Προγραμματισμός ΙΙ (ECE116)

#16 διεργασίες (μοντέλο εκτέλεσης/μνήμης)

Σπύρος Λάλης
lalis@uth.gr

Φόρτωση και εκτέλεση προγράμματος

- Ο εκτελέσιμος κώδικας αποθηκεύεται σε ένα αρχείο
  - ο κώδικας είναι «παθητικός», **δεν** εκτελείται από μόνος του

Η εκτέλεση γίνεται μέσα από μια **σύνθετη** διαδικασία

1. Φόρτωση του κώδικα από το αρχείο στην μνήμη
   - έλεγχος ότι το αρχείο όντως περιέχει εκτελέσιμο κώδικα
2. Δέσμευση πόρων και αρχικοποίηση
   - δέσμευση μνήμης
   - αρχικοποίηση globals / stack / heap
   - αρχικοποίηση παραμέτρων της `main` / περιγραφέων αρχείων Ε/Ε
3. Εκτέλεση
   - άλμα στην πρώτη εντολή της `main`
   - από εκεί και πέρα, τα περιεχόμενα της μνήμης αλλάζουν μέσα από τις εντολές του προγράμματος (και τις κλήσεις συστήματος)

Διεργασία

- Το λειτουργικό παρέχει έναν **ειδικό μηχανισμό** για την εκτέλεση προγραμμάτων
- Αυτός ο μηχανισμός ονομάζεται **διεργασία** (process)
  - οι διεργασίες είναι οι **οντότητες εκτέλεσης** των προγραμμάτων

- Κάθε διεργασία εκτελείται **ταυτόχρονα** και **ανεξάρτητα** από τις υπόλοιπες διεργασίες
- 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

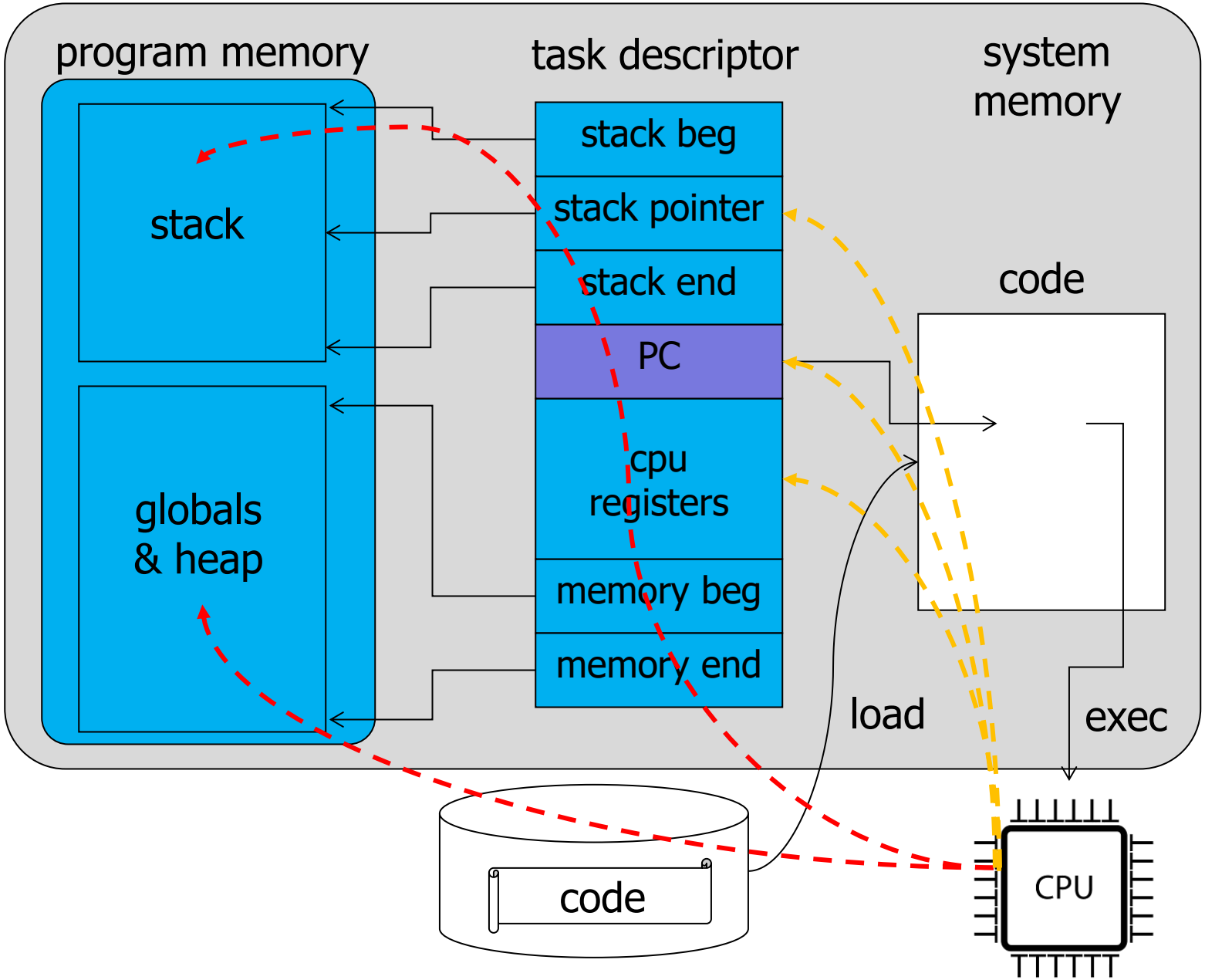
- Το **ίδιο** πρόγραμμα μπορεί να εκτελείται πολλές φόρες **ταυτόχρονα** από **διαφορετικές** διεργασίες

Πλαίσιο εκτέλεσης

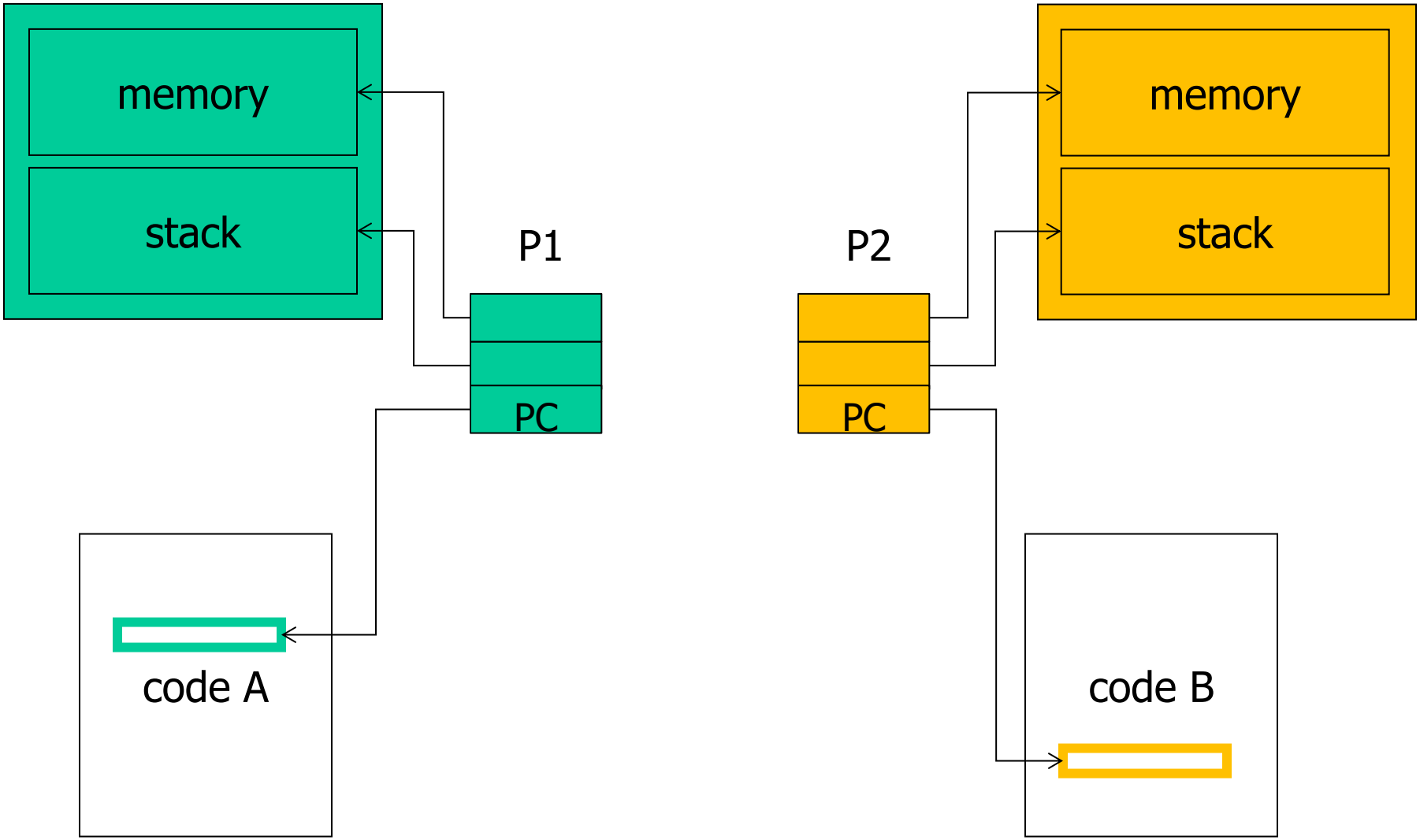
- Κάθε διεργασία αντιστοιχεί σε ένα ξεχωριστό **πλαίσιο εκτέλεσης (execution context)**

- Περιλαμβάνει την πλήρη κατάσταση εκτέλεσης του προγράμματος
  - στατική μνήμη, δυναμική μνήμη, στοίβα, file descriptors …
  - κατάσταση του επεξεργαστή

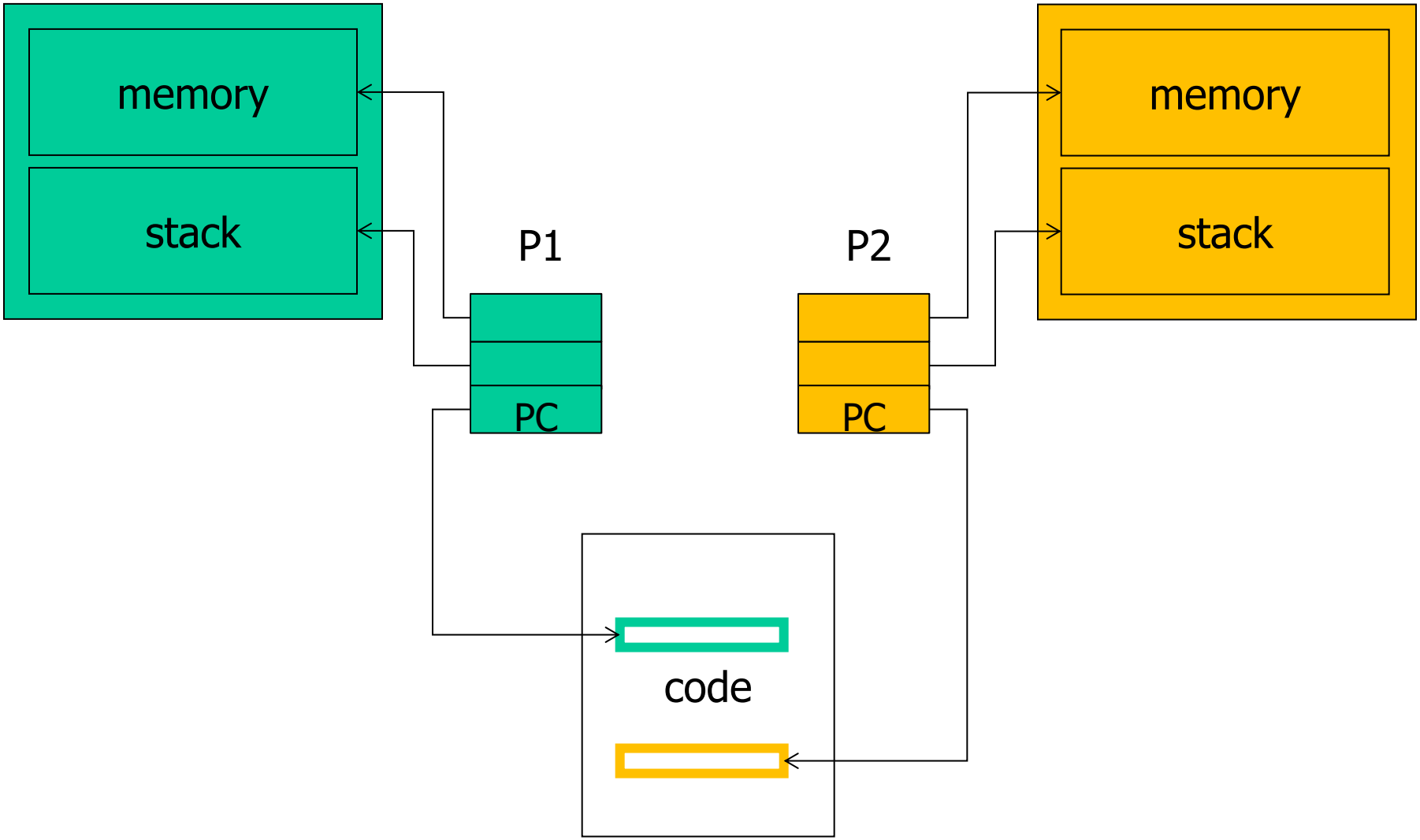
- Και άλλες πληροφορίες
  - ο χρήστης στο όνομα του οποίου γίνεται η εκτέλεση
  - προτεραιότητα εκτέλεσης
  - χρόνος που έχει αφιερωθεί για αυτή την εκτέλεση


program memory
task descriptor
system memory
stack
globals & heap
stack beg
stack pointer
stack end
PC
cpu registers
memory beg
memory end
code
load
exec
code
CPU

