Διεργασίες
(μοντέλο μνήμης & εκτέλεσης)

Ο κώδικας δεν εκτελείται «μόνος του»

- Ο εκτελέσιμος κώδικας αποθηκεύεται σε ένα αρχείο
- Το αρχείο είναι μια «παθητική» οντότητα μόνιμης αποθήκευσης δεδομένων
- Ένα αρχείο **δεν** εκτελείται «από μόνο του»

- Υπάρχει εντελώς **ξεχωριστός μηχανισμός** για την εκτέλεση του κώδικα (που βρίσκεται στο αρχείο)
- Αυτός ο μηχανισμός ονομάζεται **διεργασία** (process)
- Το λειτουργικό αναλαμβάνει την **διαχείριση** και την **χρονοδρομολόγηση** των διεργασιών (process management & process scheduling)

Φόρτωση, αρχικοποίηση, εκτέλεση

- Έλεγχος ότι το αρχείο περιέχει εκτελέσιμο κώδικα
  - πως γίνεται αυτό;
- Φόρτωμα του κώδικα στην κυρίως μνήμη
  - όχι απαραίτητα ολόκληρου του κώδικα ...
- Αρχικοποίηση περιβάλλοντος / πλαισίου εκτέλεσης
  - δέσμευση καθολικής μνήμης και της στοίβας
  - αρχικοποίηση των καθολικών μεταβλητών και της στοίβας
  - αρχικοποίηση των παραμέτρων της `main`
  - αρχικοποίηση των περιγραφέων αρχείων για Ε/Ε
- Εκτέλεση
  - άλμα στην πρώτη εντολή της `main`
  - τα περιεχόμενα της μνήμης/στοίβας αλλάζουν μέσα από τις εντολές του προγράμματους (και τις κλήσεις συστήματος)

Διεργασία

- Οι διεργασίες είναι οι **οντότητες εκτέλεσης** των προγραμμάτων (εκτελέσιμου κώδικα)
- Κάθε διεργασία έχει **ξεχωριστή/ιδιωτική** μνήμη
  - αν μια διεργασία αλλάξει τιμή σε μια θέση της μνήμης της, αυτή η αλλαγή **δεν** είναι ορατή στις υπόλοιπες διεργασίες
- Κάθε διεργασία εκτελείται **ταυτόχρονα** και **ανεξάρτητα** από τις υπόλοιπες διεργασίες
  - κάθε μια από τις διεργασίες έχει την ψευδαίσθηση ότι έχει την αποκλειστική χρήση του επεξεργαστή

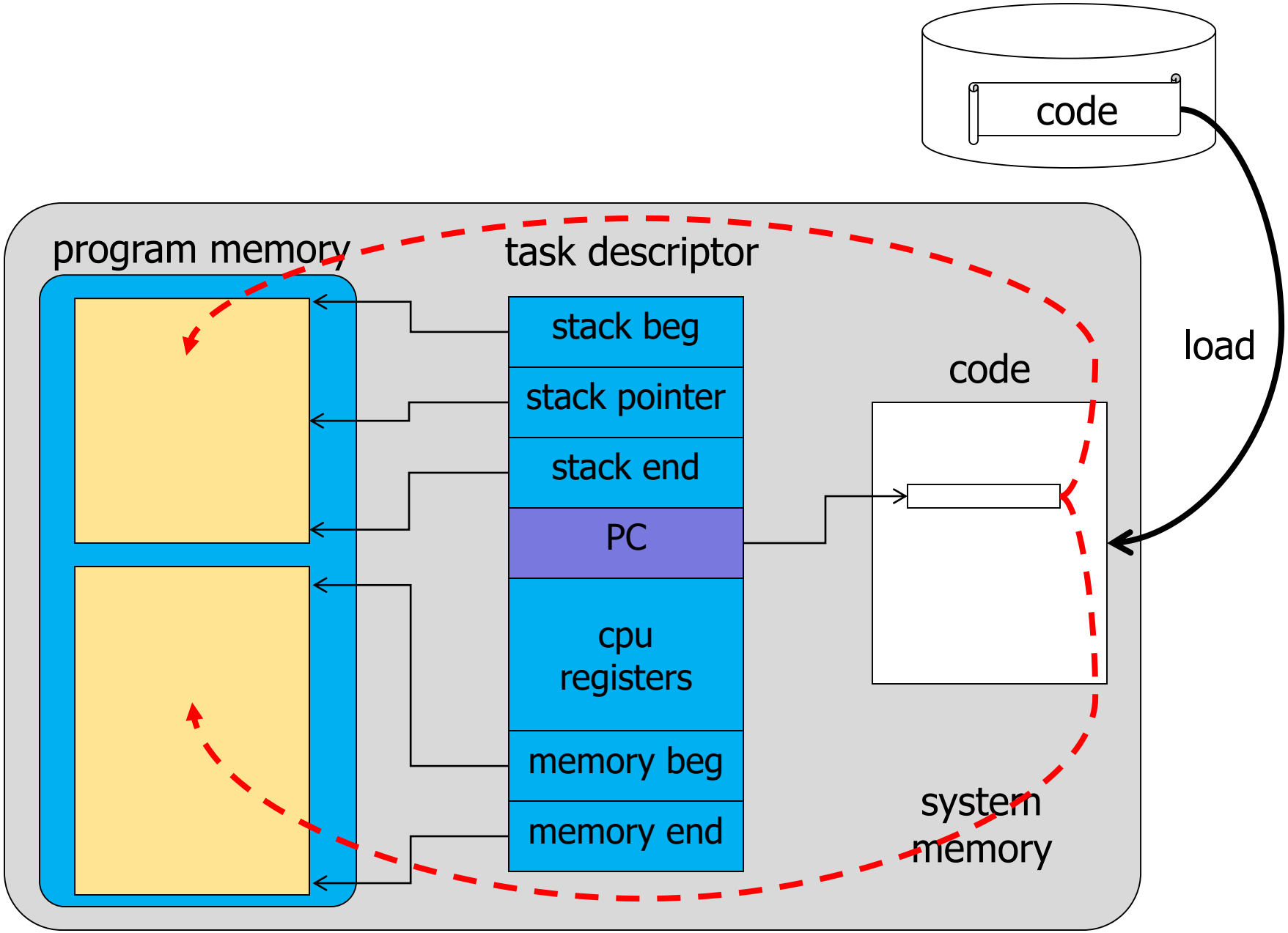
- Το **ίδιο** πρόγραμμα/κώδικας μπορεί να εκτελείται **ταυτόχρονα** από **διαφορετικές** διεργασίες

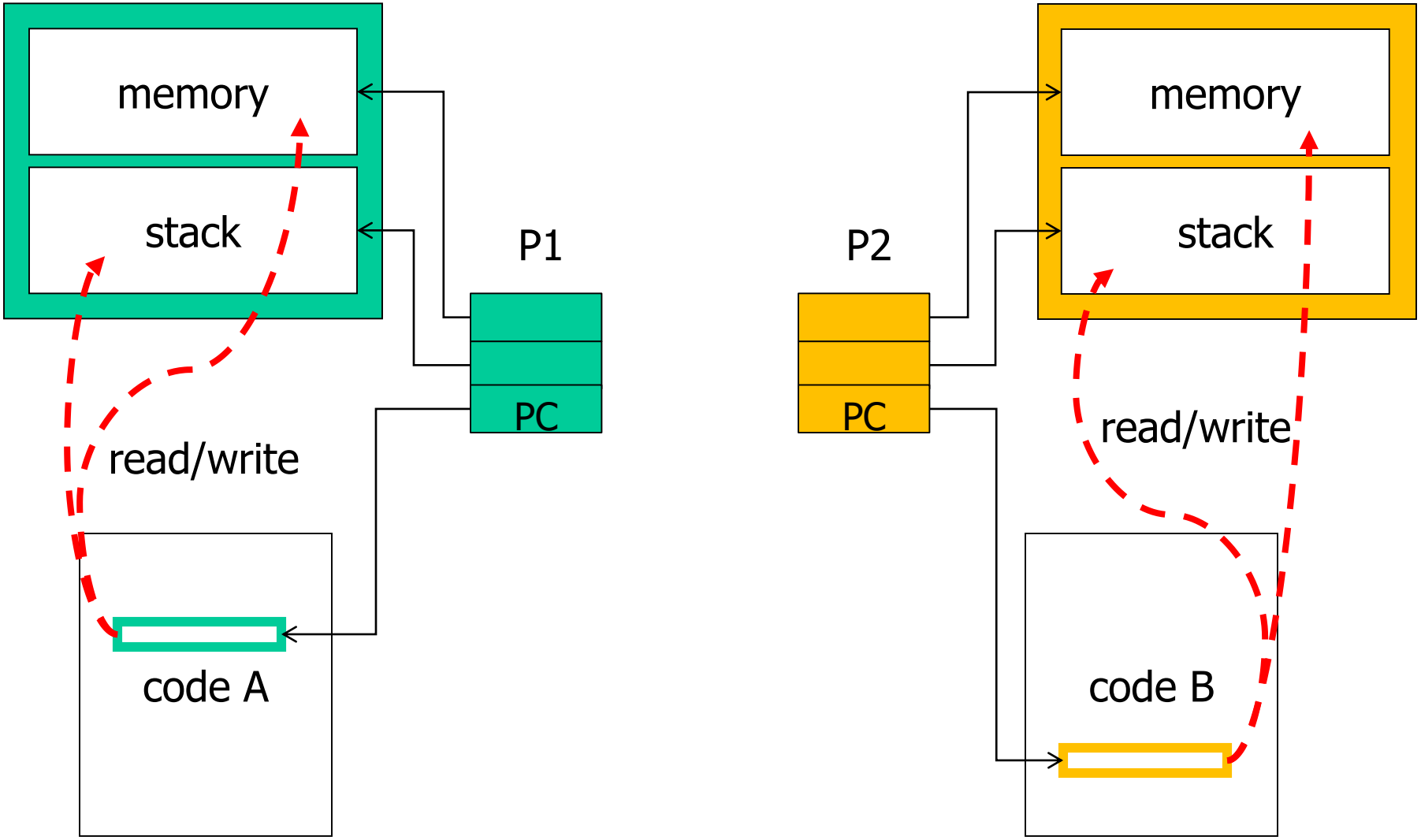
Πλαίσιο εκτέλεσης

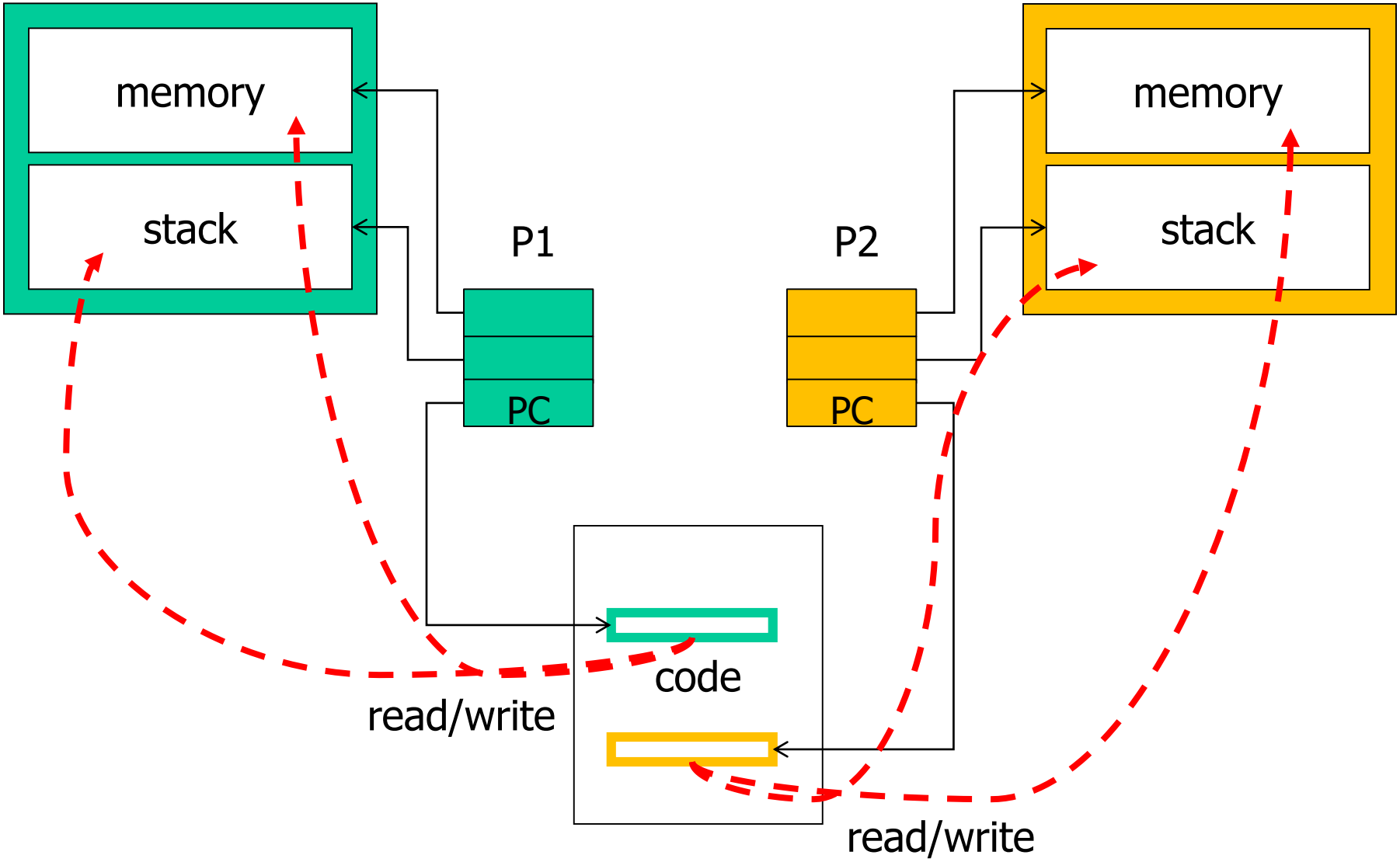
- Κάθε διεργασία αντιστοιχεί σε ένα ξεχωριστό **πλαίσιο εκτέλεσης (execution context)**

