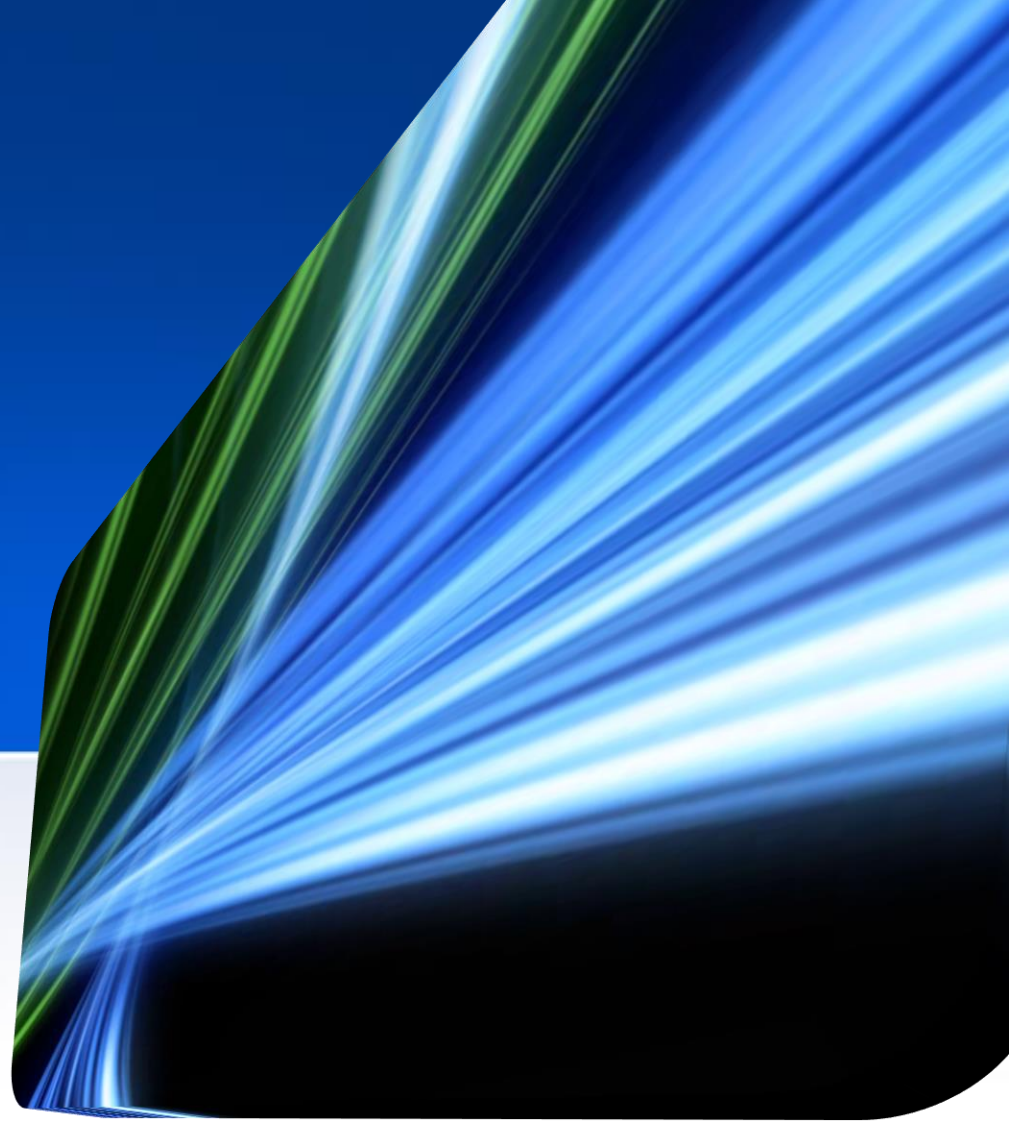
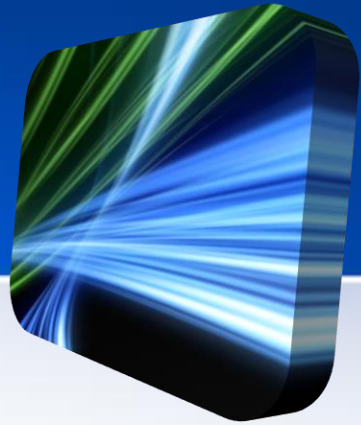
Λειτουργικά Συστήματα
6ο εξάμηνο ΣΗΜΜΥ
Ακ. έτος 2020-2021



Χρονοπρογραμματισμός CPU

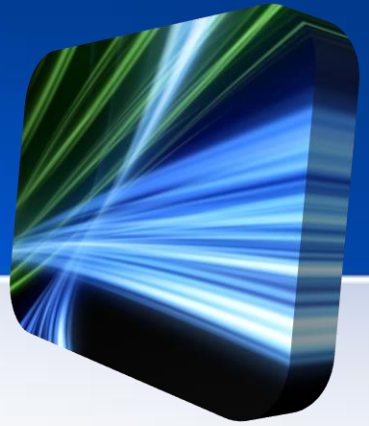
Παναγιώτης Τσανάκας

Στόχοι Παρουσίασης



- Βασικές **έννοιες** χρονοπρογραμματισμού
- **Κριτήρια** χρονοπρογραμματισμού
- **Αλγόριθμοι** χρονοπρογραμματισμού
- Χρονοπρογραμματισμός σε συστήματα **πολλαπλών επεξεργαστών**
- Χρονοπρογραμματισμός εφαρμογών **πραγματικού χρόνου**
- **Παραδείγματα** Λειτουργικών Συστημάτων
- **Αξιολόγηση** Αλγορίθμων

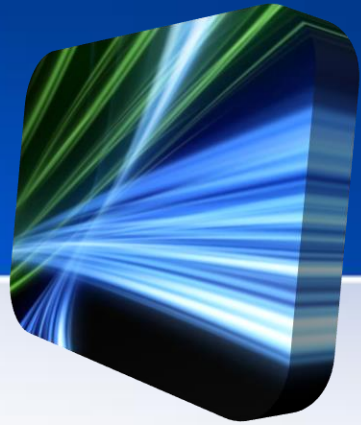
Βασικές έννοιες

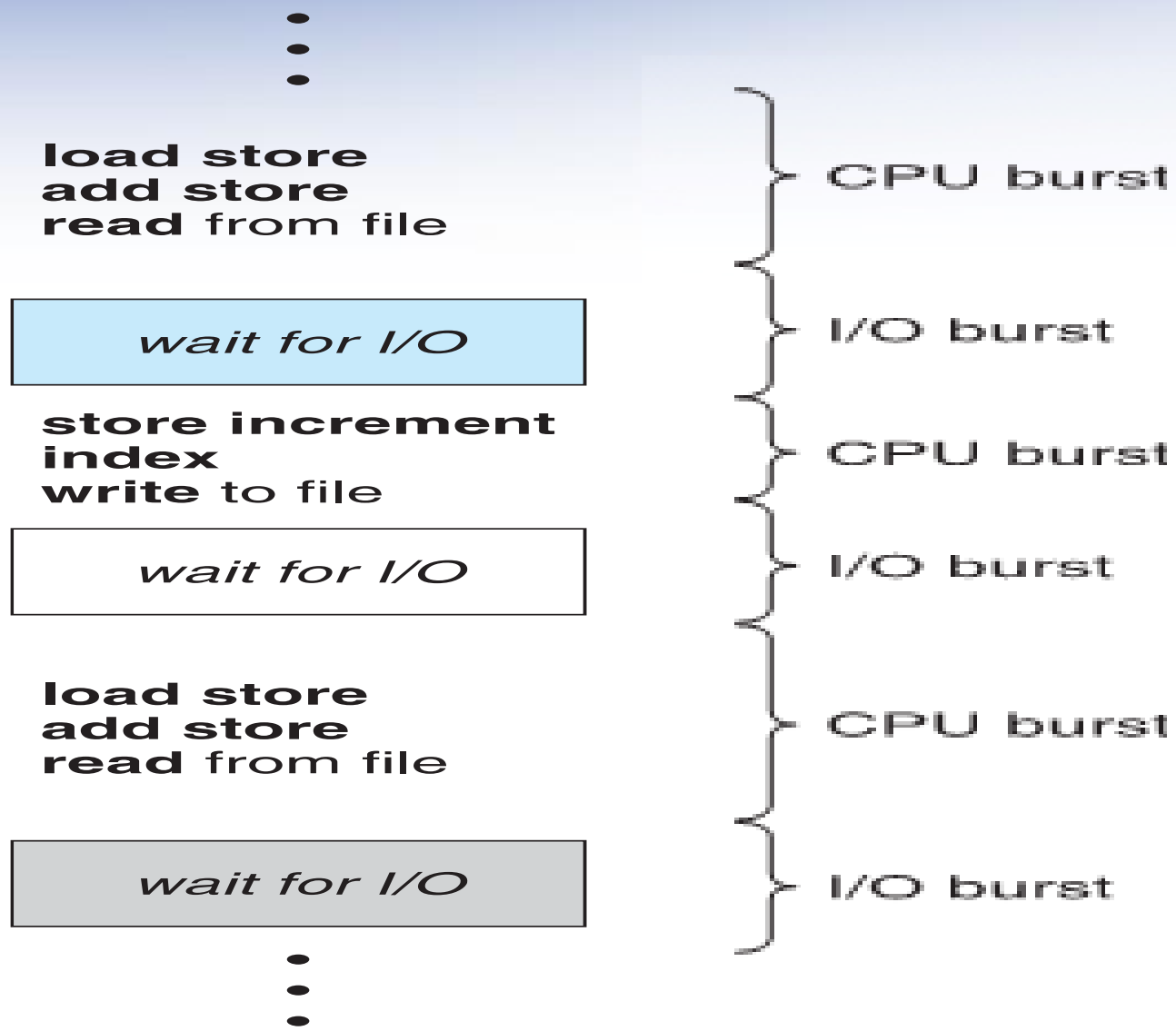


- Μέγιστη αξιοποίηση της CPU που επιτυγχάνεται με τον πολυπρογραμματισμό
- Κύκλοι CPU - I/O

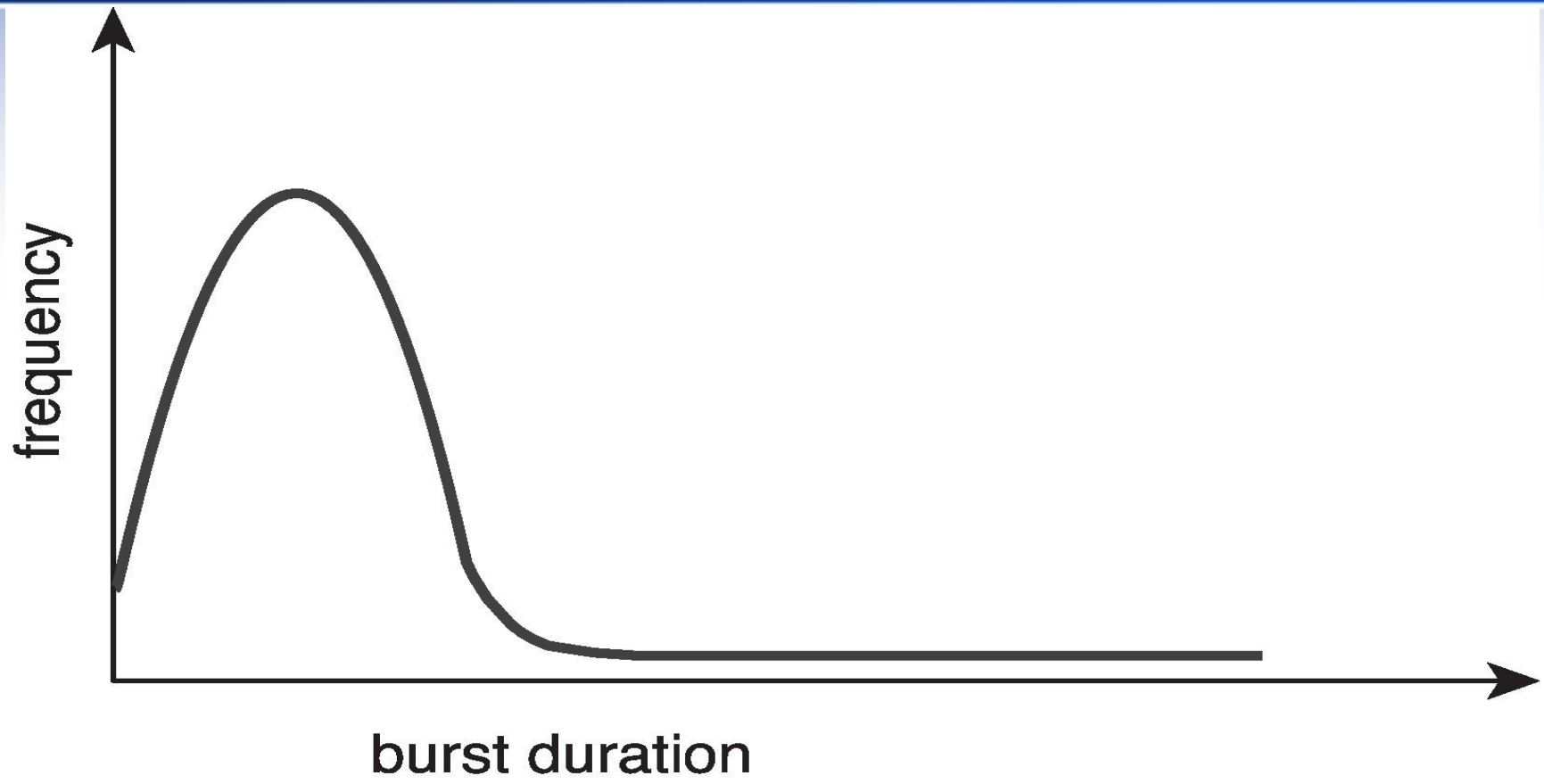
  Η εκτέλεση της διεργασίας αποτελείται από **κύκλους** με εκτέλεση CPU και αναμονή Ε/Ε
- **Ξέσπασμα CPU** (CPU burst) ακολουθείται από **ξέσπασμα Ε/Ε** (I/O Burst)

