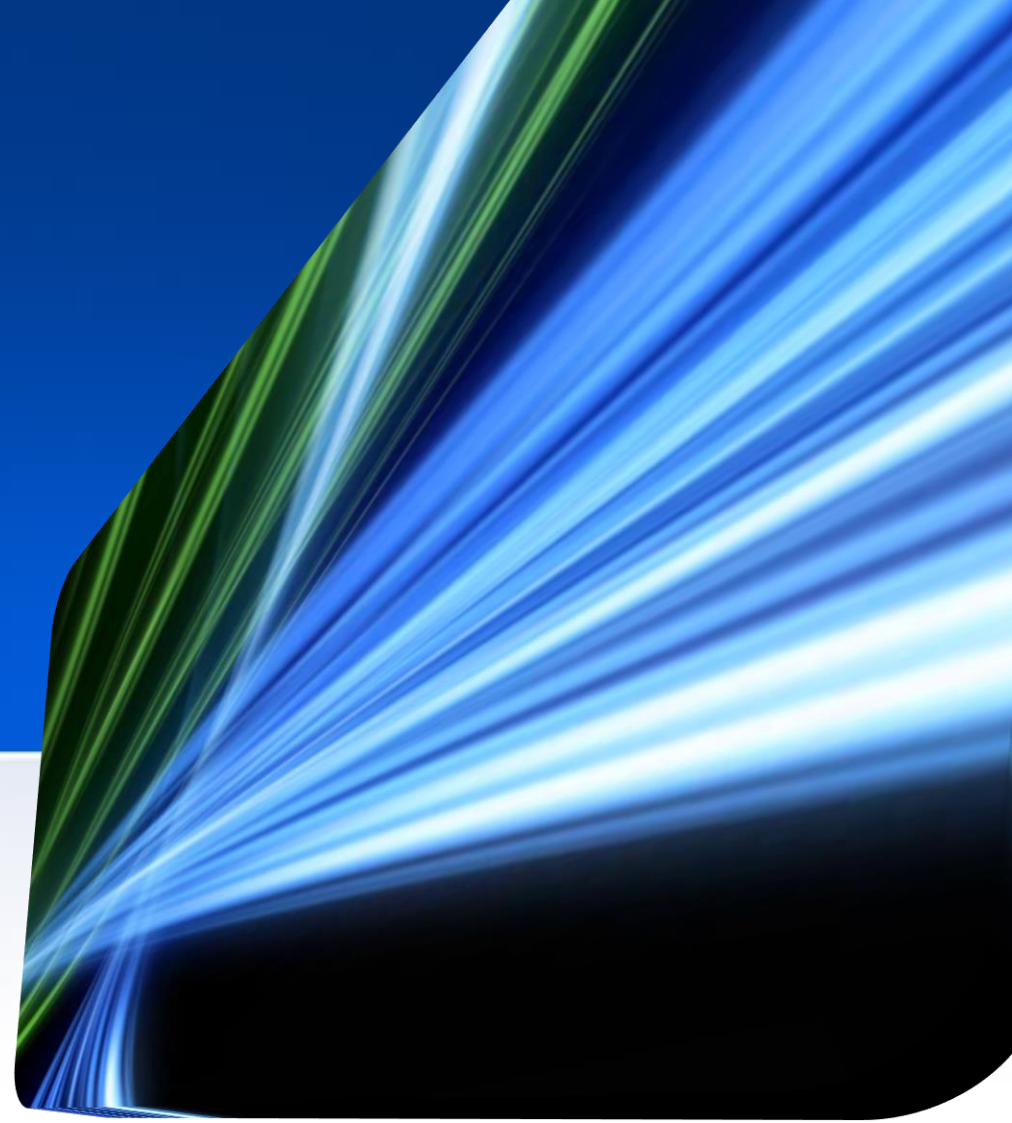
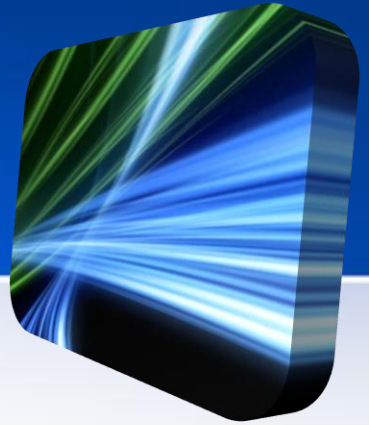
Λειτουργικά Συστήματα
6ο εξάμηνο ΣΗΜΜΥ
Ακ. έτος 2020-2021



Νήματα και Ταυτοχρονισμός

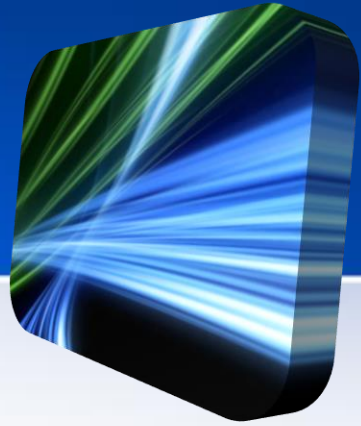
Παναγιώτης Τσανάκας

Στόχοι Παρουσίασης



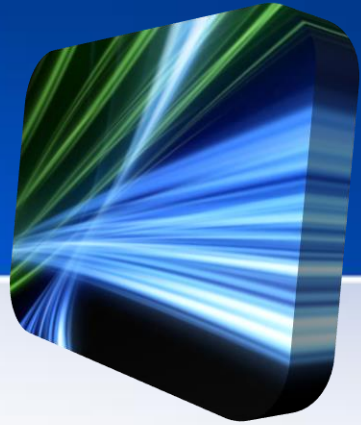
- Εισαγωγή στην **έννοια του νήματος (thread)**
- Παρουσίαση των **πολυνηματικών συστημάτων**
- Παρουσίαση **API για βιβλιοθήκες νημάτων**
- Εξέταση θεμάτων **πολυνηματικού προγραμματισμού**
- **Παραδείγματα** σε Λειτουργικά Συστήματα

Κίνητρα



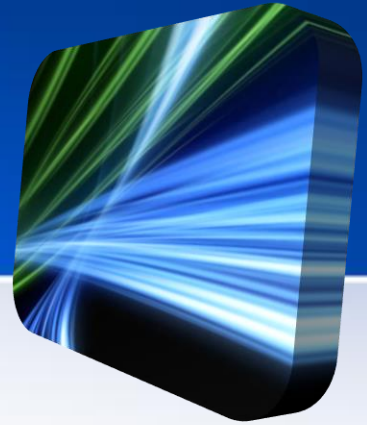
- Μια εφαρμογή υλοποιείται ως μια ξεχωριστή διεργασία με αρκετά νήματα
- Πολλαπλές λειτουργίες **μιας** εφαρμογής μπορούν να υλοποιηθούν από **ξεχωριστά νήματα**. Σε έναν κειμενογράφο μπορούμε να έχουμε διακριτά νήματα για
  - Παρουσίαση γραφικών
  - Ανταπόκριση στο πάτημα πλήκτρων
  - Ορθογραφικός έλεγχος

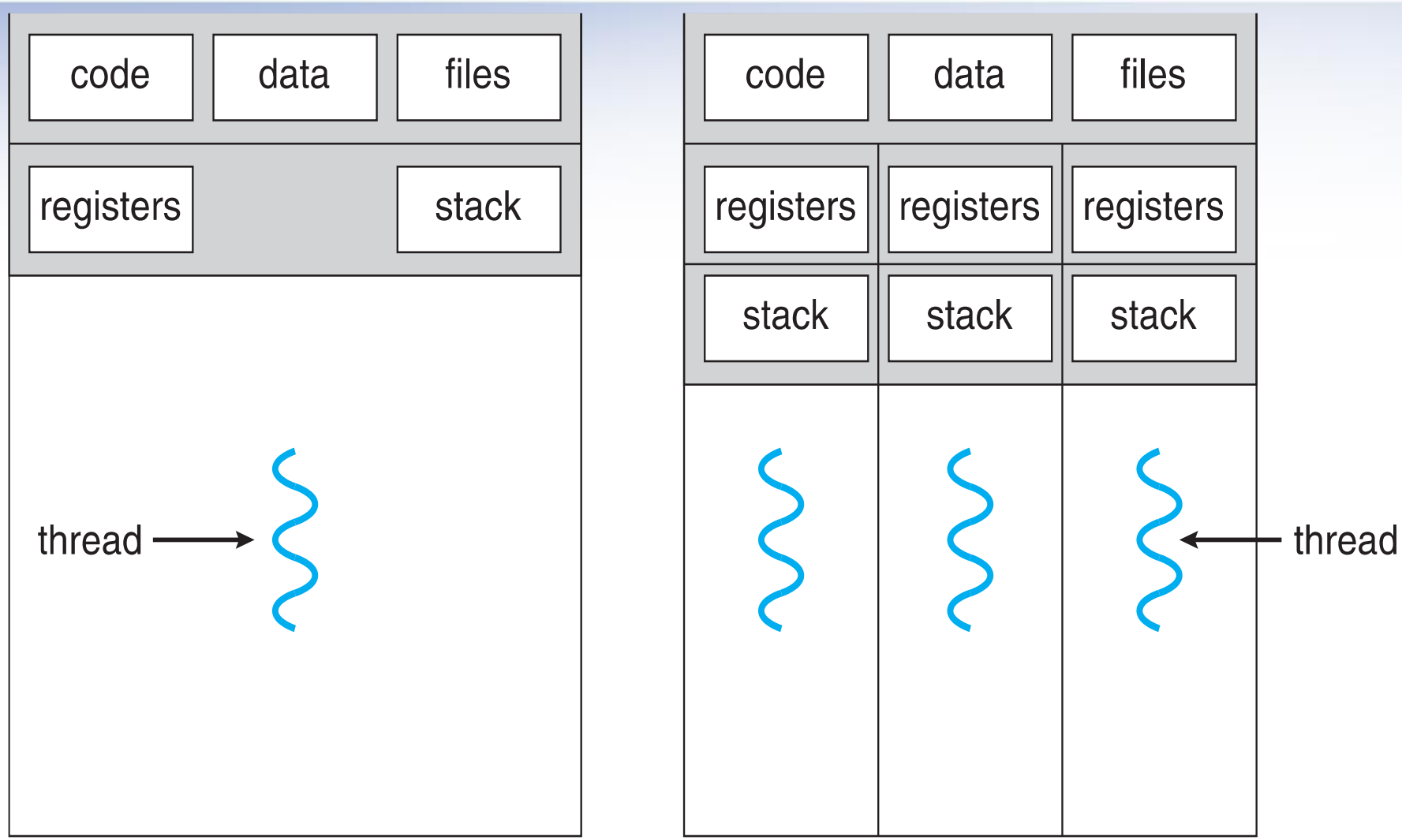
Κίνητρα



- Η δημιουργία διεργασίας είναι χρονοβόρα και απαιτεί πόρους, ενώ η δημιουργία νήματος είναι ελαφριά
- Χρήση νημάτων μπορεί να απλοποιήσει τον κώδικα, να αυξήσει την αποδοτικότητα
- Οι πυρήνες CPU είναι γενικά πολυνηματικοί