- Το πλαίσιο εκτέλεσης περιλαμβάνει την πλήρη κατάσταση εκτέλεσης ενός προγράμματος
  - στατική μνήμη, δυναμική μνήμη, στοίβα, file descriptors …
  - κατάσταση του επεξεργαστή

- Καθώς και πολλές άλλες πληροφορίες
  - ο χρήστης στο όνομα του οποίου γίνεται η εκτέλεση
  - προτεραιότητα εκτέλεσης
  - χρόνος που έχει αφιερωθεί για αυτή την εκτέλεση


code
program memory
task descriptor
stack beg
stack pointer
stack end
PC
cpu
registers
memory beg
memory end
code
load
system
memory

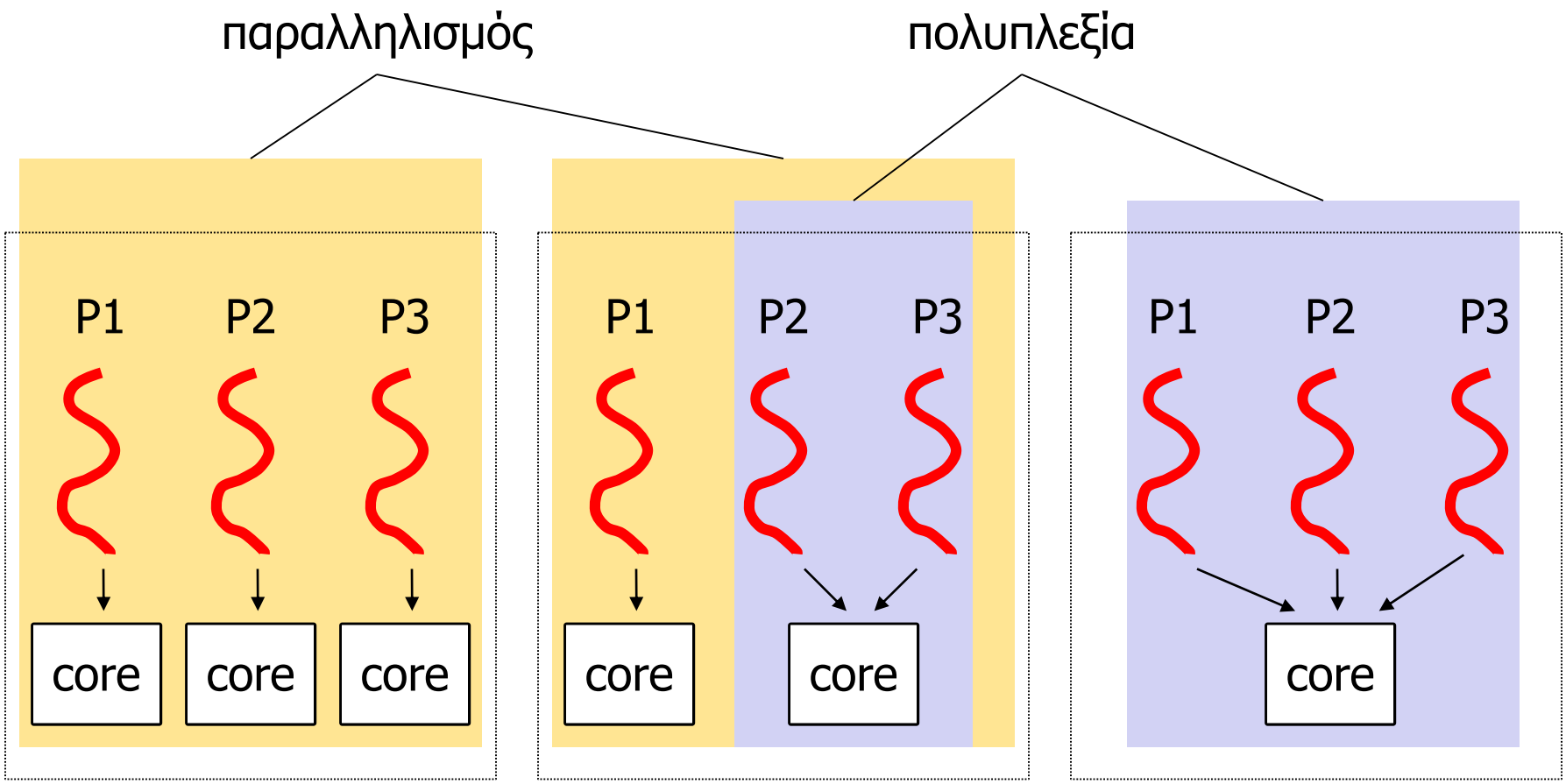

memory
stack
P1
PC
read/write
code A
P2
PC
memory
stack
read/write
code B

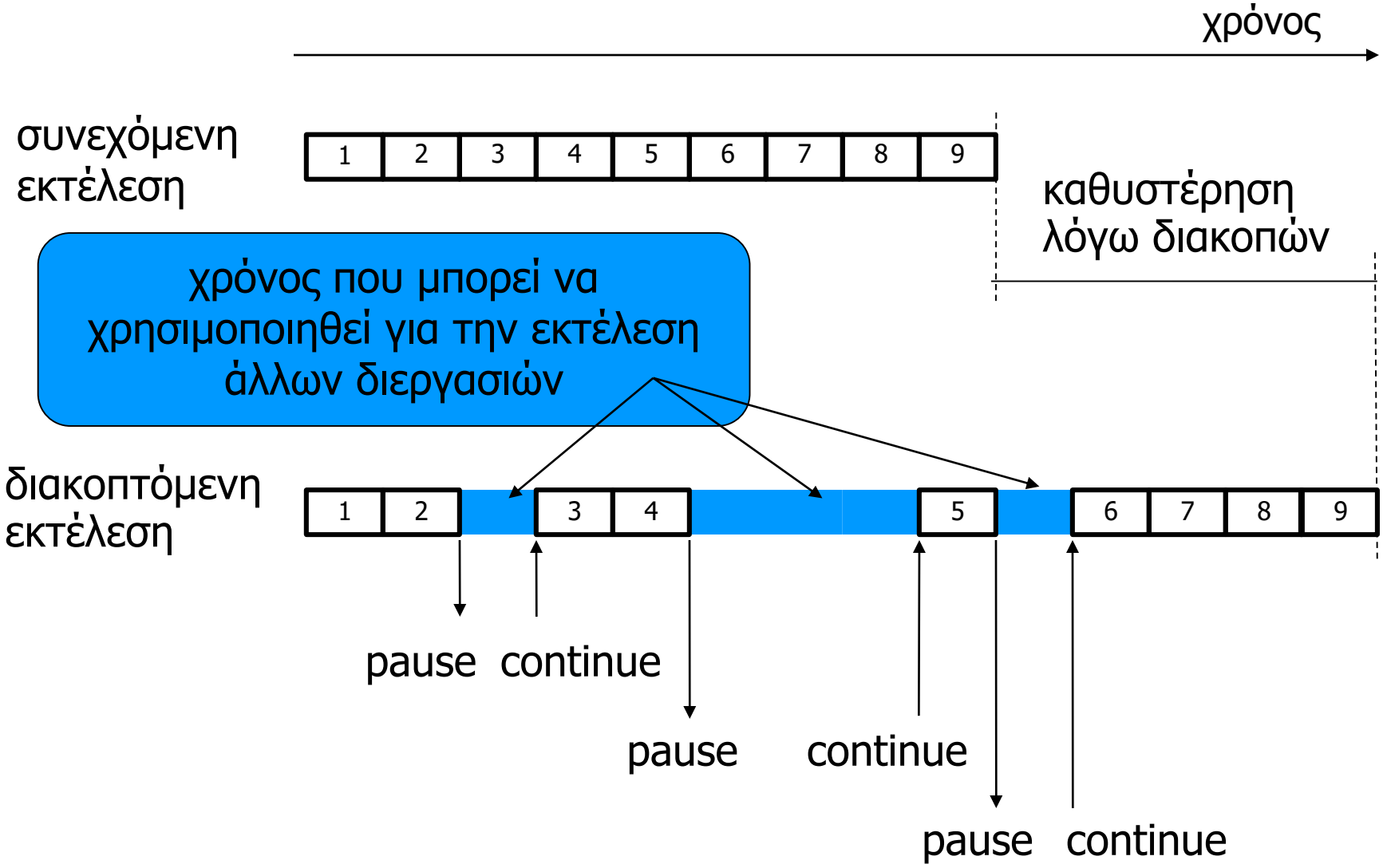

memory
stack
P1
PC
P2
PC
memory
stack
code
read/write
read/write

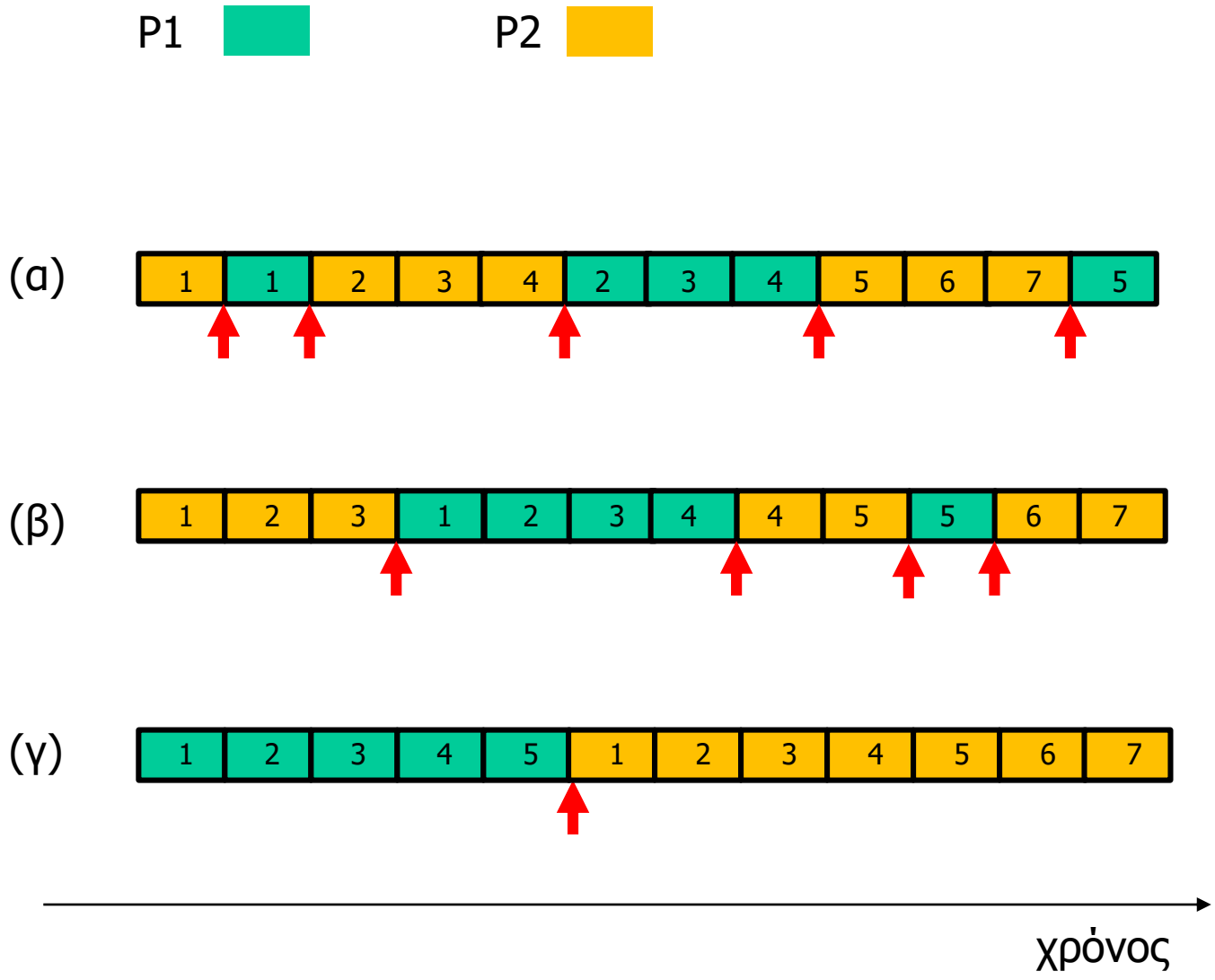
Γιατί πολλές διεργασίες;