Βασικές έννοιες



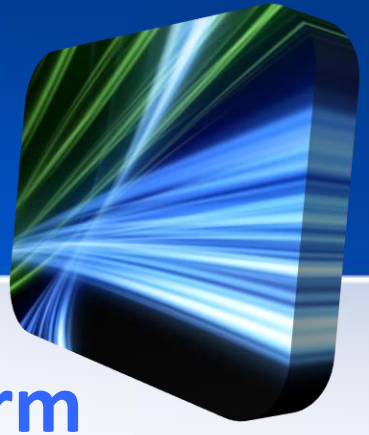


Ιστόγραμμα κύκλων CPU



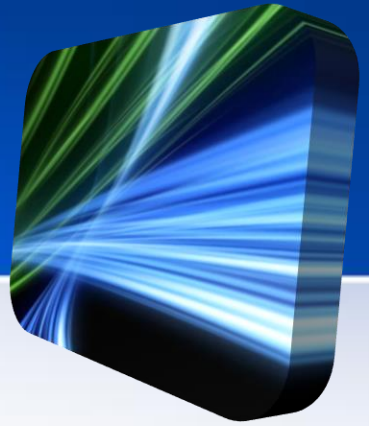
- **Πολλοί σύντομοι κύκλοι επεξεργασίας CPU**
- **Λίγοι μακροσκελείς κύκλοι επεξεργασίας CPU**

Χρονοπρογραμματιστής ΚΜΕ



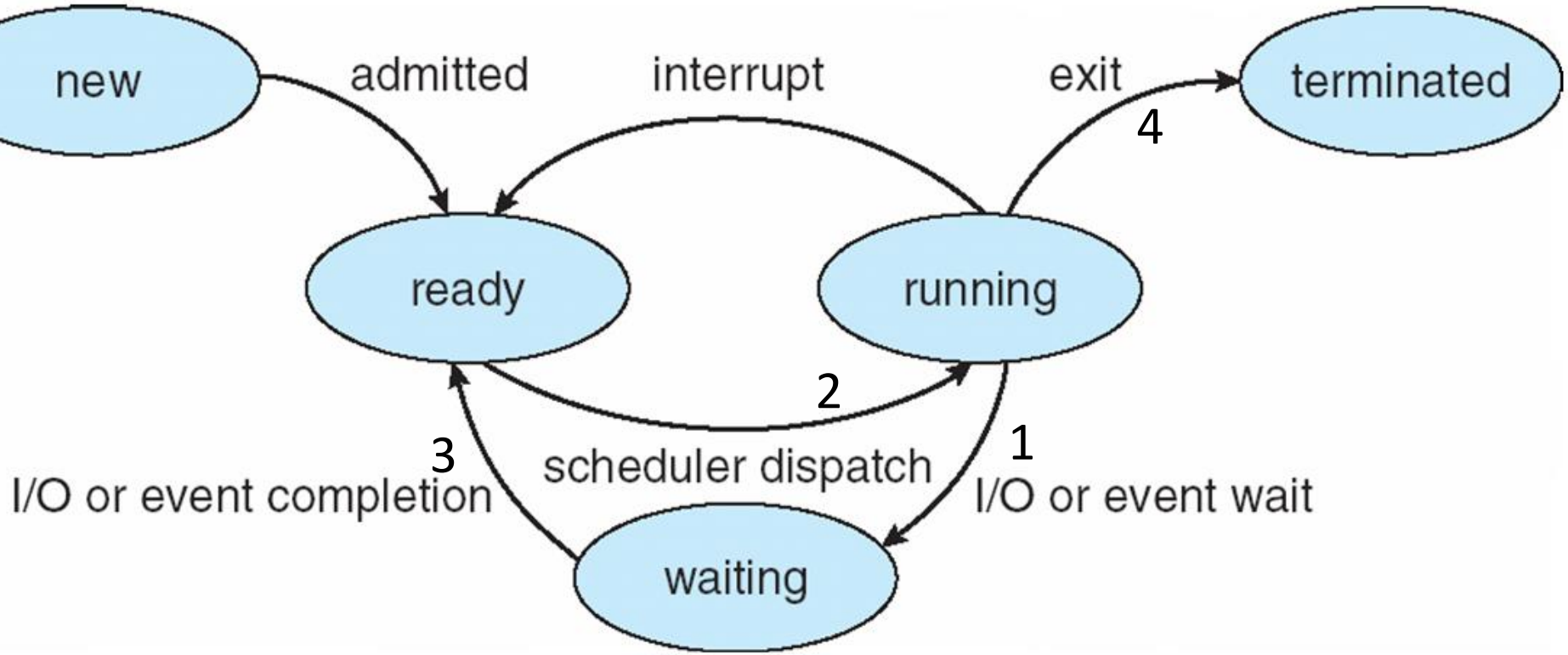
- Ο βραχυπρόθεσμος χρονοπρογραμματιστής (**Short-term scheduler**) επιλέγει μεταξύ των διεργασιών στην **ουρά έτοιμων διεργασιών** (**ready queue**) και αναθέτει την CPU σε μία από αυτές
- Η ουρά μπορεί να ταξινομηθεί με διάφορους τρόπους

Χρονοπρογραμματιστής CPU

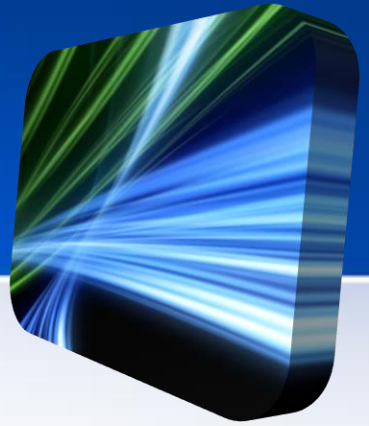


- Ο χρονοπρογραμματιστής CPU περεμβαίνει όταν μια διεργασία:
  1. Μεταβαίνει από κατάσταση εκτέλεσης (running state) στην κατάσταση αναμονής (waiting state)
  2. Μεταβαίνει από κατάσταση εκτέλεσης (running state) στην κατάσταση έτοιμη (ready state)
  3. Μεταβαίνει από κατάσταση αναμονής (waiting state) στην κατάσταση έτοιμη (ready state)
  4. Τερματίζει
- Ο χρονοπρογραμματισμός στις περιπτώσεις 1 και 4 είναι μονοσήμαντος. Στις 2 και 3, υπάρχει επιλογή.

Καταστάσεις Διεργασιών



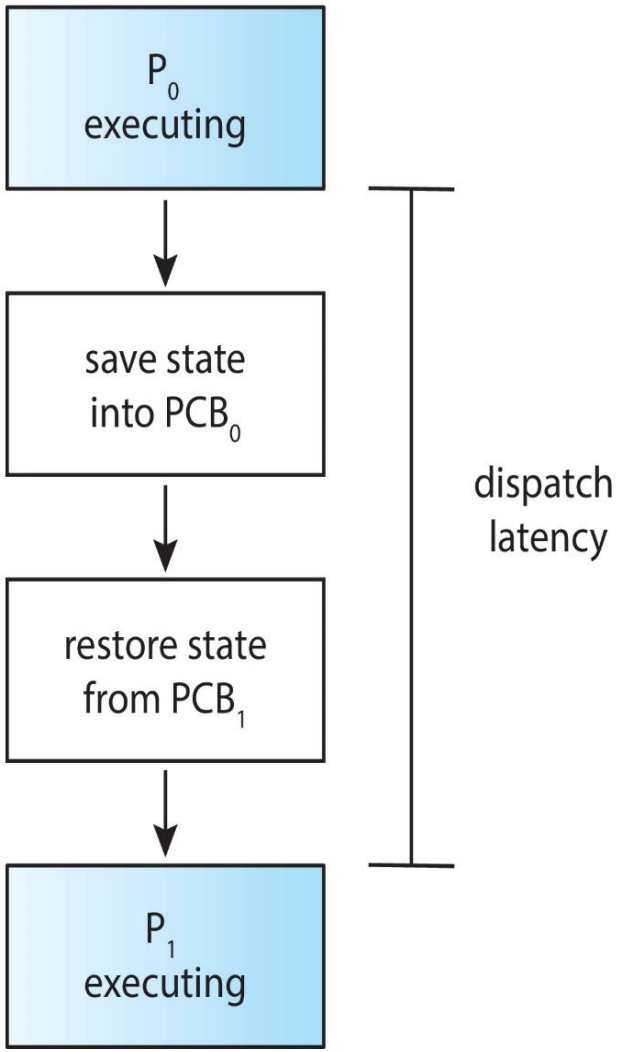
Χρονοπρογραμματιστής ΚΜΕ



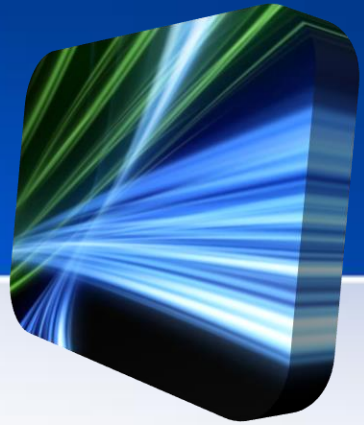
- Στις περιπτώσεις 2, 3, ο χρονοπρογραμματισμός είναι διακοπτός (**preemptive**)
  - Περίπτωση πρόσβασης στα κοινά δεδομένα (shared data) - Συνθήκες Ανταγωνισμού.
  - Περίπτωση λειτουργίας πυρήνα (kernel mode)
  - Περίπτωση διακοπών (interrupts) που προκύπτουν κατά τη διάρκεια δραστηριοτήτων του Λειτουργικού Συστήματος

Αλλαγή διεργασίας (Dispatcher)

- Δίνει τον έλεγχο της CPU σε άλλη διεργασία που επιλέχθηκε από τον βραχυπρόθεσμο Χ/Π:
  - Εναλλαγή περιβάλλοντος (context switching)
  - Εναλλαγή σε τρόπο λειτουργίας χρήστη (user-mode)
  - Μετάβαση στη σωστή θέση του προγράμματος (program counter register)

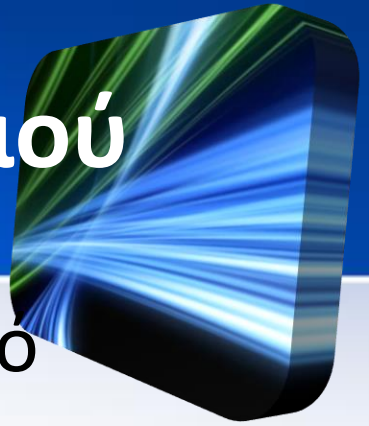


Αλλαγή διεργασίας (Dispatcher)



- Καθυστέρηση αλλαγής (**dispatch latency**) είναι ο χρόνος που απαιτείται για τον dispatcher να σταματήσει μια διεργασία και να ξεκινήσει μία άλλη

Κριτήρια Αξιολόγησης Χρονοπρογραμματισμού



- Χρησιμοποίηση Επεξεργαστή (**cpu utilization**): ποσοστό χρόνου που είναι απασχολημένη η ΚΜΕ
- Ρυθμός Διεκπεραίωσης (**throughput**): πλήθος διεργασιών που ολοκληρώνονται στη μονάδα χρόνου
- Χρόνος Ολοκλήρωσης (**turnaround time**): χρόνος ολοκλήρωσης μιας συγκεκριμένης διεργασίας
- Χρόνος Αναμονής (**waiting time**): χρόνος που μια διεργασία βρίσκεται σε κατάσταση αναμονής
- Χρόνος Απόκρισης (**response time**): χρόνος από την υποβολή ενός αιτήματος μέχρι να παραχθεί η πρώτη απόκριση

Βελτιστοποίηση Κριτηρίων Χρονοδρομολόγησης

- Μεγιστοποίηση
  - βαθμού χρησιμοποίησης ΚΜΕ
  - ρυθμού διεκπεραίωσης
- Ελαχιστοποίηση
  - χρόνου ολοκλήρωσης διεργασιών
  - Χρόνου αναμονής
  - χρόνου απόκρισης

First- Come First-Served (FCFS)

Διεργασία	Χρόνος Ξεσπάσματος CPU
P_1	24
P_2	3
P_3	3

Σειρά άφιξης P_1 , P_2 , P_3

Χρόνοι αναμονής: $P_1 = 0$, $P_2 = 24$, $P_3 = 27$

Μέσος Χρόνος Αναμονής: (0 + 24 + 27)/3 = 17

First- Come First-Served (FCFS)

