ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

# ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΕΠΙΣΤΗΜΗ ΤΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

http://eclass.aueb.gr/courses/INF511/

## Λειτουργικά Συστήματα (ΚΕΦΑΛΑΙΟ 3)


Αλκμήνη Σγουρίτσα

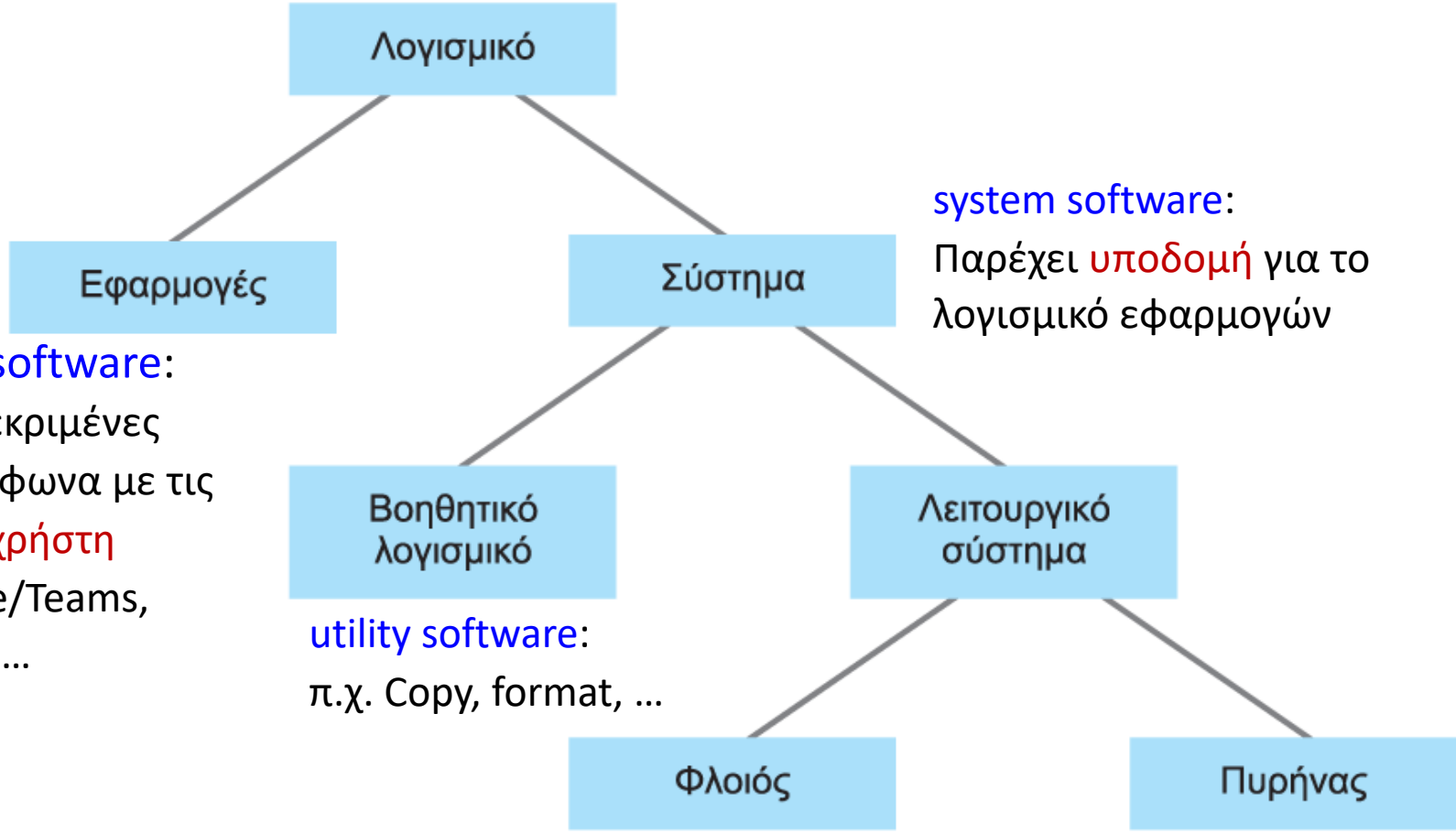
Κοδριγκτώνος 12, 2ος όροφος

E-mail: alkmini@aueb.gr

# ΚΕΦΑΛΑΙΟ 3: Λειτουργικά Συστήματα

- Δομή και βασικά στοιχεία Λειτουργικών Συστημάτων
- Διεργασίες και συντονισμός τους από το ΛΣ
  - Χρονοπρογραμματισμός
  - Διεκπεραίωση
  - Χρονο-μερισμός (time-slicing)
- Εισαγωγή στις πολιτικές χρονοπρογραμματισμού (scheduling)
  - Στατικές πολιτικές: FIFO, Shortest-Job-First, Round Robin, προτεραιότητες
  - Δυναμικές πολιτικές: εισαγωγή στις ουρές αναμονής
- Ανταγωνισμός μεταξύ διεργασιών

# Είδη λογισμικού στον υπολογιστή




application software:
Εκτελεί συγκεκριμένες εργασίες σύμφωνα με τις ανάγκες του χρήστη
Π.χ. MS Office/Teams, video games, …

system software:
Παρέχει υποδομή για το λογισμικό εφαρμογών

utility software:
π.χ. Copy, format, …

# Λειτουργικό Σύστημα

- **Λειτουργικό Σύστημα:** λογισμικό που
  - επιτρέπει σε πολλά προγράμματα (διαδικασίες, processes) να μοιράζονται ταυτόχρονα τους πόρους του υπολογιστή
    - Πόροι: CPU, Μνήμη, Δίαυλος, Χρόνος
  - Παρέχει τα μέσα για αποθήκευση και ανάκτηση πληροφορίας
  - Παρέχει τη διασύνδεση (interface) μέσω της οποίας ο χρήστης ζητάει την εκτέλεση προγραμμάτων
  - Εκτελεί προγράμματα
  - Ασφάλεια από επιθέσεις
- Γιατί είναι απαραίτητο;
  - Ο υπολογιστής σαν “σκέτο” hardware έχει περιορισμένη χρησιμότητα: τρέχει μόνο κώδικα μηχανής

# Παραδείγματα Λειτουργικών Συστημάτων

- Για υπολογιστές:
  - Windows (Microsoft): PC
  - Mac OS (Apple)
  - Solaris (Sun Microsystems, τώρα Oracle)
  - Linux
  - UNIX: για μεγαλύτερα συστήματα υπολογιστών
- Για κινητά / PDAs:
  - iPhone OS (Apple)
  - Windows Phone (Microsoft)
  - Symbian OS (Nokia)
  - Android (Google)
- Για φορητές συσκευές και Internet of Things
  - RIOT (αισθητήρες, για χαμηλή κατανάλωση ισχύος)

    https://www.riot-os.org/

# Δομή λειτουργικού συστήματος

- **Φλοιός** ή **κέλυφος (shell)**: software μέσω του οποίου γίνεται η επικοινωνία του χρήστη με τη μηχανή
  - Γραφικό περιβάλλον διεπαφής με χρήστη (Graphical User Interface, GUI)
  - Διαχειριστής παραθύρων (windows manager)
- **Πυρήνας (kernel)**: περιέχει software που εκτελεί τις βασικές λειτουργίες που απαιτούνται από το σύστημα
  - Διαχειριστής αρχείων (File manager)
  - Οδηγοί συσκευών (Device drivers)
  - Διαχειριστής μνήμης (Memory manager)
  - Χρονοπρογραμματιστής (Scheduler) και Διεκπεραιωτής (Dispatcher):
    - Ποιες διεργασίες θα εκτελεστούν
    - Πότε θα εκτελεστούν: ανάθεση χρόνου σε αυτές