- Πολλά προγράμματα
  - που θέλουμε να «τρέχουν» την ίδια στιγμή
- Πολλοί χρήστες
  - μοιράζονται ένα μηχάνημα
  - ο καθένας πρέπει να μπορεί να εκτελεί τα δικά του προγράμματα, ανεξάρτητα από τους άλλους
- Παραλληλισμός
  - εκτέλεση διαφορετικών τμημάτων μιας επεξεργασίας ή ενός υπολογισμού από ξεχωριστές διεργασίες
  - αποφυγή μπλοκαρίσματος εν' αναμονή κάποιου γεγονότος
- Καλύτερη δόμηση προγράμματος
  - μια πολύπλοκη διαδικασία ίσως μπορεί να μοντελοποιηθεί πιο απλά αν δομηθεί ως ένα «σύστημα» από διεργασίες

Εκτέλεση: παραλληλισμός ή πολυπλεξία




χρόνος
συνεχόμενη εκτέλεση
1 2 3 4 5 6 7 8 9
καθυστέρηση λόγω διακοπών
χρόνος που μπορεί να χρησιμοποιηθεί για την εκτέλεση άλλων διεργασιών
διακοπτόμενη εκτέλεση
1 2 3 4 5 6 7 8 9
pause
continue
pause
continue
pause
continue


P1
P2
(α)
1 1 2 3 4 2 3 4 5 6 7 5
(β)
1 2 3 1 2 3 4 4 5 5 6 7
(γ)
1 2 3 4 5 1 2 3 4 5 6 7
χρόνος

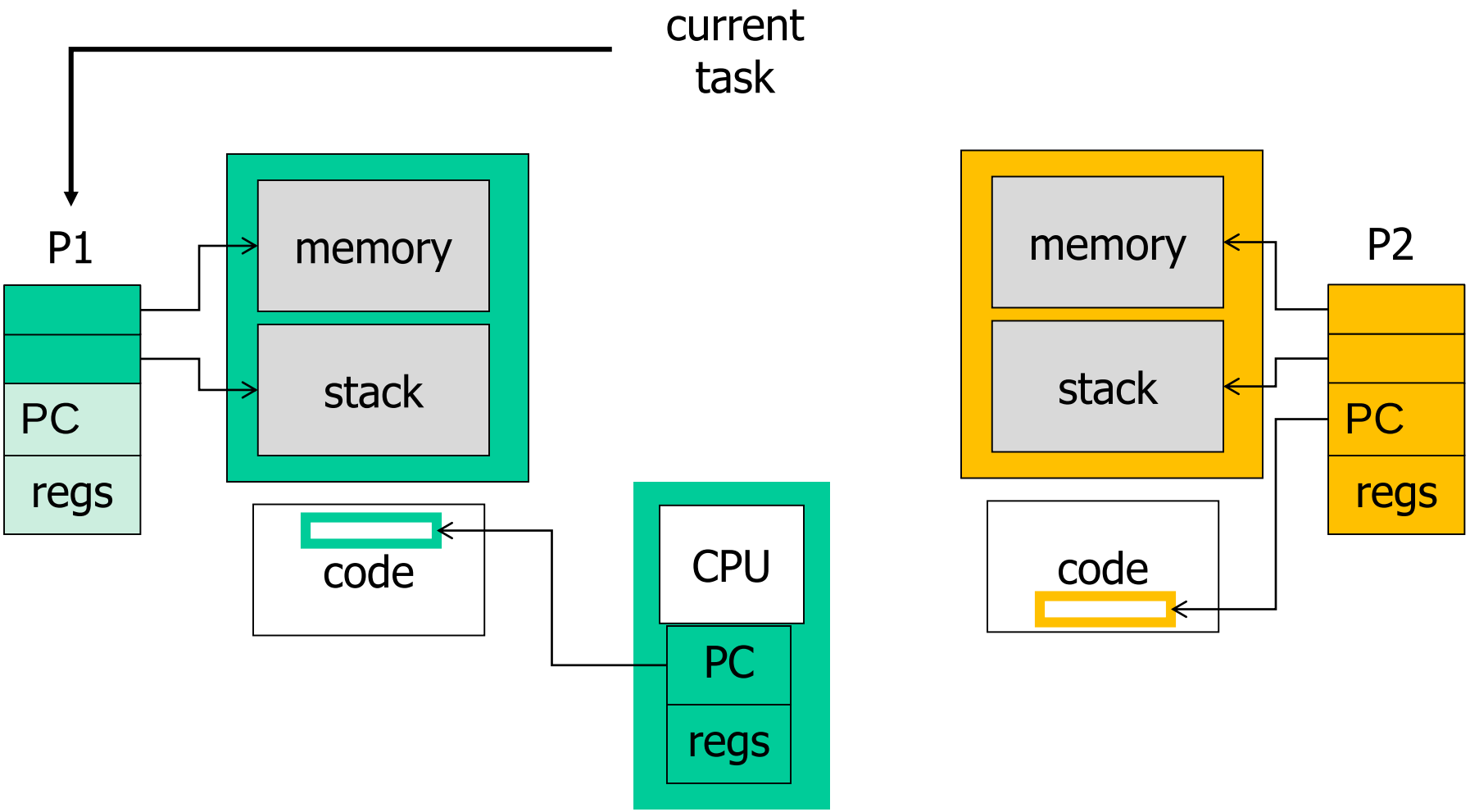
Εναλλαγή διεργασιών (context switch)

- Το λειτουργικό **σώνει** την κατάσταση εκτέλεσης της τρέχουσας διεργασίας
  - αντιγραφή όλης της απαραίτητης πληροφορίας του επεξεργαστή μέσα στην δομή του task descriptor που κρατά το λειτουργικό για την διεργασία

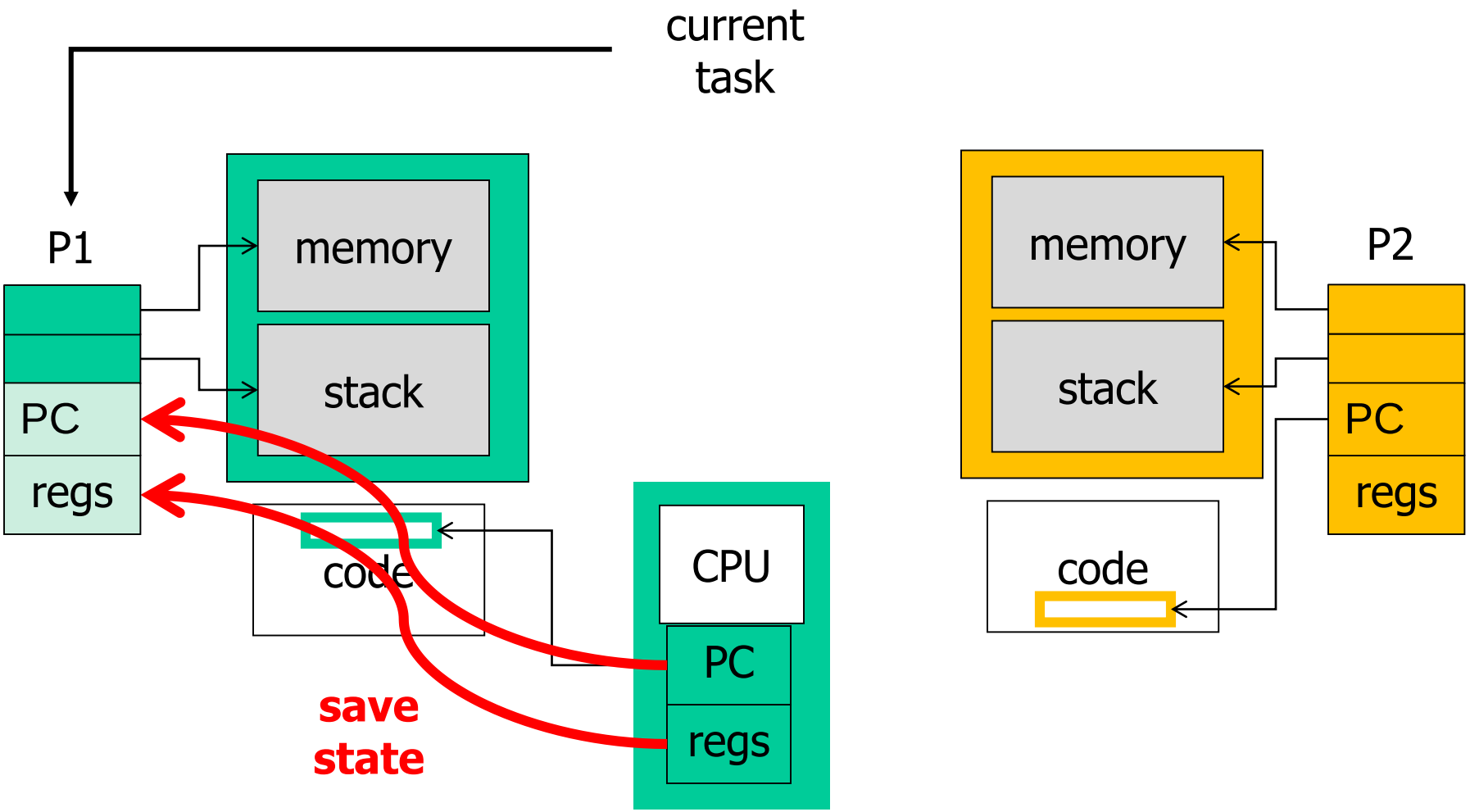
- Επιλέγεται η διεργασία που θα συνεχίσει την εκτέλεση της

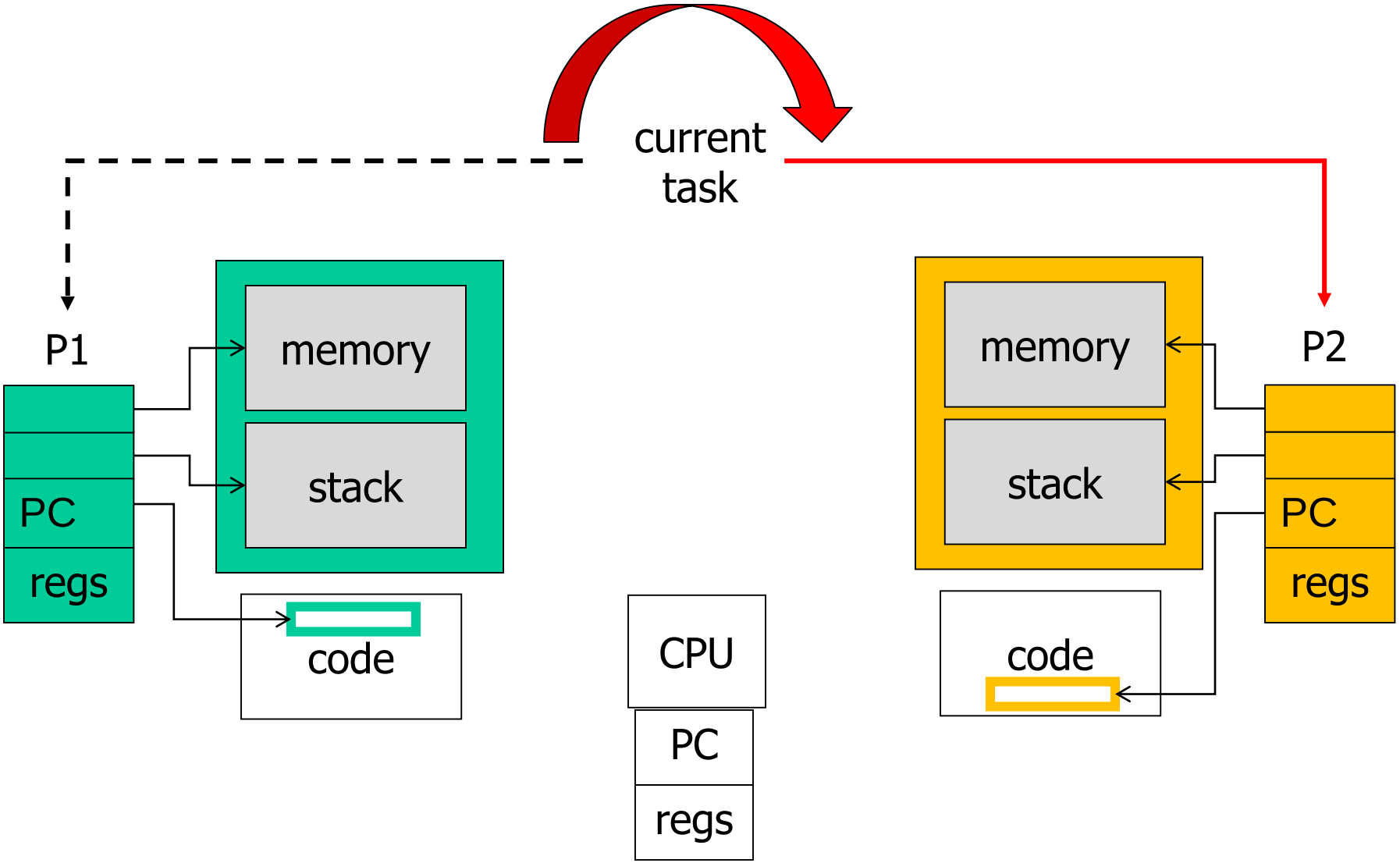
- Το λειτουργικό **επαναφέρει** την κατάσταση εκτέλεσης της διεργασίας
  - αντιγραφή όλης της απαραίτητης πληροφορίας από τον task descriptor πίσω στον επεξεργαστή