Μονονηματική και Πολυνηματική διεργασία

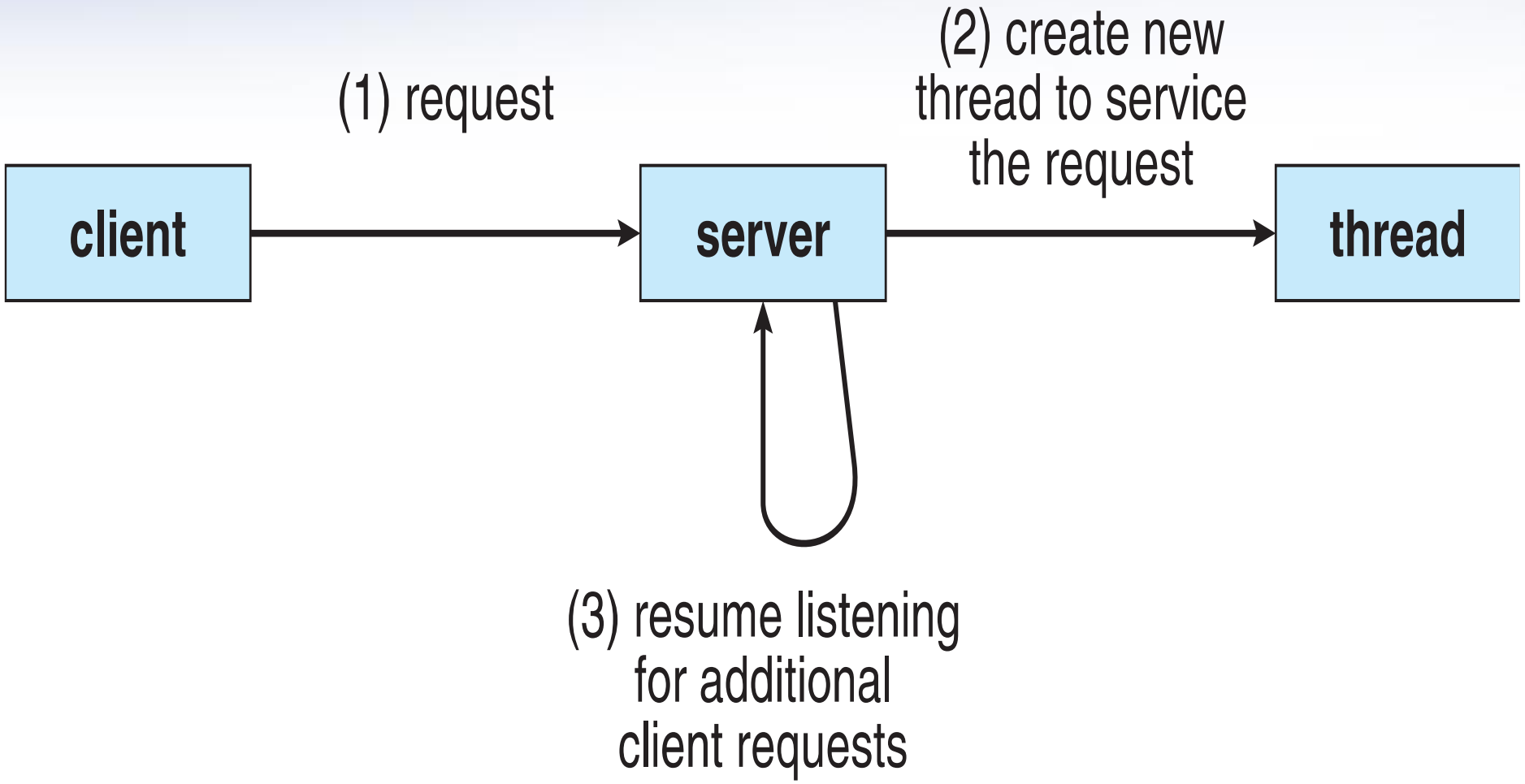




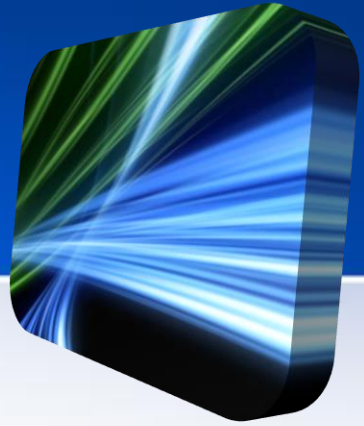
single-threaded process

multithreaded process

Αρχιτεκτονική Πολυνηματικού εξυπηρετητή



Πλεονεκτήματα νημάτων 1/2



- **Ικανότητα απόκρισης (Responsiveness) -**

Μπορεί να συνεχίζεται η εκτέλεση μιας εφαρμογής εάν κάποιο τμήμα της διεργασίας έχει αποκλειστεί, ιδιαίτερα σημαντικό για τις διεπαφές χρήστη

- **Διαμοιρασμός πόρων (Resource Sharing) -**

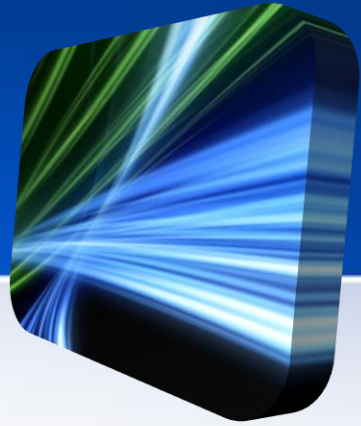
Τα νήματα μοιράζονται πόρους της διεργασίας, πιο εύκολα από την κοινή μνήμη ή το πέρασμα μηνυμάτων

Πλεονεκτήματα νημάτων 2/2

- **Οικονομία (Economy)** – μικρότερο κόστος σε μνήμη και πόρους από τη δημιουργία της διεργασίας, η εναλλαγή περιβάλλοντος λειτουργίας των νημάτων απαιτεί **λιγότερο χρόνο** από την εναλλαγή διεργασιών

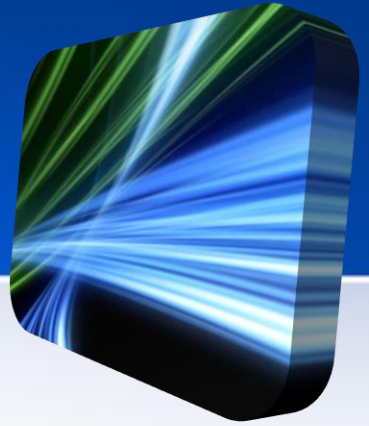
- **Κλιμάκωση (Scalability)**- η διεργασία μπορεί να επωφεληθεί από τις αρχιτεκτονικές πολλών επεξεργαστών

Πολυπύρηνος προγραμματισμός …



- Προκλήσεις σε πολυπύρηνα συστήματα (multicore):
  - Αναγνώριση – Διαίρεση ενεργειών (Dividing activities)
  - Εξισορρόπηση φορτίου (Load Balance)
  - Διαίρεση δεδομένων (Data splitting)
  - Εξάρτηση Δεδομένων (Data dependency)
  - Έλεγχος και αποσφαλμάτωση (Testing and debugging)

… Πολυπύρηνος προγραμματισμός …



- Τύποι παραλληλισμού

Παραλληλισμός Δεδομένων (Data parallelism) –

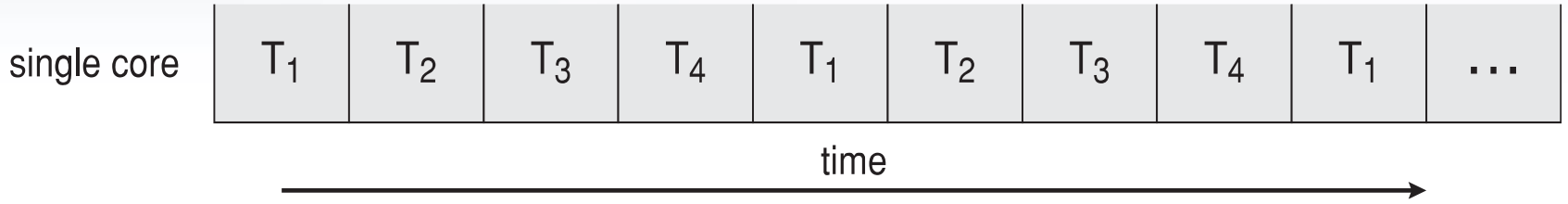
- διανέμει υποσύνολα των ίδιων δεδομένων σε πολλαπλούς πυρήνες,
- εκτέλεση της **ίδιας λειτουργίας** σε κάθε πυρήνα