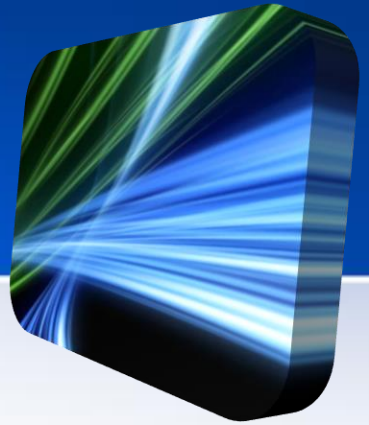
Διεργασία	Χρόνος Ξεσπάσματος
P_1	24
P_2	3
P_3	3

Σειρά άφιξης P_2 , P_3 , P_1

Χρόνοι αναμονής: P_1 = 6, P_2 = 0, P_3 = 3

Μέσος Χρόνος Αναμονής: (6 + 0 + 3)/3 = 3

First- Come First-Served (FCFS)

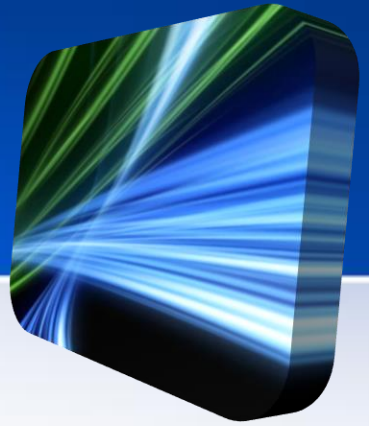


- Στην δεύτερη περίπτωση έχουμε σημαντικά χαμηλότερο Μέσο Χρόνο Αναμονής

- Φαινόμενο Φάλαγγας (**convoy effect***):*

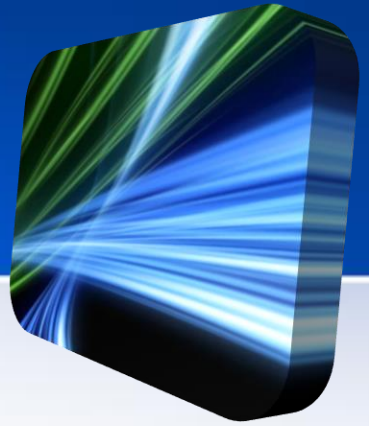
όταν μικρές διεργασίες είναι **πίσω** από μεγάλες

Shortest-Job-First (SJF)

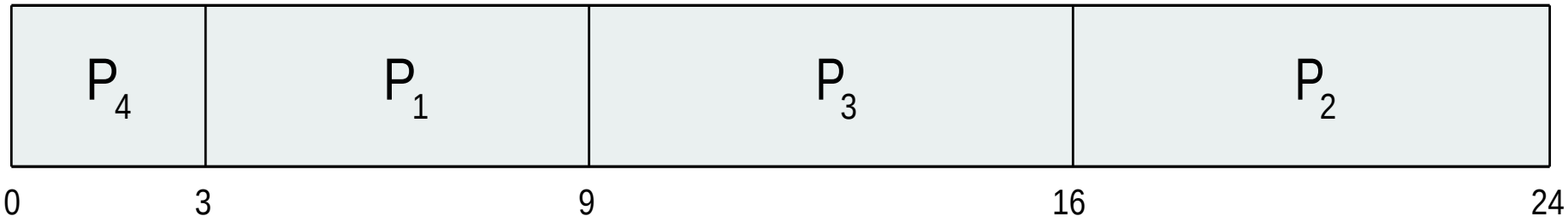


- Σύνδεση κάθε διεργασίας με βάση την διάρκεια του επόμενου ξεσπάσματος CPU
- Χρησιμοποίηση αυτού του χρόνου για να επιλεγεί η διεργασία με τον συντομότερο χρόνο CPU
- Ο μέθοδος SJF είναι βέλτιστη - δίνει τον **ελάχιστο μέσο χρόνο αναμονής** για ένα δεδομένο σύνολο διεργασιών
- Η δυσκολία είναι να **γνωρίζουμε** τη διάρκεια τού επόμενου αιτήματος χρήσης CPU –
  - **Να ζητηθεί από τον χρήστη**
  - **Να γίνει πρόβλεψη**

Παράδειγμα SJF



Διεργασία	Χρόνος Ξεσπάσματος CPU
P_1	6
P_2	8
P_3	7
P_4	3



Μέσος Χρόνος Αναμονής: (3 + 16 + 9 + 0) / 4 = 7

Εκτίμηση επόμενου ξεσπάσματος CPU

- Δεν είναι γνωστή η διάρκεια του επόμενου ξεσπάσματος κάθε διεργασίας
- Μπορεί να γίνει μόνο μια **εκτίμηση,** στηριζόμενοι στην διάρκεια του προηγούμενου ξεσπάσματος
  - επιλογή διεργασίας με την **μικρότερο εκτιμώμενο** ξέσπασμα CPU
  - μεθοδολογία του **Εκθετικού μέσου όρου**

Υπολογισμός εκτίμησης διάρκειας επόμενου ξεσπάσματος

Εκθετικός μέσος όρος:

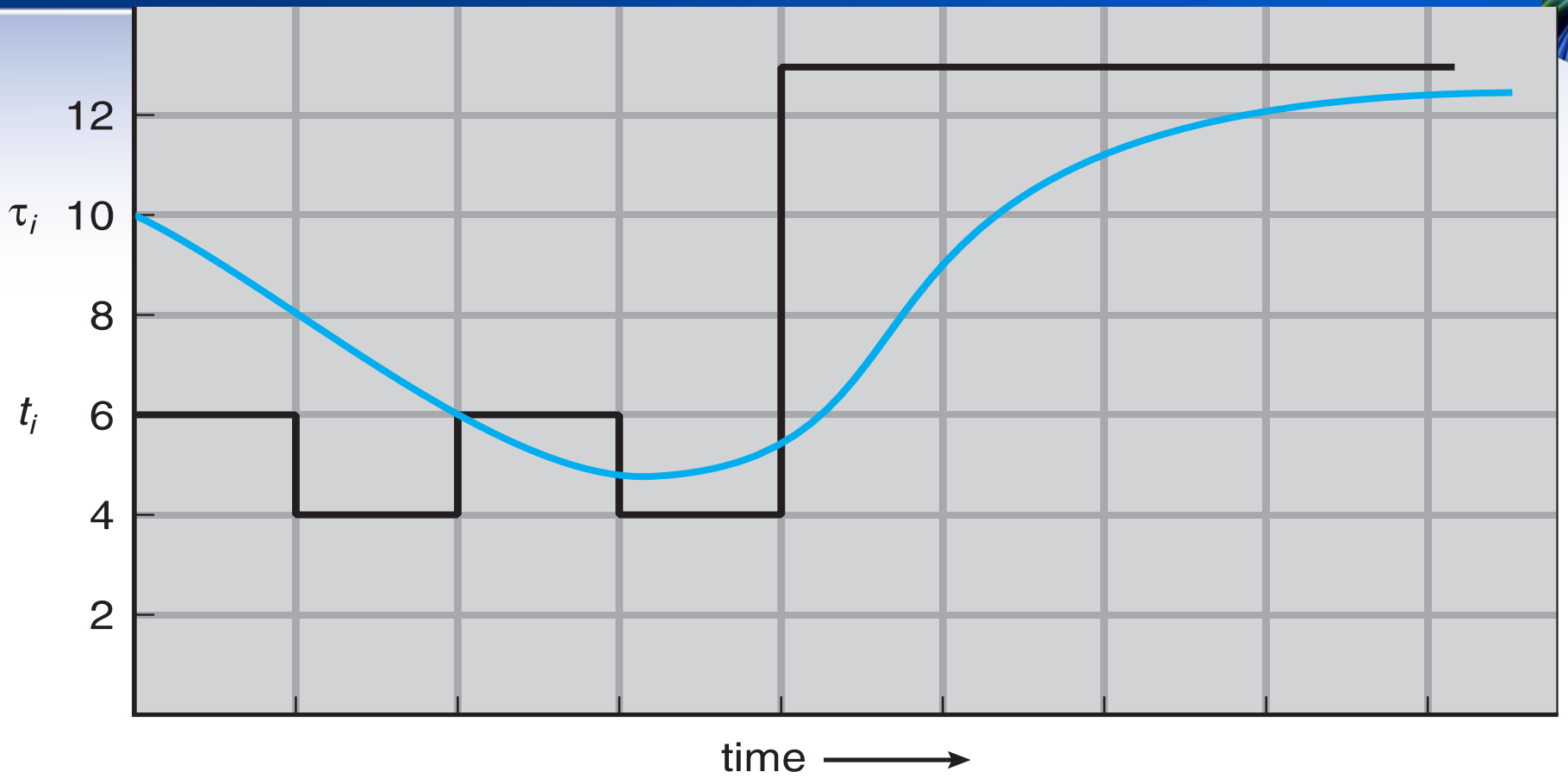
$$\tau_{n+1} = \alpha\, t_n + (1-\alpha)\tau_n$$

$t_n = \text{πραγματικό μέγεθος } n^{th} \text{ CPU ξεσπάσματος}$

$\tau_{n+1} = \text{εκτιμώμενο επόμενο CPU ξέσπασμα}$

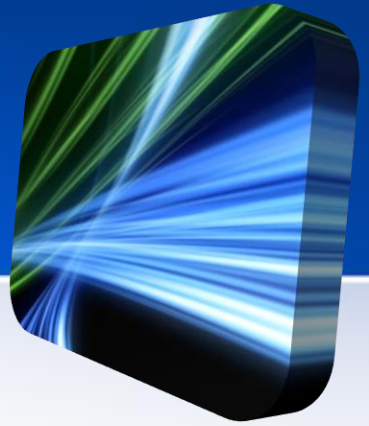
$\alpha, 0 \le \alpha \le 1, \text{ συνήθως } 0.5$

Εκτίμηση επόμενου ξεσπάσματος



CPU burst (t_i)		6	4	6	4	13	13	13	...
"guess" (τ_i)	10	8	6	6	5	9	11	12	...

Μορφές SJF

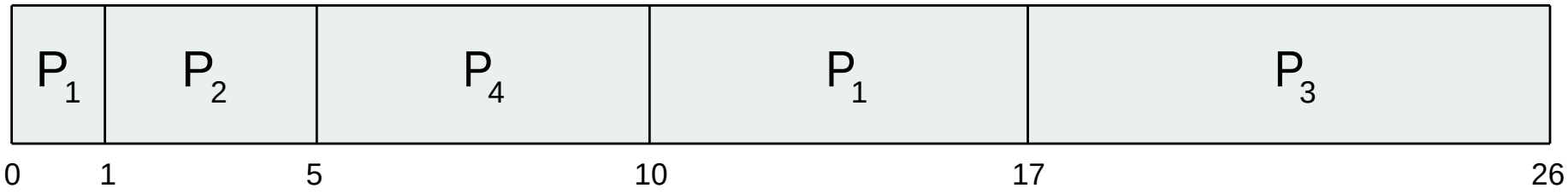


- **Non preemptive** (Μη-διακοπτός)
  Όταν παραχωρηθεί η CPU σε μία διεργασία, θα πρέπει να ολοκληρώσει τη χρήση της CPU, έως ότου παρθεί νέα απόφαση χρονοδρομολόγησης
- **Preemptive** (Διακοπτός)
  Αν εισέλθει στην ουρά έτοιμων διεργασιών μια **νέα** διεργασία με **μικρότερο** αναμενόμενο χρόνο εκτέλεσης από τον εναπομείναντα της τρέχουσας, η τρέχουσα θα αντικατασταθεί.

Ο διακοπτός SJF αλγόριθμος λέγεται **shortest-remaining-time-first**

Shortest-remaining-time-first

Διεργασία	Χρόνος Άφιξης	Χρόνος Ξεσπάσματος
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

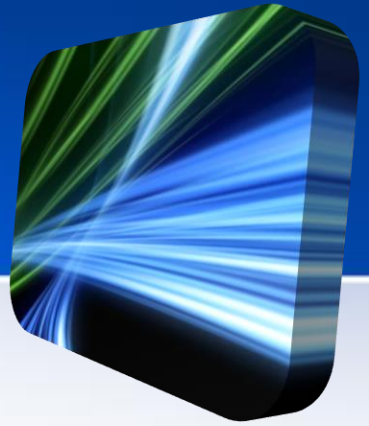


Μέσος Χρόνος Αναμονής: [(10-1)+(1-1)+(17-2)+(5-3)]/4 = 26/4 = 6.5

Χρονοπρογραμματισμός με Προτεραιότητες Priority Scheduling

- Σε κάθε διεργασία αντιστοιχίζεται ένας ακέραιος αριθμός **προτεραιότητας**
- Επιλέγεται η διεργασία με την υψηλότερη προτεραιότητα (μικρότερος ακέραιος)
  - Διακοπτός
  - Μη-διακοπτός
- SJF Ειδική περίπτωση χρονοπρογραμματισμού με προτεραιότητες σύμφωνα με το εκτιμώμενο επόμενο ξέσπασμα CPU

Χρονοπρογραμματισμός με Προτεραιότητες Priority Scheduling



- Πρόβλημα **Λιμοκτονίας (Starvation)**

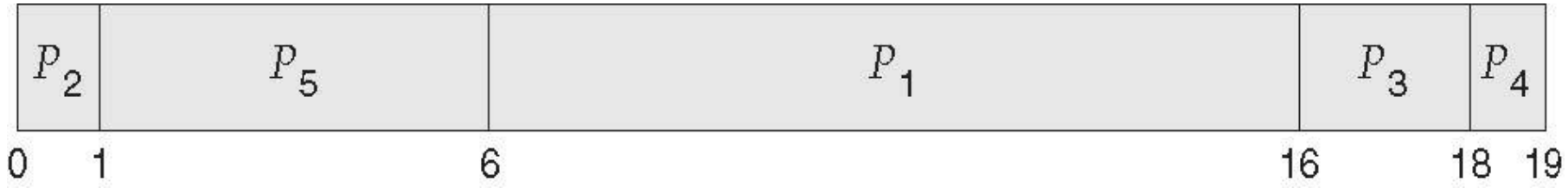
  Διεργασίες με χαμηλή προτεραιότητα μπορεί να μην εκτελεσθούν ποτέ