P1 εκτελείται

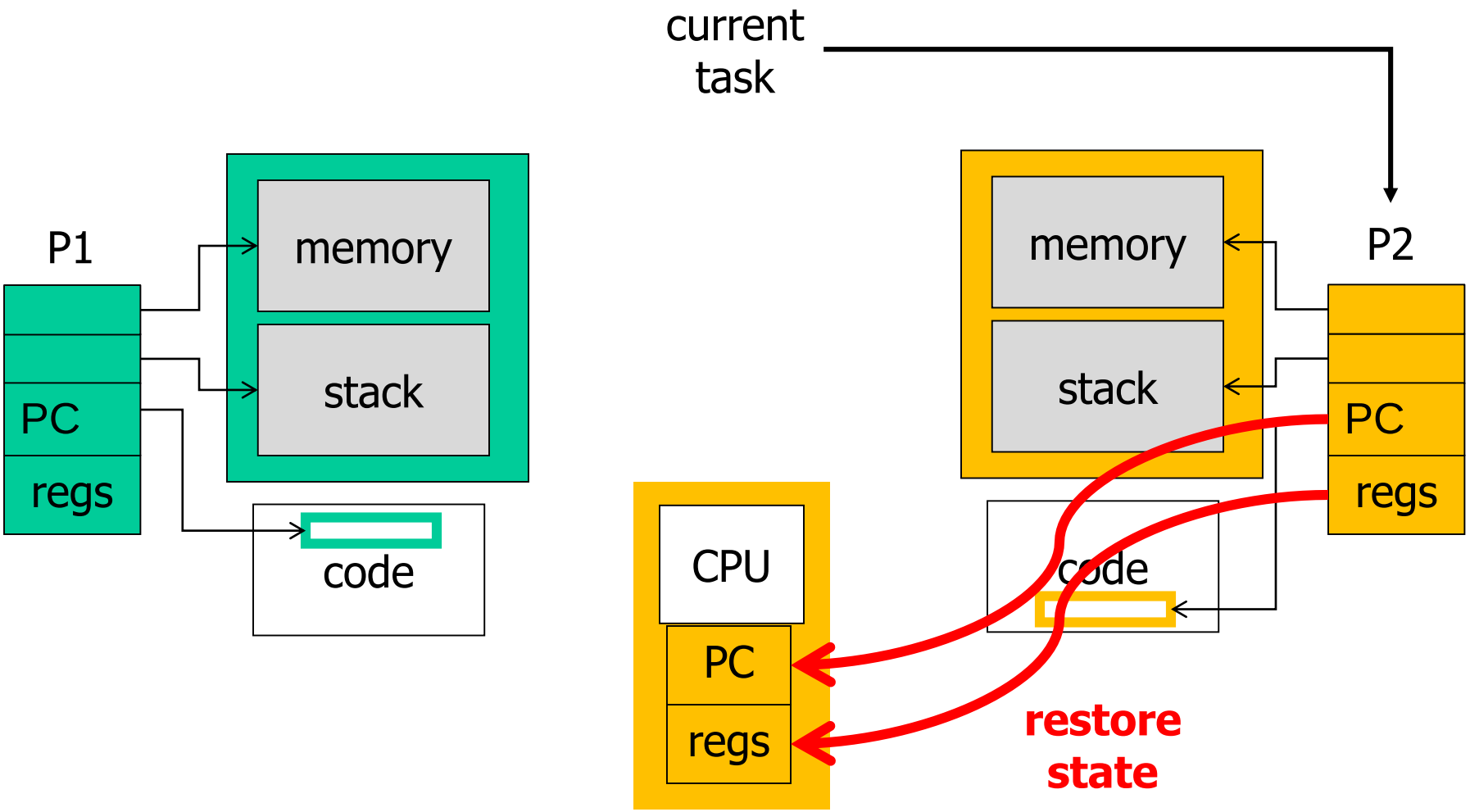


σώσιμο κατάστασης P1

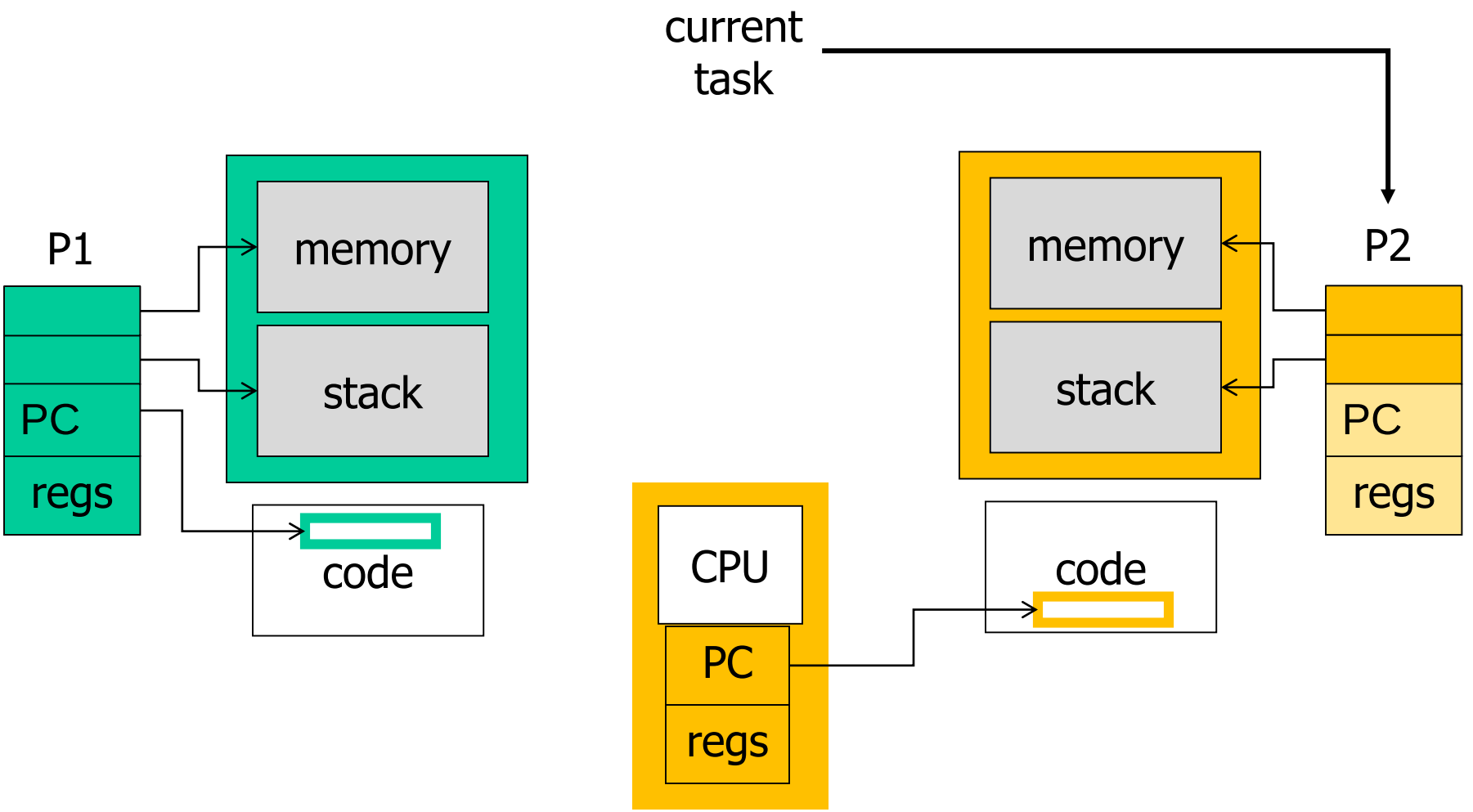



current
task
P1
memory
stack
PC
regs
code
CPU
PC
regs
memory
stack
P2
PC
regs
code