# Διαχειριστής αρχείων (File manager) και Οδηγοί Συσκευών (Device driver)

- Ο Διαχειριστής Αρχείων συντονίζει τη χρήση του μέσου μαζικής αποθήκευσης (π.χ. σκληρός δίσκος) και περιέχει πληροφορίες πρόσβασης για τα αρχεία
  - Κατάλογοι (directories) ή φάκελοι (folders)**:** ομάδες αρχείων που δημιουργούνται από το χρήστη
    - Ιεραρχική οργάνωση
  - Διαδρομή (path)**:** η θέση ενός αρχείου στην ιεραρχία του καταλόγου. Π.χ: `animals/prehistoric/dinosaurs`
  - Λειτουργίες: δημιουργία (create), διαγραφή (delete), προσπέλαση (open, close, read, write)
- Οδηγοί Συσκευών: Μονάδες λογισμικού για επικοινωνία με τους ελεγκτές (controllers) των περιφερειακών συσκευών ή απευθείας με αυτές
  - Π.χ. Οδηγός (driver) για εκτυπωτή, για οθόνη, για σκληρό δίσκο

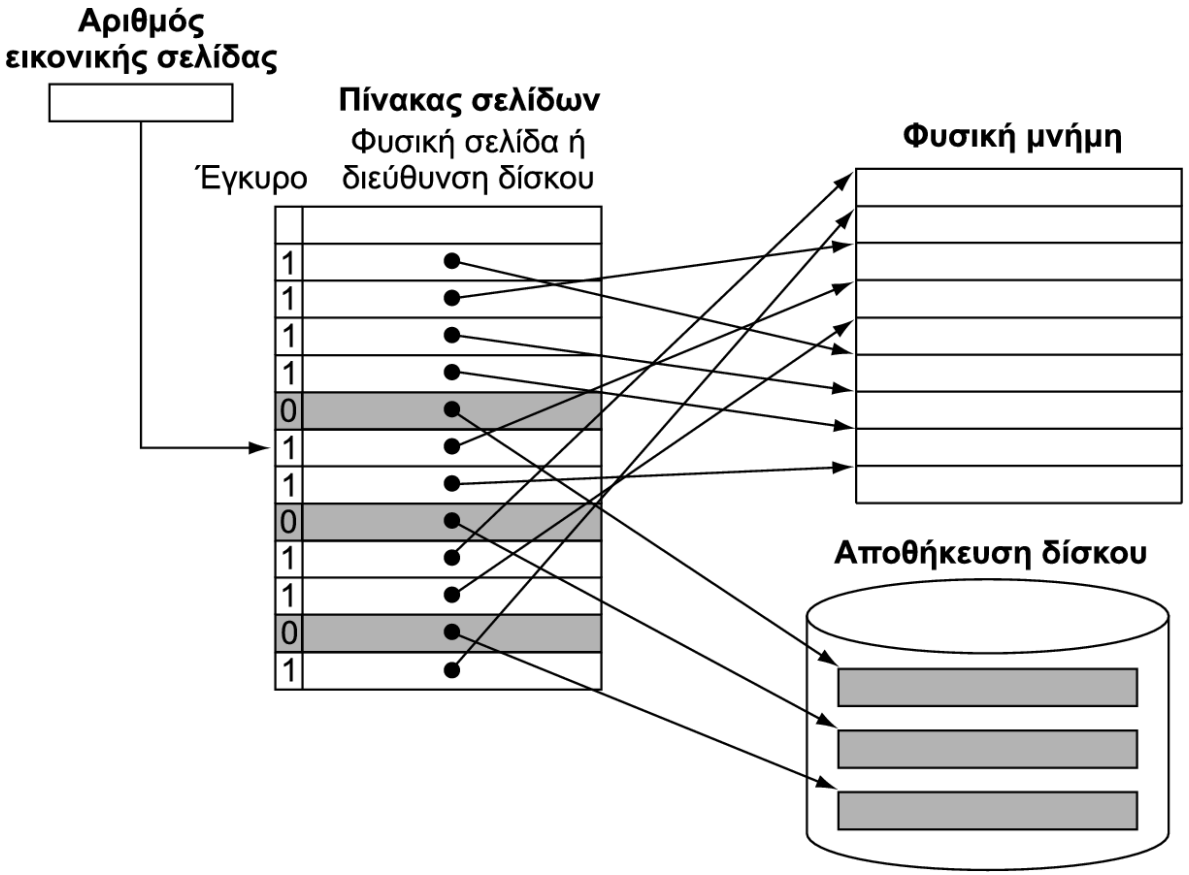
# Διαχειριστής μνήμης

- Ο Διαχειριστής μνήμης συντονίζει τη χρήση της κύριας μνήμης
  - Σημαντικός για την περίπτωση πολλών χρηστών ή πολλών εργασιών (multi-tasking) που εκτελούνται παράλληλα
  - Συνύπαρξη προγραμμάτων και δεδομένων στην κύρια μνήμη
- Εικονική μνήμη (virtual memory): προσομοίωση επιπλέον χώρου μνήμης
  - Τι γίνεται αν ένα πρόγραμμα χρειάζεται 8GB μνήμης αλλά η κύρια μνήμη είναι 4GB;
  - Ο διαχειριστής μνήμης βρίσκει και χρησιμοποιεί τα άλλα 4GB χώρου μνήνης από τον σκληρό δίσκο !
  - Αποθηκεύει εκεί τα 4GB δεδομένων - τα διαιρεί σε μονάδες δεδομένων (σελίδες, pages)
  - Εναλλάσσει τις σελίδες μεταξύ μνήμης και μέσου αποθήκευσης, ώστε οι σελίδες που κάθε φορά πρέπει να εκτελεστούν από το πρόγραμμα να βρίσκονται στην κύρια μνήμη
  - Δημιουργείται η ψευδαίσθηση ότι υπάρχουν 8GB κύριας μνήμης



Link: https://en.wikipedia.org/wiki/Virtual_memory

# Εικονική Μνήμη




**Χώρος εναλλαγής (swap space):** Χώρος στον σκληρό δίσκο που δεσμεύεται για τις θέσεις Εικονικής Μνήμης

Αν valid bit = 1, ο πίνακας δείχνει τη θέση της εικονικής μνήμης στη φυσική θέση μνήμης

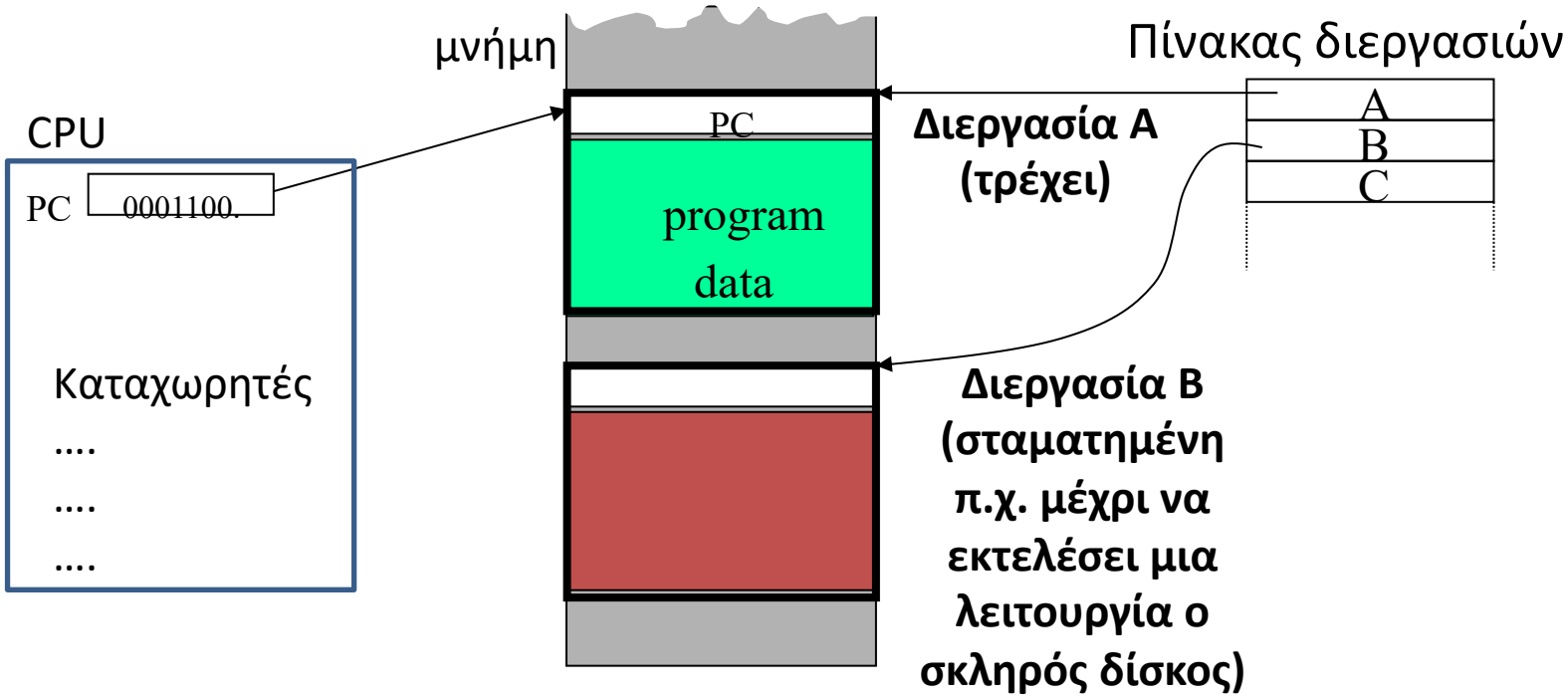
Αν valid bit = 0, ο πίνακας δείχνει τη θέση της εικονικής μνήμης στο δίσκο