- Λύση μέσω **Γήρανσης (Aging)**

  Με την πάροδο του χρόνου μεγαλώνει η προτεραιότητα των διεργασιών που δεν εκτελούνται

Priority Scheduling

Διεργασία	Χρόνος Ξεσπάσματος	Προτεραιότητα
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

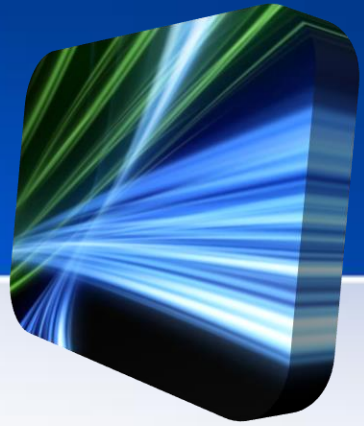


Μέσος Χρόνος Αναμονής = 8.2

Round Robin - RR…

- Κάθε διεργασία λαμβάνει μια μικρή μονάδα χρόνου CPU (**time quantum** *q*), συνήθως 10-100 ms. Μετά την πάροδο αυτού του χρόνου, η διαδικασία **διακόπτεται** και τοποθετείται στο τέλος της ουράς.
  - Εάν υπάρχουν **n** διεργασίες στην ουρά και
  - …time quantum είναι **q**,
  - …κάθε διαδικασία παίρνει **1/n** του χρόνου τής CPU
  - …σε κομμάτια χρόνου ίσα με το q.

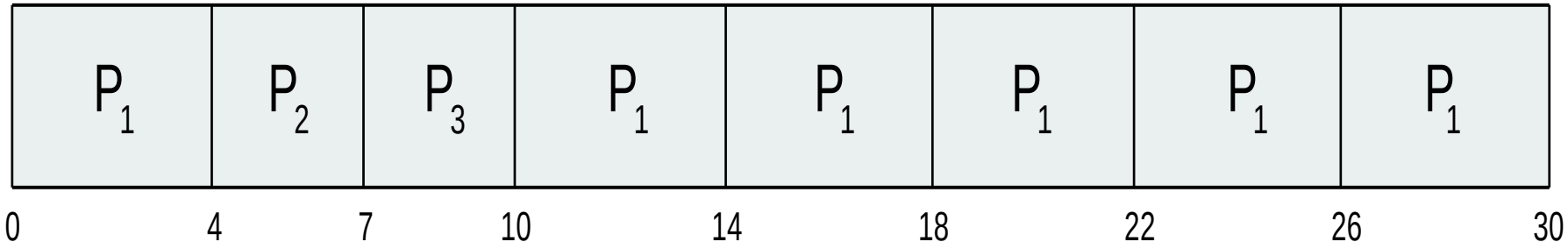
Round Robin - RR



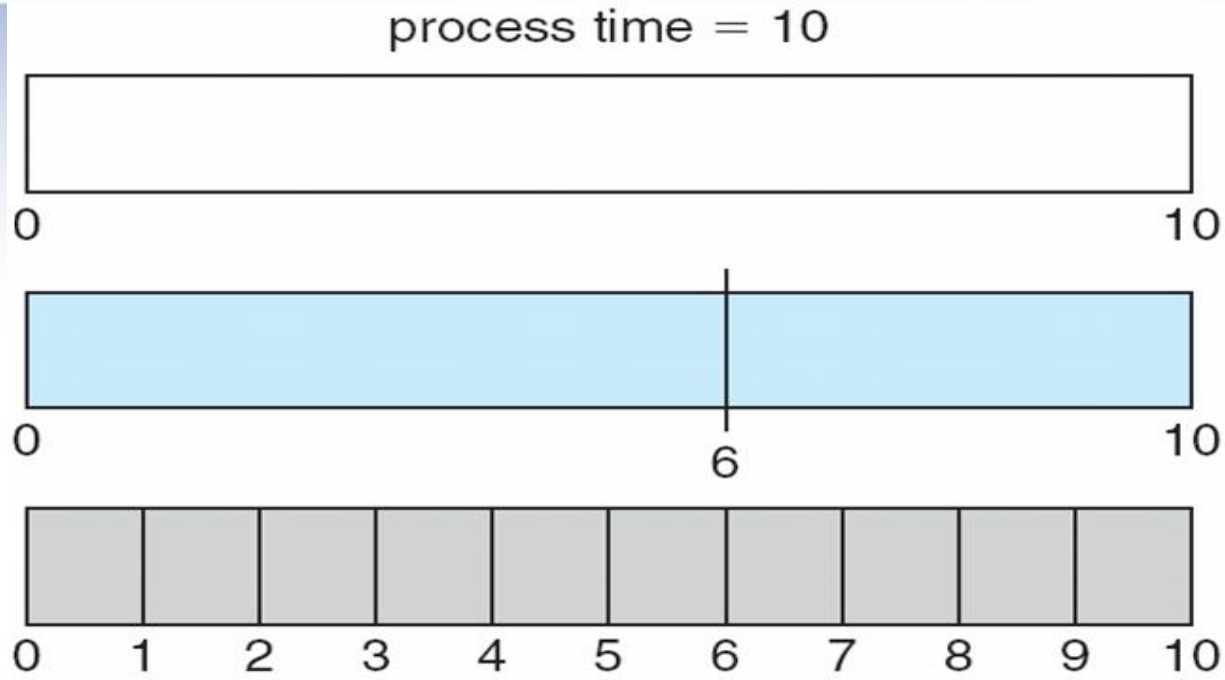
- Καμία εργασία δεν περιμένει περισσότερο από (n-1) q μονάδες χρόνου.
- Ο χρονοπρογραμματιστής διακόπτει κάθε q μονάδες χρόνου, για να δρομολογήσει την επόμενη εργασία
- Απόδοση
  - q μεγάλο ⟶ FIFO
  - q μικρό⟶ Το σύστημα αναλώνεται σε εναλλαγές περιεχομένου (context switch).
  - Το q πρέπει να είναι μεγάλο σε σχέση με τη διάρκεια του context switching. Διαφορετικά, η χρονική επιβάρυνση είναι μεγάλη

RR με Time Quantum = 4

Διεργασία	Χρόνος Ξεσπάσματος
P_1	24
P_2	3
P_3	3



Κβάντο χρόνου και Μεταγωγή περιβάλλοντος

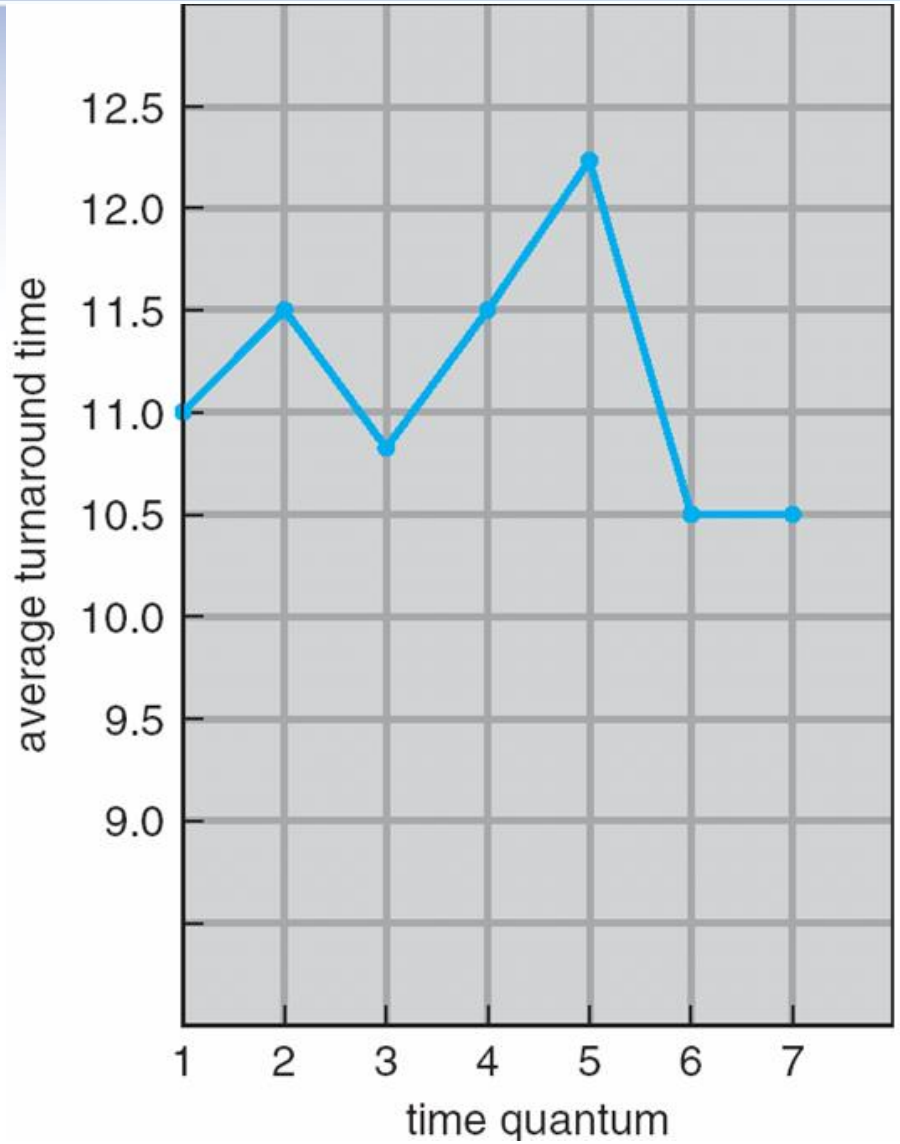


quantum	context switches
12	0
6	1
1	9

Το κβάντο χρόνου πρέπει να είναι (σημαντικά) μεγαλύτερο από τον χρόνο μεταγωγής(Context Switch), γενικά:

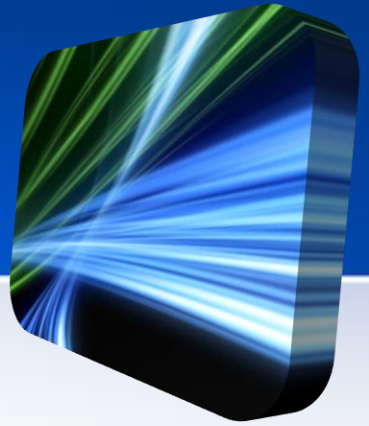
- Context Switch < 10μs
- Time Quantum : 10 έως 100 ms