Παραλληλισμός Ενεργειών (Task parallelism) –

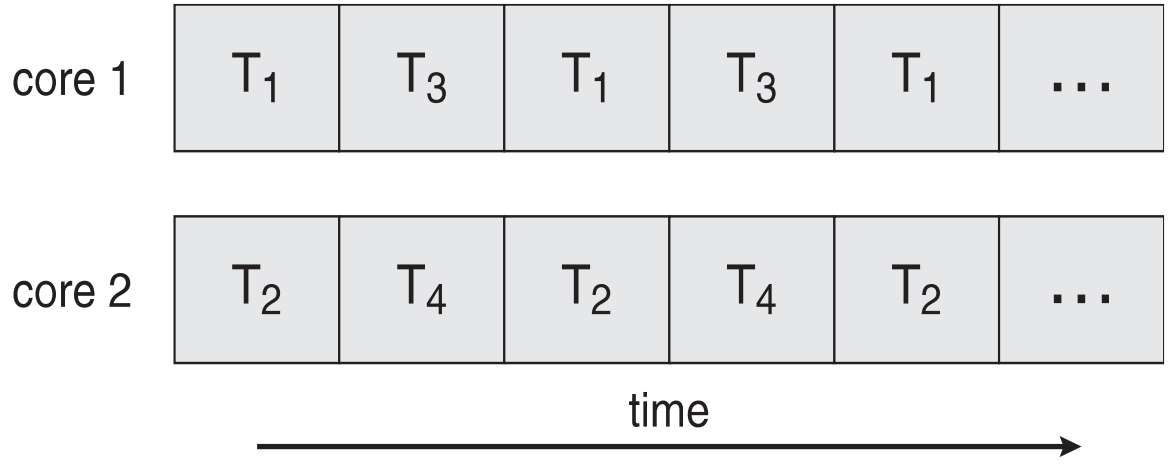
- κατανομή των νημάτων σε πυρήνες,
- κάθε νήμα εκτελεί **διαφορετική** λειτουργία

Συνδρομικότητα vs Παραλληλισμός
Concurrency vs. Parallelism

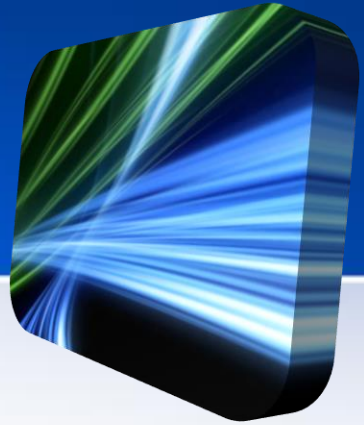
- Συνδρομική εκτέλεση (Concurrent execution) σε έναν πύρηνα.

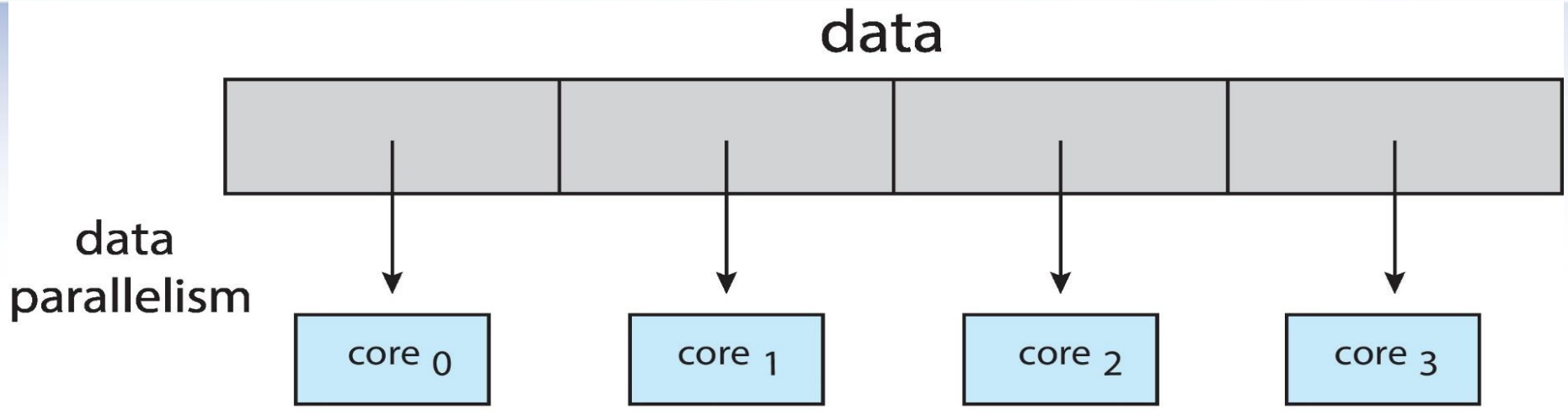


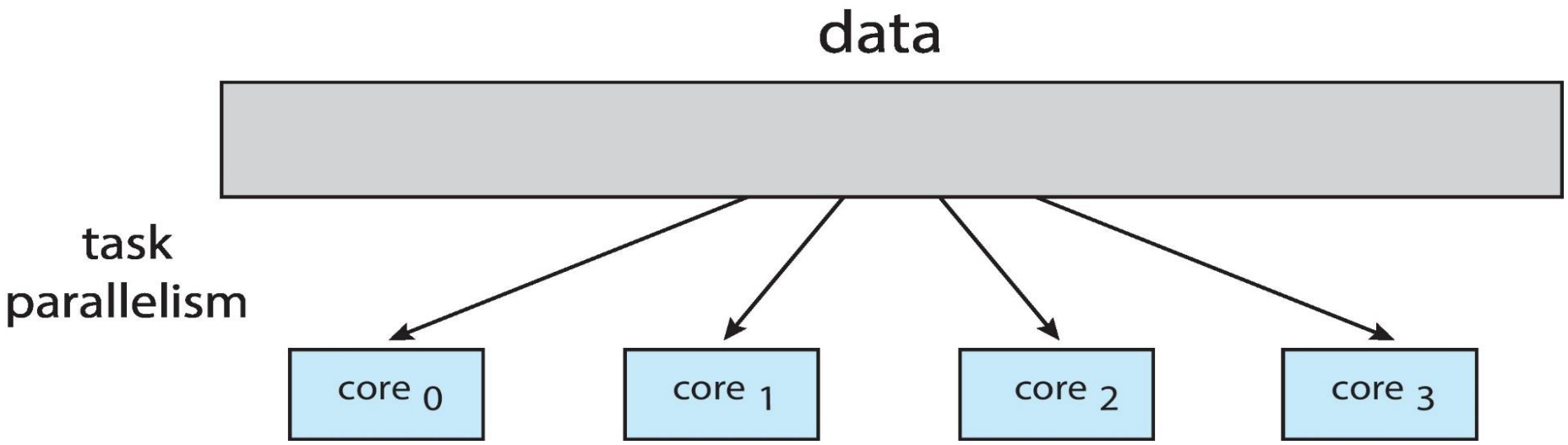
- Παράλληλη εκτέλεση(Parallel execution) σε ένα πολυπύρηνο σύστημα.



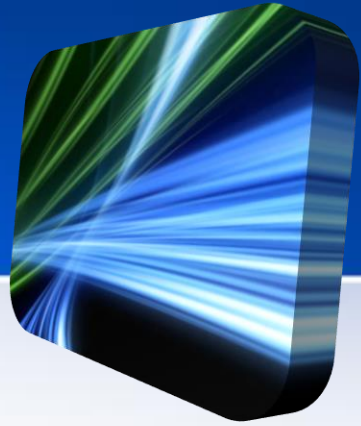
Τύποι παραλληλισμού







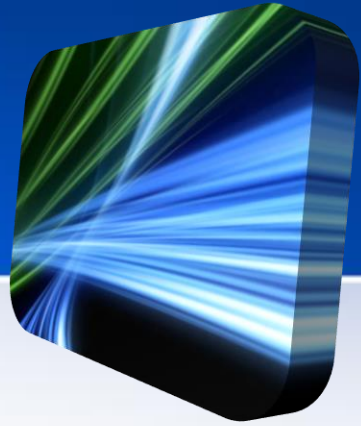
Ο νόμος του Amdahl



- Προσδιορίζει τα πιθανά οφέλη απόδοσης από την προσθήκη πρόσθετων πυρήνων υπολογισμού σε μια εφαρμογή που διαθέτει ακολουθιακά (σειριακά) και παράλληλα τμήματα
- Αν S είναι σειριακό τμήμα και
- N οι πυρήνες επεξεργασίας