Διεργασίες που εκτελούν διαφορετικό κώδικα



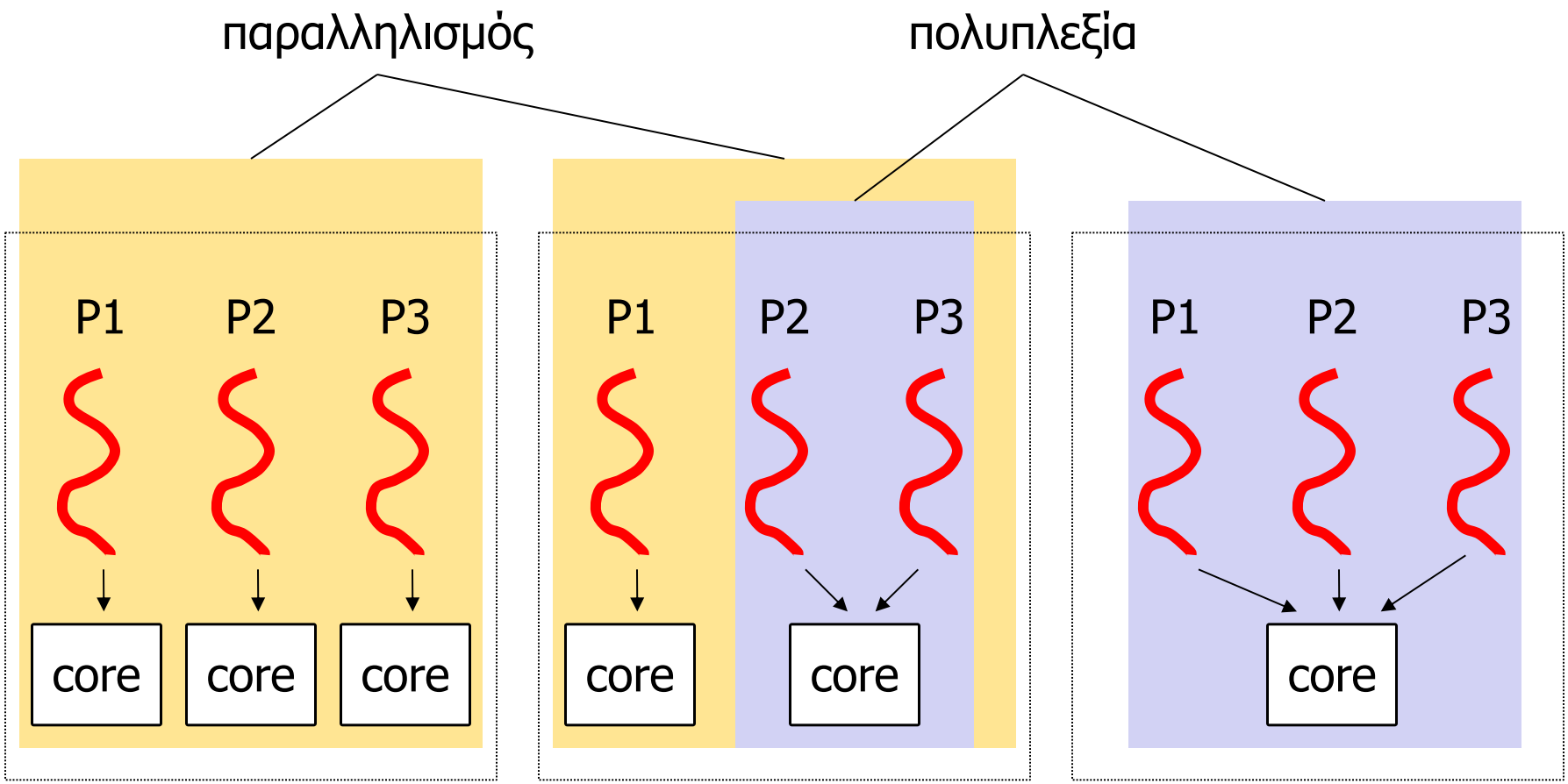
Διεργασίες που εκτελούν τον ίδιο κώδικα

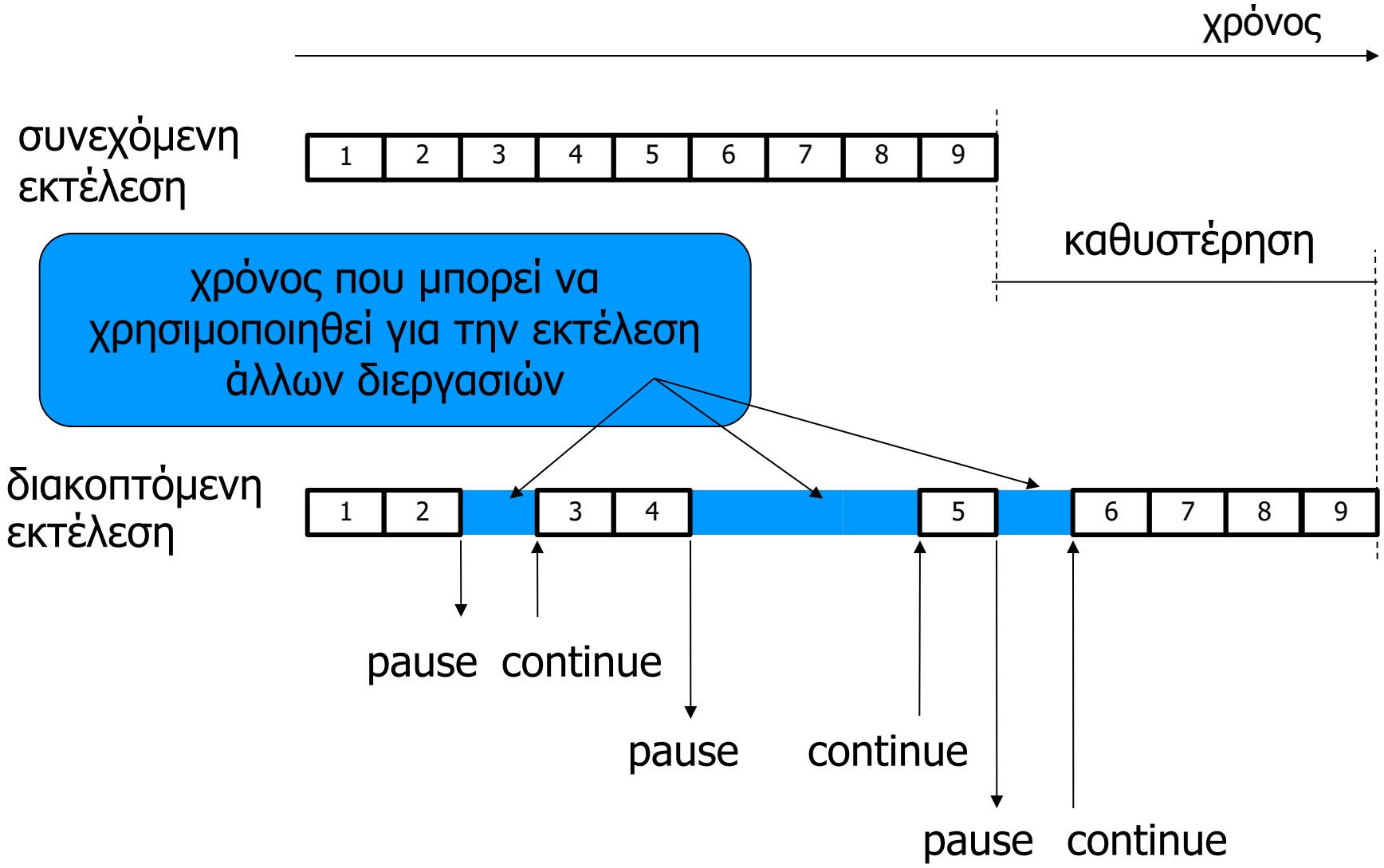


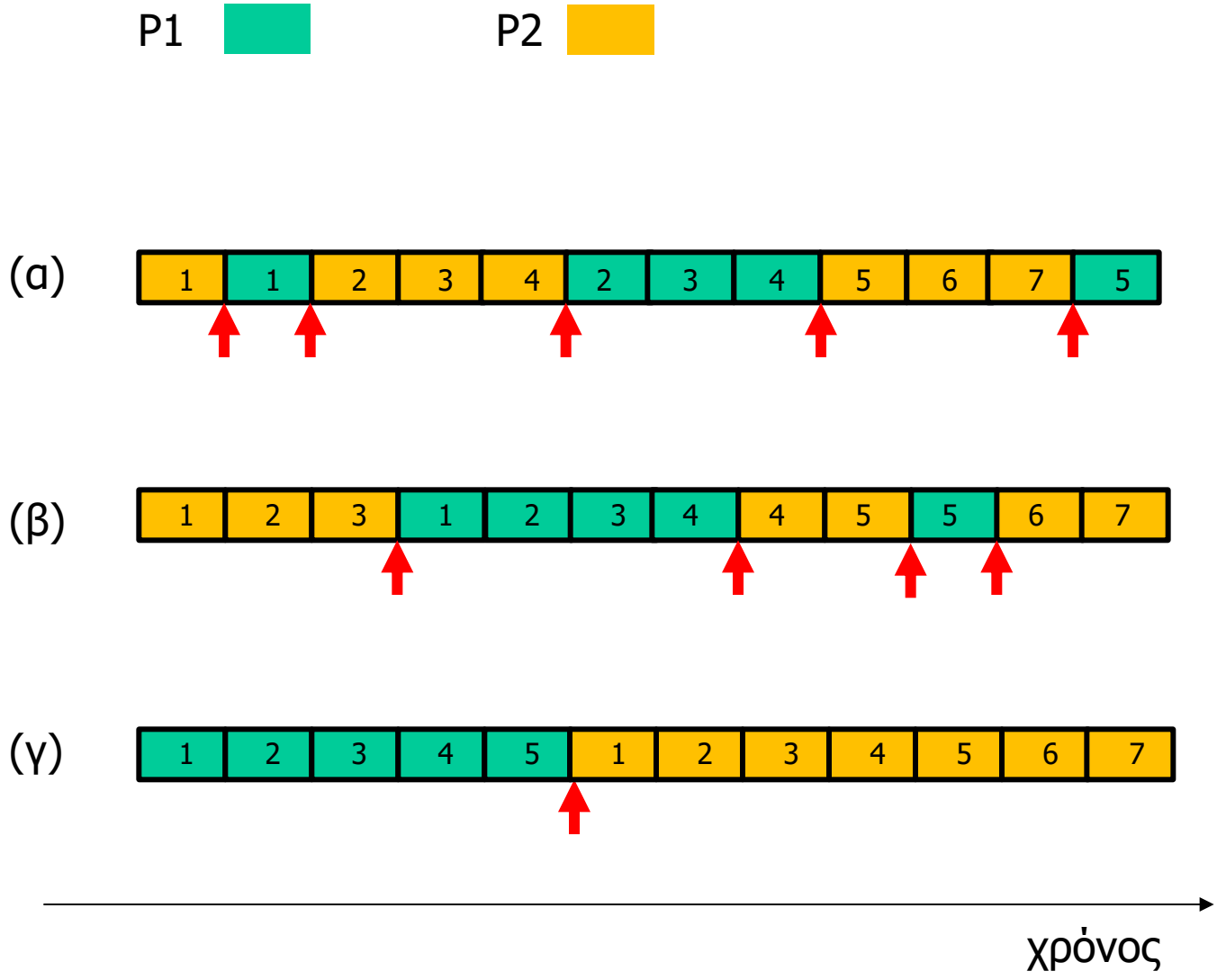
Γιατί χρειαζόμαστε πολλές διεργασίες;