Ο χρόνος ολοκλήρωσης εξαρτάται από το Time Quantum



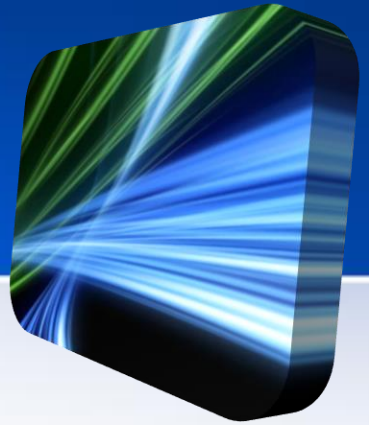
process	time
P_1	6
P_2	3
P_3	1
P_4	7

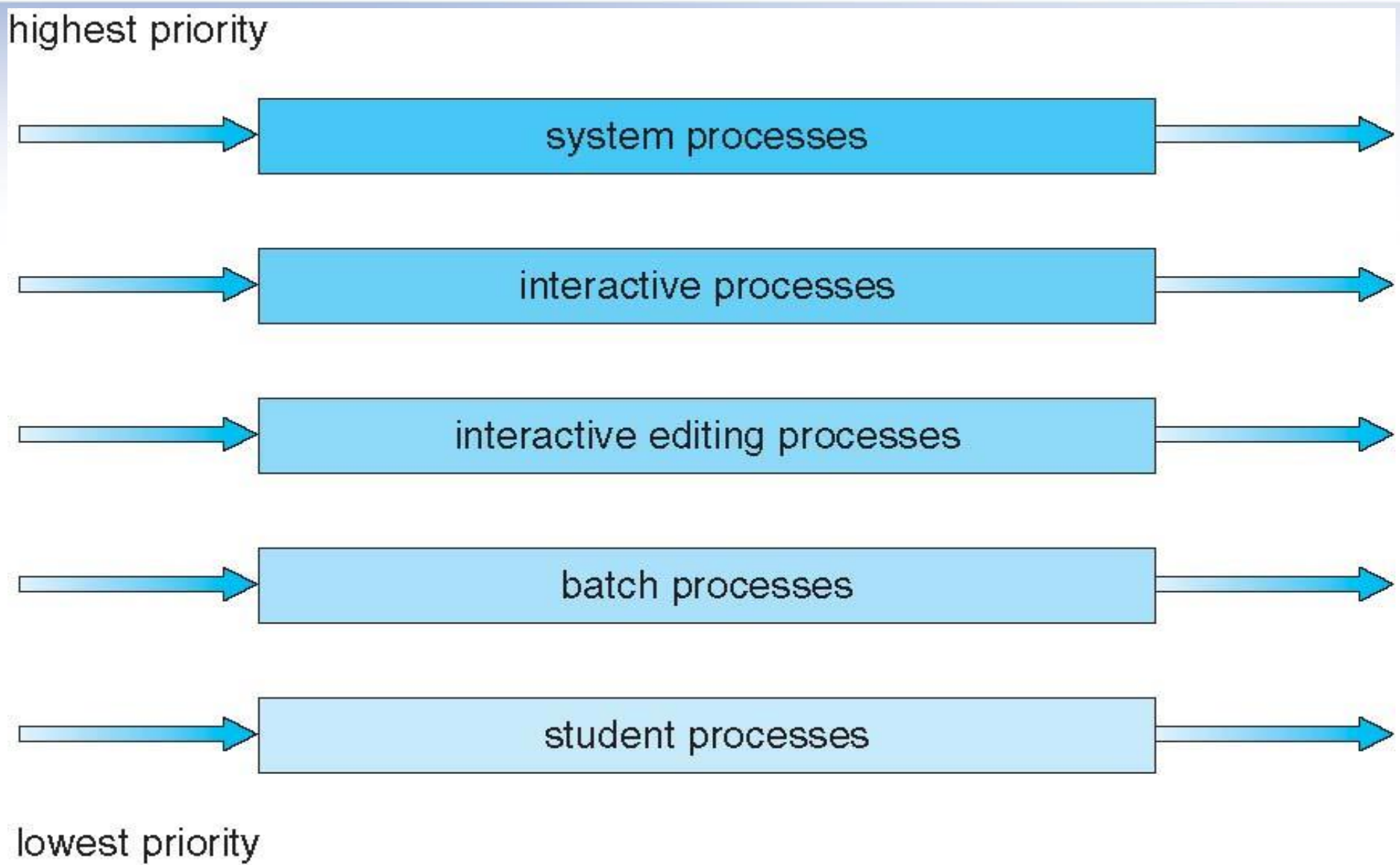
Πολυεπίπεδες Ουρές (Multilevel Queue)



- Η ουρά με τις έτοιμες διεργασίες χωρίζεται σε πολλαπλές ουρές,
  - Ουρά προσκηνίου **foreground** (interactive)
  - Ουρά παρασκηνίου **background** (batch)
- Κάθε ουρά έχει δικό της αλγόριθμο δρομολόγησης:
  - Ουρά προσκηνίου– RR
  - Ουρά παρασκηνίου– FCFS
- Χρονοπρογραμματισμός γίνεται μεταξύ των ουρών
  - Με Προτεραιότητες (πιθανότητα λιμοκτονίας)
  - Κάθε ουρά λαμβάνει ποσοστό χρήσης της ΚΜΕ

Πολυεπίπεδες Ουρές (Multilevel Queue)

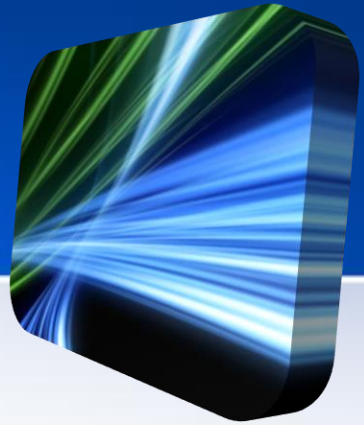




Πολυεπίπεδες Ουρές με ανατροφοδότηση

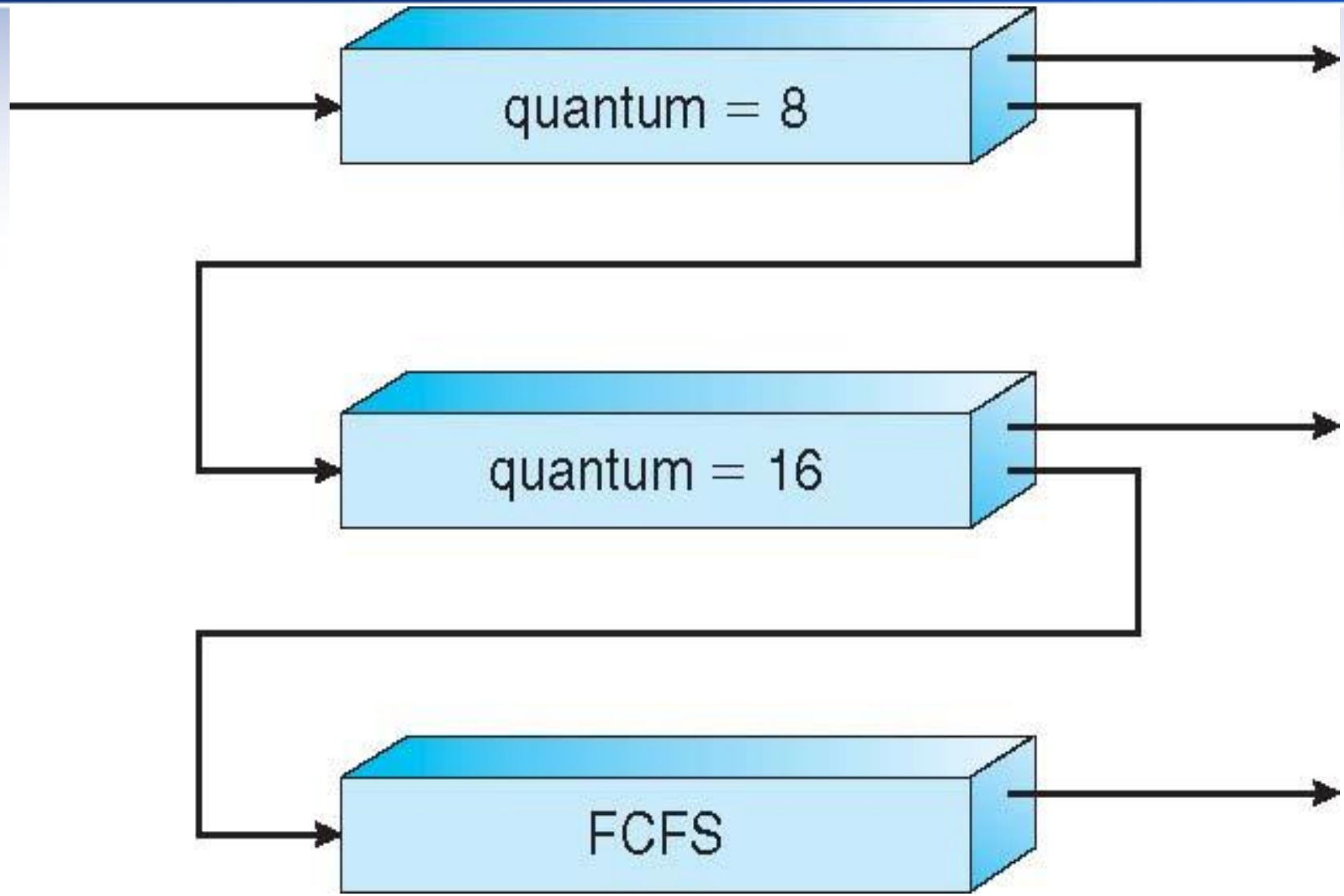
- Μετακίνηση διεργασιών μεταξύ διαφορετικών ουρών (με τον τρόπο αυτό επιτυγχάνεται η **γήρανση**)
- Αναγκαίες παράμετροι:
  - Πλήθος ουρών
  - Αλγόριθμος χρονοπρογραμματισμού για κάθε ουρά
  - Μέθοδος αναβάθμισης διεργασιών
  - Μέθοδος υποβάθμισης διεργασιών
  - Μέθοδος επιλογής ουράς για κάθε διεργασία

Πολυεπίπεδων Ουρών με ανατροφοδότηση

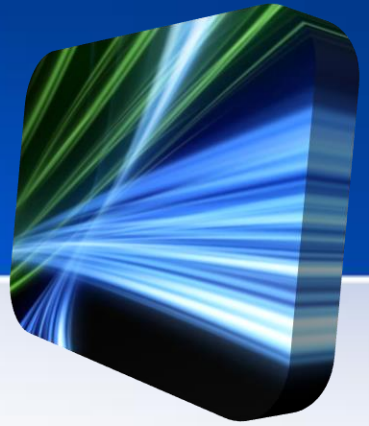


- Τρείς ουρές:
  - Q_0 – RR με time quantum 8 milliseconds
  - Q_1 – RR me time quantum 16 milliseconds
  - Q_2 – FCFS
- Χρονοπρογραμματισμός:
  - Νέα διεργασία εισέρχεται στην Q_0
  - Αν δεν ολοκληρωθεί σε 8ms μεταφέρεται στην Q_1
  - Αν δεν ολοκληρωθεί σε 16ms (8+16=24) μεταφέρεται στην Q_2

Πολυεπίπεδες Ουρές με ανατροφοδότηση

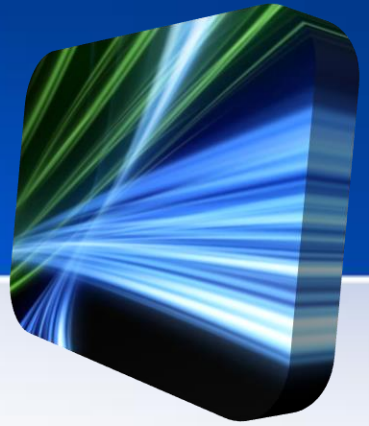


Πεδίο Ανταγωνισμού Νημάτων



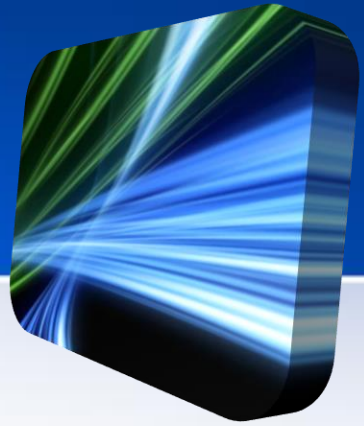
- Νήματα Χρηστών – Νήματα Πυρήνα
- Πολλά-προς-Ένα, Πολλά-προς-Πολλά / LWP
  - Ανταγωνισμός εντός Διεργασίας (PCS- Process Contention Scope)
  - PTHREAD_SCOPE_PROCESS
  - Προτεραιότητα από τον προγραμματιστή
- Νήματα Πυρήνα
  - Ανταγωνισμός εντός συστήματος (SCS- System Contention Scope)
  - PTHREAD_SCOPE_SYSTEM
  - Ένα-προς-ένα

Pthread Scheduling API



```
#define NUM_THREADS 5
int main(int argc, char *argv[]) {
   int i, scope;
   pthread_t tid[NUM_THREADS];
   pthread_attr_t attr;
   /* get the default attributes */
   pthread_attr_init(&attr);
   /* first inquire on the current scope */
   if (pthread_attr_getscope(&attr, &scope) != 0)
      fprintf(stderr, "Unable to get scheduling scope\n");
   else {
      if (scope == PTHREAD_SCOPE_PROCESS)
         printf("PTHREAD_SCOPE_PROCESS");
      else if (scope == PTHREAD_SCOPE_SYSTEM)
         printf("PTHREAD_SCOPE_SYSTEM");
      else fprintf(stderr, "Illegal scope value.\n"); }
```

Pthread Scheduling API



```
   /* set the scheduling algorithm to PCS or SCS */
   pthread_attr_setscope(&attr, PTHREAD_SCOPE_SYSTEM);
   /* create the threads */
   for (i = 0; i < NUM_THREADS; i++)
      pthread_create(&tid[i],&attr,runner,NULL);
   /* now join on each thread */
   for (i = 0; i < NUM_THREADS; i++)
      pthread_join(tid[i], NULL);
}
/* Each thread will begin control in this function */
void *runner(void *param)
{
   /* do some work ... */
   pthread_exit(0); }
```

Συστήματα Πολλαπλών Επεξεργαστών

- Ο χρονοπρογραμματισμός είναι πιο περίπλοκος, όταν έχουμε πολλαπλούς επεξεργαστές.
- **Ασύμμετρος** (Asymmetric multiprocessing)
  - Ο Χρονοπρογραμματισμός τρέχει **σε έναν** επεξεργαστή
  - Οι υπόλοιποι χρησιμοποιούνται για εκτέλεση κώδικα χρήστη
- **Συμμετρικός** (Symmetric multiprocessing)
  - Ο Χρονοπρογραμματισμός τρέχει σε **όλους** τους επεξεργαστές
  - Κάθε χρονοπρογραμματιστής επιλέγει διεργασία προς εκτέλεση στον αντίστοιχο επεξεργαστή
  - Απαιτείται συγχρονισμός

Χρονοπρογραμματισμός Συστήματα Πολλαπλών Επεξεργαστών

- **Προσκόλληση σε Επεξεργαστή** (processor affinity)