$$speedup \leq \frac{1}{S + \frac{(1-S)}{N}}$$

Ο νόμος του Amdahl

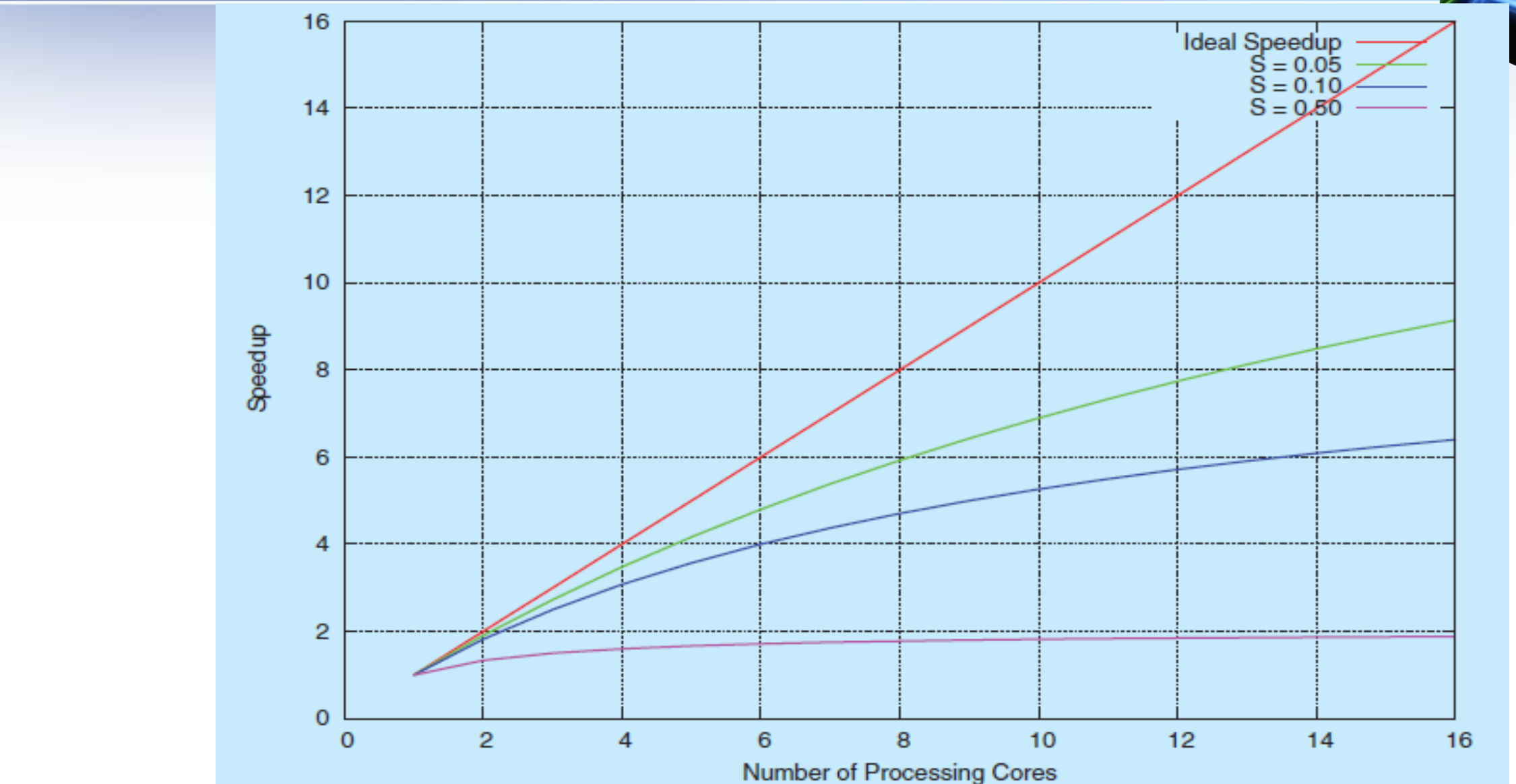


Δηλαδή, αν η εφαρμογή είναι 75% παράλληλη και 25% σειριακή, η μετάβαση από 1 σε 2 πυρήνες οδηγεί σε επιτάχυνση **1,6 φορές**

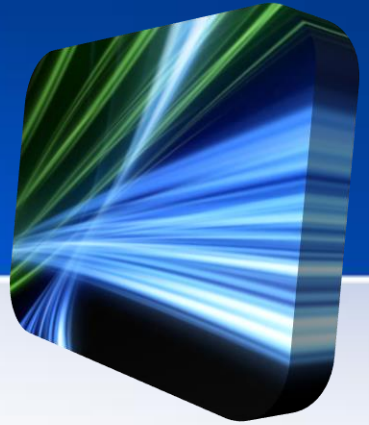
Καθώς το N πλησιάζει το άπειρο, η ταχύτητα συγκλίνει στο **1/S**

Το σειριακό τμήμα μιας εφαρμογής μπορεί να έχει μια **δυσανάλογη** επίδραση στις επιδόσεις που επιτυγχάνονται με την προσθήκη πρόσθετων πυρήνων

Ο νόμος του Amdahl

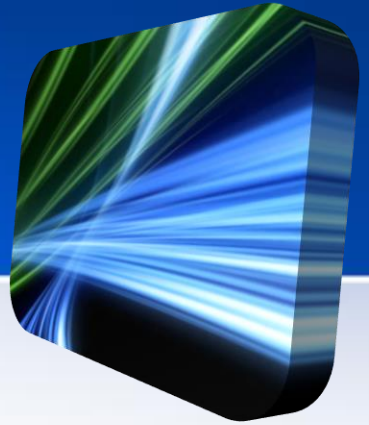


Νήματα Χρήστη (User Threads) και Νήματα Πυρήνα (Kernel Threads)



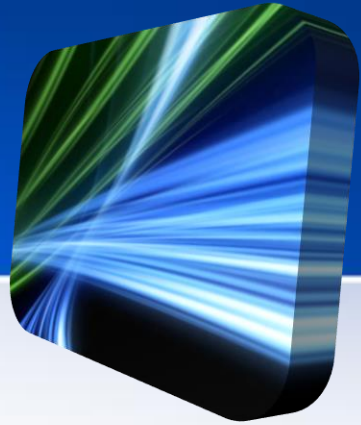
- Νήματα χρηστών (User threads)- η διαχείριση γίνεται από τη βιβλιοθήκη νήματος, σε επίπεδο χρήστη
- Τρεις κύριες βιβλιοθήκες νημάτων:
  - POSIX Pthreads
  - Τα νήματα των Windows
  - Java threads

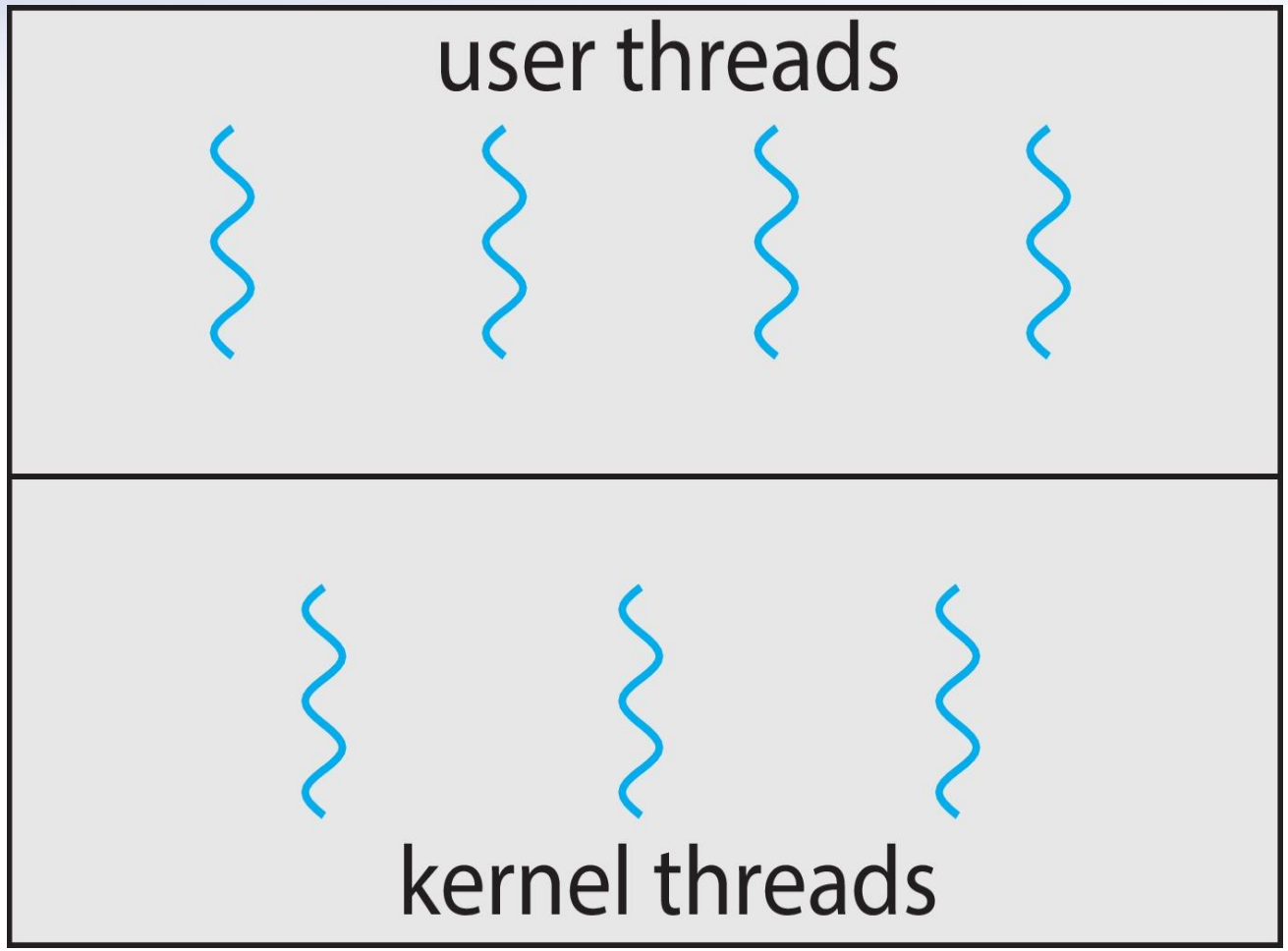
Νήματα Χρήστη (User Threads) και Νήματα Πυρήνα (Kernel Threads)



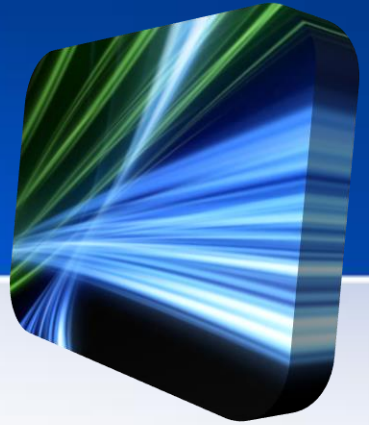
- Νήματα πυρήνα (kernel threads)- η διαχείριση γίνεται από τον **πυρήνα**
  - Windows
  - Solaris
  - Linux
  - Mac OS X
  - Android
  - iOS