- Πολλά προγράμματα
  - θέλουμε να «τρέχουν» την ίδια στιγμή
- Πολλοί χρήστες
  - θέλουμε να μπορούν να εκτελούν τα δικά τους προγράμματα, εντελώς ανεξάρτητα από τους άλλους
- Ταυτοχρονισμός / παραλληλισμός
  - εκτέλεση διαφορετικών τμημάτων μιας επεξεργασίας ή ενός υπολογισμού από ξεχωριστές διεργασίες
  - αποφυγή μπλοκαρίσματος εν' αναμονή κάποιου γεγονότος
- Καλύτερη δόμηση
  - μια πολύπλοκη διαδικασία ίσως μπορεί να μοντελοποιηθεί πιο απλά αν δομηθεί ως ένα «σύστημα» από διεργασίες

Εκτέλεση: παραλληλισμός ή πολυπλεξία



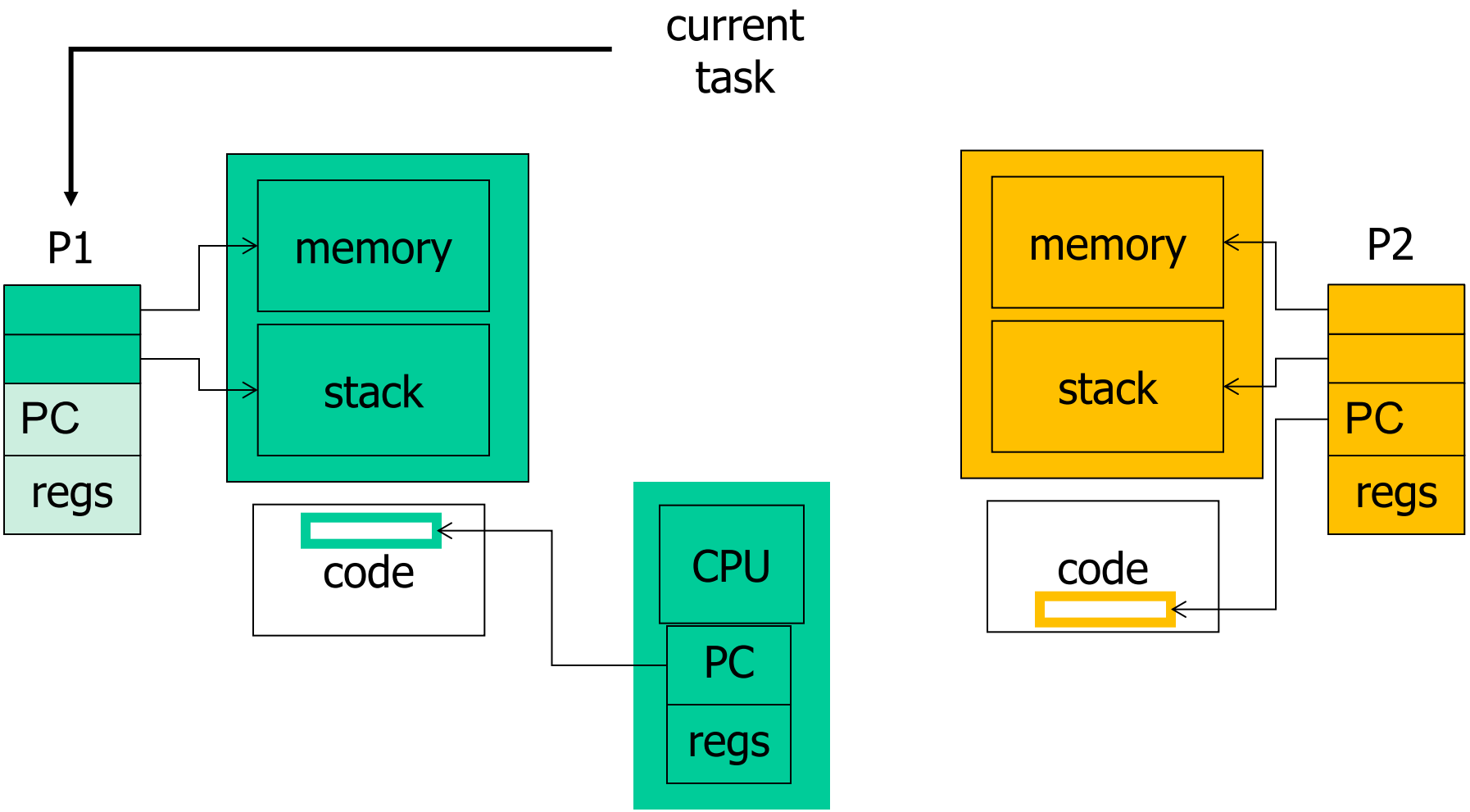

χρόνος
συνεχόμενη εκτέλεση
1 2 3 4 5 6 7 8 9
καθυστέρηση
χρόνος που μπορεί να χρησιμοποιηθεί για την εκτέλεση άλλων διεργασιών
διακοπτόμενη εκτέλεση
1 2 3 4 5 6 7 8 9
pause
continue
pause
continue
pause
continue


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

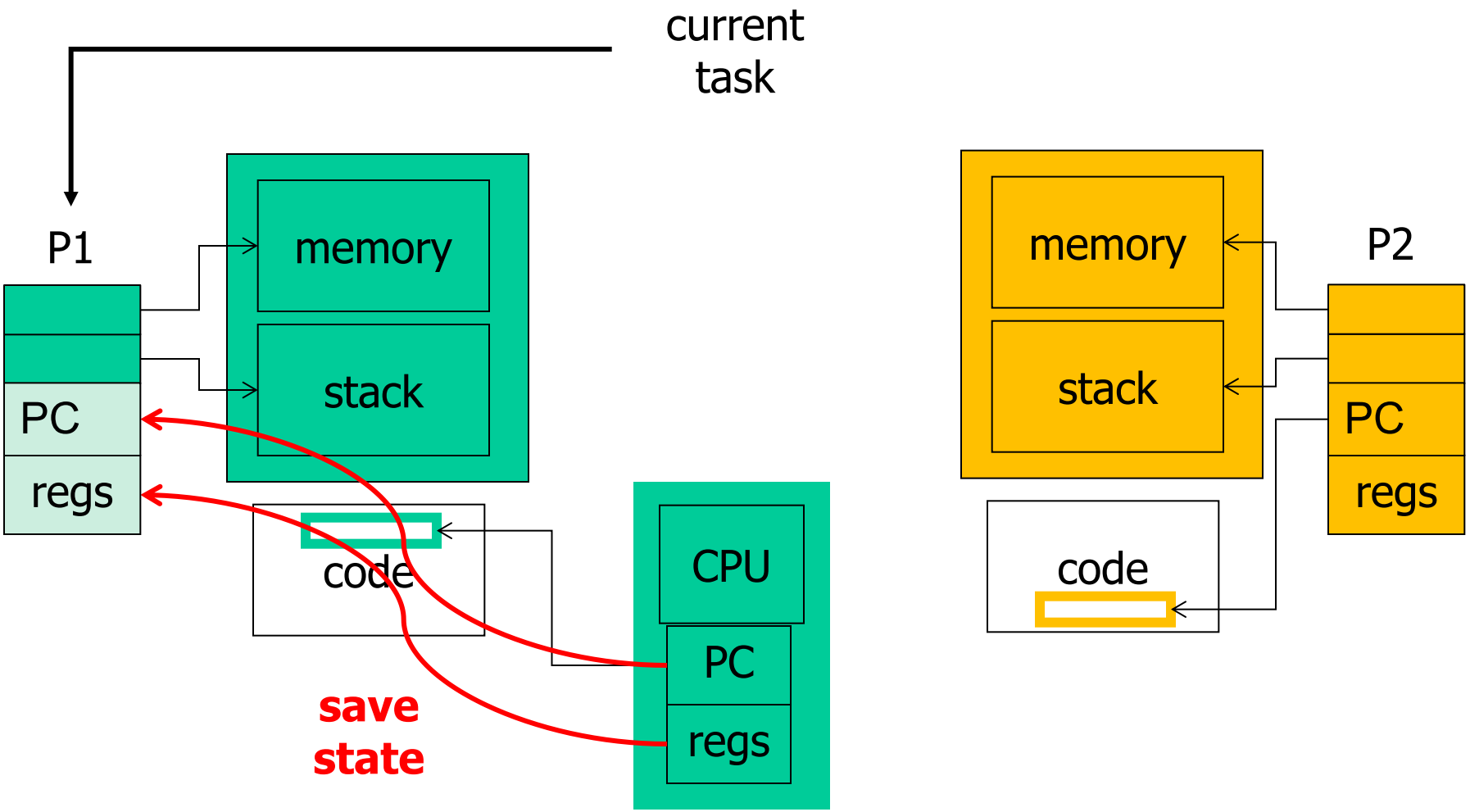
Εναλλαγή διεργασιών (context switch)

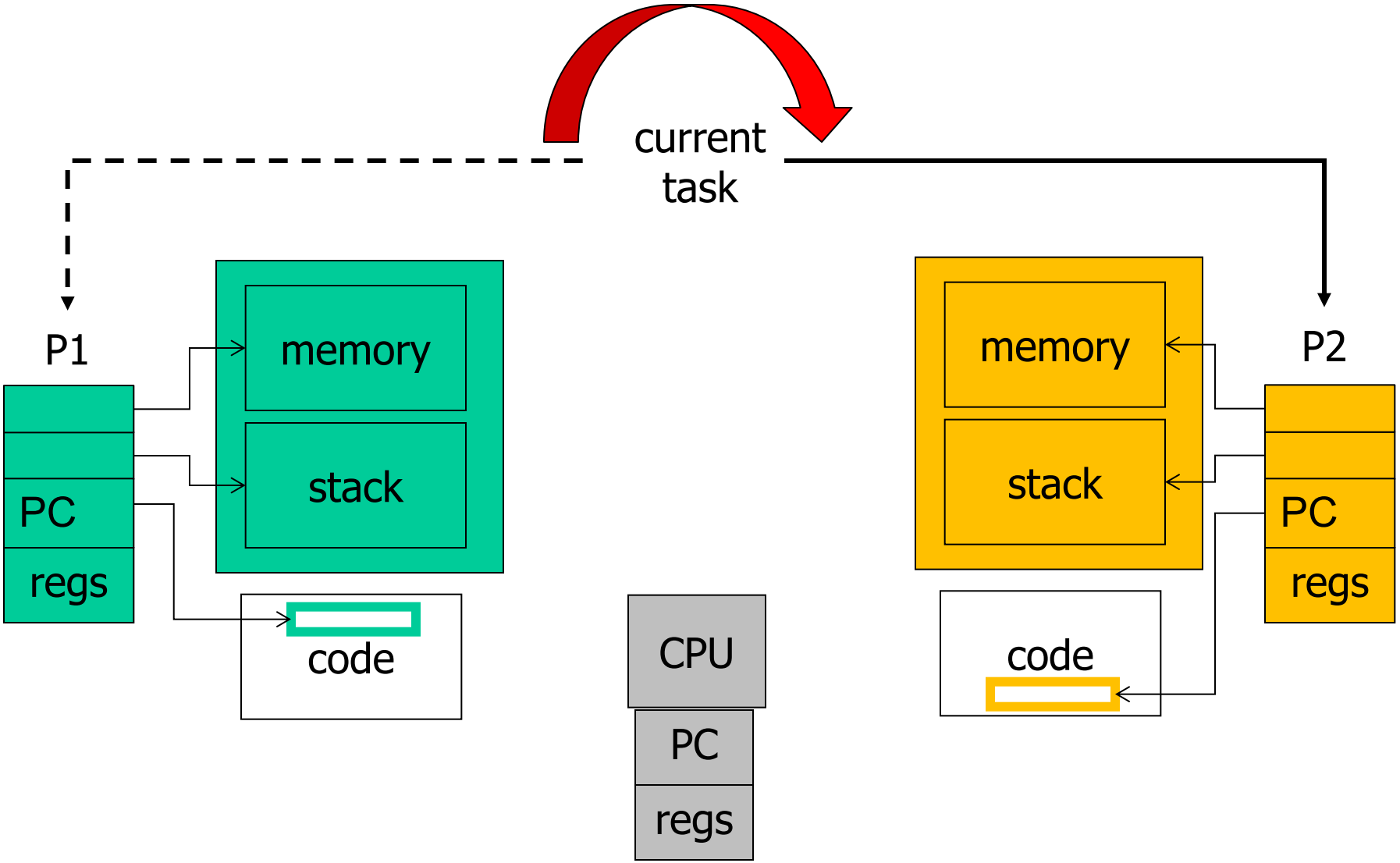
- Το λειτουργικό **σώνει** την κατάσταση εκτέλεσης της τρέχουσας διεργασίας
  - αντιγραφή της απαραίτητης πληροφορίας του επεξεργαστή μέσα στην δομή του task που κρατά το λειτουργικό για την διεργασία
- Το λειτουργικό **επιλέγει** την διεργασία που θα συνεχίσει την εκτέλεση της
- Το λειτουργικό **επαναφέρει** την κατάσταση εκτέλεσης της διεργασίας
  - αντιγραφή της απαραίτητης πληροφορίας από την δομή του task (όπου είχε αποθηκευτεί στο σώσιμο) πίσω στον επεξεργαστή