επαναφορά κατάστασης P2

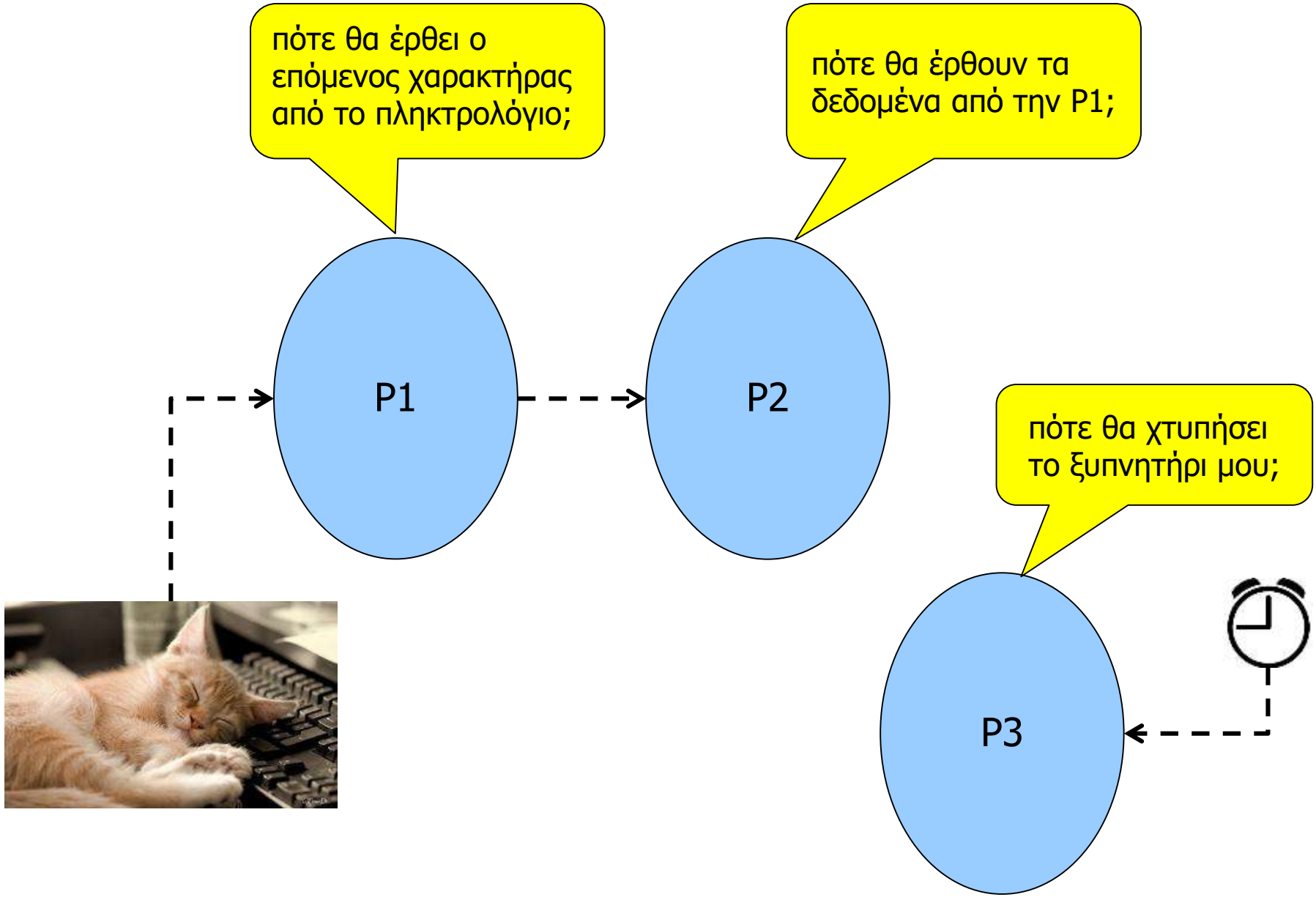


P2 εκτελείται



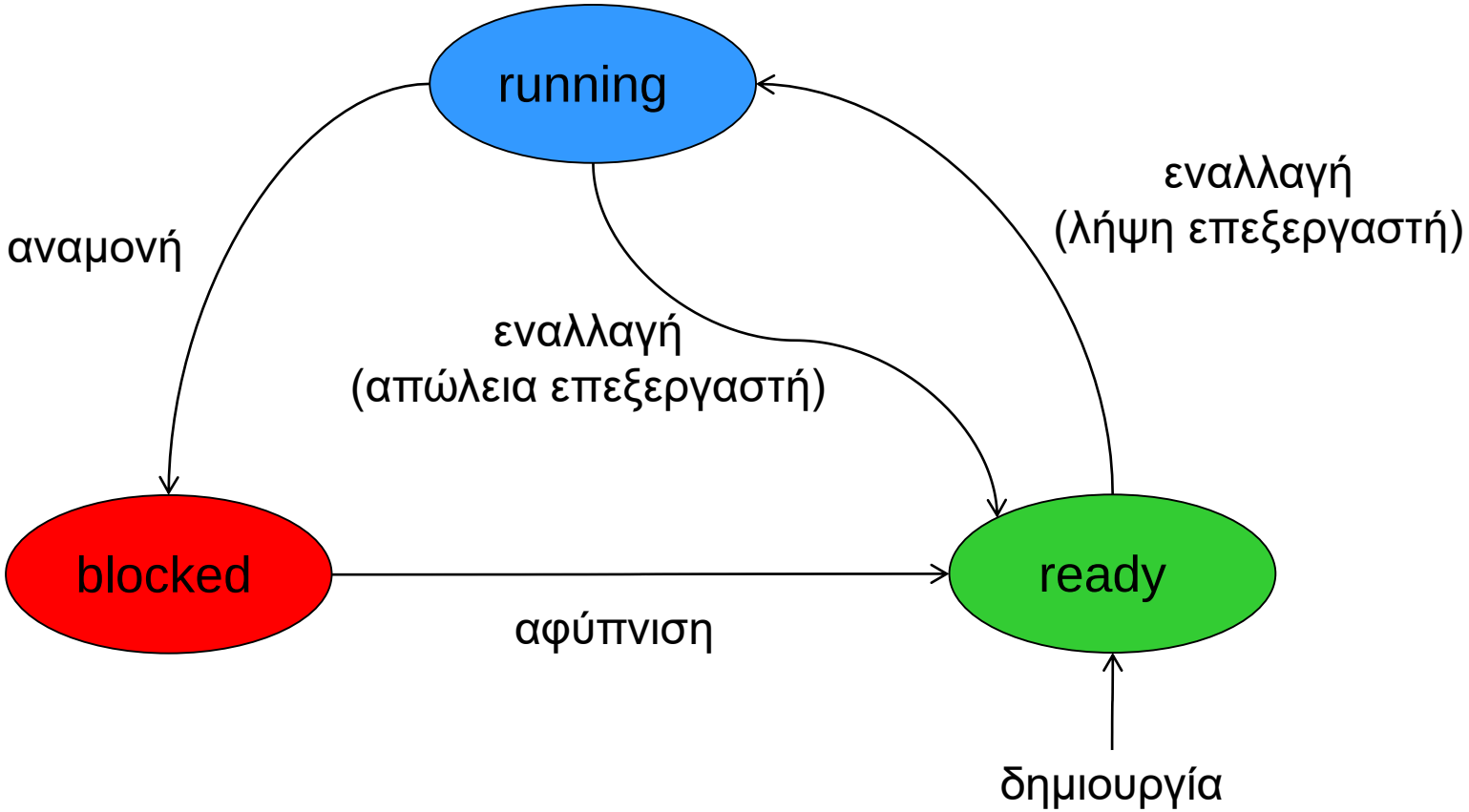
Αναμονή

- Οι περισσότερες) διεργασίες περνάνε το μεγαλύτερο μέρος «της ζωής» τους απλά ... **περιμένοντας**
- Να περάσει ένα συγκεκριμένο χρονικό διάστημα
  - time-outs
- Να παραλάβουν δεδομένα από την μνήμη, περιφερειακές συσκευές ή άλλες διεργασίες
  - input/output
- Να συγχρονιστούν με άλλες διεργασίες
  - process synchronization


πότε θα έρθει ο επόμενος χαρακτήρας από το πληκτρολόγιο;
πότε θα έρθουν τα δεδομένα από την P1;
P1
P2
πότε θα χτυπήσει το ξυπνητήρι μου;
P3

Κατάσταση αναμονής

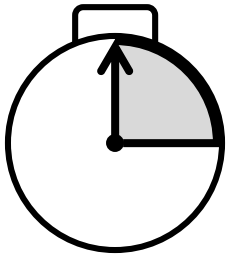
- Το λειτουργικό σύστημα πρέπει να **γνωρίζει** κατά πόσο μια διεργασία βρίσκεται σε αναμονή
  - ώστε να **μην** της δίνει χρόνο από τον επεξεργαστή
- Διαχωρισμός κατάστασης μιας διεργασίας
- **Ready:** η διεργασία είναι **έτοιμη** να λάβει τον επεξεργαστή, όταν αυτός ελευθερωθεί
- **Blocked:** η διεργασία **περιμένει** για κάτι, συνεπώς δεν είναι έτοιμη να λάβει τον επεξεργαστή
- **Running:** η διεργασία **έχει** τον επεξεργαστή
- Οι **μεταβάσεις** ανάμεσα σε αυτές τις καταστάσεις γίνονται **υπό τον έλεγχο** του λειτουργικού, μέσα από διάφορες λειτουργίες του συστήματος


running
αναμονή
εναλλαγή
(λήψη επεξεργαστή)
εναλλαγή
(απώλεια επεξεργαστή)
blocked
ready
αφύπνιση
δημιουργία

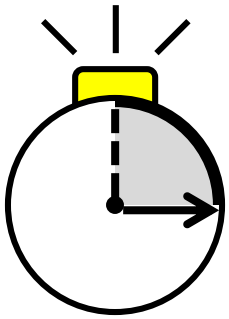
Χρονοπρογραμματισμός διεργασιών

- Μέσω της εναλλαγής, μπορεί να διακοπεί η εκτέλεση μιας διεργασίας για να συνεχιστεί μια άλλη διεργασία
- Υπάρχουν όμως κάποιες διαστάσεις **«πολιτικής»** για το **πώς** θα χρησιμοποιηθεί αυτός ο μηχανισμός

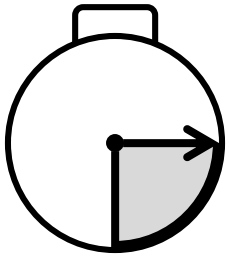
- **Πότε** διακόπτεται η τρέχουσα διεργασία;
  - αυτόματα (περιοδικά), με βάση ένα χρονόμετρο (timer)
  - όταν καλέσει μια ανασταλτική λειτουργία του συστήματος
  - αν αποφασίσει να απελευθερώσει τον επεξεργαστή (yield)
- **Ποιά** διεργασία θα συνεχίσει την εκτέλεση της;
  - αποφασίζεται με βάση τις προτεραιότητες των διεργασιών
  - ο πιο απλός αλγόριθμος: round-robin