# Διεργασίες

- **Πρόγραμμα:** στατικό σύνολο από εντολές
- **Διεργασία (process):** Η δυναμική δραστηριότητα εκτέλεσης ενός προγράμματος

# Κατάσταση Διεργασίας

- **Κατάσταση διεργασίας (process state):** Τρέχουσα κατάσταση της δραστηριότητας: αλλάζει με το χρόνο και αποτελείται από:
  - Εντολή που έπεται αυτής που εκτελείται τώρα στο πρόγραμμα – Program Counter
  - Στιγμιαίο περιεχόμενο των καταχωρητών γενικής χρήσης
  - Στιγμιαίο περιεχόμενο του σχετικού τμήματος της κύριας μνήμης
- Κατάσταση διεργασίας: πλήρης πληροφορία ώστε να μπορούμε να ξανασυνεχίσουμε την διεργασία από εκεί που σταμάτησε



# Διαχείριση Διεργασιών

- Διαχείριση διεργασιών μέσω λειτουργικού συστήματος, ώστε:
  - Κάθε διεργασία να έχει τους **απαιτούμενους πόρους** (θέσεις μνήμης, CPU, πρόσβαση σε περιφερειακές συσκευές) που χρειάζεται
  - Οι διεργασίες να **μην παρεμβάλλονται** και να μην εμποδίζουν η μια την άλλη
  - Οι διεργασίες να **ανταλλάσσουν πληροφορία** (αν χρειάζεται)

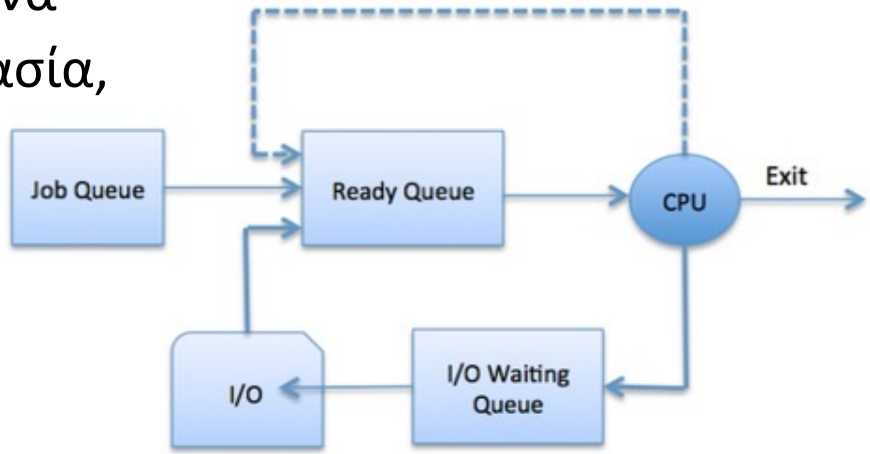
# Χρονοπρογραμματιστής (Scheduler)

α. Διατηρεί τον πίνακα διεργασιών
Ο πίνακας διατηρείται στην μνήμη

Πίνακας διεργασιών

| |
|---|
| A |
| B |
| C |
| |

- Έτοιμη (ready) διεργασία: μπορεί να ξεκινήσει ή να συνεχίσει
- Σε αναμονή (waiting) διεργασία: περιμένει κάποιο εξωτερικό γεγονός να συμβεί, π.χ. μήνυμα από άλλη διεργασία, κτύπημα στο πληκτρολόγιο κλπ.