Νήματα Χρήστη / Πυρήνα





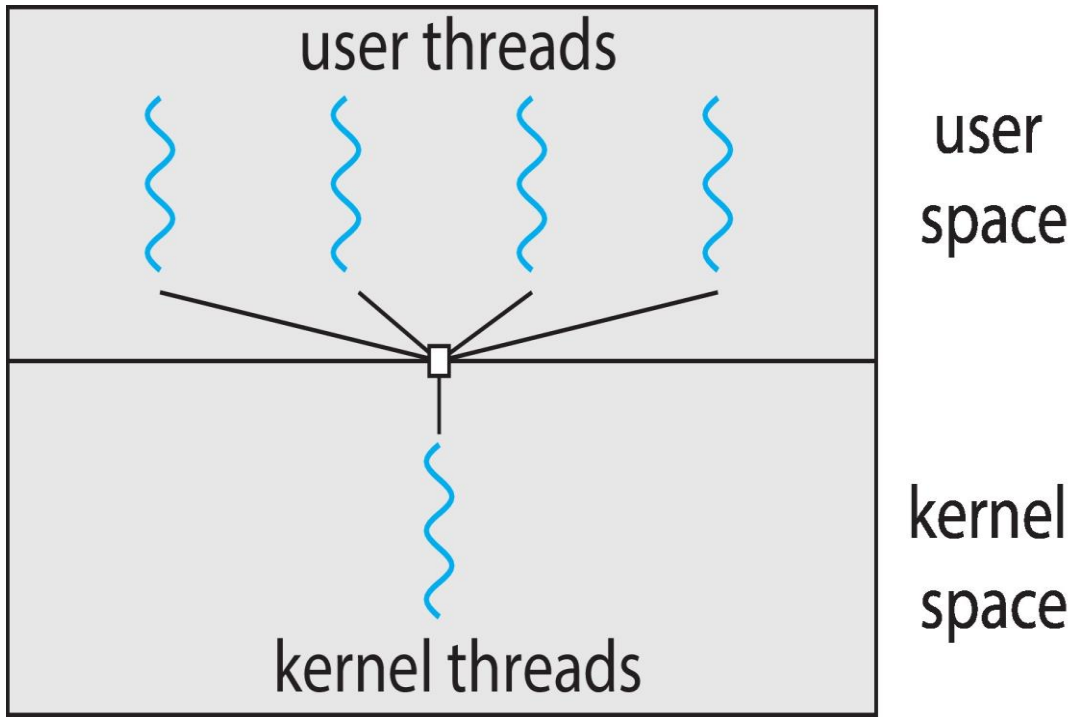
Μοντέλα Πολυ-νημάτωσης



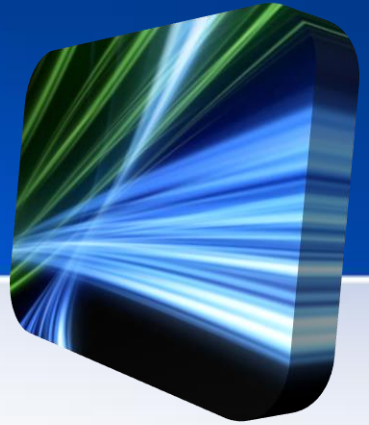
- Πολλά προς ένα
- Ένα προς ένα
- Πολλά προς πολλά

Πολλά προς Ένα

- Πολλά νήματα σε επίπεδο χρήστη έχουν αντιστοιχιστεί σε **ένα** νήμα πυρήνα
- Ο αποκλεισμός ενός νήματος προκαλεί αποκλεισμό όλων των άλλων νημάτων



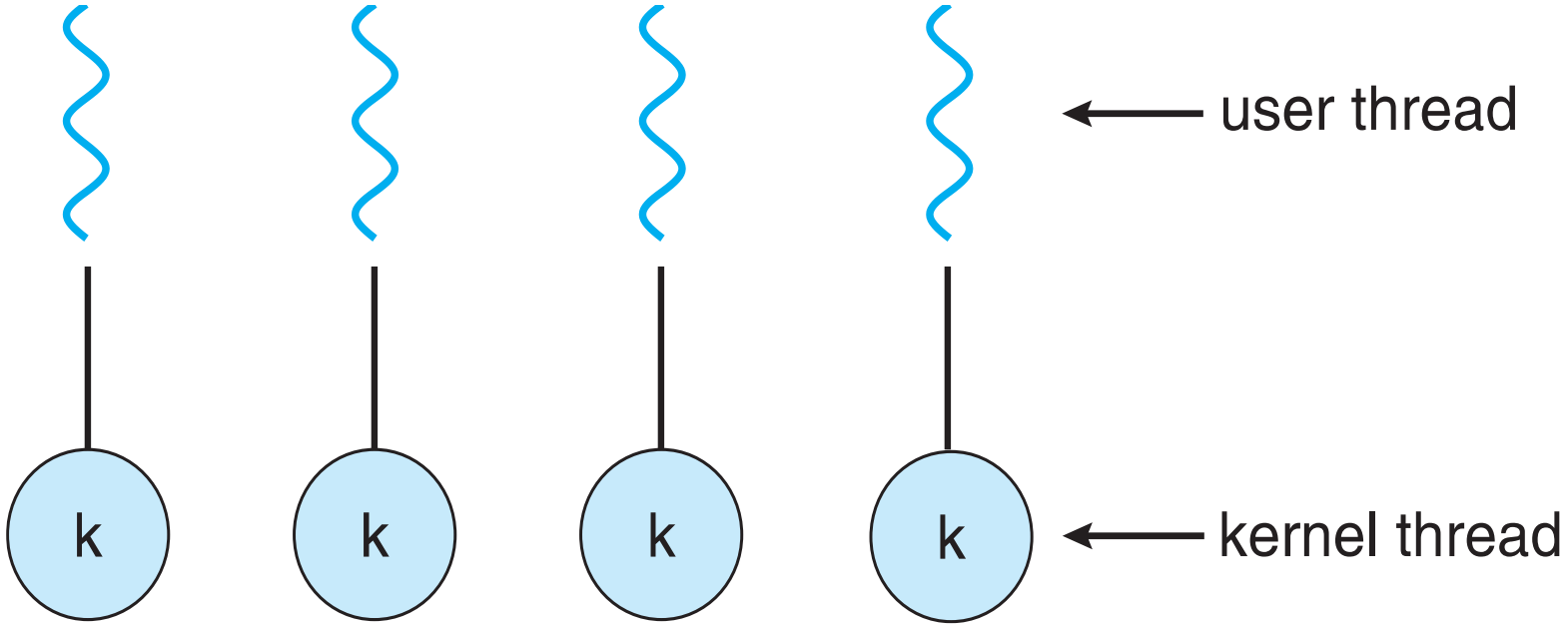
Πολλά προς Ένα



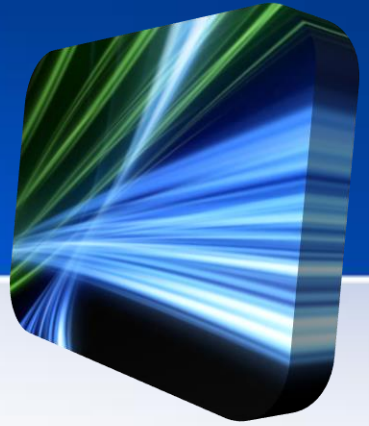
- Πολλά νήματα ενδέχεται να μην εκτελούνται παράλληλα, επειδή μόνο ένα μπορεί να είναι στον πυρήνα κάθε φορά
- **Λίγα** συστήματα χρησιμοποιούν αυτήν τη στιγμή αυτό το μοντέλο
  - Solaris Green Threads
  - GNU Portable Threads

Ένα προς ένα

- Κάθε νήμα σε επίπεδο χρήστη αντιστοιχεί σε **ένα** νήμα πυρήνα
- Η δημιουργία ενός νήματος σε επίπεδο χρήστη δημιουργεί **ένα** νήμα πυρήνα



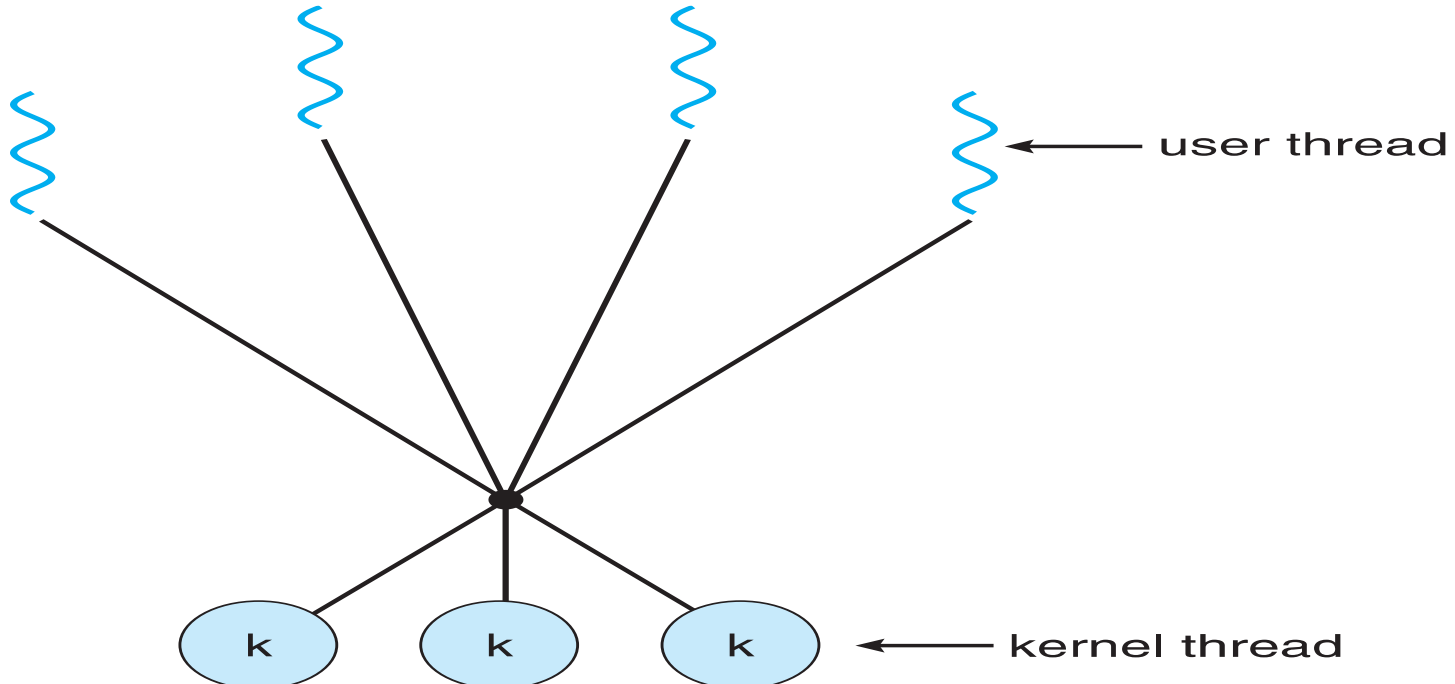
Ένα προς ένα



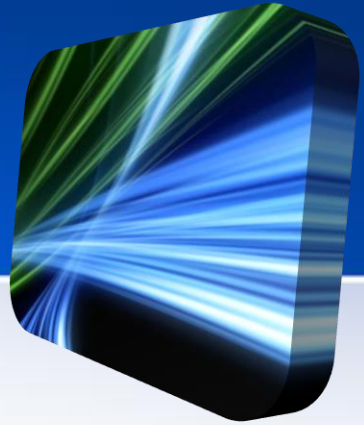
- Περισσότερη **συνδρομική** λειτουργία από το μοντέλο «πολλά προς ένα»
- Ο αριθμός των νημάτων ανά διεργασία ορισμένες φορές **περιορίζεται** λόγω της επιβάρυνσης για δημιουργία νημάτων πυρήνα
- Παραδείγματα
  - Windows
  - Linux
  - Solaris 9 και μετά