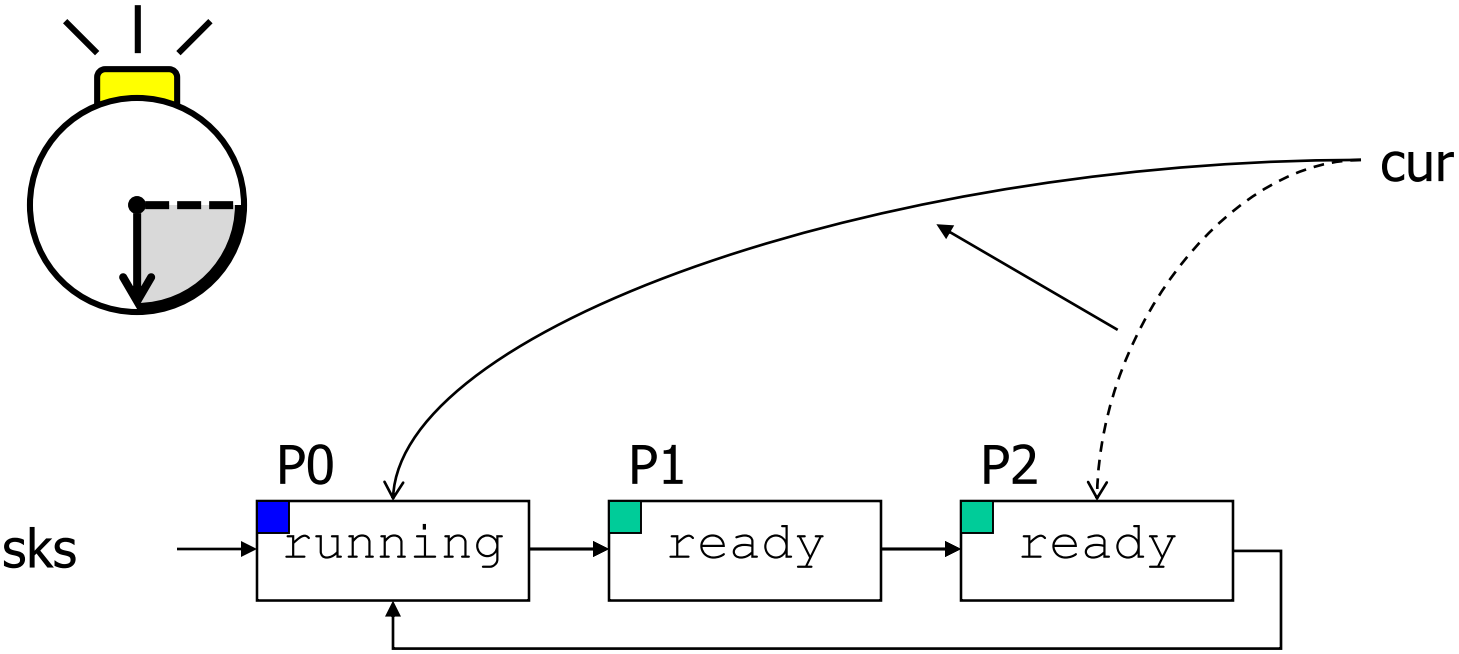
P1 εκτελείται



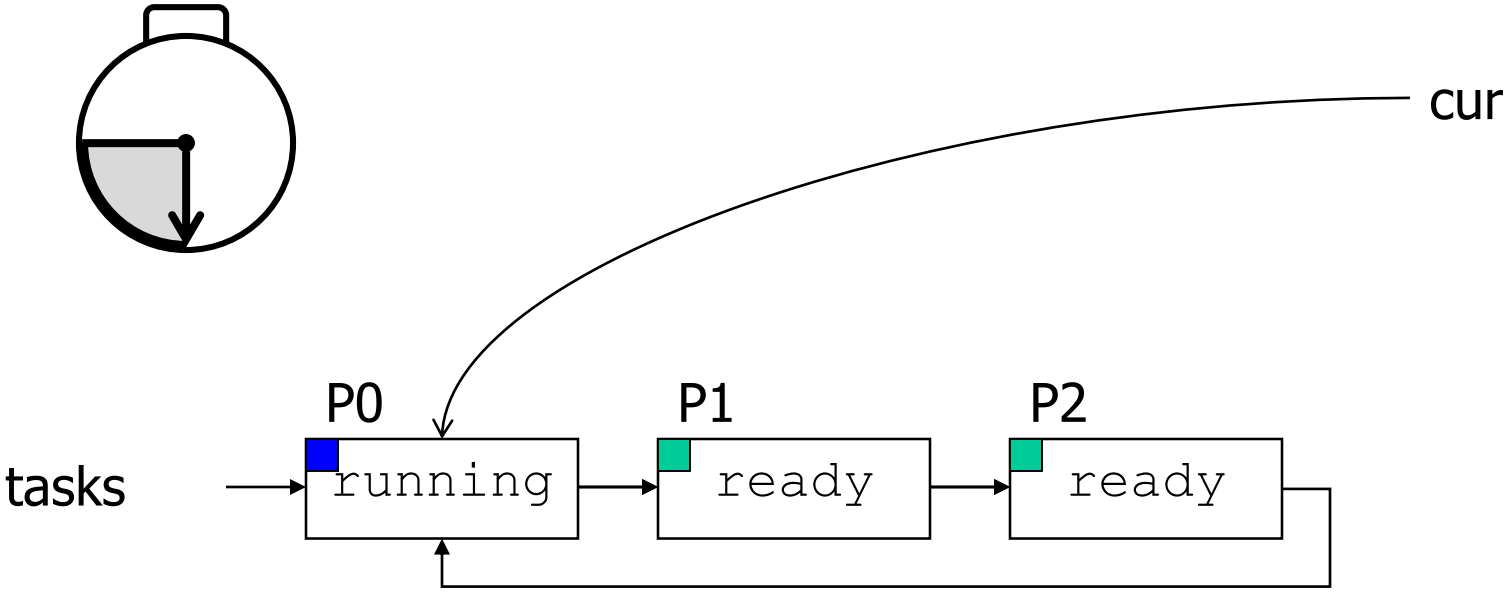
εναλλαγή



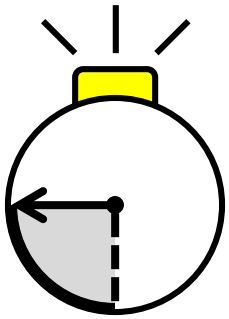
P2 εκτελείται



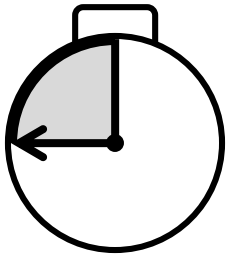
εναλλαγή



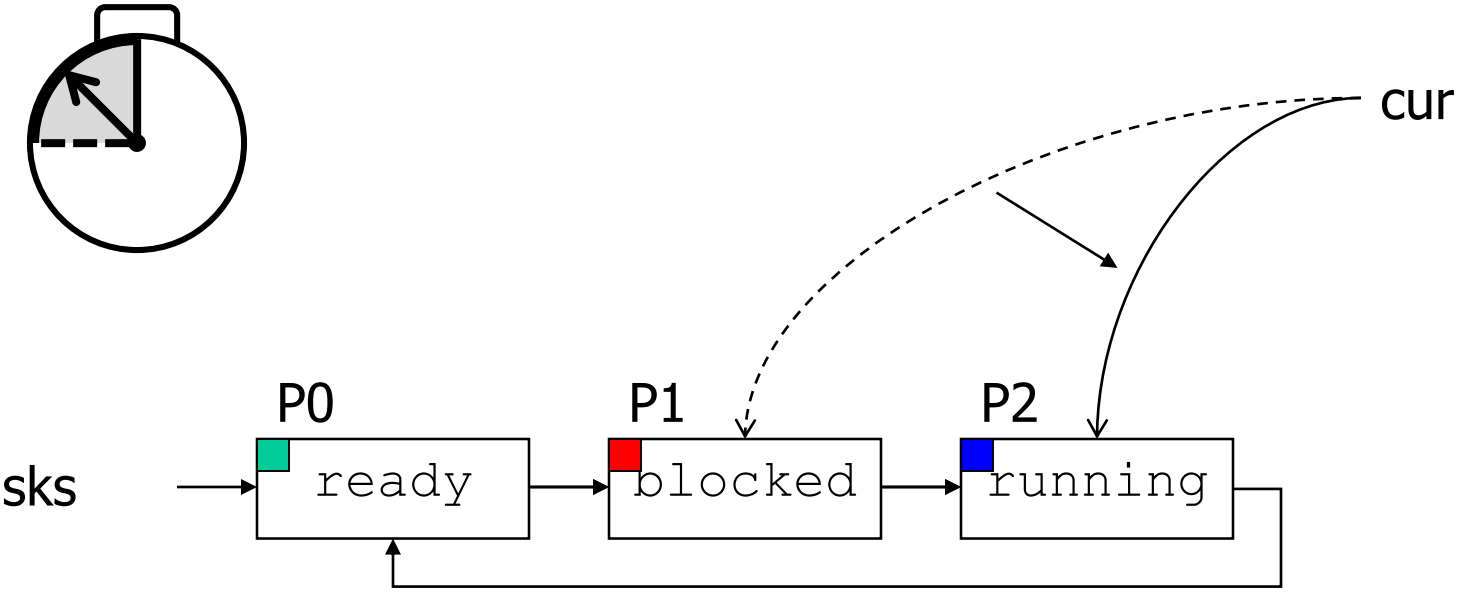
P0 εκτελείται



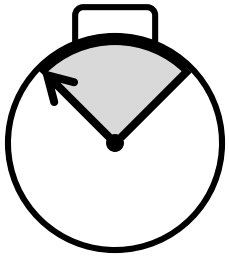
εναλλαγή



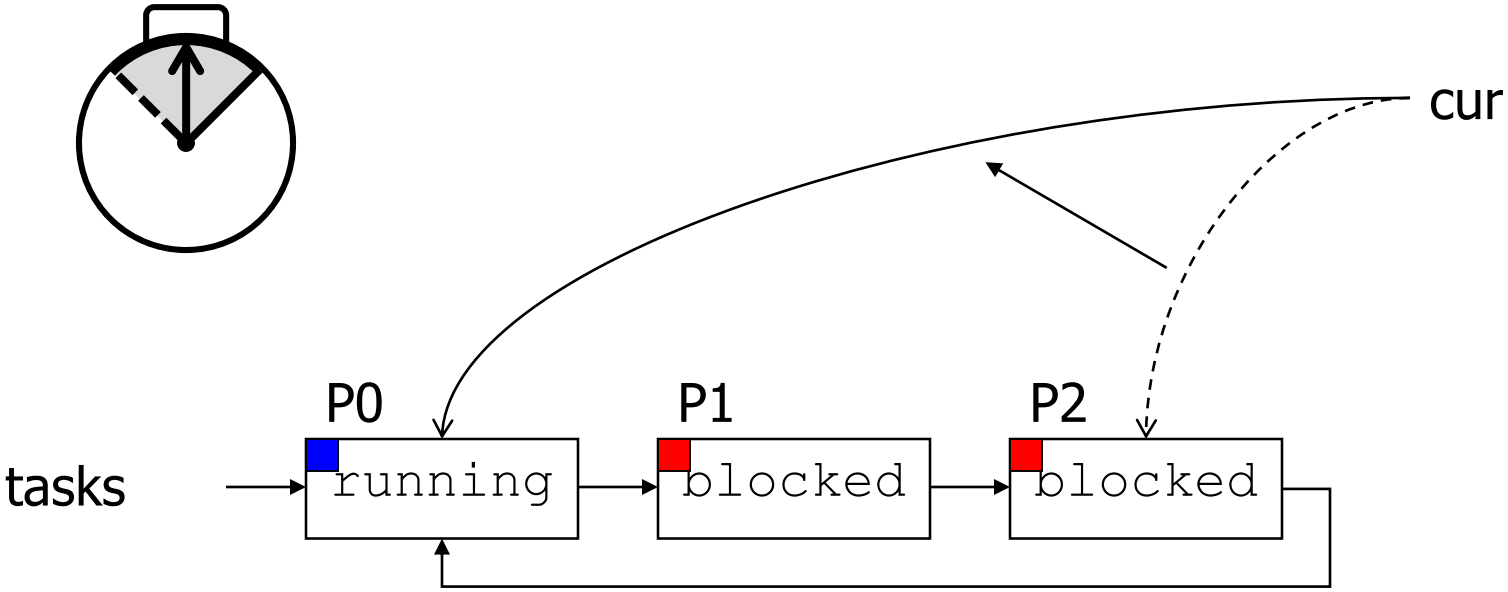
P1 εκτελείται



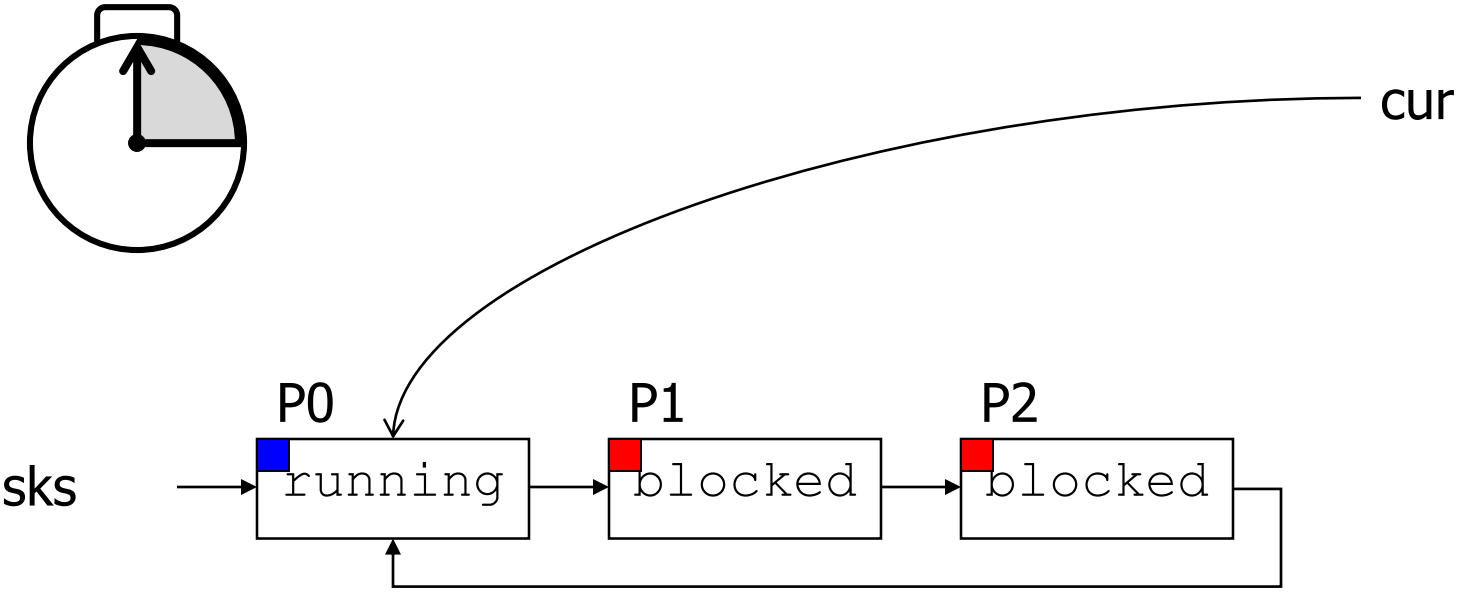
P1 μπαίνει σε αναμονή



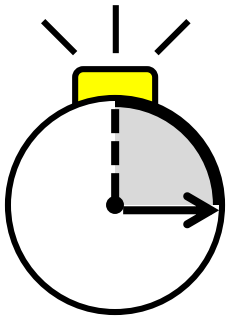
P2 εκτελείται



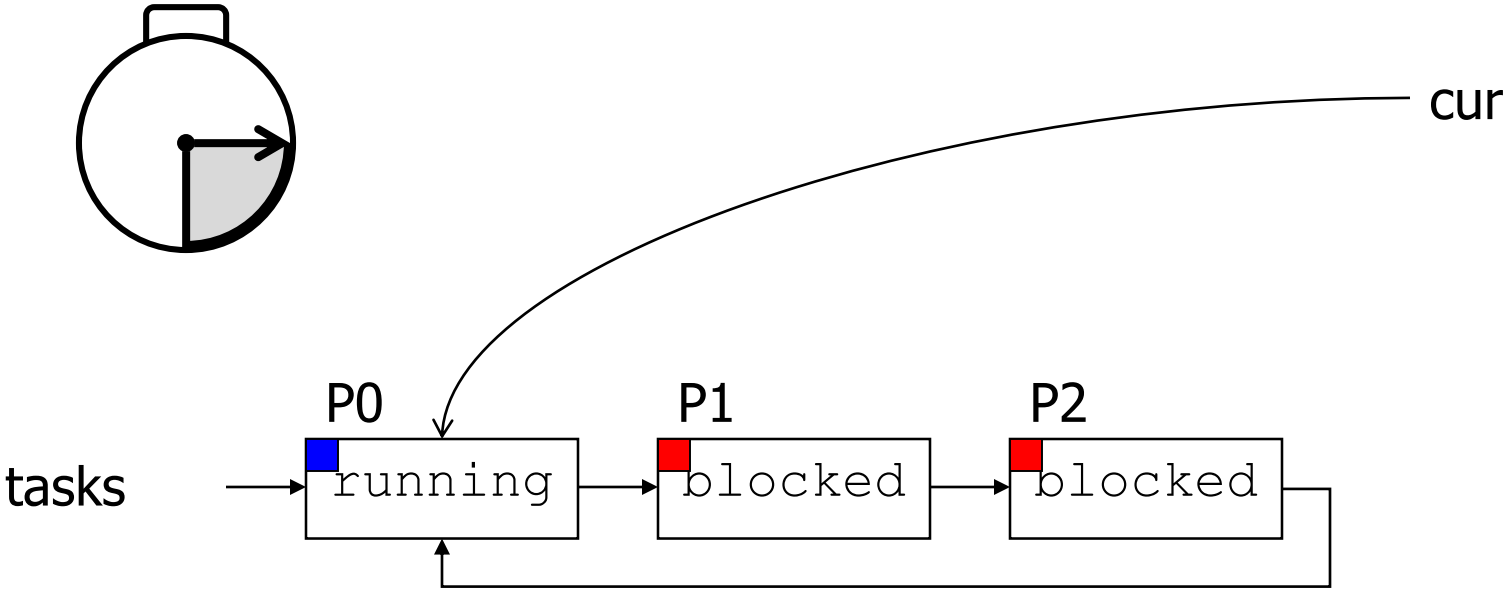
P2 μπαίνει σε αναμονή



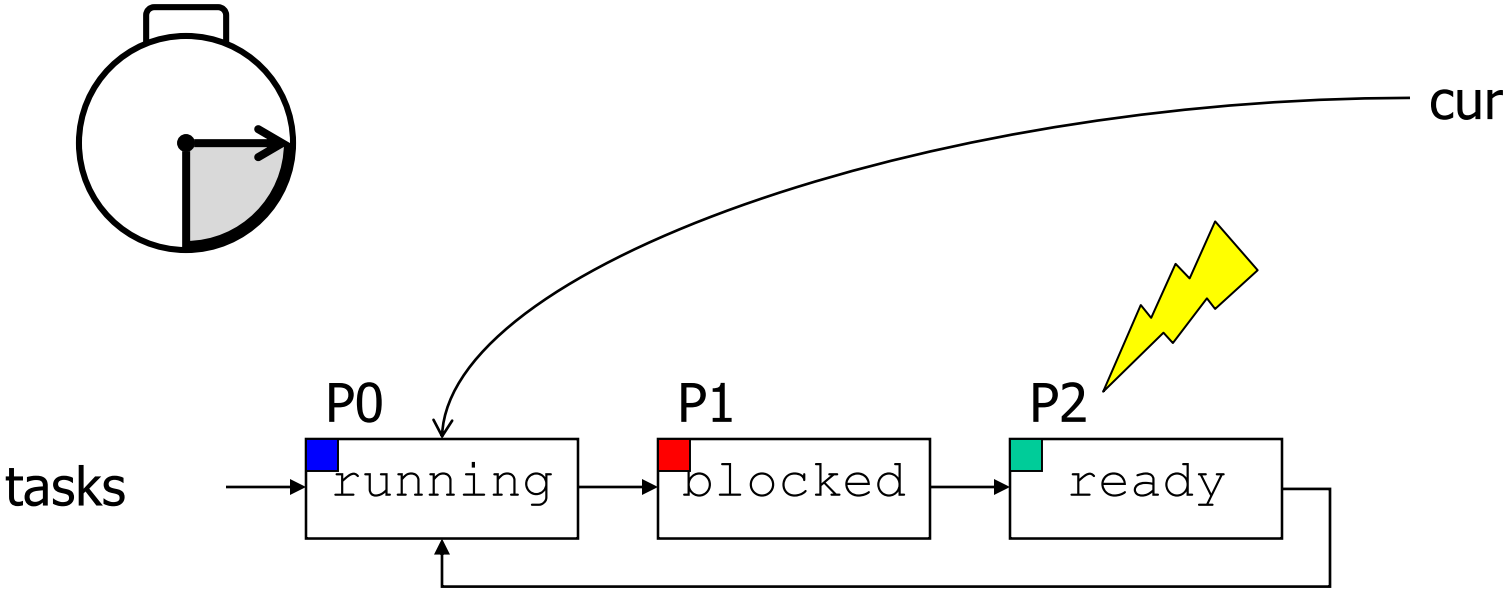
P0 εκτελείται



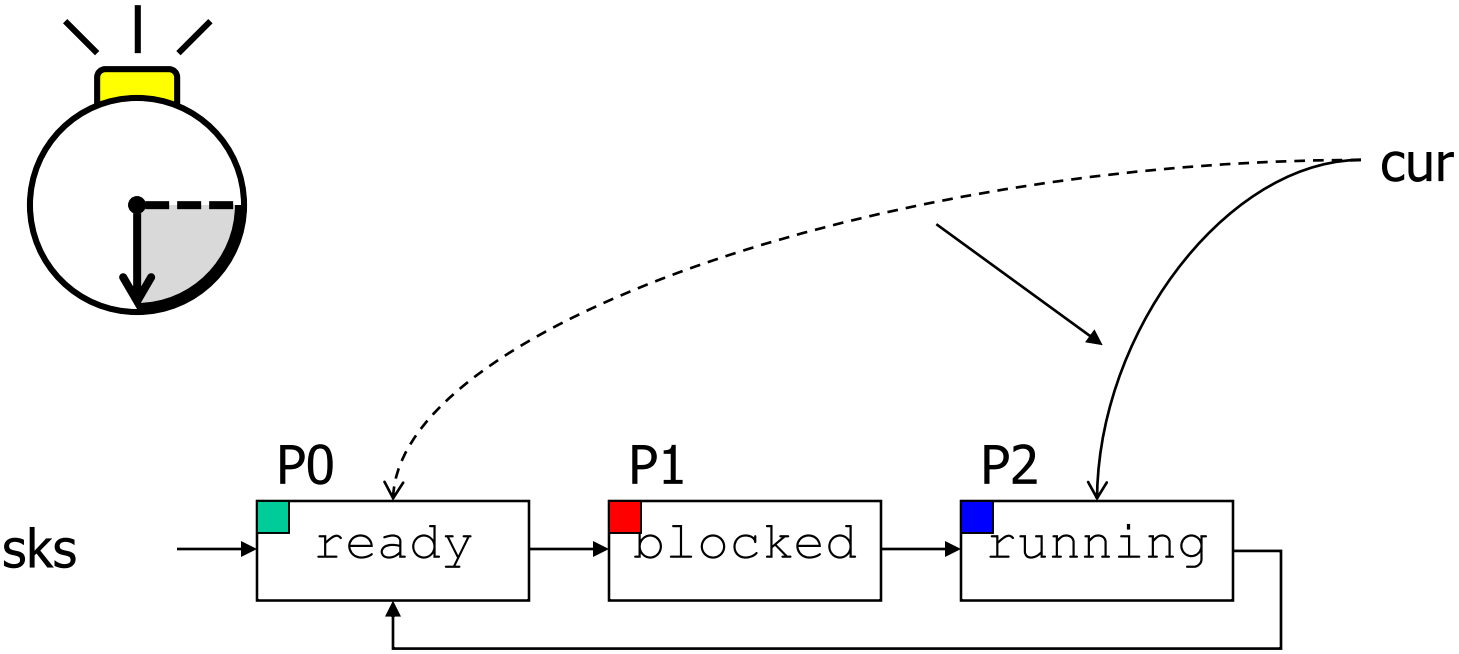
P0 εκτελείται



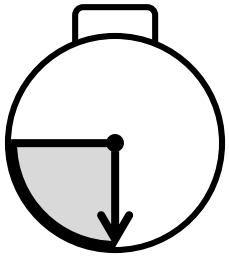
P0 εκτελείται



P2 αφυπνίζεται



εναλλαγή



P2 εκτελείται

Βελτιστοποίηση

- Αν υπάρχουν πολλές διεργασίες σε αναμονή δεν συμφέρει να τις κρατάμε στην ίδια λίστα με τις διεργασίες που είναι έτοιμες προς εκτέλεση