P1 εκτελείται

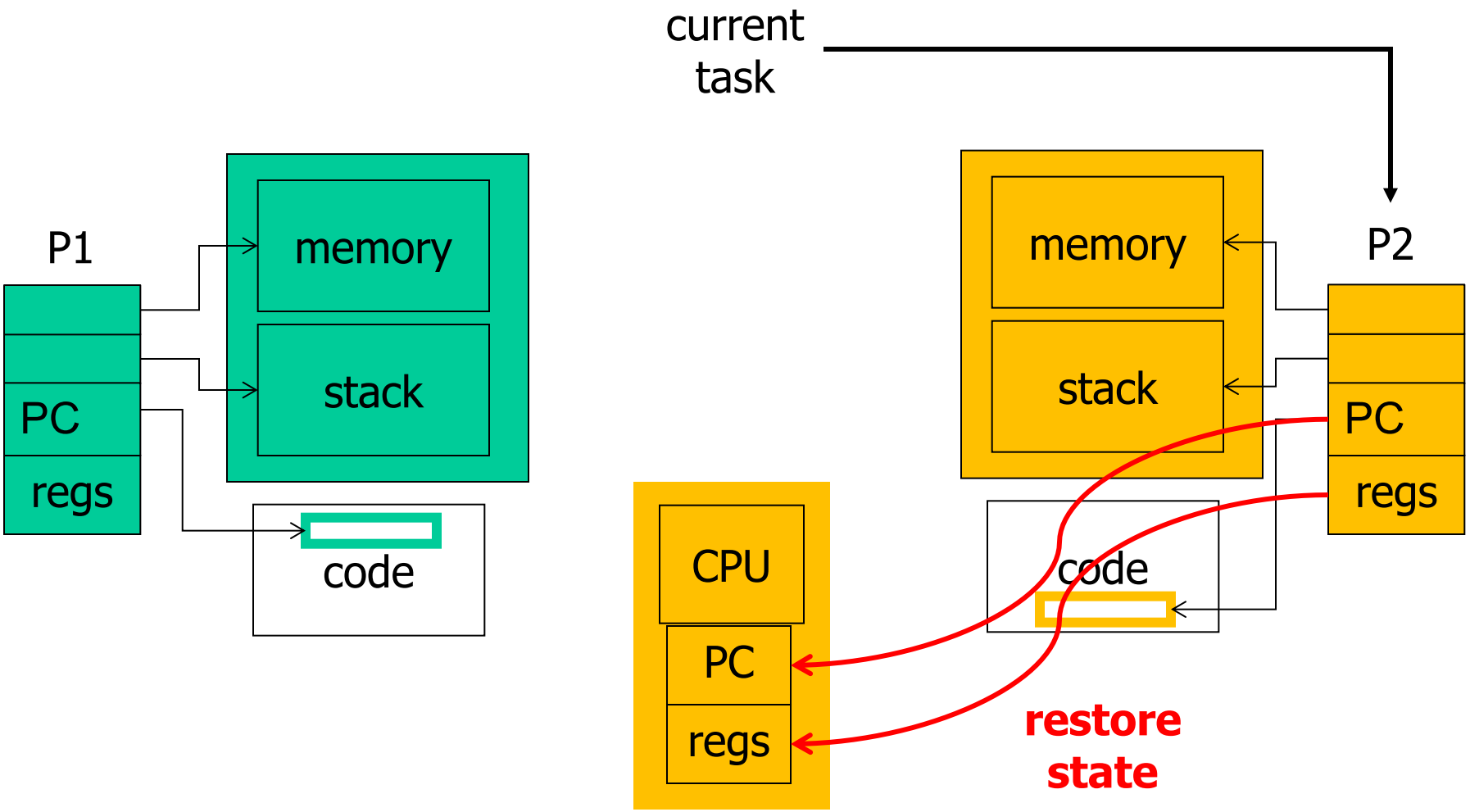


σώσιμο κατάστασης P1

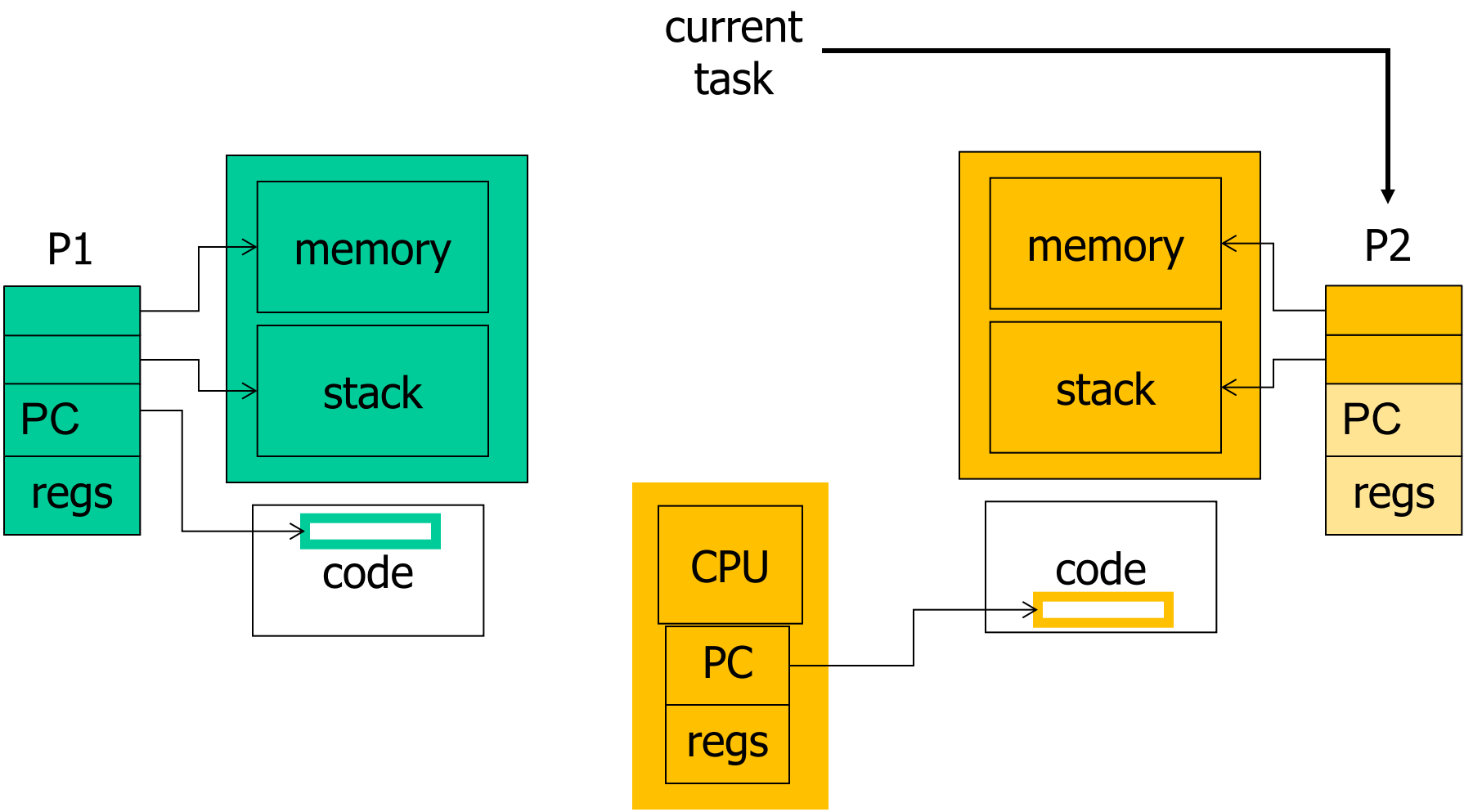



current
task
P1
memory
stack
PC
regs
code
CPU
PC
regs
memory
stack
P2
PC
regs
code

επαναφορά κατάστασης P2



P2 εκτελείται



Προβολή διεργασιών

- `ps`
  - πληροφορίες για τις διεργασίες που εκτελούνται μέσα από το τρέχον τερματικό ή/και σε όλο το σύστημα
  - υπάρχουν πολλές επιλογές, βλέπε manual
- `top`
  - στατιστικά εκτέλεσης διεργασιών και πληροφορία εκτέλεσης διεργασιών, με περιοδική ανανέωση
- `kill`
  - αποστολή σήματος (βλέπε αργότερα)
  - τερματισμός διεργασιών (συνηθισμένη χρήση)
- `pgrep/pkill`
  - αναζήτηση/τερματισμός διεργασιών με βάση χαρακτηριστικά όπως το όνομα του προγράμματος, το όνομα του χρήστη, ...

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
  - synchronization

πότε θα έρθει ο επόμενος χαρακτήρας από το πληκτρολόγιο;

πότε θα έρθουν τα δεδομένα που θέλω;

P1

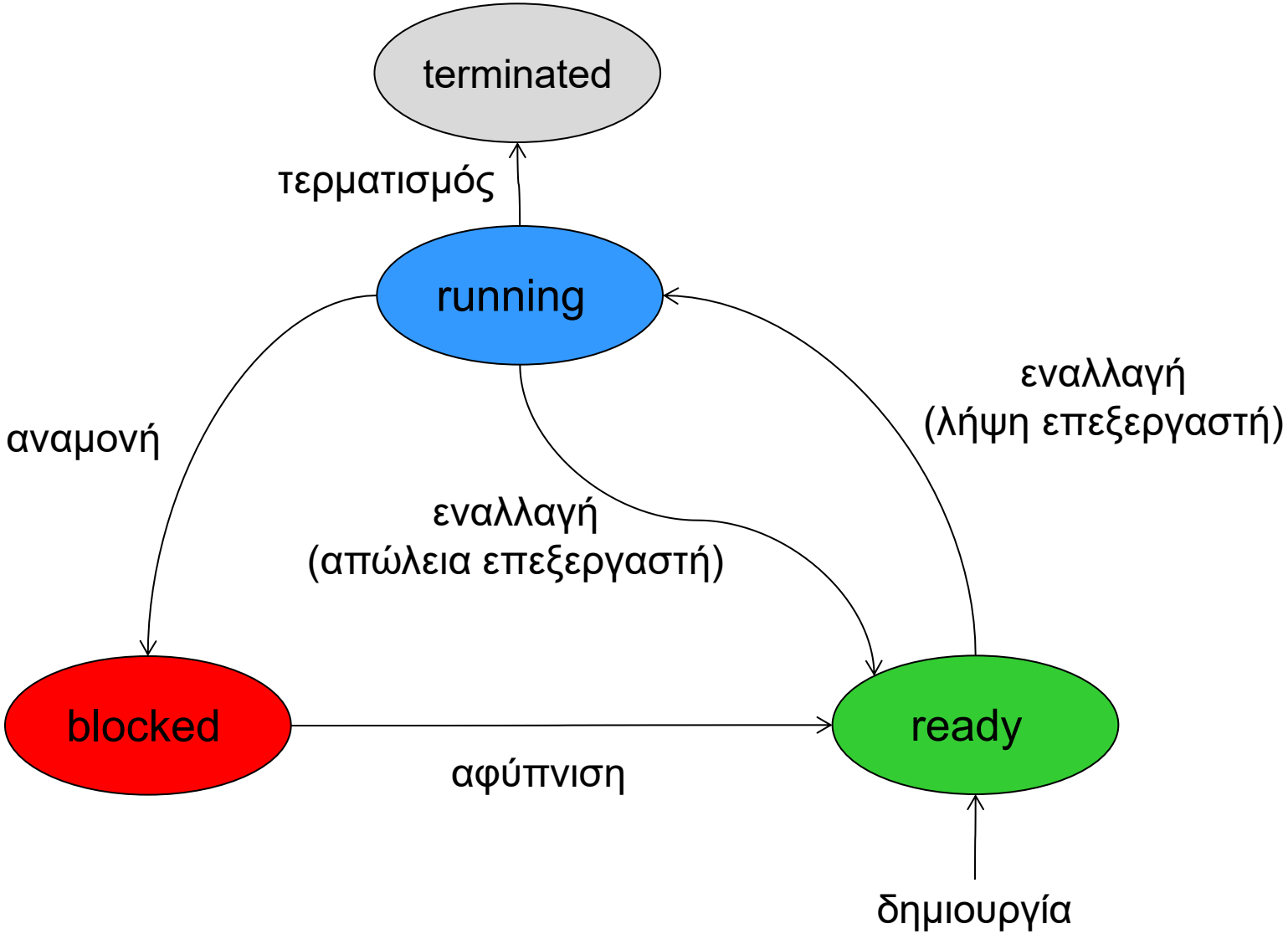
P2

πότε θα χτυπήσει το ξυπνητήρι μου;