Πολλά προς πολλά

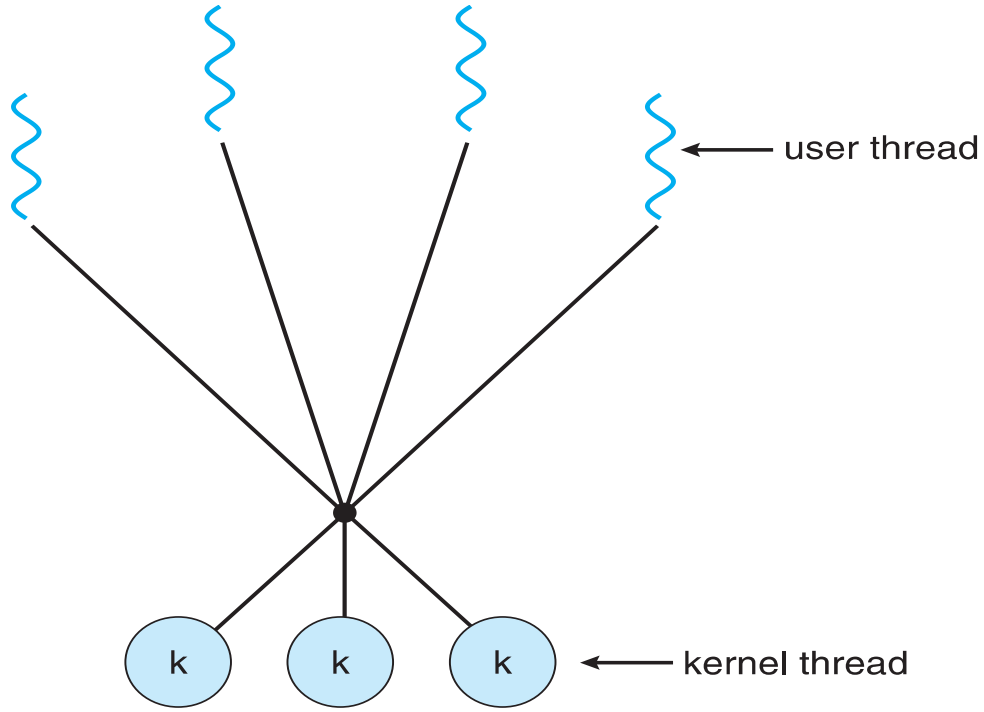
- Επιτρέπει **πολλά** νήματα επιπέδου χρήστη να αντιστοιχίζονται σε **πολλά** νήματα πυρήνα
- Επιτρέπει στο Λειτουργικό Σύστημα τη δημιουργία επαρκούς αριθμού νημάτων πυρήνα



Πολλά προς πολλά

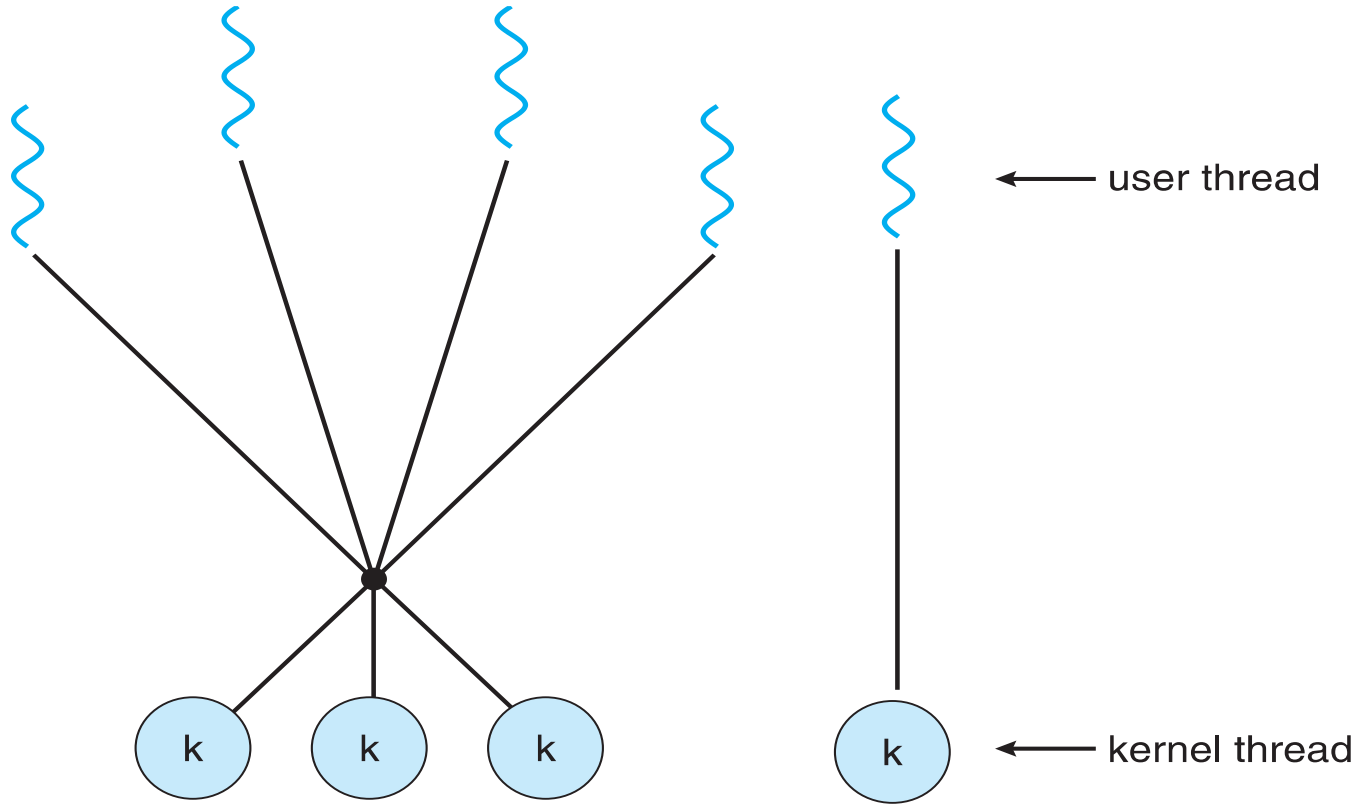


- Λειτουργικά Συστήματα
  - Solaris πριν από την έκδοση 9
  - Windows με το πακέτο ThreadFiber

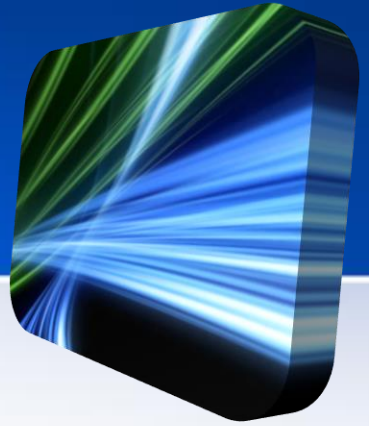


Μοντέλο «δύο επιπέδων»

- Παρόμοιο με το μοντέλο «πολλά προς πολλά», εκτός από το ότι **επιτρέπει** σε ένα νήμα χρήστη να δεσμεύεται με ένα νήμα του πυρήνα
- Παραδείγματα
  - IRIX
  - HP-UX
  - Tru64 UNIX
  - Solaris 8



Βιβλιοθήκες Νημάτων - Thread Libraries

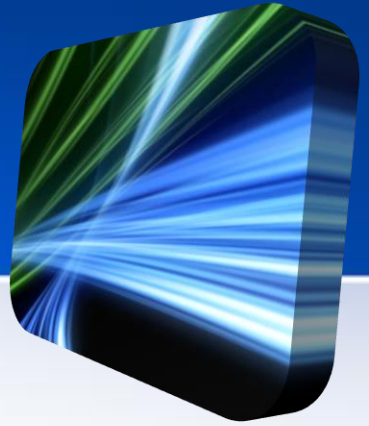


- Η βιβλιοθήκη νήματος (**Thread library**) παρέχει στον προγραμματιστή ένα API για τη δημιουργία και τη διαχείριση των νημάτων
- Δύο βασικοί τρόποι υλοποίησης
  - Βιβλιοθήκη εξ ολοκλήρου στον χώρο του **χρήστη**
  - Βιβλιοθήκη σε επίπεδο **πυρήνα** που υποστηρίζεται από το Λειτουργικό Σύστημα