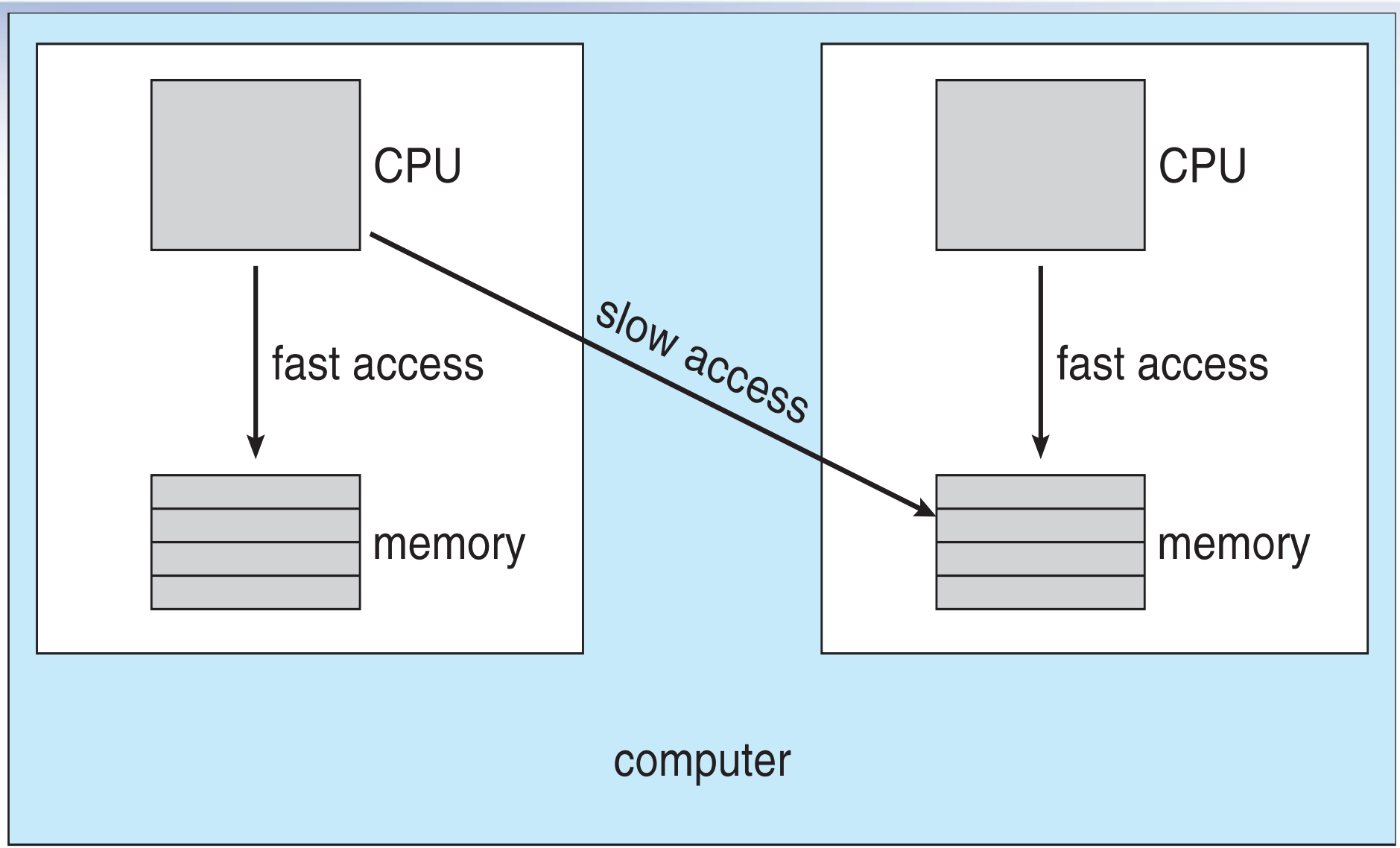
Επίδοση κρυφής μνήμης

- **Εξισορρόπηση Φορτίου** (load balancing)

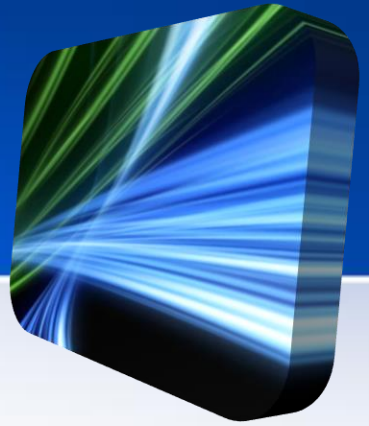
Απαιτεί μετακίνηση διεργασιών

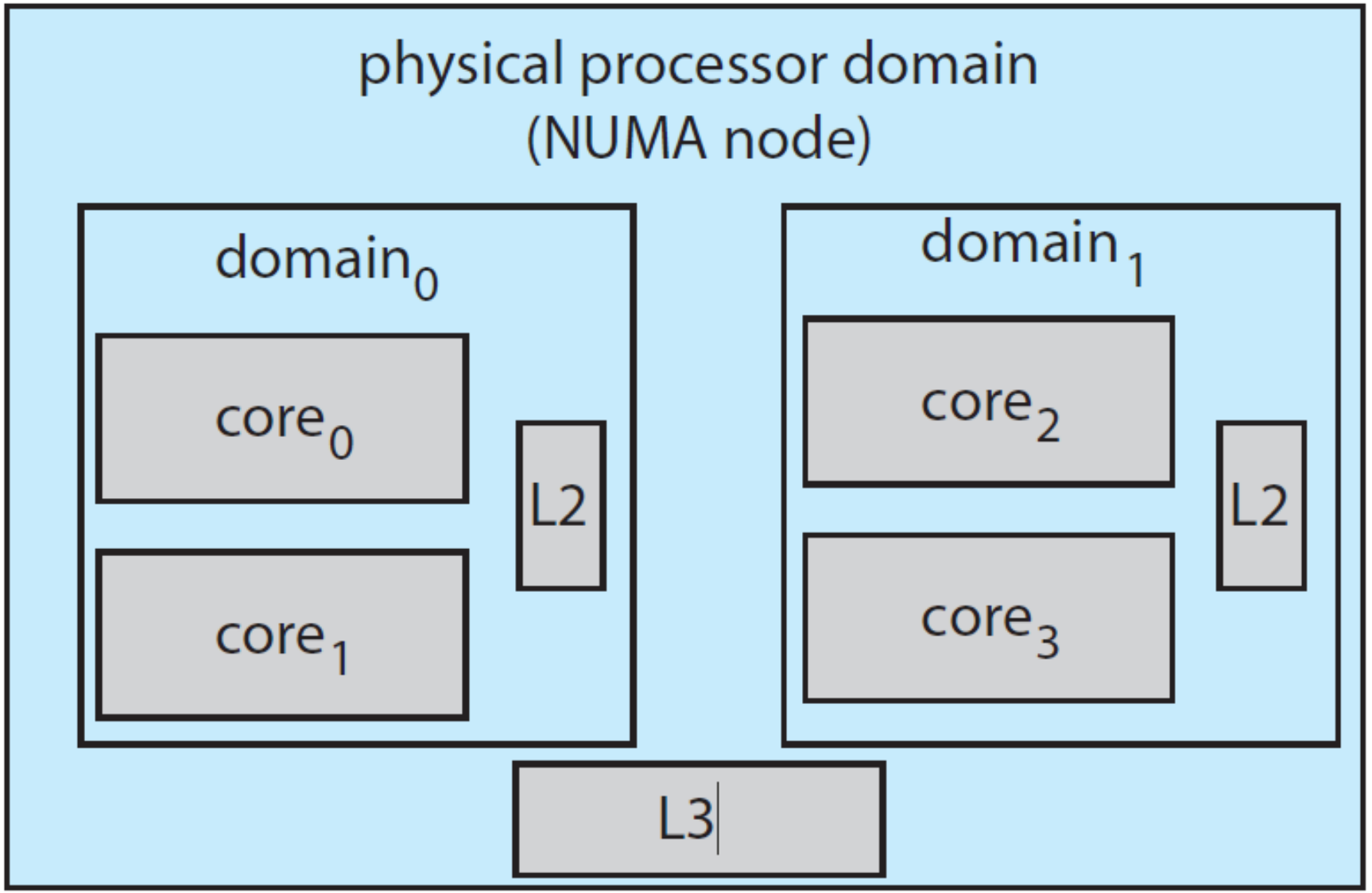
- **Μετακίνηση ώθησης** (push migration)
- **Μετακίνηση έλξης** (pull migration)
- Γνώση φυσικής τοπολογίας συστήματος, Πχ NUMA, SMT