P3

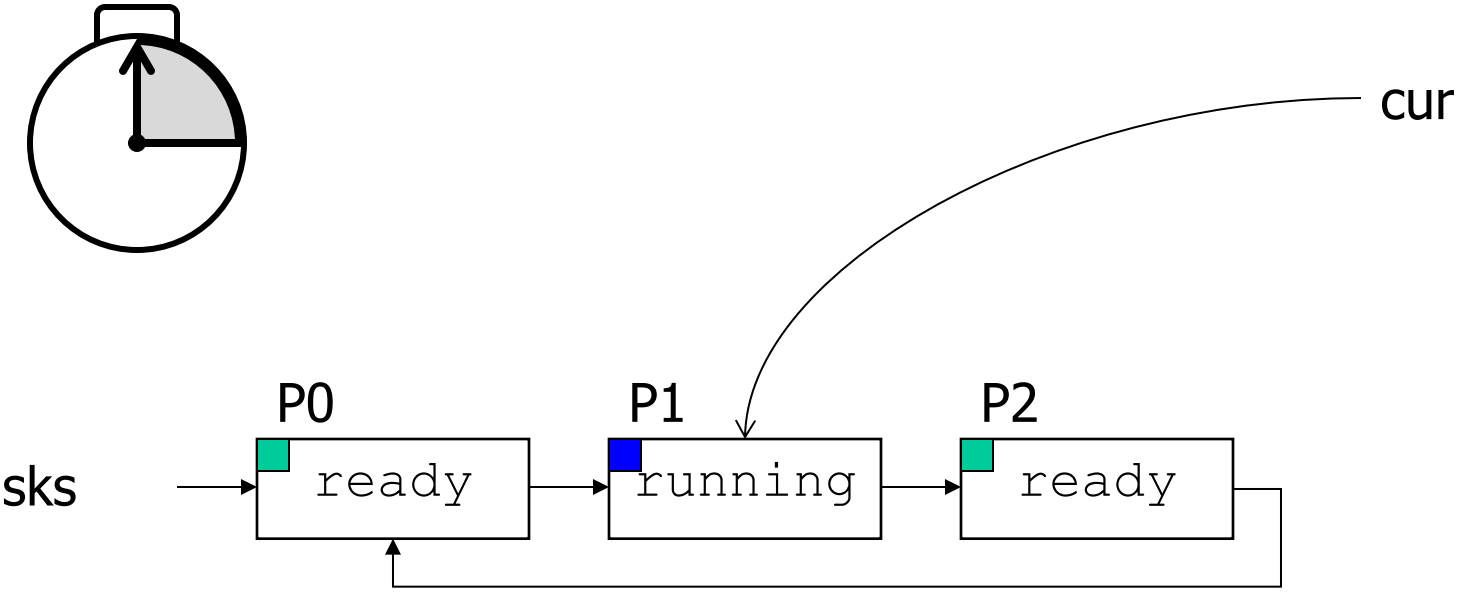
Κατάσταση αναμονής

- Το λειτουργικό σύστημα πρέπει να **γνωρίζει** κατά πόσο μια διεργασία βρίσκεται σε αναμονή
  - ώστε να **μην** κάνει **καθόλου** εναλλαγή σε αυτή
- Διαχωρισμός κατάστασης μιας διεργασίας
- **Ready: έτοιμη** να συνεχίζει την εκτέλεση της (να λάβει τον επεξεργαστή όταν αυτός ελευθερωθεί)
- **Blocked/Waiting/Sleeping:** έχει μπλοκάρει **περιμένοντας** για κάτι (δεν είναι έτοιμη να λάβει τον επεξεργαστή)
- **Running:** εκτελείται (χρησιμοποιεί τον επεξεργαστή)
- Οι **μεταβάσεις** ανάμεσα σε αυτές τις καταστάσεις γίνονται **υπό τον έλεγχο** του λειτουργικού
  - μέσα από διάφορες λειτουργίες του συστήματος

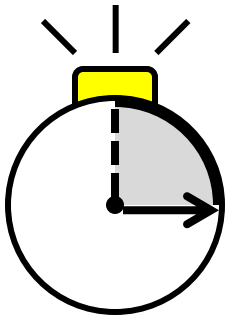

terminated
τερματισμός
running
αναμονή
εναλλαγή
(λήψη επεξεργαστή)
εναλλαγή
(απώλεια επεξεργαστή)
blocked
ready
αφύπνιση
δημιουργία

Χρονοπρογραμματισμός διεργασιών

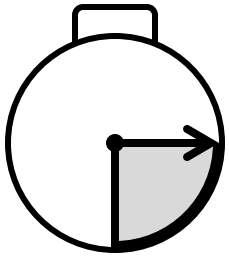
- Πως αποφασίζεται πως θα μοιραστεί ο χρόνος του επεξεργαστή ανάμεσα στις (έτοιμες) διεργασίες;
- Αυτό γίνεται με βάση μια λεγόμενη **πολιτική** που έχει δύο βασικές διαστάσεις
- **Πότε** διακόπτεται η τρέχουσα διεργασία;
  - αυτόματα (περιοδικά) αφού εκπνεύσει ο χρόνος που της δόθηκε
  - όταν καλέσει μια ανασταλτική λειτουργία του συστήματος
  - αν αποφασίσει να απελευθερώσει τον επεξεργαστή (yield)
- **Ποιά** διεργασία θα επιλεγεί για να συνεχίσει την εκτέλεση της;
  - αποφασίζεται με βάση τις προτεραιότητες των διεργασιών
- **Η πιο απλοϊκή πολιτική:** σταθερό time slice για όλες τις διεργασίες και επιλογή round-robin