Pthreads

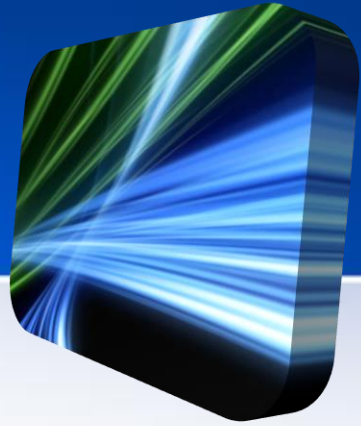
- Μπορεί να παρέχεται είτε σε επίπεδο **χρήστη** είτε σε επίπεδο **πυρήνα**
- Ένα API στο πρότυπο **POSIX** (IEEE 1003.1c) για την δημιουργία και τον συγχρονισμό νημάτων
- Πρόκειται για μια προδιαγραφή της **συμπεριφοράς** των νημάτων και **όχι** μια υλοποίηση
- Η υλοποίηση εξαρτάται από τους σχεδιαστές των Λειτουργικών Συστημάτων
- Κοινή σε λειτουργικά συστήματα UNIX (Solaris, Linux, Mac OS X)

Παράδειγμα Pthreads 1/2



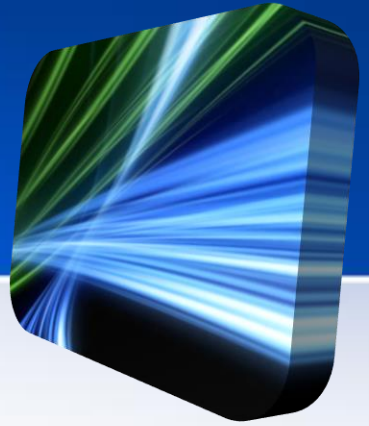
```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
int sum; /* this data is shared by the thread(s) */
void *runner(void *param); /*threads call this function*/
int main(int argc, char *argv[])
{
Pthread_t tid; /* the thread identifier */
Pthread_attr_t attr; /* set of thread attributes */
/* set the default attributes of the thread */
Pthread_attr_init(&attr);
/* create the thread */
Pthread_create(&tid, &attr, runner, argv[1]);
/* wait for the thread to exit */
Pthread_join(tid,NULL);
printf("sum = %d\n",sum);
}
```

Παράδειγμα Pthreads 2/2



```
/* The thread will execute in this function */
void *runner(void *param)
{
int i, upper = atoi(param);
sum = 0;
for (i = 1; i <= upper; i++)
sum += i;
Pthread_exit(0);
}
```

Pthreads για συνένωση 10 Νημάτων



```
#define NUM_THREADS 10

/* an array of threads to be joined upon */
pthread_t workers[NUM_THREADS];

for (int i = 0; i < NUM_THREADS; i++)
   pthread_join(workers[i], NULL);
```

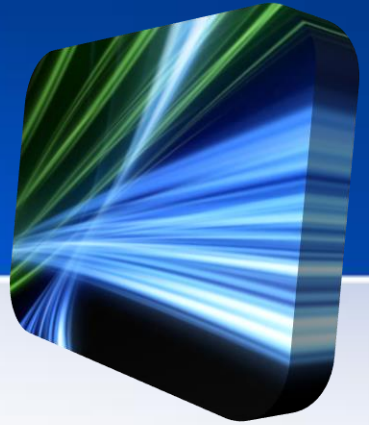
Αυτόματη Νημάτωση - Implicit Threading

- Όσο αυξάνεται ο αριθμός των νημάτων, η ορθότητα του προγράμματος είναι πιο δύσκολη.

Λύση:

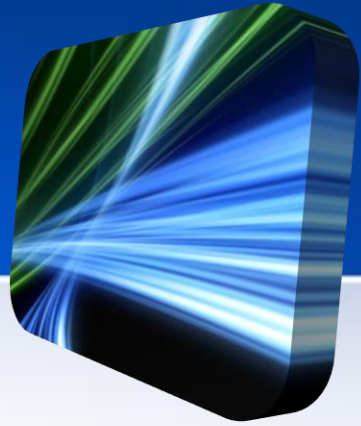
- Η δημιουργία και διαχείριση νημάτων να γίνεται από τους **μεταγλωττιστές** και σε βιβλιοθήκες κατά τον **χρόνο εκτέλεσης**, και **όχι** από τους προγραμματιστές. Η μέθοδος αυτή καλείται **Αυτόματη Νημάτωση (Implicit Threading)**
- **Ορισμός παράλληλων task και όχι thread**

Αυτόματη Νημάτωση - Implicit Threading



- Τρεις μέθοδοι διερευνήθηκαν
  - Δεξαμενές Νημάτων (Thread Pools)
  - OpenMP
  - Grand Central Dispatch

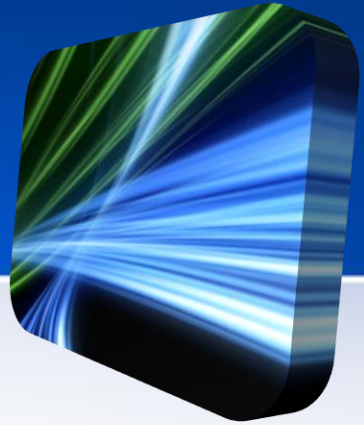
Δεξαμενές Νημάτων



- Δημιουργία ενός συνόλου νημάτων σε μια **δεξαμενή**, όπου αναμένουν εργασία
- Πλεονεκτήματα:

- Συνήθως είναι **ελαφρώς ταχύτερο** να εξυπηρετηθεί ένα αίτημα με ένα **υπάρχον** νήμα από το να δημιουργηθεί ένα νέο νήμα
- Επιτρέπει να είναι **συγκεκριμένος** ο αριθμός των νημάτων στις εφαρμογές
- Η **διαχωρισμός** της εργασίας που πρέπει να εκτελεστεί από την διαδικασία δημιουργίας των εργασιών επιτρέπει **διαφορετικές στρατηγικές** για την εκτέλεση της εργασίας

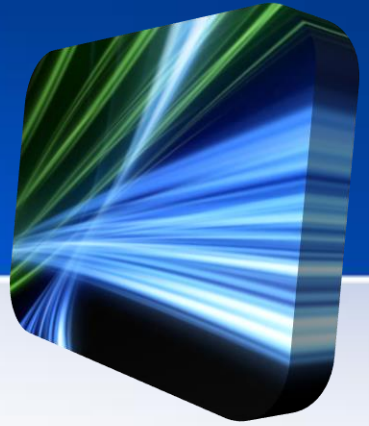
OpenMP



- Σύνολο οδηγιών (directives) μεταγλωττιστή και API για C, C ++, FORTRAN
- Παρέχει υποστήριξη για **παράλληλο προγραμματισμό** σε περιβάλλοντα **κοινής μνήμης**
- Προσδιορίζει **παράλληλες περιοχές** parallel regions - μπλοκ κώδικα που μπορούν να τρέξουν παράλληλα

```
#pragma omp parallel

#pragma omp parallel for
for(i=0;i<N;i++) {
    c[i] = a[i] + b[i];
}
```

OpenMP

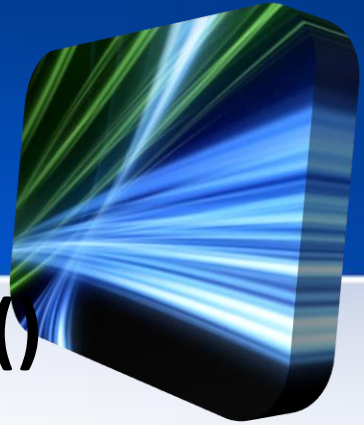


```
#include <omp.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
   /* sequential code */

   #pragma omp parallel
   {
      printf("I am a parallel region.");
   }

   /* sequential code */

   return 0;
}
```

Grand Central Dispatch

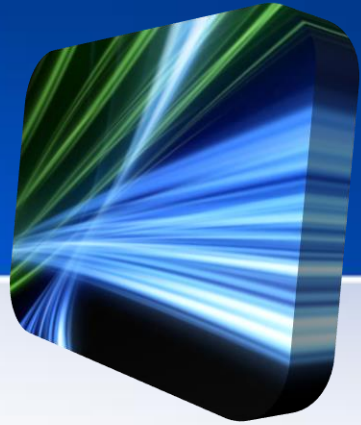
- Τεχνολογία Apple για Λειτουργικά Συστήματα Mac OS X και iOS
- Επεκτάσεις σε γλώσσες C, C ++, API και βιβλιοθήκη χρόνου εκτέλεσης
- Επιτρέπει την αναγνώριση των παράλληλων τμημάτων (blocks)
- Διαχειρίζεται τις περισσότερες λεπτομέρειες σχετικά με τα νήματα

Θέματα Νημάτων



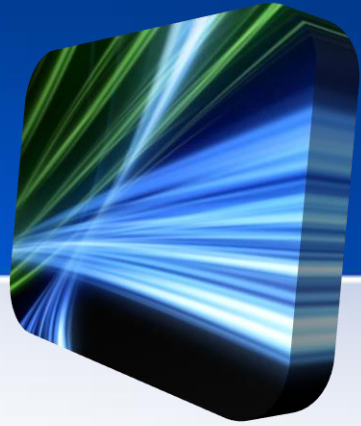
- Αλλάζει η σημασιολογία των κλήσεων συστήματος **fork()** και **exec()** στα νήματα
  - Αντιγραφή **όλων** των νημάτων;
- Χειρισμός Σημάτων (signals)
  - Synchronous και asynchronous
  - Default/User-defined Handler
  - `kill(pid t pid, int signal)`
  - `Pthread_kill(pthread_t tid, int signal)`

Θέματα νημάτων



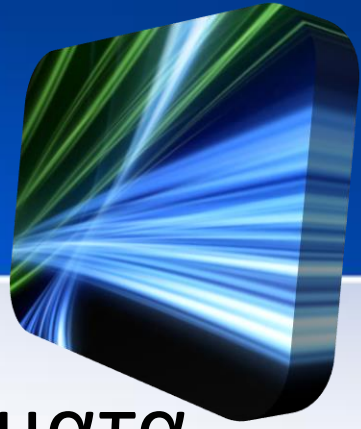
- Ακύρωση Νήματος
  - Asynchronous και
  - Deferred (με χρονική μετάθεση) Cancellation Point
- Τοπικός χώρος αποθήκευσης νήματος, επιπλέον του **μοιραζόμενου** χώρου αποθήκευσης
- Ενεργοποιήσεις Χρονοπρογραμματιστή
  - Συντονισμός με τα νήματα του πυρήνα (πολλά-προς-πολλά)
  - LWP Light-weight Process