  - κάθε φορά που επιχειρείται εναλλαγή, πραγματοποιούνται O(N) έλεγχοι για να βρεθεί η επόμενη έτοιμη διεργασία
- Μπορούμε να εισάγουμε μια **ξεχωριστή** λίστα, αποκλειστικά για τις διεργασίες εν αναμονή
  - η εναλλαγή γίνεται σε O(1)
- Οι λειτουργίες αλλαγής κατάστασης γίνονται λίγο πιο πολύπλοκες καθώς πρέπει να «μετακινούν» μια διεργασία από την μια λίστα στην άλλη
  - αλλά και αυτό γίνεται σε O(1)

Αυτόματη εναλλαγή

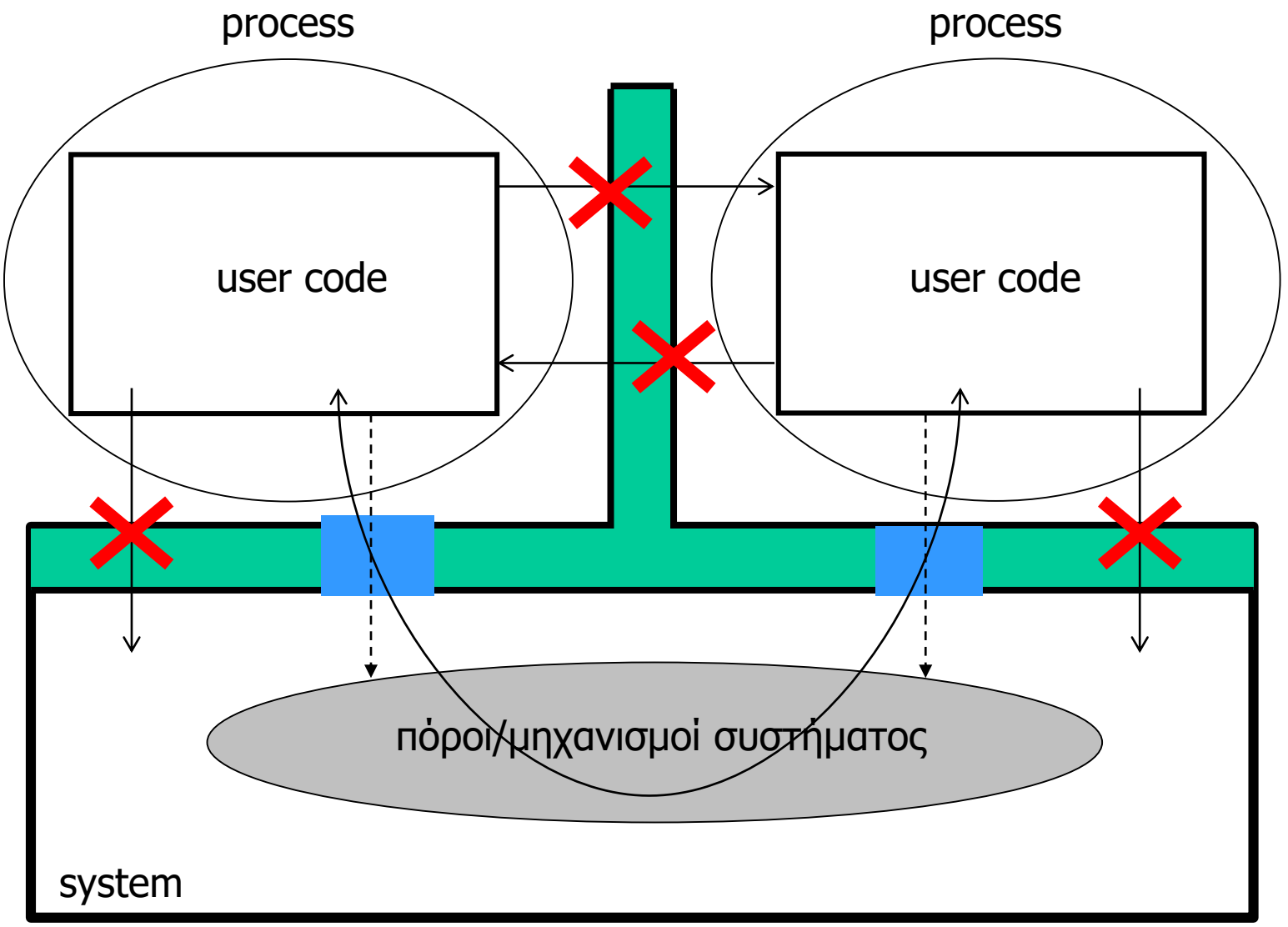
- Ο προγραμματιστής **δεν** ελέγχει την εναλλαγή
  - το λειτουργικό δεν παρέχει τέτοια λειτουργία –γιατί;

- Μπορεί να γίνει εναλλαγή, **ανά πάσα στιγμή**
  - **δεν γνωρίζουμε** αν, πότε και σε ποια σημεία του κώδικα που γράφουμε θα γίνει εναλλαγή
  - τα σημεία στα οποία γίνεται η εναλλαγή μπορεί να είναι **διαφορετικά** σε κάθε νέα εκτέλεση

- Η εναλλαγή είναι **διαφανής** για τον προγραμματιστή
  - γράφει τον κώδικα του προγράμματος του **χωρίς** να τον απασχολεί το γεγονός ότι κατά την εκτέλεση μπορεί να γίνουν εναλλαγές – εκτός αν ...