NUMA and CPU Scheduling

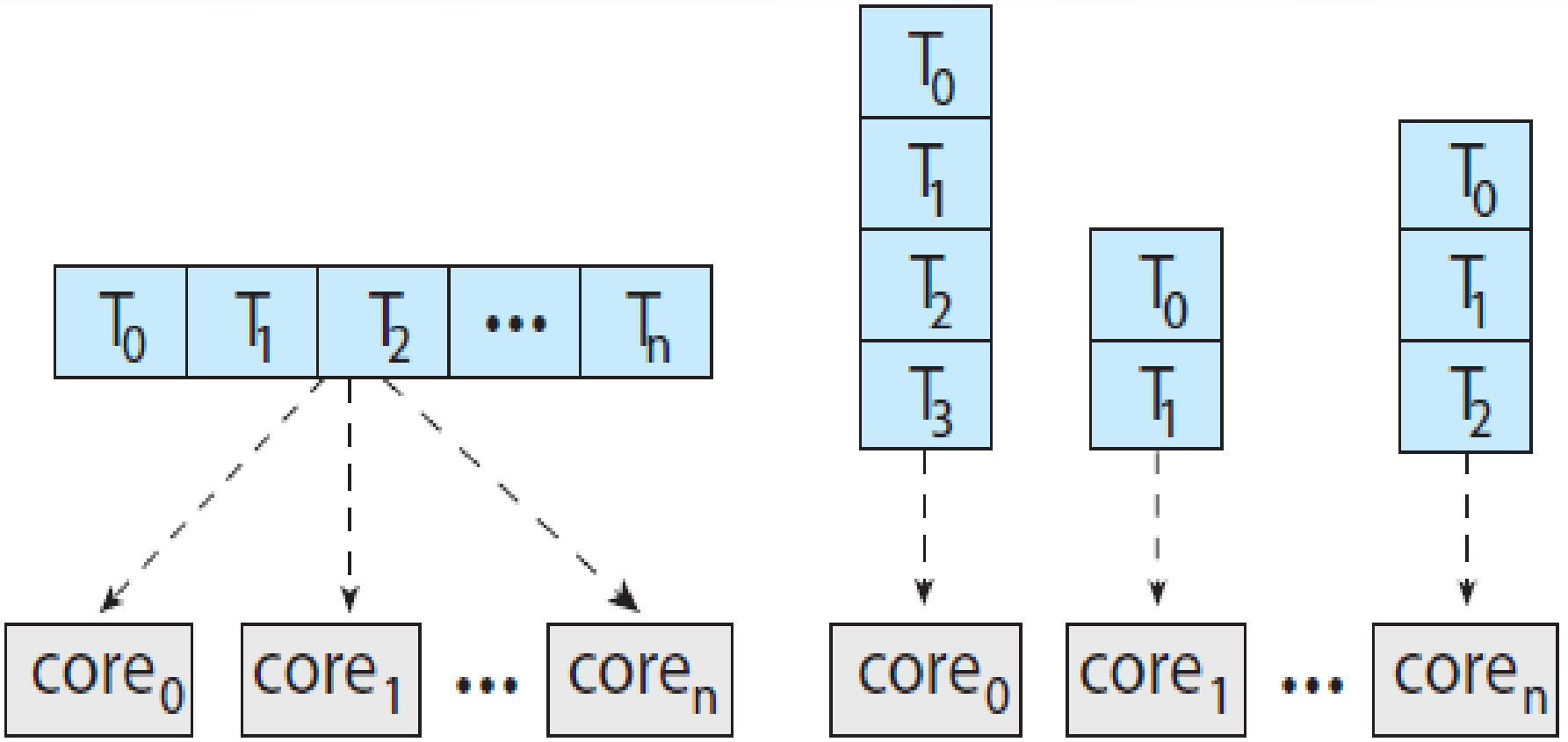


NUMA and CPU Scheduling



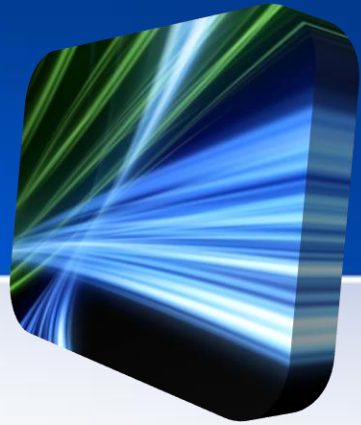


Πολύ-πύρηνοι επεξεργαστές

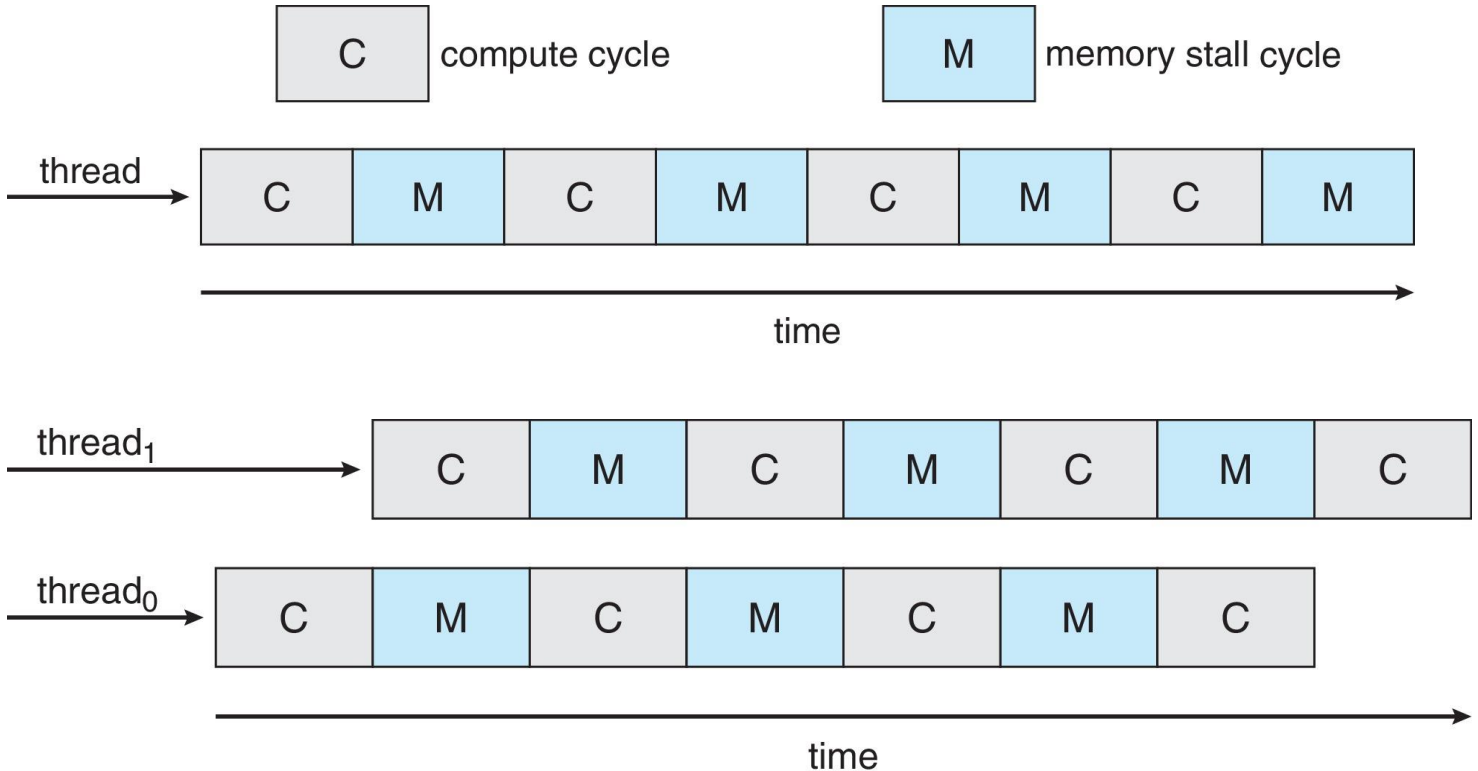


Κοινή/Ανεξάρτητες λίστες έτοιμων διεργασιών

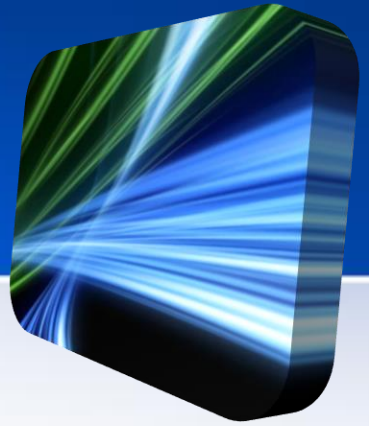
Πολλαπλά νήματα/πυρήνα



- Πολλαπλοί πυρήνες ανα CPU
- Λιγότερη κατανάλωση ενέργειας
- Πολλαπλά νήματα ανά πυρήνα
- Επικάλυψη καθυστερήσεων μνήμης (Memory Stall)



Συστήματα Πραγματικού Χρόνου

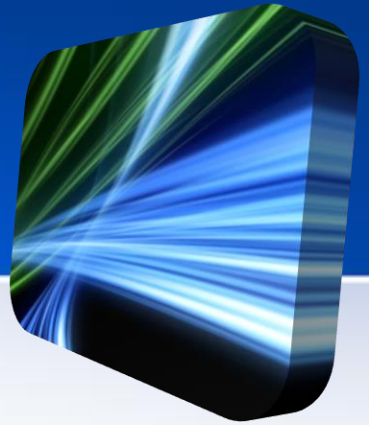


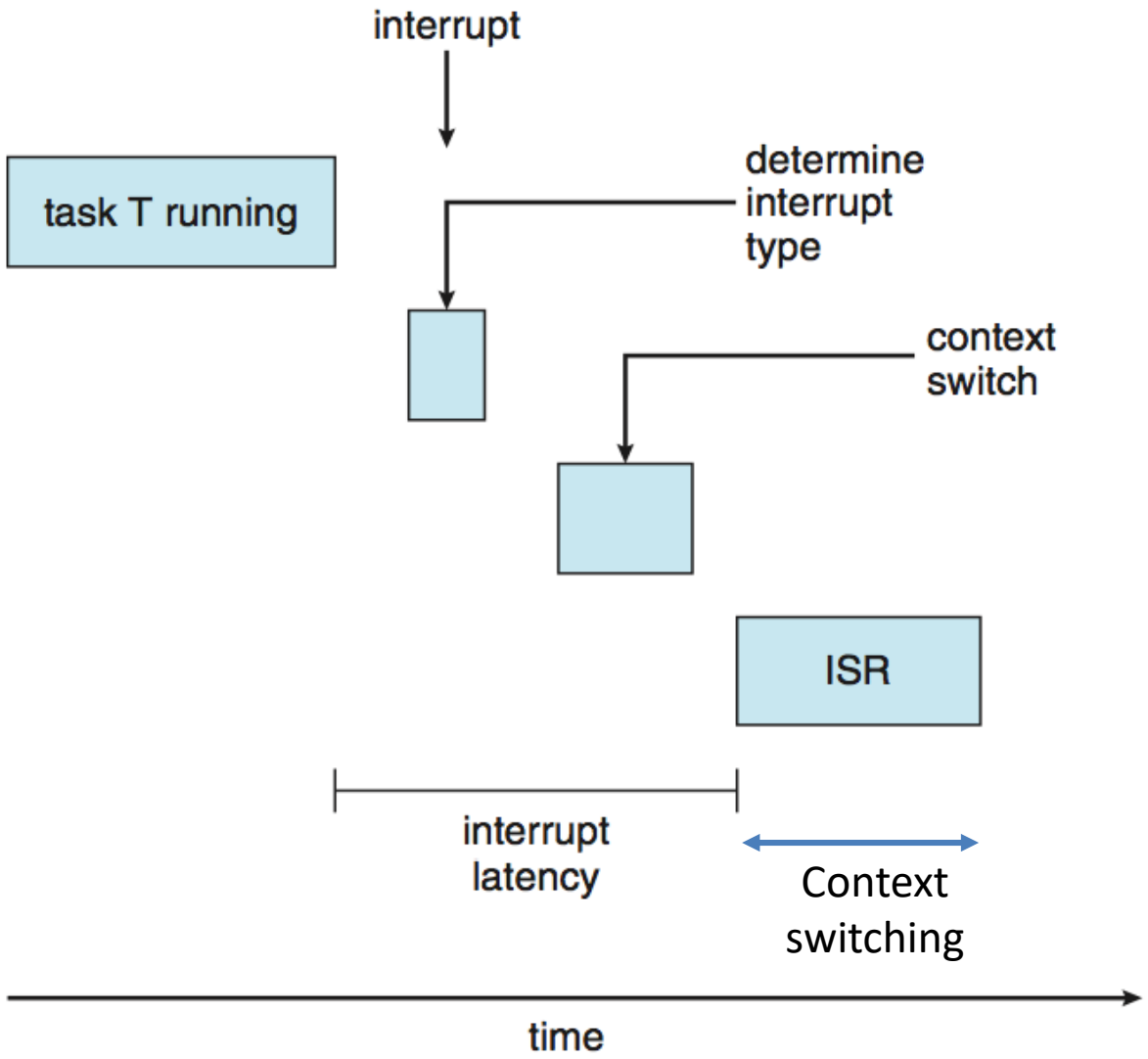
- Χαλαρά (**Soft real-time systems**)

  καμία εγγύηση σχετικά με το πότε θα προγραμματιστεί η κρίσιμη διεργασία σε πραγματικό χρόνο

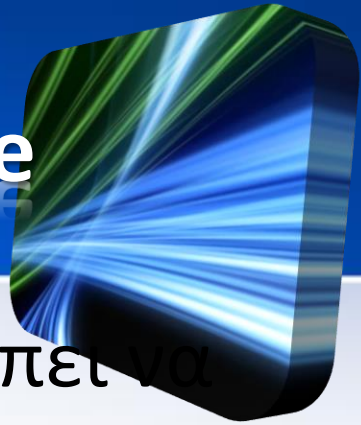
- Αυστηρά (**Hard real-time systems**) Πρέπει να ολοκληρωθεί μια διεργασία σε αυστηρά χρονικά περιθώρια (deadline). Δύο τύποι καθυστερήσεων:
  - Interrupt latency - χρόνος από την άφιξη της διακοπής μέχρι την έναρξη της ρουτίνας που διακόπτει τις υπηρεσίες
  - Dispatch latency - χρόνος για context switching

Συστήματα Πραγματικού Χρόνου



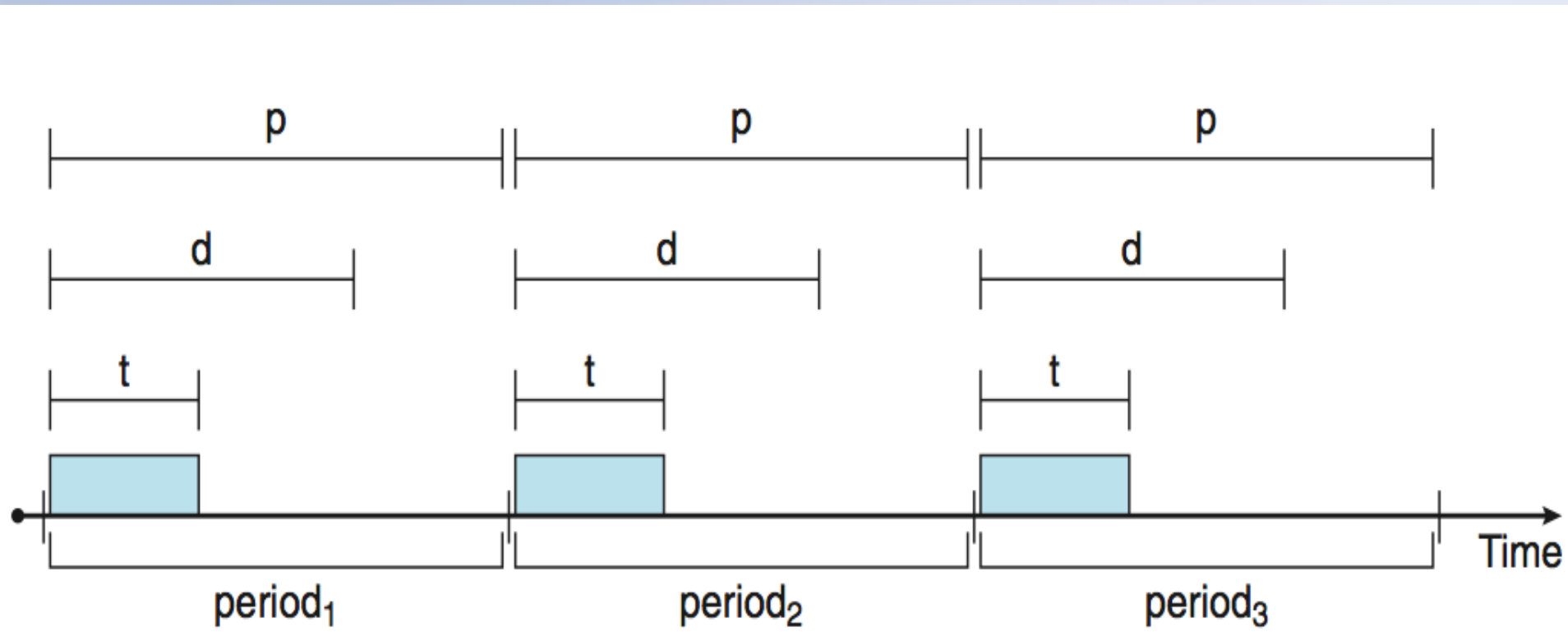


Χρονοπρογραμματισμός σε Συστήματα Real–time

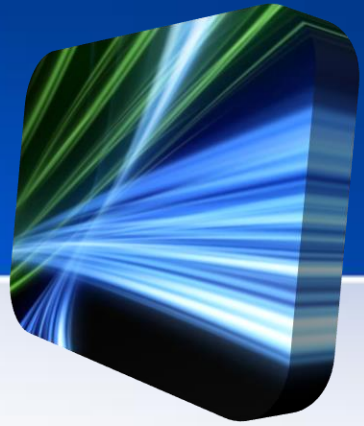


- Για χρονοπρογραμματισμό σε **πραγματικό χρόνο**, πρέπει να υποστηρίζονται **διακοπτοί** αλγόριθμοι, με **προτεραιότητες**
- Οι περιοδικές (**periodic**) διεργασίες απαιτούν χρόνο CPU, σε σταθερά διαστήματα
- Οι διεργασίες έχουν νέα χαρακτηριστικά:
  - Χρόνο επεξεργασίας **t**, προθεσμία **d**, περίοδο **p**
  - $0 \leq \mathbf{t} \leq \mathbf{d} \leq \mathbf{p}$
  - Ο ρυθμός της περιοδικής εργασίας είναι **1/p**