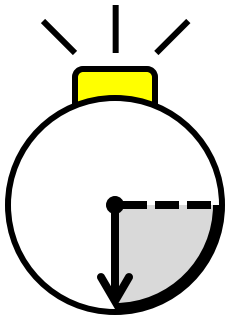
P1 εκτελείται



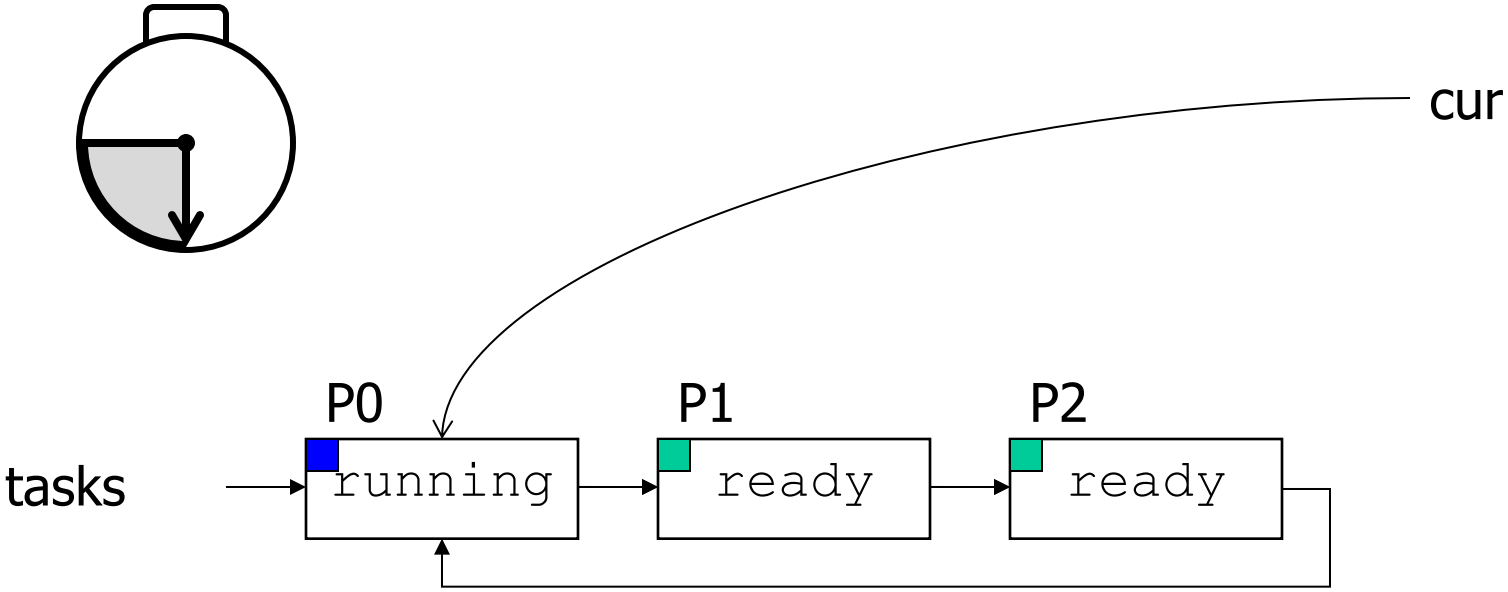
εναλλαγή



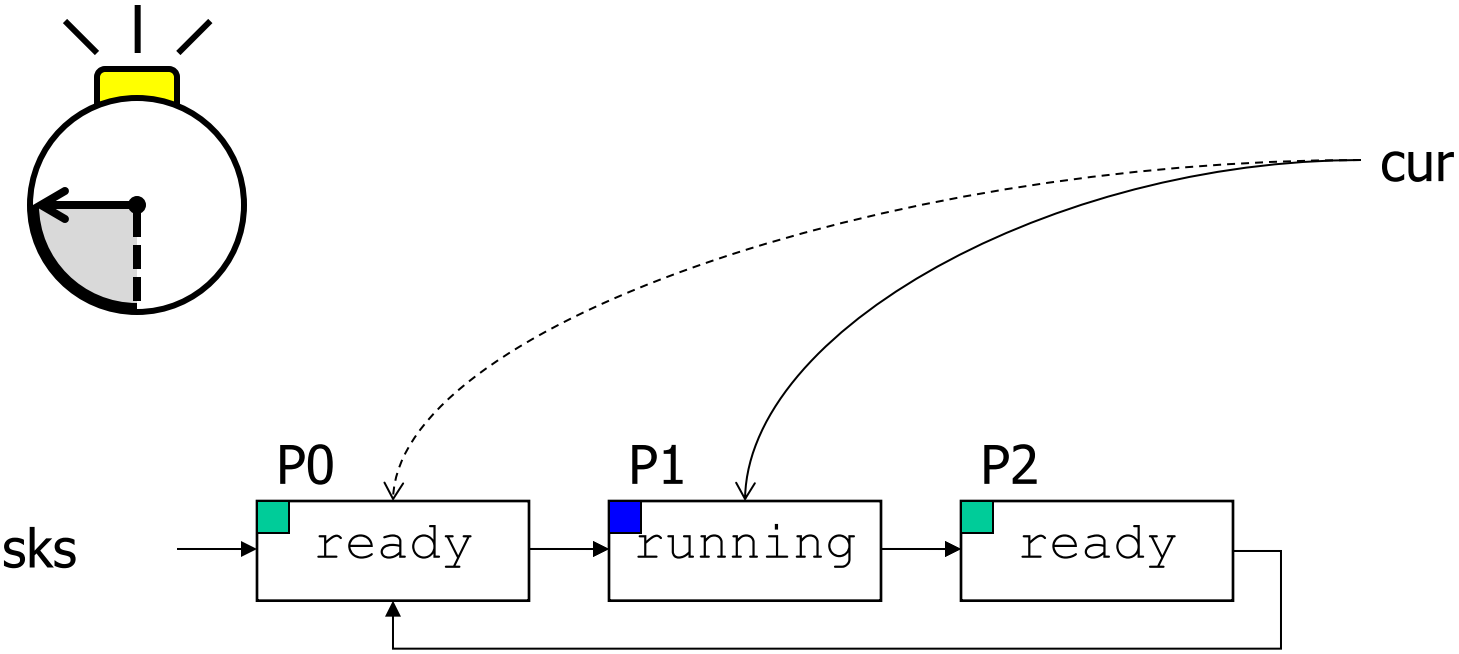
P2 εκτελείται



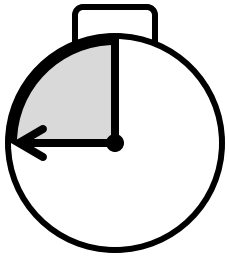
εναλλαγή



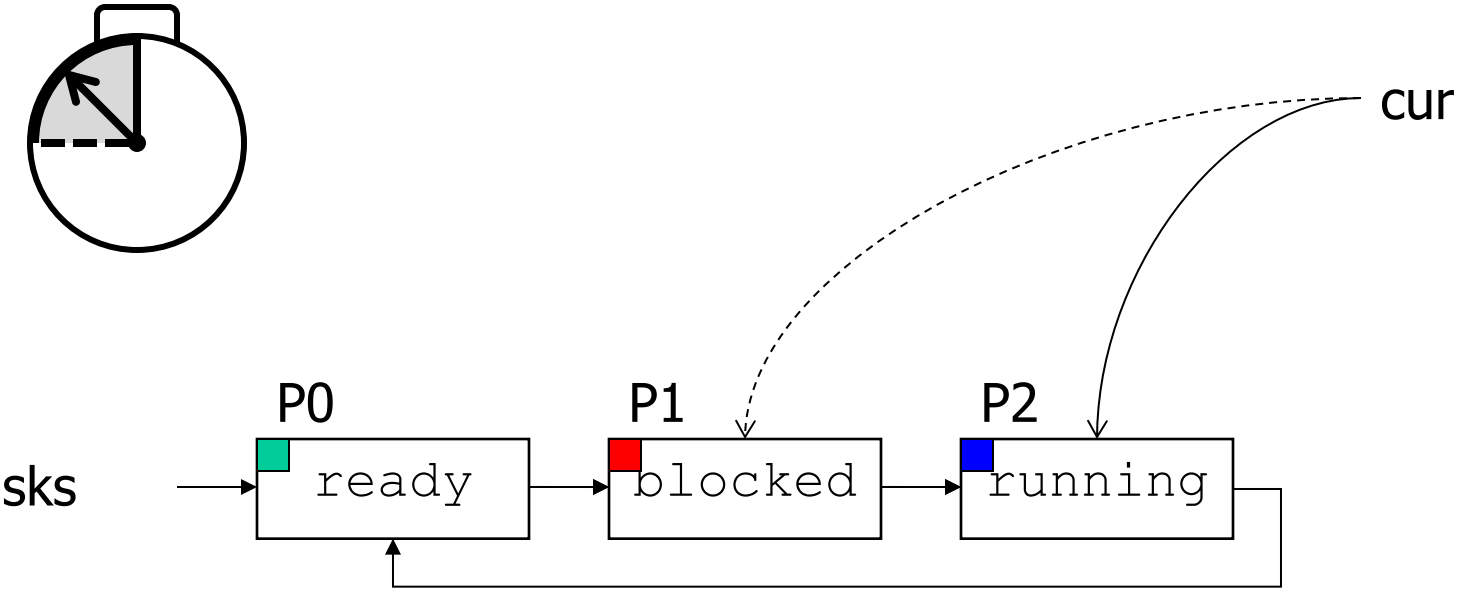
P0 εκτελείται



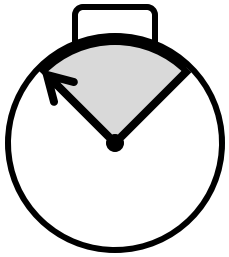
εναλλαγή



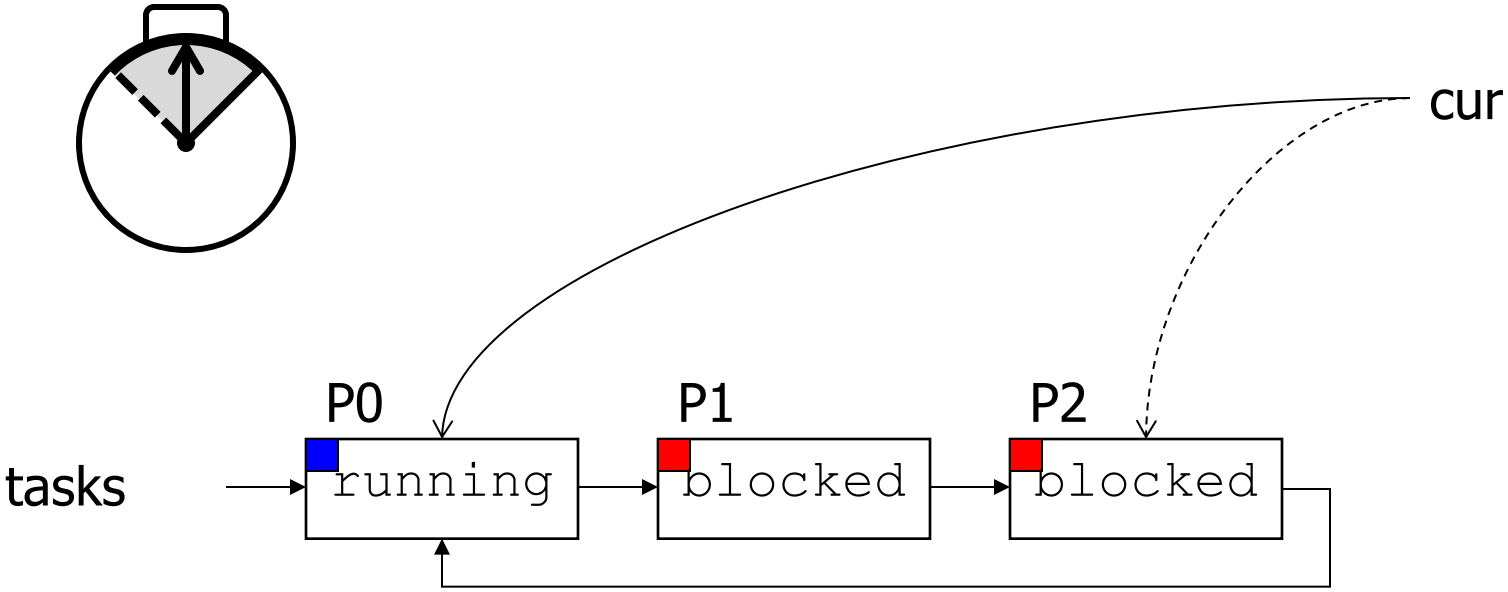
P1 εκτελείται



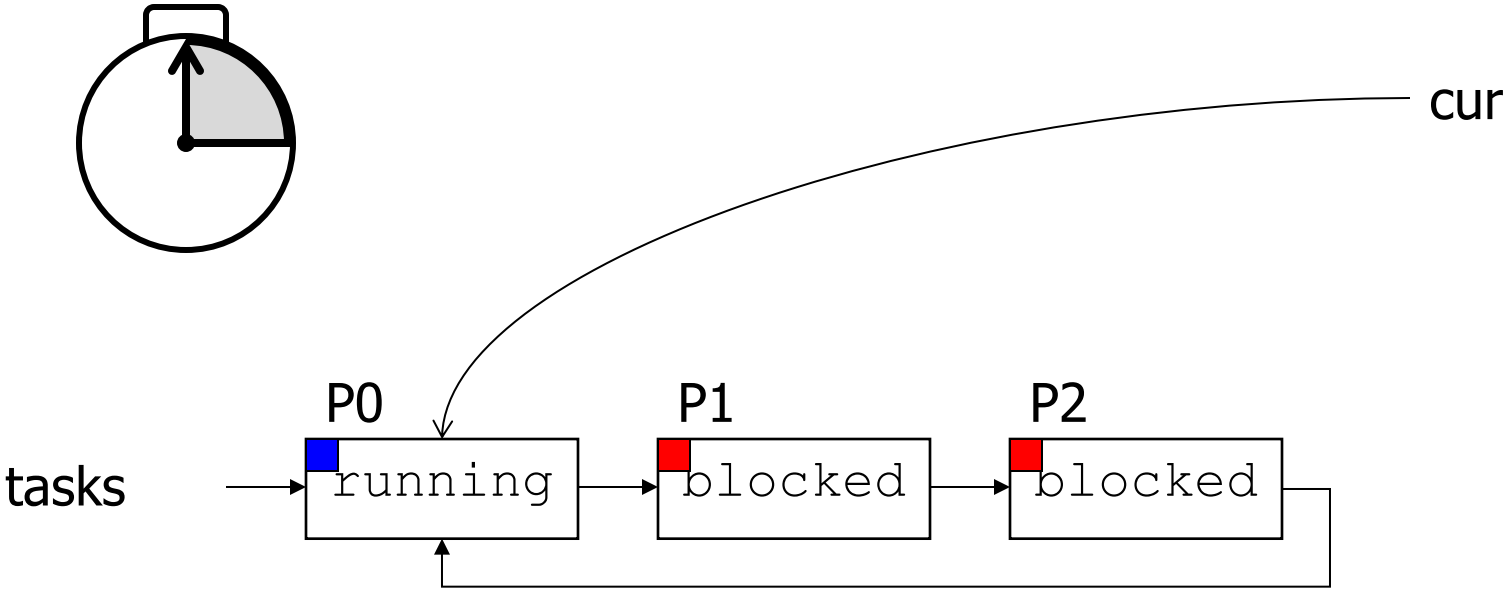
P1 μπαίνει σε αναμονή



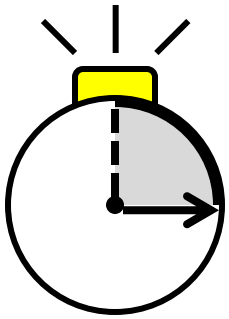
P2 εκτελείται



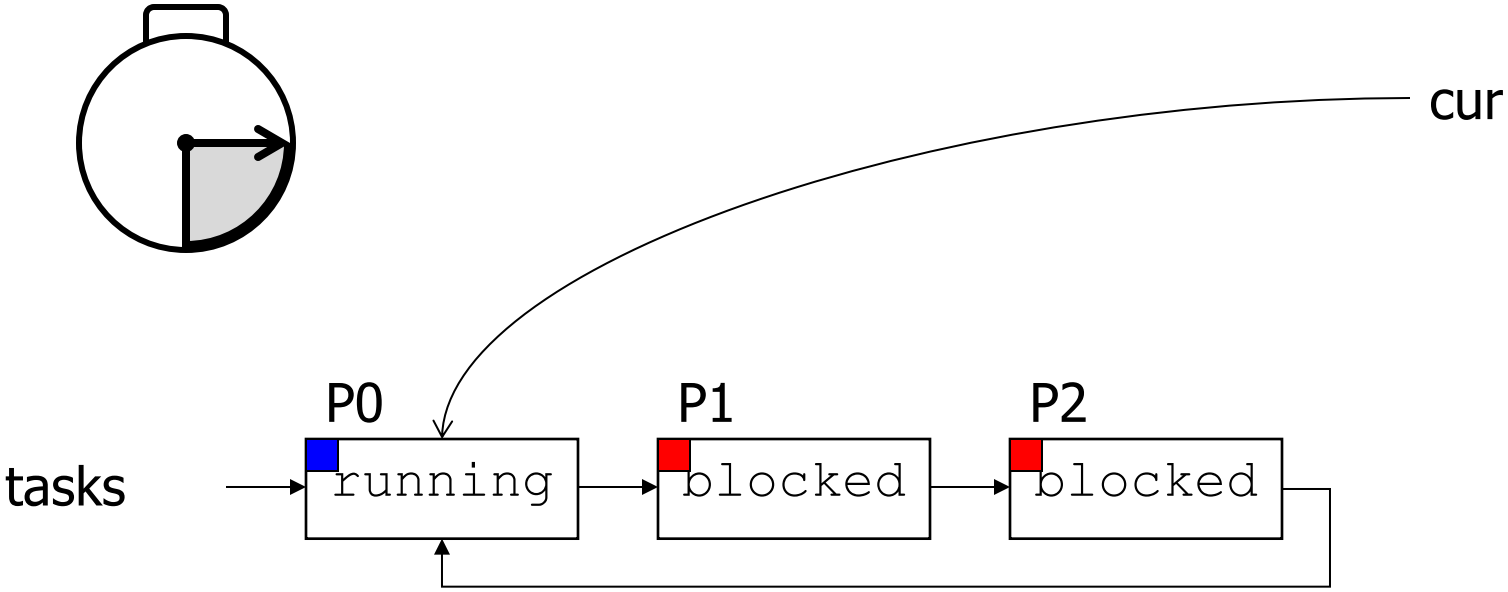
P2 μπαίνει σε αναμονή



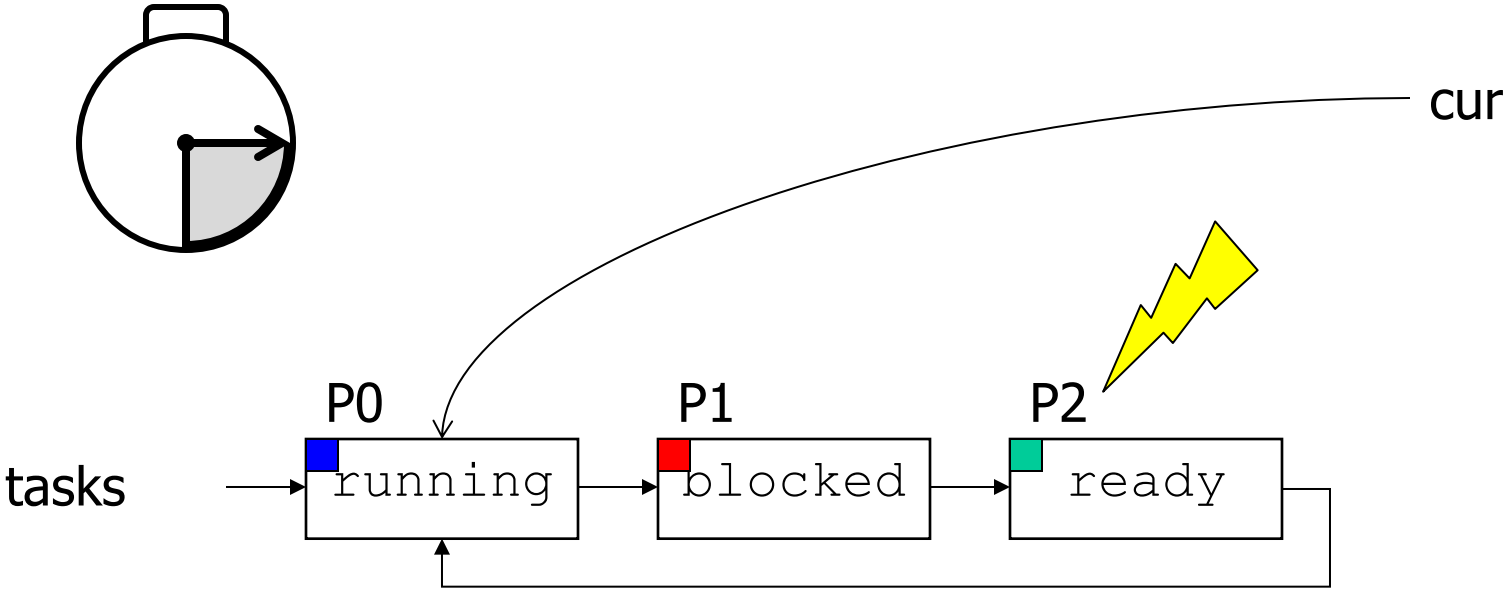
P0 εκτελείται



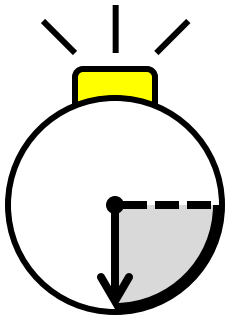
P0 εκτελείται



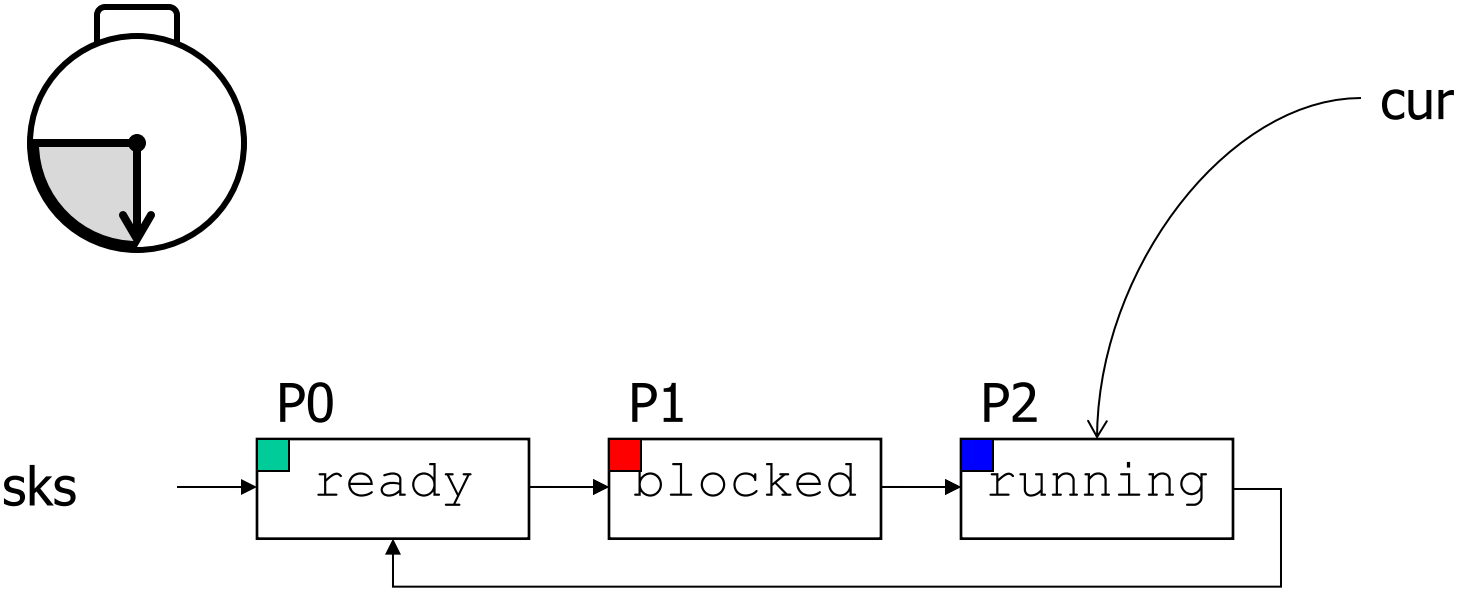
P0 εκτελείται



P2 αφυπνίζεται



εναλλαγή



P2 εκτελείται

Βελτιστοποίηση

- Δεν συμφέρει να κρατάμε τις διεργασίες σε αναμονή στην ίδια λίστα με όσες είναι έτοιμες προς εκτέλεση