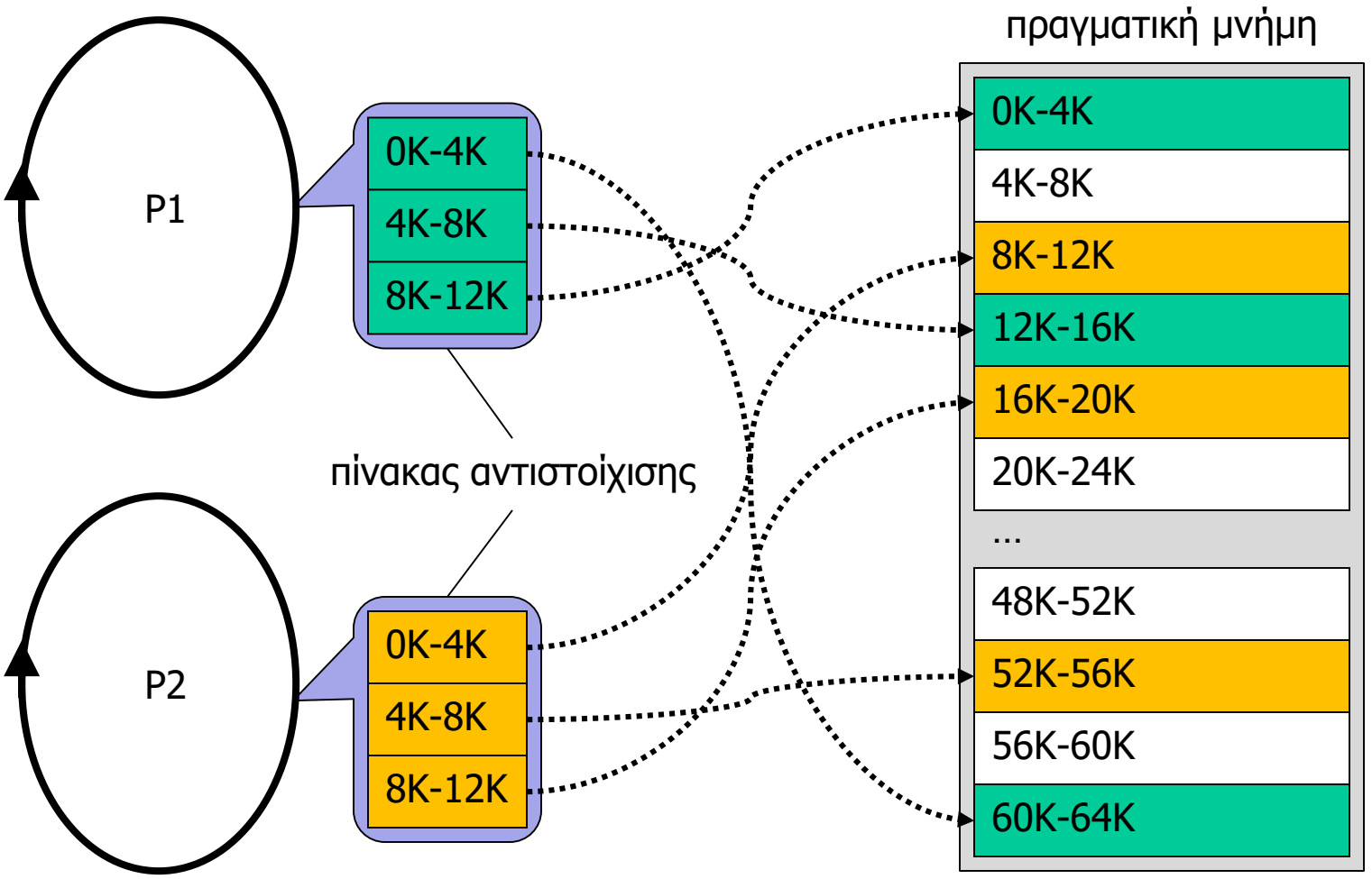
Μνήμη και επικοινωνία διεργασιών

- Κάθε διεργασία έχει **δική** της **ιδιωτική** μνήμη
- Μια διεργασία **δεν** μπορεί να γράψει/διαβάσει από/σε θέσεις μνήμης άλλων διεργασιών
- Η επικοινωνία / μεταφορά δεδομένων μεταξύ των διεργασιών γίνεται μέσα από συγκεκριμένους μηχανισμούς επικοινωνίας
- Μέσω του λειτουργικού (κλήσεις συστήματος)


process
process
user code
user code
πόροι/μηχανισμοί συστήματος
system

Εικονική μνήμη – virtual memory

- Οι διευθύνσεις μνήμης που χρησιμοποιεί ο κώδικας που εκτελούν οι διεργασίες ... είναι **εικονικές!**
- Κάθε διεργασία έχει την ψευδαίσθηση ότι έχει στην διάθεση της **ολόκληρη** την μνήμη του Η/Υ
  - ή ακόμα **περισσότερη** ...
- Το λειτουργικό πραγματοποιεί την **αντιστοίχιση** μεταξύ εικονικών και πραγματικών διευθύνσεων, εξασφαλίζοντας ξεχωριστή μνήμη για κάθε διεργασία
- Αυτό γίνεται με **διαφανή** τρόπο, χωρίς να εμπλέκεται ο προγραμματιστής ... ούτε ο μεταγλωττιστής

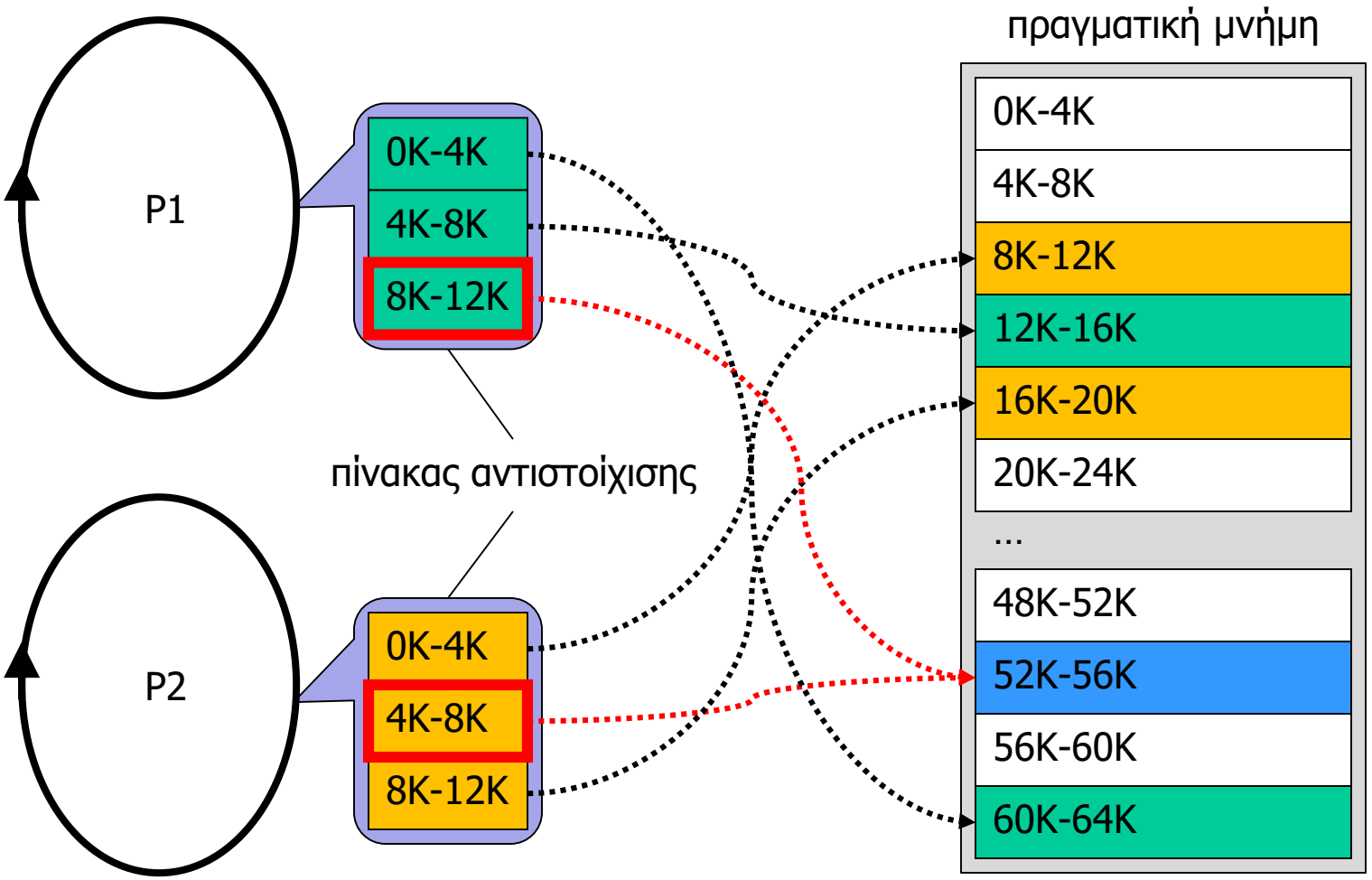

πραγματική μνήμη
P1
0K-4K
4K-8K
8K-12K
πίνακας αντιστοίχισης
P2
0K-4K
4K-8K
8K-12K
0K-4K
4K-8K
8K-12K
12K-16K
16K-20K
20K-24K
...
48K-52K
52K-56K
56K-60K
60K-64K

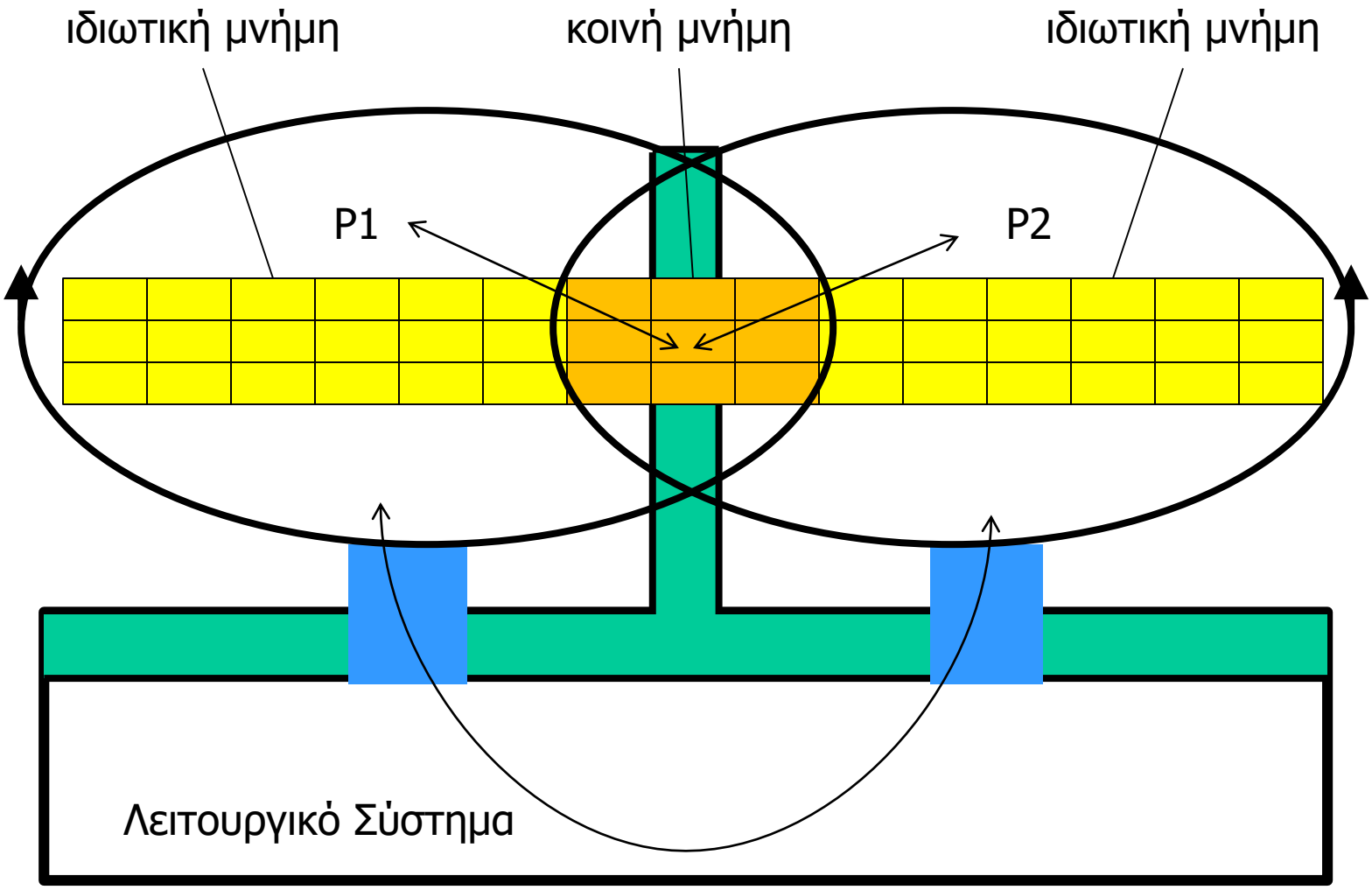
Δημιουργία κοινόχρηστης μνήμης

- **Ρύθμιση** της εικονικής μνήμης έτσι ώστε κάποιες εικονικές διευθύνσεις διαφορετικών διεργασιών να αντιστοιχηθούν στις **ίδιες** φυσικές διευθύνσεις

- Προκύπτουν **«κοινόχρηστα»** τμήματα μνήμης

- Σε αυτή την περίπτωση, οι διεργασίες μπορεί να επικοινωνούν μεταξύ τους γράφοντας/διαβάζοντας απ' ευθείας σε/από αυτές τις **κοινές** θέσεις μνήμης
- **Χωρίς να πραγματοποιούν κλήσεις συστήματος**


πραγματική μνήμη
P1
0K-4K
4K-8K
8K-12K
πίνακας αντιστοίχισης
P2
0K-4K
4K-8K
8K-12K
0K-4K
4K-8K
8K-12K
12K-16K
16K-20K
20K-24K
...
48K-52K
52K-56K
56K-60K
60K-64K


ιδιωτική μνήμη
κοινή μνήμη
ιδιωτική μνήμη
P1
P2
Λειτουργικό Σύστημα