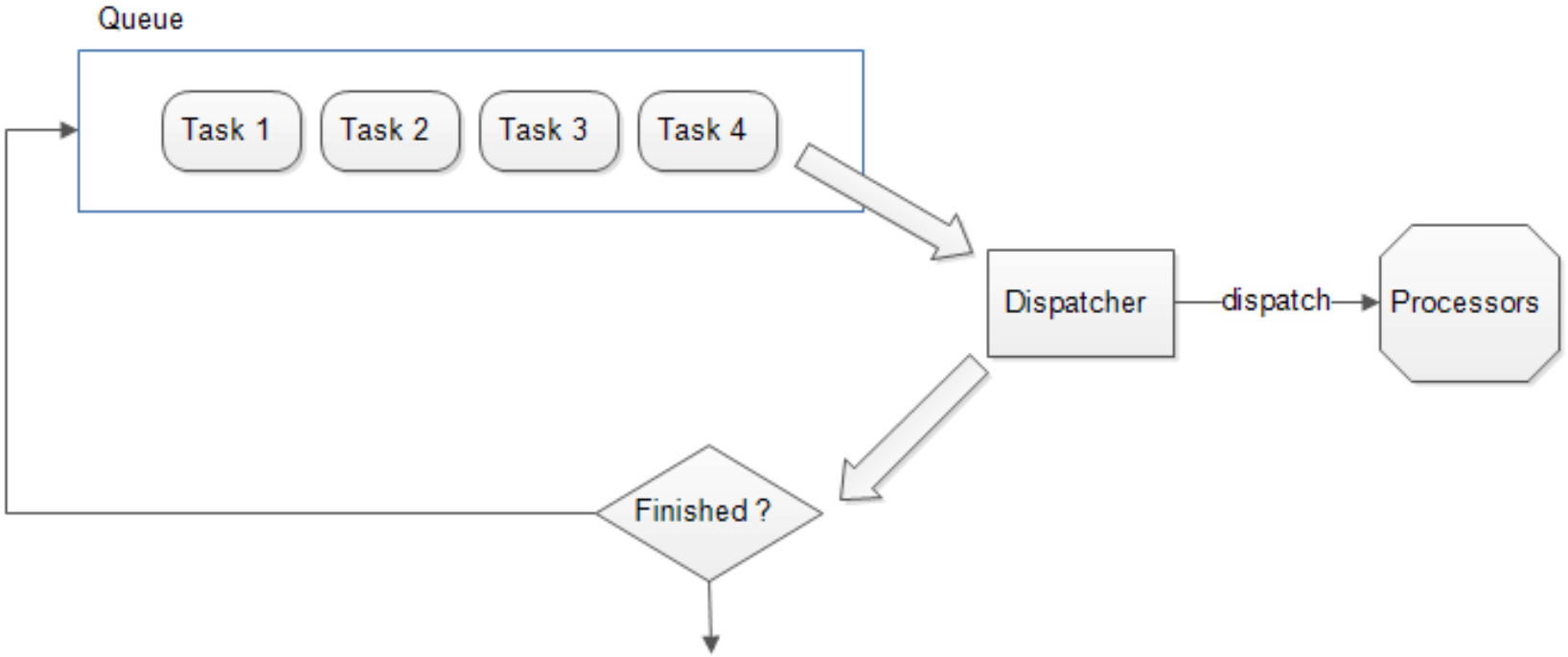



β. Αναθέτει **προτεραιότητες** για την εκτέλεση

γ. Εισάγει νέες διεργασίες στον πίνακα, βγάζει διεργασίες που ολοκληρώθηκαν

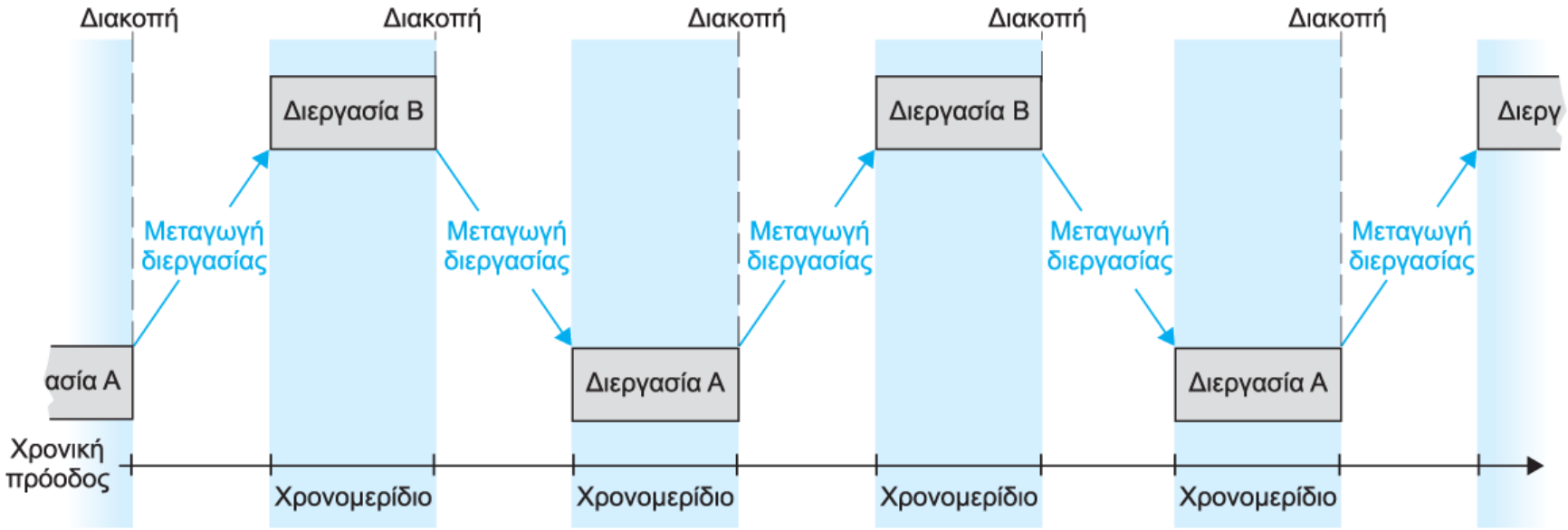
# Διεκπεραιωτής (Dispatcher)

- Χρησιμοποιεί την πληροφορία από τον πίνακα διεργασιών.
- Δίνει ένα χρονο-μερίδιο (time slice ή time slot) σε μία διεργασία που είναι έτοιμη
  - Χρονο-μερίδιο: milli-seconds ή micro-seconds
- Πραγματοποιεί τις προγραμματισμένες διεργασίες



# Χρονομερισμός (time-slicing) μεταξύ διεργασιών (πολυπρογραμματισμός)

Χρονομερισμός: κάθε διεργασία τρέχει για ένα χρονομερίδιο (time slice)




Ο διεκπεραιωτής εκτελεί την μεταγωγή (switching) διεργασιών από την Α στην Β

- Σώζει την κατάσταση της Α (για να την συνεχίσει στο μέλλον)
- Επαναφέρει την κατάσταση της Β
- Ξεκινά την Β

# Διεκπεραιωτής (Dispatcher)

Έστω ότι τώρα ξεκινά να τρέχει η διεργασία Α

1. Ξεκινά ένας **μετρητής** που μετράει αντίστροφα
   - Μετρητής: υπόλοιπος χρόνος για να τελειώσει το χρονομερίδιο της Α
2. Η διεργασία Β (και άλλες ενδεχομένως) γίνεται έτοιμη για εκτέλεση.

Ο **Διεκπεραιωτής επιλέγει μια διεργασία** από τον πίνακα. π.χ. Β

   - Το ποια διεργασία θα επιλέξει, εξαρτάται από την προτεραιότητά της (που την παρέχει ο **χρονοπρογραμματιστής** - scheduler)

3. Ο Διεκπεραιωτής ανακτά την κατάστασή της Β
4. Όταν μετρητής = 0, παράγεται σήμα διακοπής (interrupt)
5. Η CPU σταματά την Α και σώζει την τρέχουσα κατάστασή της
6. Εκτελεί το πρόγραμμα «χειριστής διακοπών» (interrupt handler) (μέρος του διεκπεραιωτή) που βρίσκεται στη μνήμη και μεταφέρει τον έλεγχο στον διεκπεραιωτή
7. Εκτελείται η **μεταγωγή** (switching) στην Β
8. Ξεκινά ένας μετρητή (διάρκεια = time slice) και ξεκινά η νέα διεργασία Β. Πήγαινε στο Βήμα 2.

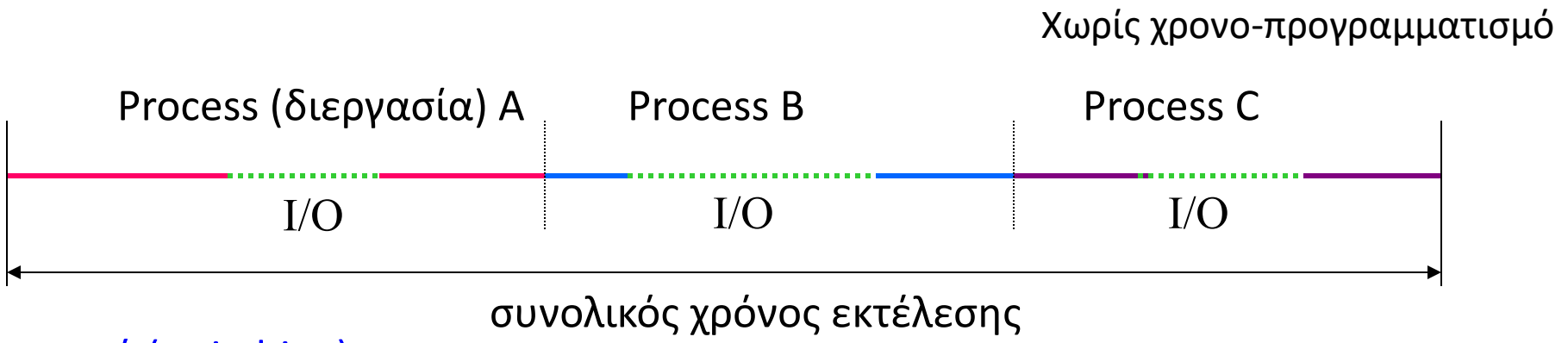
# Πολυπρογραμματισμός

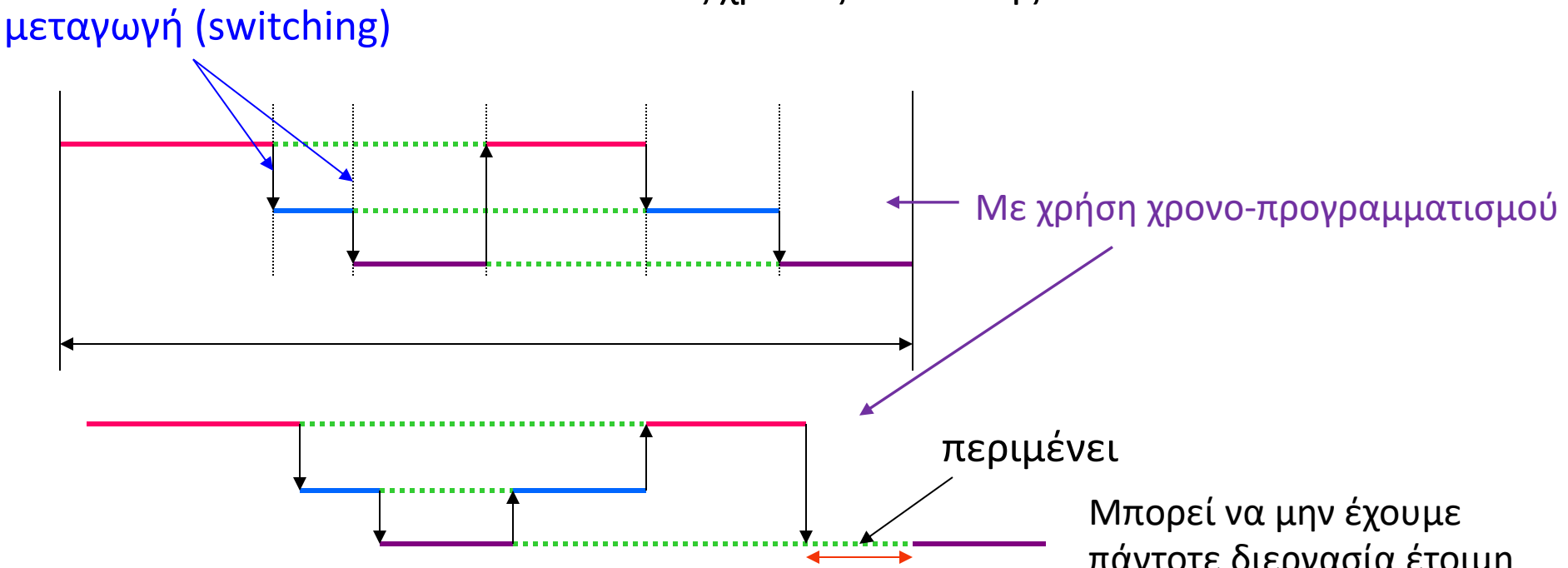
Είναι τελικά καλός ο πολυπρογραμματισμός;