  - κάθε φορά που επιχειρείται εναλλαγή, πραγματοποιούνται $O(\mathrm{N})$ έλεγχοι για να βρεθεί η επόμενη έτοιμη διεργασία

- Μπορούμε να εισάγουμε μια **ξεχωριστή** λίστα, αποκλειστικά για τις διεργασίες εν αναμονή
  - η εναλλαγή μπορεί να γίνει σε $O(1)$

- Οι λειτουργίες αλλαγής κατάστασης πρέπει να **μετακινούν** τις διεργασίες ανάμεσα στις λίστες
  - και αυτό γίνεται σε $O(1)$

Διαφάνεια εναλλαγής

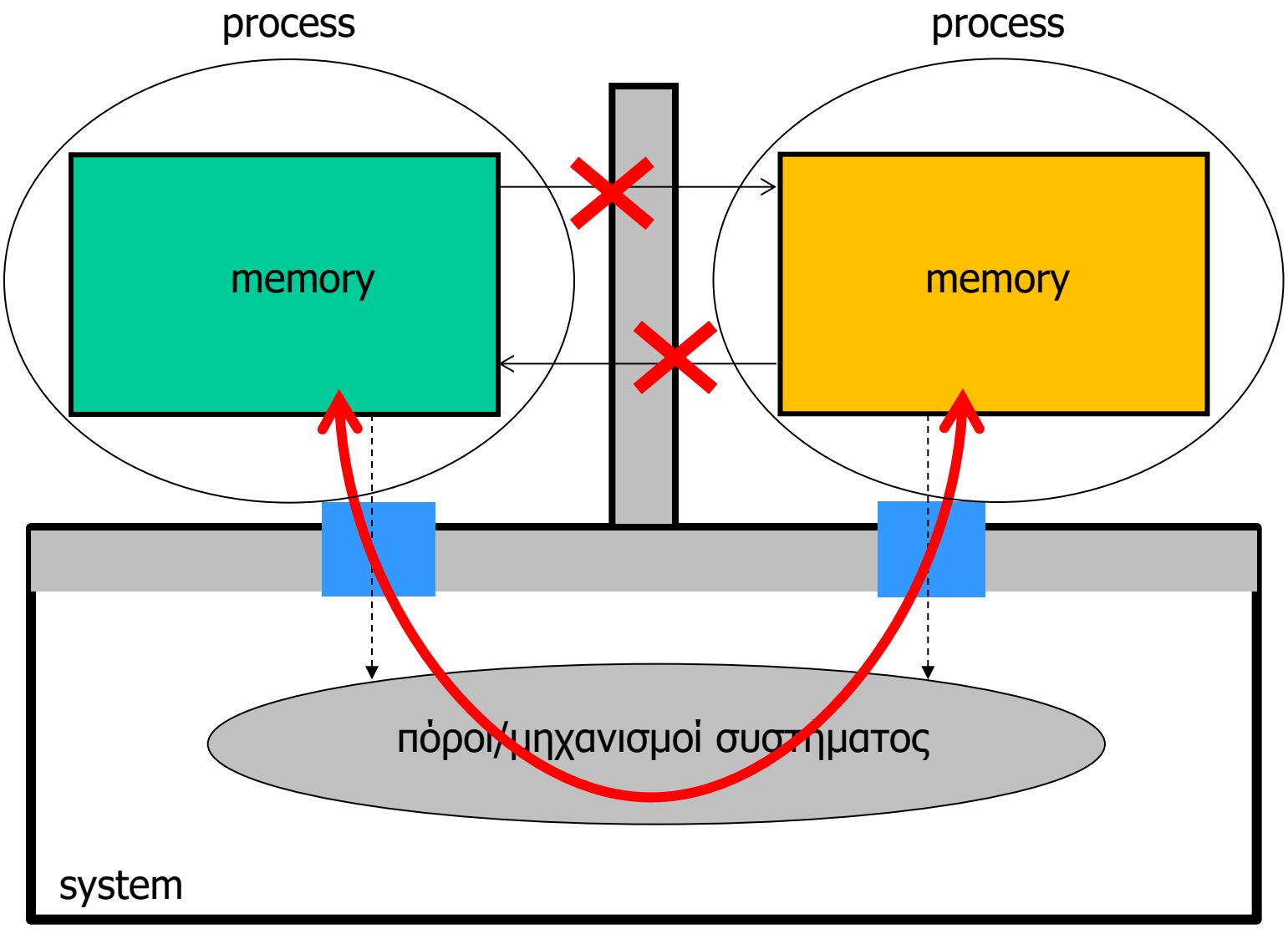
- Ο χρήστης/προγραμματιστής **δεν** ελέγχει το πότε και πόσο συχνά γίνεται εναλλαγή ανάμεσα στις διεργασίες που «τρέχουν» στο σύστημα

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

- Η εναλλαγή είναι **διαφανής** για τον προγραμματιστή
  - ο κώδικας του προγράμματος γράφεται (συνήθως) **χωρίς** να μας απασχολεί το ότι μπορεί να εκτελεστεί ταυτόχρονα με άλλο κώδικα
  - εκτός αν υπάρχει κάποια αλληλεπίδραση μεταξύ των διεργασιών