Χρονοπρογραμματισμός πραγματικού χρόνου

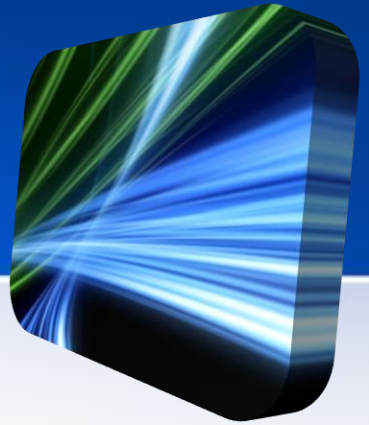


Linux version 2.6.23 +



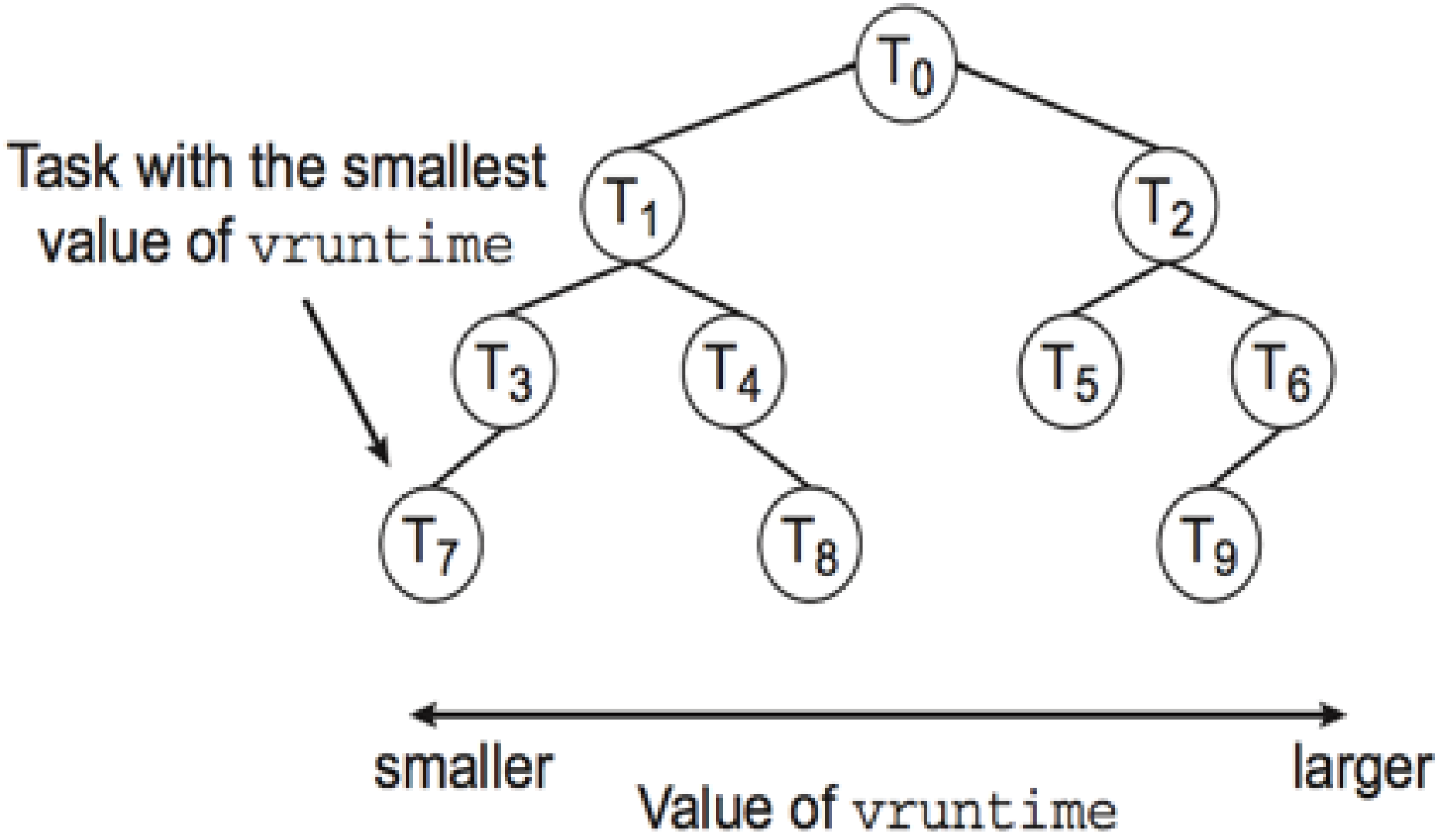
- ***Completely Fair Scheduler*** (CFS)
- **Κλάσεις Χρονοδρομολόγησης**
  - Κάθε κλάση έχει μία προτεραιότητα
  - Ο Χ/Δ επιλέγει την **εργασία** με τη **μέγιστη** προτεραιότητα, από την **κλάση μέγιστης** προτεραιότητας
  - Χρόνος CPU, ανάλογος με την προτεραιότητα
  - Δύο βασικές κλάσεις
    1. default
    2. real-time

Linux Version 2.6.23 +

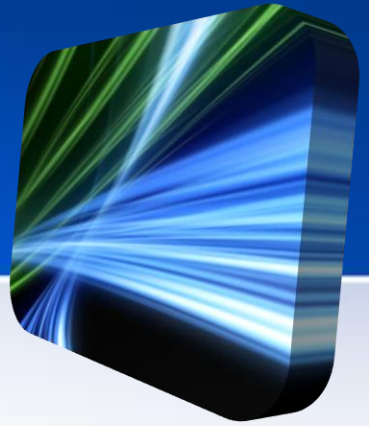


- Ο CFS υπολογίζει έναν **virtual run time (`vruntime`)**
- Επιλέγει για εκτέλεση την εργασία με τον μικρότερο vruntime.

CFS: Δομή εκτελέσιμων εργασιών

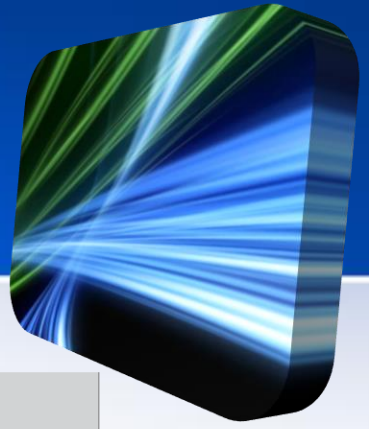


Χρονοπρογραμματισμός Windows



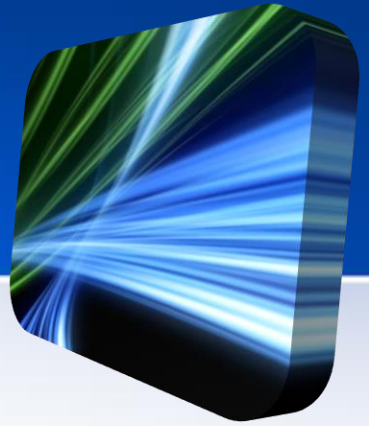
- Τα Windows χρησιμοποιούν **διακοπτό** αλγόριθμο στηριζόμενο σε **προτεραιότητες**
- Κάθε φορά εκτελείται η διεργασία με την **μεγαλύτερη** προτεραιότητα
- Dispatcher και scheduler είναι το **ίδιο** πρόγραμμα
- Τα νήματα πραγματικού χρόνου μπορούν να διακόψουν τα μη-πραγματικού χρόνου
- Σχήμα προτεραιότητας **32 επιπέδων**
- Ουρά για **κάθε** προτεραιότητα

Προτεραιότητες στα Windows



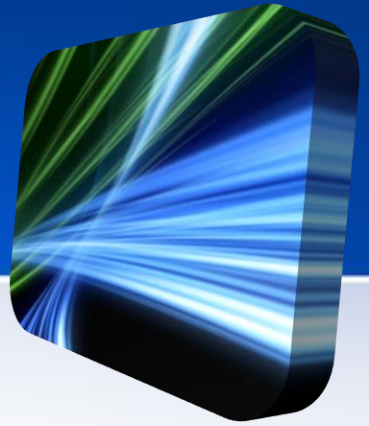
	real-time	high	above normal	normal	below normal	idle priority
time-critical	31	15	15	15	15	15
highest	26	15	12	10	8	6
above normal	25	14	11	9	7	5
normal	24	13	10	8	6	4
below normal	23	12	9	7	5	3
lowest	22	11	8	6	4	2
idle	16	1	1	1	1	1

Αξιολόγηση Αλγορίθμων



- Πώς επιλέγεται ένας αλγόριθμος χρονοπρογραμματισμού CPU;
  - Καθορισμός κριτηρίων και
  - Αξιολόγηση του αλγορίθμου
- Για συγκεκριμένο φόρτο εργασίας, εκτιμάται η απόδοση κάθε αλγορίθμου

Παράδειγμα Αξιολόγησης Αλγορίθμων



Για 5 διεργασίες που εμφανιζονται τον χρόνο 0:

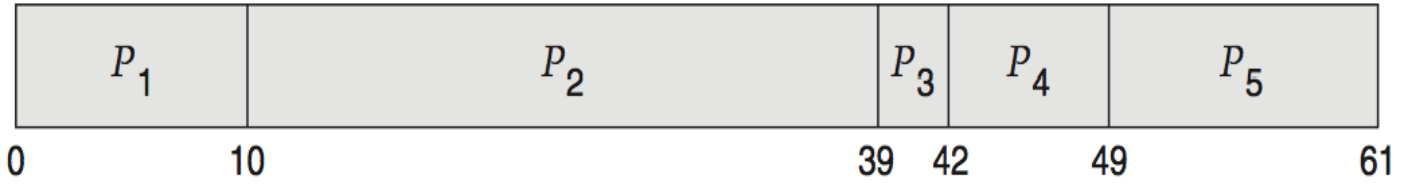
Process	Burst Time
P_1	10
P_2	29
P_3	3
P_4	7
P_5	12

Ντετερμινιστική αποτίμηση

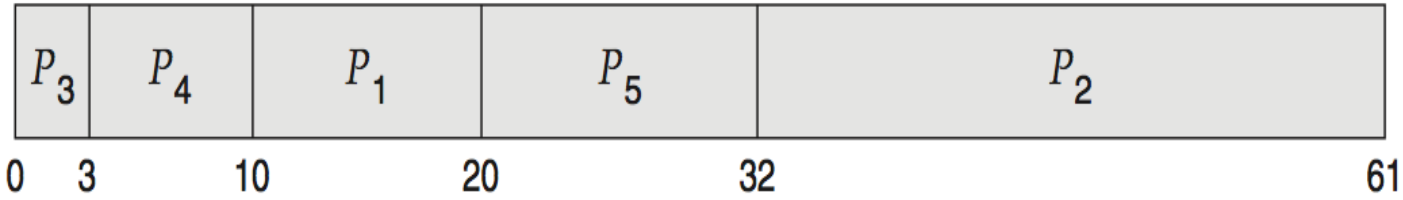
Υπολογισμός **ελάχιστου μέσου χρόνου αναμονής**

Process	Burst Time
P_1	10
P_2	29
P_3	3
P_4	7
P_5	12

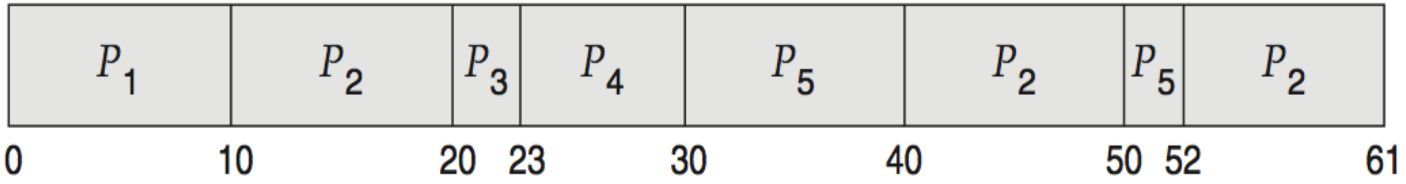
- FCFS χρειάζεται **28ms:**



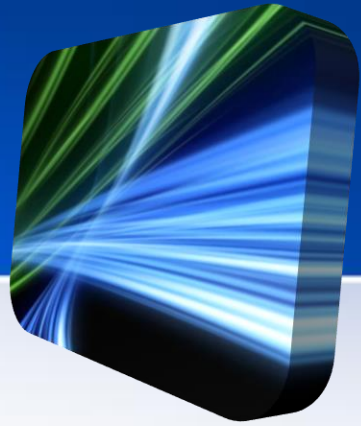
- Non-preemptive SFJ χρειάζεται **13ms:**



- RR χρειάζεται **23ms:**



Θεωρία αναμονής



- n = μέσο μήκος ουράς
- W = μέσος χρόνος αναμονής στην ουρά
- λ = μέσος ρυθμός άφιξης στην ουρά
- **Νόμος του Little** – σε σταθερή κατάσταση,
  - αποχωρούσες=νέες διεργασίες, ήτοι:

    $$n = \lambda \times W$$
- Εάν προσέρχονται κατά μ.ο. **7** διεργασίες/s, και παραμένουν **14** διεργασίες στην ουρά κατά μ.ο., **τότε** ο μέσος χρόνος αναμονής ανά διεργασία είναι **2 s**.

Εξομοίωση - Simulation

Τα συστήματα εξομοίωσης είναι πιο πρακτικά & ακριβή