- **Αρνητικά**: Απαιτείται χρόνος για την εναλλαγή των διεργασιών
- **Θετικά**: Ο χρόνος που περιμένει μία διεργασία (π.χ. δεδομένα από περιφερειακές συσκευές) αξιοποιείται για την εκτέλεση άλλης διεργασίας

Συνολικά η αποδοτικότητα του υπολογιστή είναι καλύτερη με τον πολυπρογραμματισμός από ότι με τη σειριακή εκτέλεση των εργασιών.

# Χρονοπρογραμματισμός



# Διαχείριση διεργασιών (1)

- Διεργασία Α (διάρκεια 100 sec, έτοιμη τη στιγμή 0)
- Διεργασία Β (διάρκεια 10 sec, έτοιμη τη στιγμή 0+dt)
- Αν:


- Χρόνος ολοκλήρωσης Α: στα 100 sec
- Χρόνος ολοκλήρωσης Β: στα 110 sec
- Μέσος χρόνος ολοκλήρωσης: (100+110)/2 = 105 sec.

- Αν εκτελώ κάθε μια εναλλάξ για 5 sec


- Χρόνος ολοκλήρωσης Α: στα 110 sec
- Χρόνος ολοκλήρωσης Β: στα 20 sec
- Μέσος χρόνος ολοκλήρωσης: (110+20)/2 = 65 sec.

# Διαχείριση διεργασιών (2)

- Αν υπάρχουν περισσότερες από 1 μονάδες εκτέλεσης (π.χ. πολλές CPU): Παραλληλισμός
  - Πολλοί ταμίες (δηλ. εξυπηρετητές) εξυπηρετούν πολλούς πελάτες, έναν πελάτη την φορά καθένας
- Πόσο γρηγορότερη γίνεται η εκτέλεση τελικα;
- Ιδανικά Ν φορές πιο γρήγορα, αν έχουμε Ν παράλληλες μονάδες εκτέλεσης
  - Αλλά δεν είναι τόσο απλά τα πράγματα ΑΝ υπάρχουν αλληλεξαρτήσεις μεταξύ των διεργασιών
- Π.χ. συγκρίνετε το χρόνο που θα κάνατε ένα project αν
  - Α. Δουλεύατε μόνος/η σας
  - Β. Με άλλο ένα άτομο
  - Γ. Με μια ομάδα 10 ατόμων

# Διαχείριση διεργασιών (3)

- Στόχος: καλή διαχείριση των πόρων του συστήματος (μνήμη, CPU, δίσκος)
- Ελαχιστοποίηση χρόνου απόκρισης / χρόνου ολοκλήρωσης
  - χρόνος από τότε που υποβλήθηκε μια εργασία μέχρι την ολοκλήρωσή της
  - π.χ. ο χρόνος από το κλικ του ποντικιού μέχρι το κλείσιμο του παραθύρου
- Μεγιστοποίηση ρυθμού διεκπεραίωσης (ρυθμοαπόδοση, throughput)
  - Ρυθμός με τον οποίο διεκπεραιώνονται εργασίες (αριθμός εργασιών ανά μονάδα χρόνου)
- 2 κατηγορίες πολιτικών διαχείρισης (χρονοπρογραμματισμού) διεργασιών
  - Στατικές: υποθέτουν ένα **συγκεκριμένο** σύνολο διεργασιών, βρίσκουν με τι σειρά θα εκτελεστούν αυτές
  - Δυναμικές: υποθέτουν μια διαρκή ροή (stream) διεργασιών

# Στατικός Χρονοπρογραμματισμός: Πολιτική FCFS ή FIFO

- FCFS = First-come First-serve ή FIFO = First-in First-out
- Εκτέλεση διεργασιών με την σειρά άφιξής τους
- Πλεονέκτημα: απλή μέθοδος
- Μειονέκτημα: Η απόδοση εξαρτάται από την σειρά άφιξης
  - Όχι καλή (όχι δίκαιη) για εργασίες που φτάνουν αργότερα ή αν μια μεγάλη εργασία φτάνει νωρίς

# Στατικός Χρονοπρογραμματισμός: Πολιτική FCFS ή FIFO

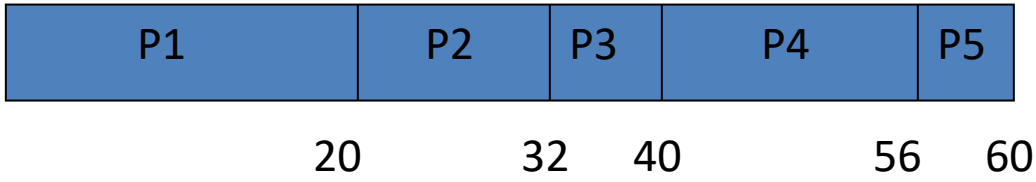
- FCFS = First-come First-serve ή FIFO = First-in First-out
- Εκτέλεση διεργασιών με την σειρά άφιξής τους
- Πλεονέκτημα: απλή μέθοδος
- Μειονέκτημα: Η απόδοση εξαρτάται από την σειρά άφιξης
  - Όχι καλή (όχι δίκαιη) για εργασίες που φτάνουν αργότερα ή αν μια μεγάλη εργασία φτάνει νωρίς
- Διεργασία Α διάρκειας 100sec, Β διάρκειας 2 sec, Γ διάρκειας 3 sec
  - Αν φτάνουν σχεδόν ταυτόχρονα (η Α λίγο πριν την Β, και η Β λίγο πριν την Γ)


- Αν φτάνουν με σειρά Β,Γ,Α:


- Πρόβλημα  αν μια υπολογιστικά μεγάλη διεργασία φτάσει πρώτη (μπλοκάρει όλες τις άλλες που φτάνουν μετά από αυτήν)

# Παράδειγμα FIFO

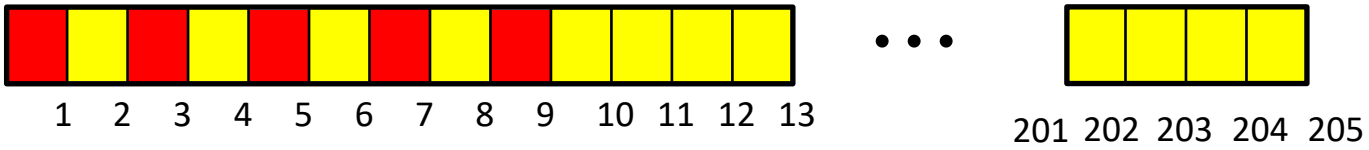
- Να υπολογιστεί ο μέσος χρόνος αναμονής για διεργασίες που έρχονται με σειρά άφιξης P1, P2, P3, P4, P5 και έχουν την παρακάτω διάρκεια
- P1: 20, P2: 12, P3: 8, P4: 16, P5: 4




- Χρόνος αναμονής μιας διεργασίας: ο χρόνος μέχρι αυτή να εκκινήσει
- Χρόνοι αναμονής:
  - P1: 0
  - P2: 20
  - P3: 32
  - P4: 40
  - P5: 56
- Μέσος χρόνος αναμονής:  29.6