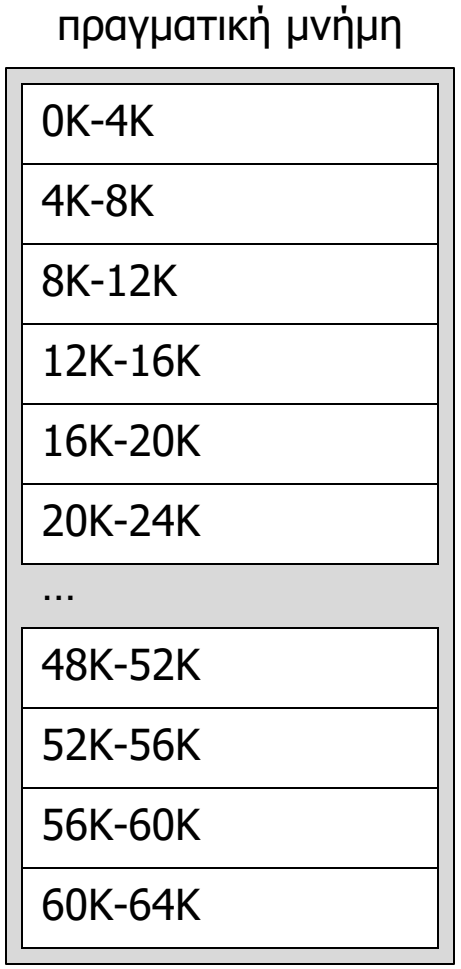
Μνήμη και επικοινωνία διεργασιών

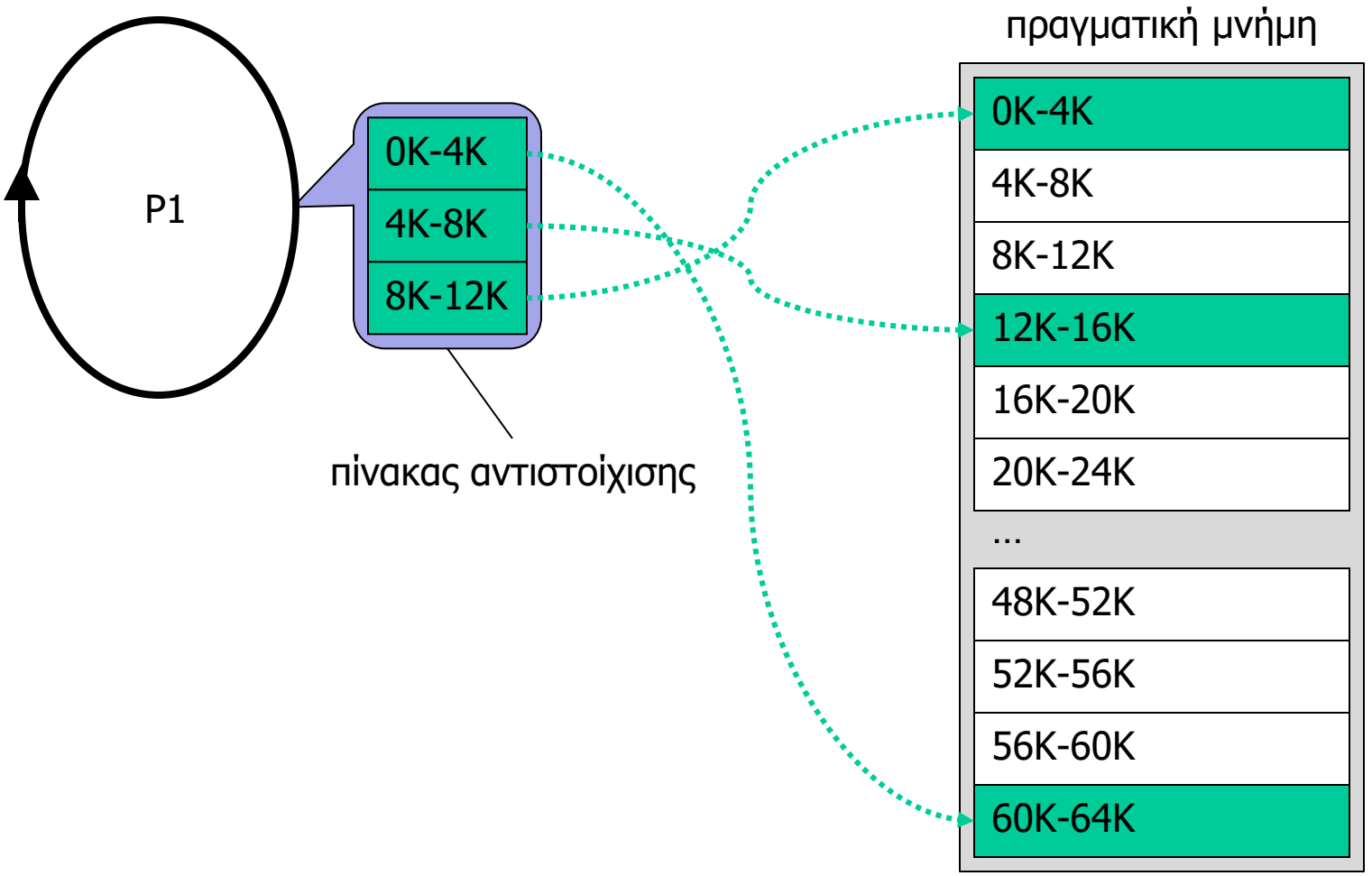
- Κάθε διεργασία έχει **δική** της **ιδιωτική** μνήμη
  - μια διεργασία **δεν** μπορεί να γράψει/διαβάσει από/σε μνήμη άλλων διεργασιών (υπάρχουν εξαιρέσεις)
- Βασική προϋπόθεση για την **απομονωμένη** και **ασφαλή** εκτέλεση προγραμμάτων στον Η/Υ
  - ακόμα και αν μια διεργασία που δυσλειτουργήσει (έχει bugs ή είναι κακόβουλη) **δεν** μπορεί να καταστρέψει τα δεδομένα που κρατάει μια άλλη διεργασία στην μνήμη της
- Τι γίνεται αν κάποιες διεργασίες επιθυμούν να αλληλεπιδράσουν / συνεργαστούν μεταξύ τους;
- Το λειτουργικό σύστημα προσφέρει συγκεκριμένους **μηχανισμούς επικοινωνίας**
  - χρησιμοποιούνται μέσω κλήσεων συστήματος
  - κάποιες από τις οποίες θα δούμε πιο αναλυτικά στην συνέχεια

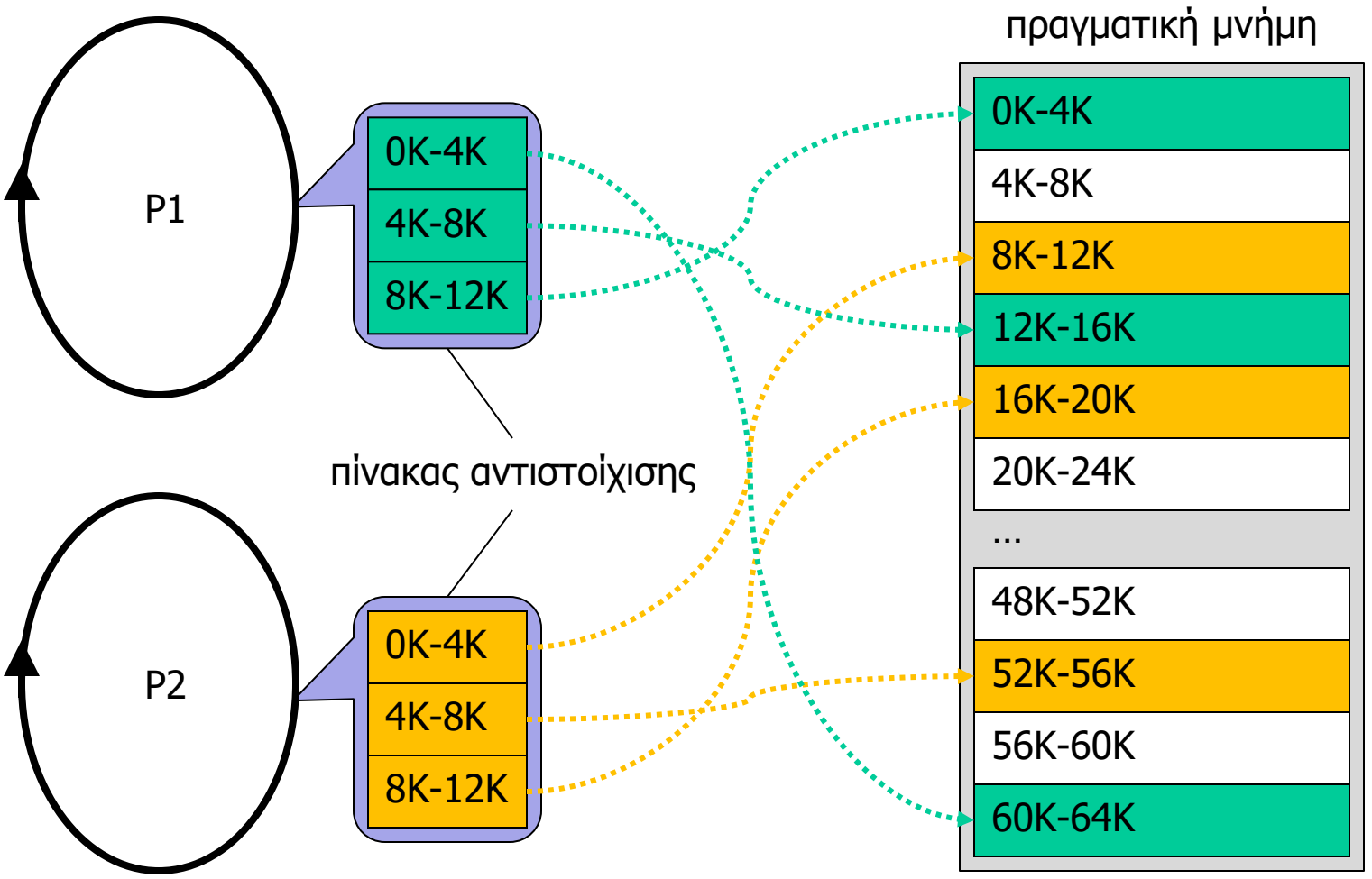

process
process
memory
memory
πόροι/μηχανισμοί συστήματος
system

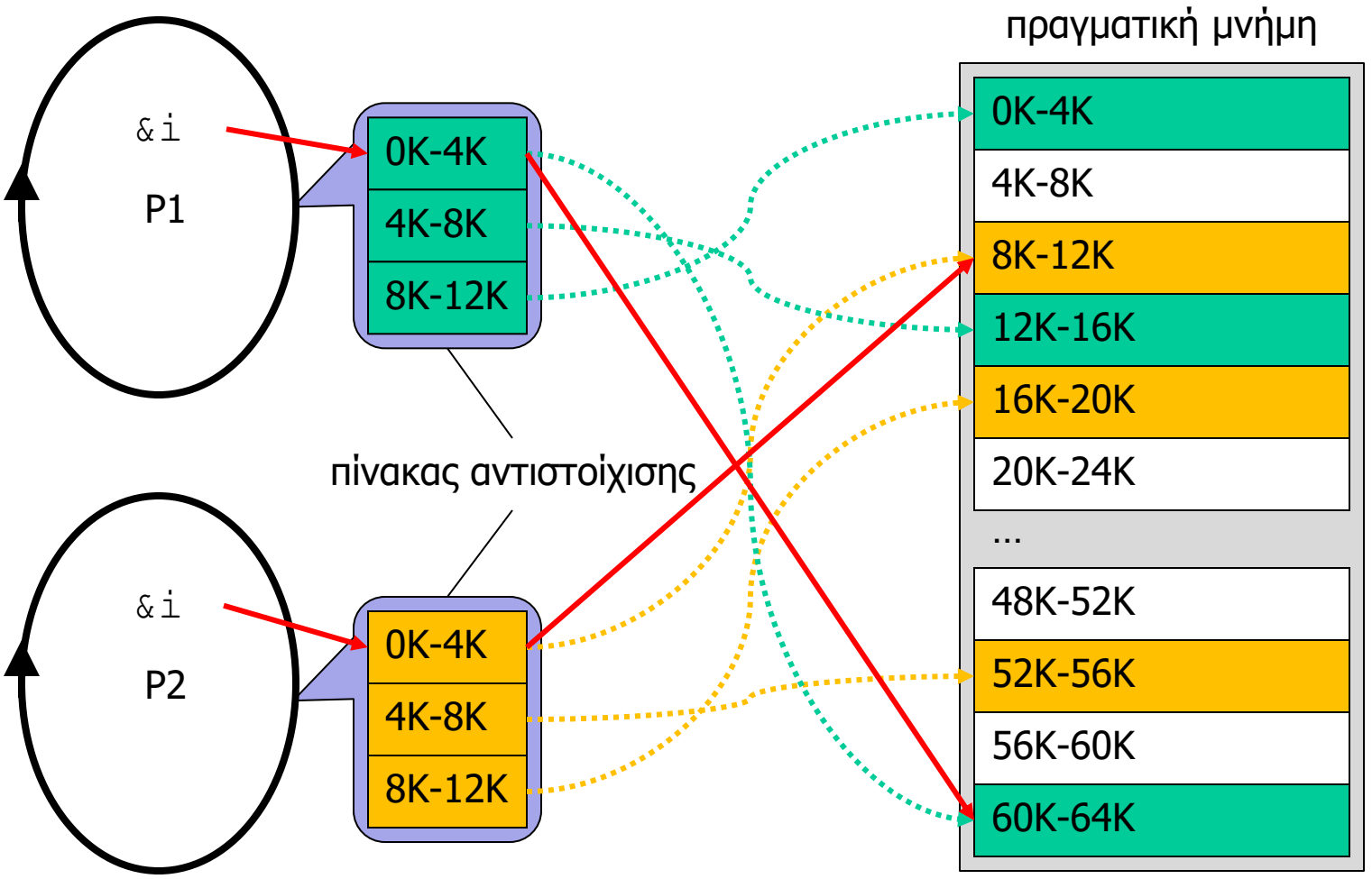
Εικονική μνήμη – virtual memory

- Οι διευθύνσεις μνήμης που χρησιμοποιεί ο κώδικας που εκτελούν οι διεργασίες είναι **εικονικές!**
- Το λειτουργικό κάνει την **αντιστοίχιση** μεταξύ εικονικών και φυσικών/πραγματικών διευθύνσεων
  - γίνεται σε επίπεδο τμημάτων μνήμης – σελίδες / memory pages
- Διαφορετικές διεργασίες μπορεί να χρησιμοποιούν ταυτόχρονα τις **ίδιες** εικονικές διευθύνσεις
  - αντιστοιχίζονται σε **διαφορετικές** φυσικές διευθύνσεις
- Το λειτουργικό **υπολογίζει** και προσπελάζει τις **αντίστοιχες φυσικές διευθύνσεις** στην μνήμη
- Αυτό γίνεται με **διαφανή** τρόπο
  - υποστήριξη από υλικό (memory management unit – MMU)


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


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


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


πραγματική μνήμη
&i
P1
0K-4K
4K-8K
8K-12K
πίνακας αντιστοίχισης
&i
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

Μέγεθος εικονικής μνήμης

- Μια διεργασία μπορεί να έχει την **ψευδαίσθηση** ότι έχει στην διάθεση της **ολόκληρη** την μνήμη του Η/Υ
  - ή ακόμα περισσότερη
- Πως υποστηρίζεται αυτή η ψευδαίσθηση;
- Κάποιες σελίδες μνήμης αποθηκεύονται **εκτός** της φυσικής μνήμης – στον **δίσκο** (swap space)
- Το λειτουργικό ξεφορτώνει τις σελίδες από την μνήμη στον δίσκο (και το αντίστροφο) με **διαφανή** τρόπο
  - page swapping
- Αν πολλές ενεργές διεργασίες προσπελάζουν συνεχώς την μνήμη, μπορεί να προκληθεί **συνεχές** swapping
  - thrashing