



και στη λίστα:

OS@lists.cslab.ece.ntua.gr