# Χρονοδρομολόγηση με Εναλλαγές (Round Robin Scheduling , RR)

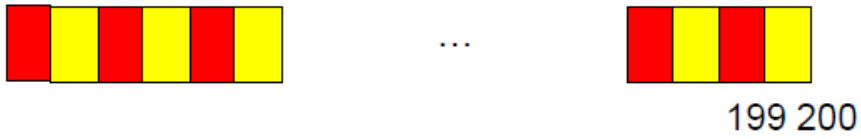
- Το προηγούμενο πρόβλημα δεν υφίσταται αν οι υπολογιστικά μεγάλες διεργασίες δεν μονοπωλούν τον επεξεργαστή
- Εκτέλεση μέρους κάθε διεργασίας σε γύρους
- Σε κάθε γύρο, η CPU εκτελεί διαδοχικά και κυκλικά όλες τις εργασίες, την καθεμία για ένα συγκεκριμένο χρονικό διάστημα (περίοδο) T
  - Δίκαιος τρόπος
- Ικανοποιητικός μέσος χρόνος ολοκλήρωσης για εργασίες αρκετά διαφορετικού μεγέθους μεταξύ τους π.χ. A=5, B=200




- Ποιος ο μέσος χρόνος ολοκλήρωσης;
  - (9+205)/2=107
- Μέσος χρόνος ολοκλήρωσης της FIFO;
  - Αν A πριν τη B, (5+205)/2=105. Αν B πριν την A, (200+205)/2=202.5

# Χρονοδρομολόγηση με Εναλλαγές (Round Robin Scheduling , RR)

- Το προηγούμενο πρόβλημα δεν υφίσταται αν οι υπολογιστικά μεγάλες διεργασίες δεν μονοπωλούν τον επεξεργαστή
- Εκτέλεση μέρους κάθε διεργασίας σε γύρους
- Σε κάθε γύρο, η CPU εκτελεί διαδοχικά και κυκλικά όλες τις εργασίες, την καθεμία για ένα συγκεκριμένο χρονικό διάστημα (περίοδο) Τ
  - Δίκαιος τρόπος
- Για διεργασίες παρόμοιου μεγέθους π.χ. Α=100, Β=100




- Ποιος ο μέσος χρόνος ολοκλήρωσης;
  - (199+200)/2=199.5
- Μέσος χρόνος ολοκλήρωσης της FIFO;
  - (100+200)/2=150
- Συμπέρασμα;

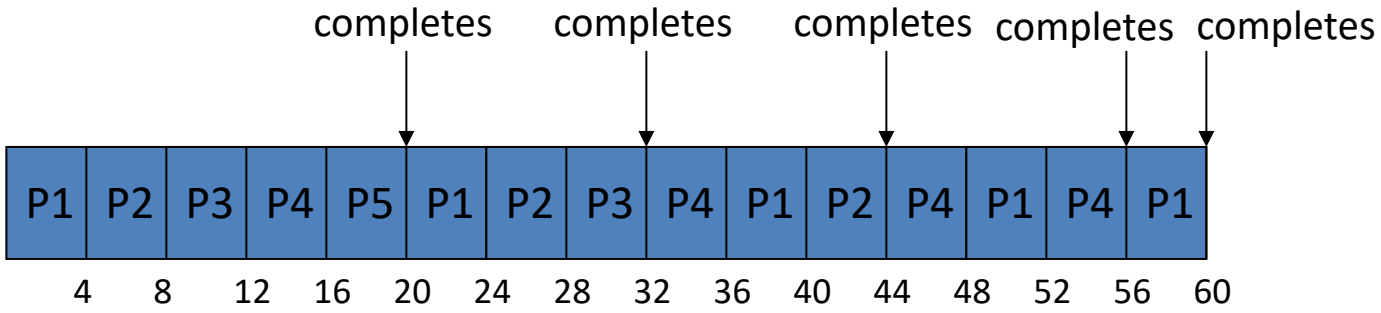
# RR: Σχεδιαστικά ζητήματα

- Περίοδος Τ εκτέλεσης κάθε διεργασίας:
  - Μεγάλη περίοδος: οι διεργασίες ίσως τερματίσουν πριν τελειώσει η περίοδος που εκτελούνται
    - Καταναλώνεται κάποιος χρόνος άσκοπα
  - Μικρή περίοδος: καλύτερος μέσος χρόνος αναμονής, αλλά υπάρχει συνεχές context-switching

- Επιθυμούμε περίοδο αρκετά μεγαλύτερη από το χρόνο μεταγωγής (έστω δ)
  - Σύνηθες κόστος μεταγωγής: δ < 1-5% της περιόδου Τ
  - περίοδος Τ~100msec ή 200msec

# Παράδειγμα Round Robin (δ=0)

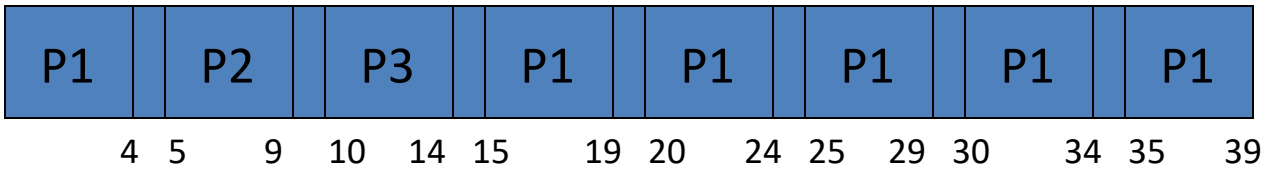
- Διάρκεια της διεργασίας στην CPU και σειρά:
  - P1: 20, P2: 12, P3: 8, P4: 16, P5: 4 (έστω ότι εμφανίζονται όλες τη στιγμή 0)
  - Περίοδος RR: T = 4




- Συνολικός χρόνος αναμονής για κάθε διεργασία :
  - P1: 0 + 16 + 12 + 8 + 4 = 40
  - P2: 4 + 16 + 12 = 32
  - P3: 8 + 16 = 24
  - P4: 12 + 16 + 8 + 4= 40
  - P5: 16
- Μέσος συνολικός χρόνος αναμονής: 30.4.

*Σημ: Για RR, ο συνολικός χρόνος αναμονής για μια διεργασία είναι ο χρόνος μέχρι να ξεκινήσει συν το συνολικό χρόνο που μεσολαβεί μεταξύ κάθε 2 διαδοχικών εκτελέσεων της σε RR*

# Παράδειγμα με Context-switching (δ ≠ 0)



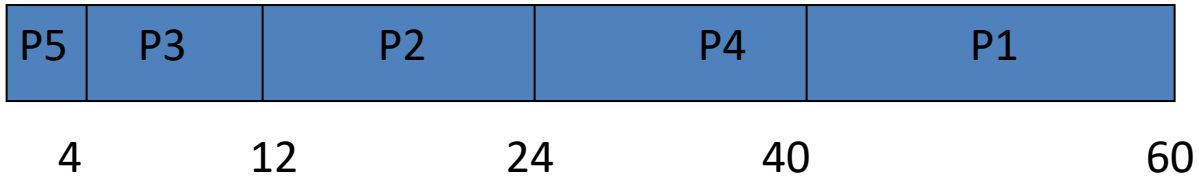

- Μέσος χρόνος αναμονής για RR (round robin) με:
  - Διεργασίες: P1: 24, P2: 4, P3: 4 με αυτή τη σειρά άφιξης
  - Περίοδος T = 4, χρόνος για context-switching δ = 1
  - Συνολικός χρόνος αναμονής για κάθε διεργασία:
  - P1: 0 + 11 + 1 + 1 + 1 + 1 = 15
  - P2: 5
  - P3:10
- Μέσος συνολικός χρόνος αναμονής: 10