Σημασιολογία των κλήσεων συστήματος fork() και exec()



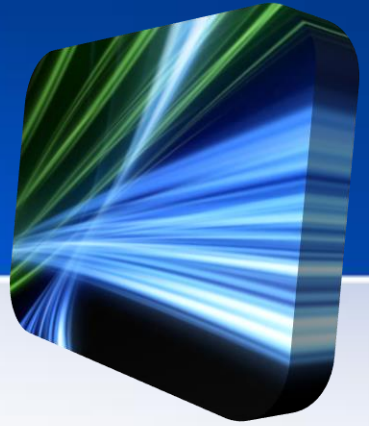
- Η **`fork()`** δημιουργεί αντίγραφο μόνο για το thread που την καλεί ή για όλα τα νήματα;
  - Μερικές διανομές UNIX έχουν δύο εκδόσεις της fork
- Η **`exec()`** λειτουργεί ως συνήθως– αντικαθιστά την τρέχουσα διεργασία, συμπεριλαμβανομένων όλων των νημάτων

Χειρισμός Σημάτων



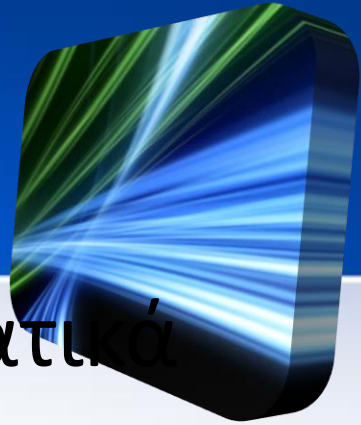
- Τα σήματα (signals) χρησιμοποιούνται σε συστήματα UNIX για την κοινοποίηση σε μια διεργασία ότι έχει συμβεί ένα συγκεκριμένο συμβάν
- Για την επεξεργασία σημάτων χρησιμοποιείται χειριστής σημάτων (handler)
- Το σήμα παράγεται από ένα συγκεκριμένο γεγονός

Χειρισμός Σημάτων



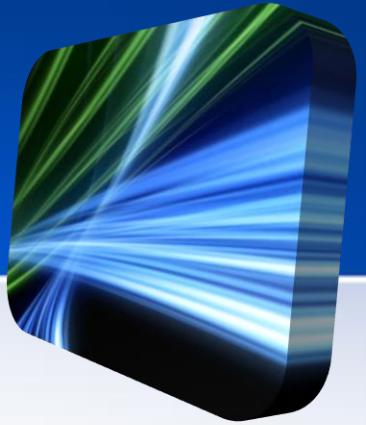
- Το σήμα παραδίδεται σε μια διεργασία
- Ο χειρισμός κάθε σήματος γίνεται από τους δύο πιθανούς χειριστές σημάτων:
  - Προκαθορισμένος
  - Ορισμένος από τον χρήστη
- Κάθε σήμα έχει προεπιλεγμένο χειριστή που τρέχει στον πυρήνα
- Ο χειριστής σημάτων που έχει οριστεί από τον χρήστη μπορεί να αντικαταστήσει τον προκαθορισμένο (default)

Χειρισμός Σημάτων



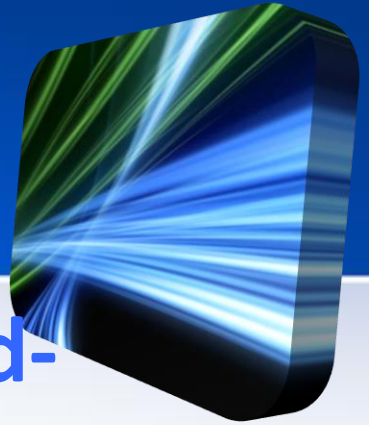
- Πού πρέπει να παραδοθεί ένα σήμα σε πολυνηματικά προγράμματα;
  - Παράδοση του σήματος στο νήμα στο οποίο απευθύνεται το σήμα
  - Παράδοση του σήματος σε κάθε νήμα της διεργασίας
  - Παράδοση του σήματος σε ορισμένα συγκεκριμένα νήματα της διεργασίας
  - Ανάθεση σε ένα συγκεκριμένο νήμα που λαμβάνει όλα τα σήματα για τη διεργασία

Ακύρωση νήματος



- Είναι δυνατόν να τερματιστεί ένα νήμα πριν τελειώσει. Το νήμα που ακυρώνεται είναι το νήμα-στόχος (**target thread**)
- Δύο γενικές προσεγγίσεις:
  - Η **ασύγχρονη ακύρωση** (**asynchronous cancellation**) τερματίζει αμέσως το νήμα στόχευσης
  - Η **μετατοπισμένη ακύρωση** (**Deferred cancellation**) επιτρέπει στο νήμα στόχο να ελέγχει περιοδικά αν πρέπει να ακυρωθεί

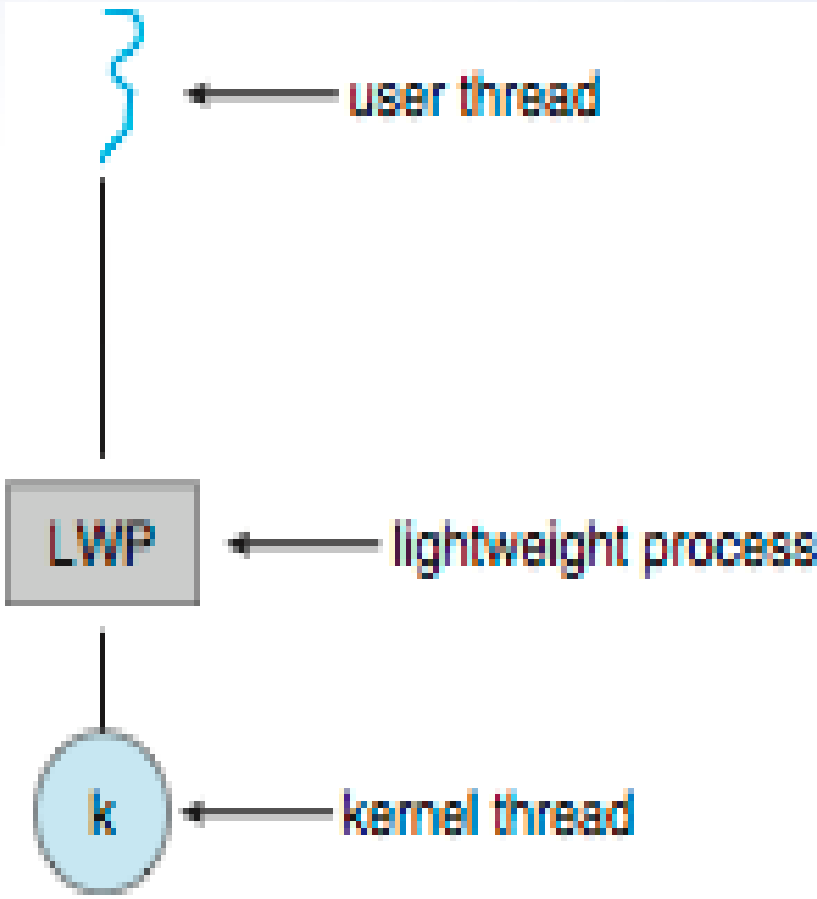
Τοπικός Χώρος αποθήκευσης Νημάτων



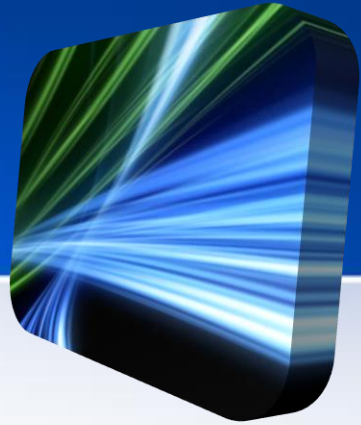
- Ο τοπικός χώρος αποθήκευσης νημάτων (**Thread-local storage TLS**) επιτρέπει σε κάθε νήμα να έχει το δικό του χώρο δεδομένων
- Χρήσιμο όταν δεν υπάρχει έλεγχος στη διαδικασία δημιουργίας νήματος (δηλ. Όταν χρησιμοποιείται μια δεξαμενή νημάτων)
- Διαφορετικός από τις τοπικές μεταβλητές
- Παρόμοιος με **`static`** δεδομένα
- Ο TLS είναι μοναδικός σε κάθε νήμα

Ενεργοποιήσεις χρονοπρογραμματιστή

- Τα μοντέλα νημάτων «πολλά προς πολλά» και «δύο επιπέδων» απαιτούν επικοινωνία για τον έλεγχο των νημάτων πυρήνα που διατίθενται στην εφαρμογή
- Συνήθως χρησιμοποιείται μια ενδιάμεση δομή (σαν **εικονικός επεξεργαστής**) μεταξύ του χρήστη και του πυρήνα του πυρήνα - **lightweight process** (**LWP**)

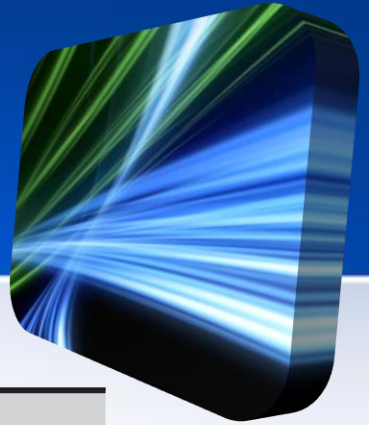


Νήματα στο Linux



- Στο Linux αναφέρονται ως **tasks** αντί για **threads**
- Η δημιουργία τους γίνεται με την κλήση συστήματος **`clone()`**
- Η **`clone()`** επιτρέπει σε μια διεργασία-παιδί να μοιραστεί το χώρο διευθύνσεων της γονικής διεργασίας

Νήματα στο Linux



flag	meaning
CLONE_FS	File-system information is shared.
CLONE_VM	The same memory space is shared.
CLONE_SIGHAND	Signal handlers are shared.
CLONE_FILES	The set of open files is shared.