# Χρονοδρομολόγηση με Προτεραιότητες

- **Ιεράρχηση** διεργασιών: ανάθεση προτεραιότητας σε κάθε διεργασία
- Κάθε φορά επιλέγεται η έτοιμη (Ready) διεργασία με μεγαλύτερη προτεραιότητα
  - Διεργασίες ίδιας προτεραιότητας εκτελούνται με RR
- Οι προτεραιότητες μπορεί να είναι ανάλογα με:
  - Τον χρόνο που μεσολάβησε από την τελευταία εκτέλεση
  - Την αξιοποίηση συσκευών I/O: π.χ. υψηλή προτεραιότητα σε διεργασίες που εκτελούν πολύ I/O
- Οι προτεραιότητες μπορεί να είναι:
  - Εσωτερικές: Π.χ. Σύμφωνα με παράγοντες σχετικούς με το Λ/Σ (απαιτήσεις σε μνήμη)
  - Εξωτερικές, π.χ. σπουδαιότητα για τον χρήστη
  - Στατικές: σταθερές για όλη τη διάρκεια της διεργασίας
  - Δυναμικές: μπορεί να αλλάζουν κατά τη διάρκεια εκτέλεσης της διεργασίας
    - Π.χ. ως συνάρτηση της χρήσης της CPU ή του χρόνου αναμονής
- FIFO: η προτεραιότητα, και άρα η σειρά εκτέλεσης, καθορίζεται από τη σειρά με την οποία είναι διαθέσιμες
- Shortest Job First, SJF (επόμενη διαφάνεια): υψηλή προτεραιότητα σε μικρής διάρκειας διεργασίες

# SJF: Shortest Job First (Πάραδειγμα)

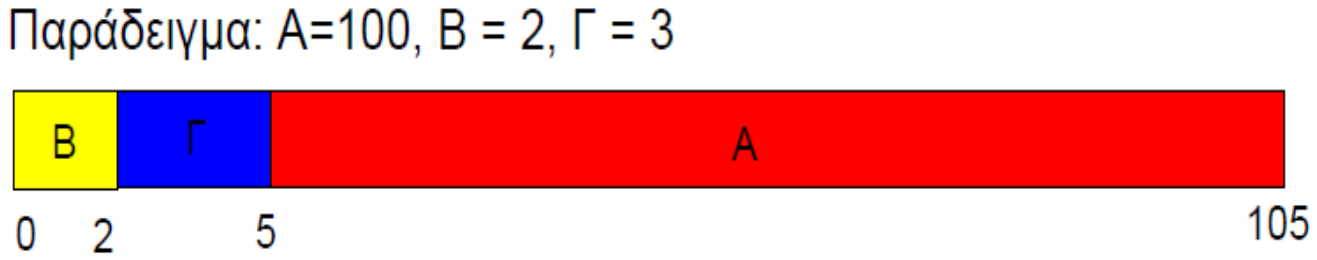
- Διεργασίες και διάρκειες: P1: 20, P2: 12, P3: 8, P4: 16, P5: 4




- Χρόνοι αναμονής για κάθε διεργασία:
  - P1: 40
  - P2: 12
  - P3: 4
  - P4: 24
  - P5: 0
- Μέσος χρόνος αναμονής:  16

# Πρώτα η Συντομότερη Διεργασία (Shortest Job First, SJF)

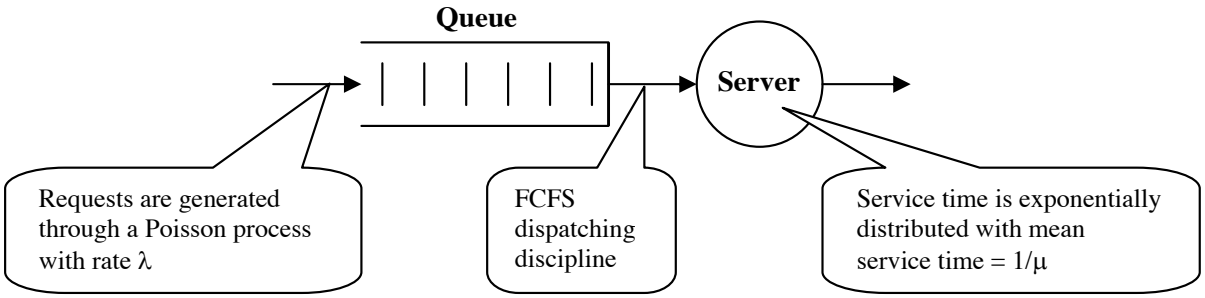
- Πολιτική SJF: Βάλε στη σειρά τις διεργασίες σε αύξουσα σειρά διάρκειας (αρχίζοντας από την πιο σύντομη)
- Εκτέλεσε τις εργασίες με αυτή τη σειρά




- Θεώρημα: Η πολιτική SJF είναι βέλτιστη ως προς τον συνολικό χρόνο αναμονής
  - Άρα και ως προς τον μέσο χρόνο αναμονής
  - Δηλ. η SJF είναι η πολιτική που ελαχιστοποιεί το συνολικό (άρα και τον μέσο) χρόνο αναμονής) – ανάμεσα σε όλες τις δυνατές πολιτικές
  - Γιατί;
- Απόδειξη (hint): Αλλάζοντας τη σειρά εκτέλεσης, πάντα αυξάνεται ο συνολικός χρόνος αναμονής
  - Βάζοντας τη σύντομη διεργασία μετά την μεγάλη ευνοούμε τη μεγάλη διεργασία λιγότερο από ότι επιβαρύνουμε την σύντομη

# Δυναμικές Πολιτικές Χρονοπρογραμματισμού: Θεωρία Ουρών Αναμονής

- Ουρά: δυναμικό σύστημα με διαρκείς αφίξεις και εξυπηρετήσεις διεργασιών



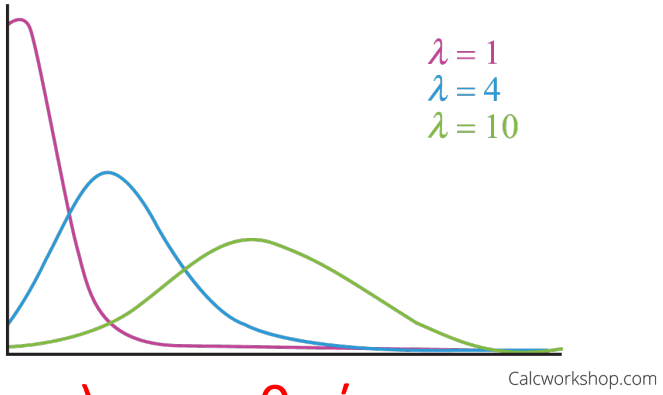

- Δείτε link από το μάθημα στο MIT:

Link: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-263j-data-communication-networks-fall-2002/lecture-notes/Lectures5_6.pdf

# Ουρές Αναμονής (Queues)

- Θεωρία ουρών αναμονής (queueing systems)
- Αφίξεις εργασιών: με ρυθμό λ εργασίες/sec
  - Ο αριθμός των εργασιών που φτάνουν προς εκτέλεση σε κάθε time slot είναι τυχαίος (τυχαία μεταβλητή)
  - κατανομή Poisson με παράμετρο λ,
    - $f(k,\lambda) = Pr(X=k) = \lambda^k e^{-\lambda} / k!$
  - μέσος αριθμός άφιξης εργασιών ανά sec,

    μέσος ρυθμός άφιξης λ εργασίες/sec




- Αναχωρήσεις εργασιών (δηλ. εξυπηρετήσεις) με ρυθμό μ εργασίες/sec
  - Ο αριθμός των εργασιών που εξυπηρετούνται σε κάθε slot είναι τυχαίος (τυχαία μεταβλητή). Οι εργασίες έχουν διαφορετικό μέγεθος και άρα και χρόνο εκτέλεσης.
  - κατανομή Poisson με παράμετρο μ (μέσος ρυθμός εξυπηρέτησης μ -> μ εργασίες ανά sec): Μέσος χρόνος εξυπηρέτησης: 1/μ



Link: http://www.cs.bu.edu/fac/matta/Teaching/cs556/archive/2003-07-MM1.pdf

# Ουρές Αναμονής (Queues)

- Μέτρα επίδοσης
  - Μέσος χρόνος αναμονής στο σύστημα = μέσος χρόνος αναμονής στην ουρά Q + μέσος χρόνος εξυπηρέτησης
    - $T = Q + (1/\mu)$
- Βασικός τύπος: Νόμος του Little
  - N’ (Μέσος αριθμός εργασιών στην ουρά) = λ * Q (Μέσος χρόνος αναμονής στην ουρά)
    - $N' = \lambda Q$
  - N (Μέσος αριθμός εργασιών στο σύστημα) = λ * T (Μέσος χρόνος αναμονής στο σύστημα)
    - $N = \lambda T$



Link: http://www.cs.bu.edu/fac/matta/Teaching/cs556/archive/2003-07-MM1.pdf

# Δυναμικές πολιτικές: Προεκτοπισιμότητα και Μη

- Έστω Διεργασίες Α,Β,Γ,... με διαφορετική προτεραιότητα
- Προεκτοπίσιμη (pre-emptive) πολιτική χρονοδρομολόγησης με προτεραιότητες: Αν εκτελείται μια διεργασία x και καταφτάσει μια διεργασία y, η x διακόπτεται μόνο αν η y είναι μεγαλύτερης προτεραιότητας από την x
- Εφόσον δεν φτάσει άλλη διεργασία μεγαλύτερης προτεραιότητας από την y:
  - Η εκτέλεση της y συνεχίζεται μέχρι την ολοκλήρωσή της
  - Όταν ολοκληρωθεί η y, συνεχίζεται η εκτέλεση της x
- Δηλ. σε κάθε χρονική στιγμή εκτελείται η μεγαλύτερης προτεραιότητας εργασία από αυτές που είναι έτοιμες προς εκτέλεση

# Προεκτοπισιμότητα και Μη (2)

- Μη προεκτοπίσιμη (non preemptive) πολιτική χρονοδρομολόγησης με προτεραιότητες: Αν εκτελείται μια διεργασία x και φτάσει μια διεργασία y (οποιασδήποτε προτεραιότητας) η x ΔΕΝ διακόπτεται σε καμιά περίπτωση
  - Η y θα περιμένει
  - Η y θα εκτελεστεί μόλις ολοκληρωθεί η x (εφόσον μέχρι τότε δεν έχει φτάσει άλλη διεργασία μεγαλύτερης προτεραιότητας από την y)
- Ποιες πολιτικές χρονοδρομολόγησης μπορεί να είναι προεκτοπίσιμες;
- FIFO
  - Μη προεκτοπίσιμη (non-preemptive)
  - Η προτεραιότητα είναι ανάλογα με την σειρά άφιξης - δεν αλλάζει
- Δυναμική SJF (Shortest Job First) και άλλες πολιτικές με προτεραιότητες:
  - Pre-emptive ή μη, ανάλογα με το τι γίνεται όταν καταφτάνει μια μεγαλύτερης προτεραιότητας διεργασία ενώ εξυπηρετείται μια άλλη

# Ανταγωνισμός μεταξύ διεργασιών (1)

- Ανταγωνισμός διεργασιών για πόρους (resources)
  - Πόροι: Μνήμη, CPU, χώρος στον πίνακα διεργασιών, χρονο-μερίδια, περιφερειακές συσκευές
- Σηματοφορέας (semaphore): σημαία (flag) ελέγχου που λέει εάν κάποιος πόρος είναι σε χρήση
  - Flag: Clear (ελεύθερος), set (σε χρήση)
- Πρόβλημα σύγκρουσης διεργασιών:
  - Έστω η διεργασία Α ζητά πρόσβαση στον εκτυπωτή με flag = clear
  - Μπορεί η Α να διακοπεί (interrupt) αφού έχει ανιχνευτεί clear αλλά πριν γίνει set
  - Στο επόμενο χρονομερίδιο, μια άλλη διεργασία Β ξεκινά, ζητά και αυτή τον εκτυπωτή. Της παρέχεται πρόσβαση γιατί το flag είναι clear
  - → σύγκρουση, όταν η Α επανέλθει στο επόμενο χρονομερίδιο
  - Λύση: Ο έλεγχος και η ενεργοποίηση flag γίνονται χωρίς διακοπή
    - (test & set flag instruction) σε μια εντολή
    - Απενεργοποίηση διακοπών πριν και ενεργοποίηση τους μετά

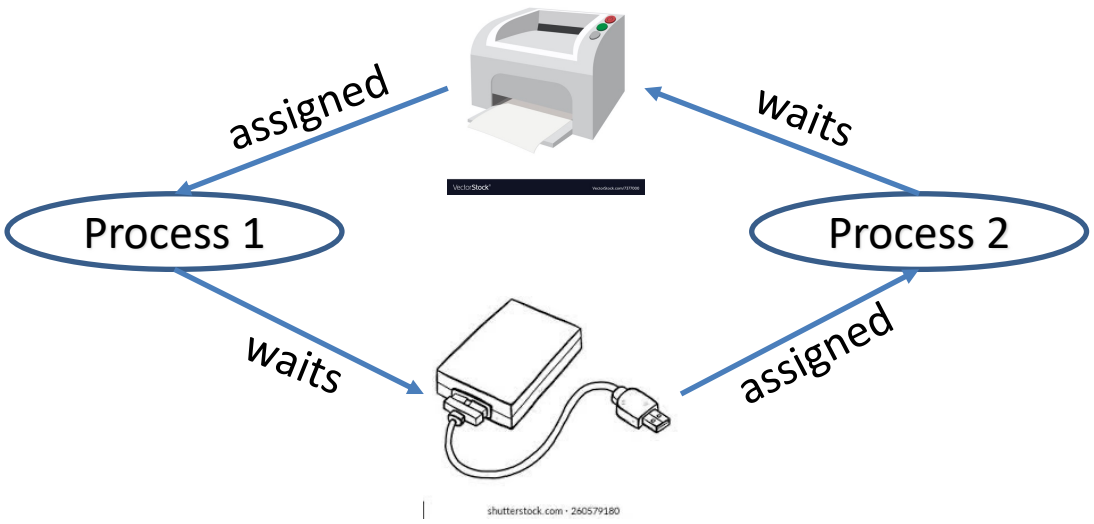
# Ανταγωνισμός μεταξύ διεργασιών (2)

- **Κρίσιμη περιοχή (critical region)**: ακολουθία εντολών που μπορεί να εκτελούνται από μόνο μία διεργασία τη φορά (συνήθως προστατεύεται από σηματοφορέα)

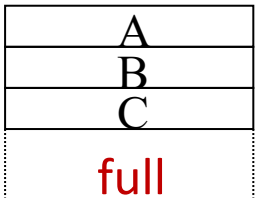
- **Αμοιβαίος αποκλεισμός (mutual exclusion)**: εκτέλεση κρίσιμης περιοχής από μία μόνο διεργασία τη φορά
  - Για να μπει στην κρίσιμη περιοχή, μια διεργασία πρέπει να βρει το σηματοφορέα clear. Θα τον κάνει set πριν μπει
  - Όταν βγει από την κρίσιμη περιοχή, πρέπει να τον ξαναθέσει σε clear
  - Αν μια διεργασία δει το σηματοφορέα set, περιμένει μέχρι αυτός να γίνει clear

# Ανταγωνισμός μεταξύ διεργασιών (3)

- **Αδιέξοδο** (deadlock): Η εκτέλεση 2 ή περισσοτέρων διεργασιών εμποδίζεται από το να συνεχιστεί (γιατί η μία περιμένει την άλλη)




Πίνακας διεργασιών




Κάθε διεργασία πρέπει να δημιουργήσει μία νέα διεργασία για να μπορέσει να ολοκληρωθεί.

- Μέθοδοι αποτροπής, ανίχνευσης και αντιμετώπισης του αδιεξόδου
  - Συνήθης λύση για άρση αδιεξόδου είναι το σκότωμα (killing) της διεργασίας

# Για όποιον/α ενδιαφέρεται να το ψάξει περισσότερο:

- Τι χρονοπρογραμματισμό εφαρμόζει το λειτουργικό σύστημα Linux;
- Πως γίνεται αυτός σε σχέση με αυτά που αναφέραμε στο μάθημα
- Ομοίως για Windows και macOS

Χρήσιμο link: https://en.wikipedia.org/wiki/Scheduling_(computing)

# Τέλος Κεφαλαίου 3

- Είδαμε για ποιες λειτουργείες είναι υπεύθυνο το λειτουργικό σύστημα, πώς διαχειρίζεται και συντονίζει τις διεργασίες και τι είναι ο χρονοπρογραμματισμός.

- Στο επόμενο κεφάλαιο θα ασχοληθούμε με δίκτυα και το διαδίκτυο, και θα μελετήσουμε πρωτόκολλα πολλαπλής πρόσβασης, δρομολόγησης και μεταφοράς