ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΕΠΙΣΤΗΜΗ ΤΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

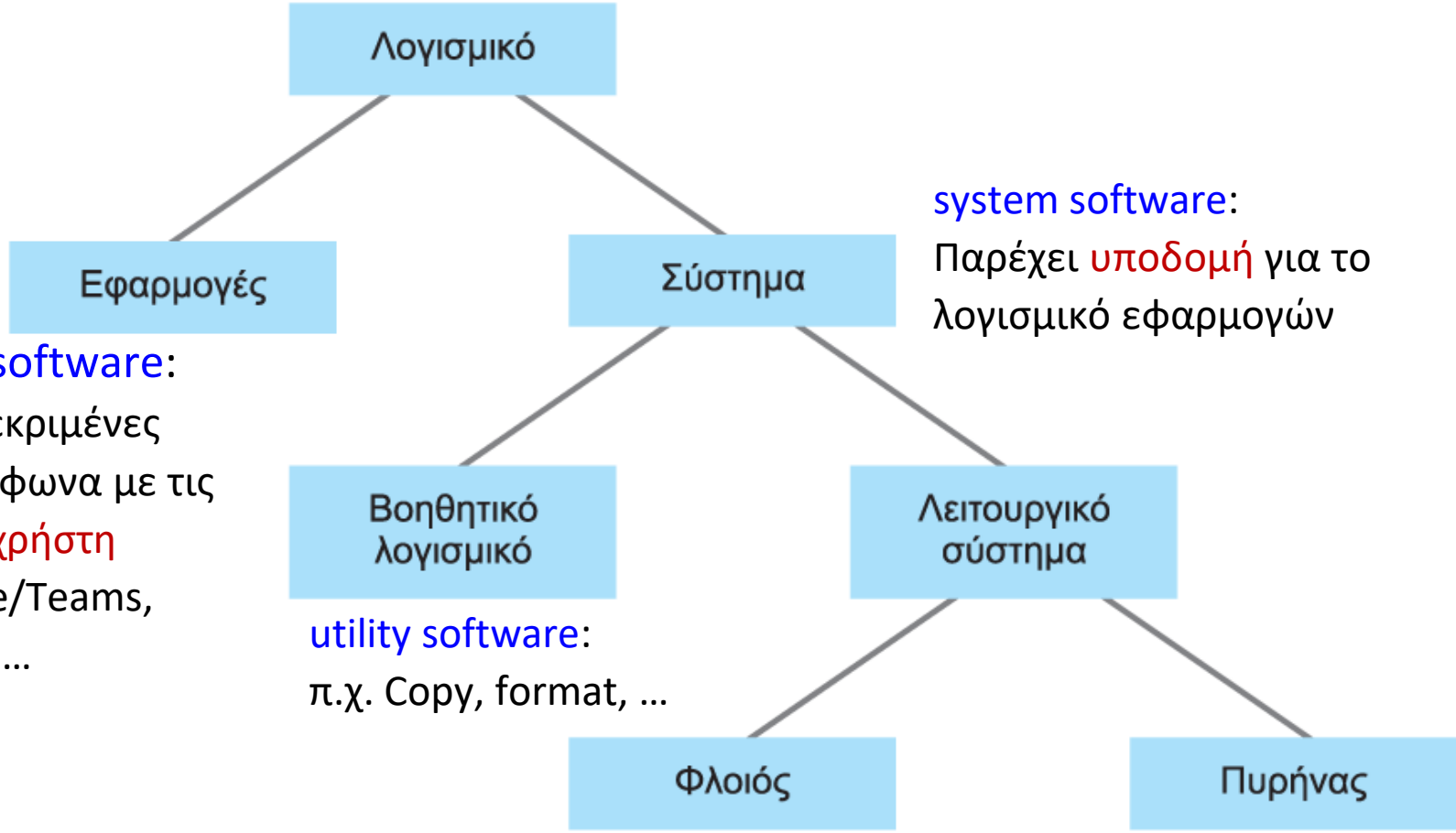
http://eclass.aueb.gr/courses/INF511/

Λειτουργικά Συστήματα (ΚΕΦΑΛΑΙΟ 3)

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



system software:
Παρέχει υποδομή για το λογισμικό εφαρμογών

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

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:
  - 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

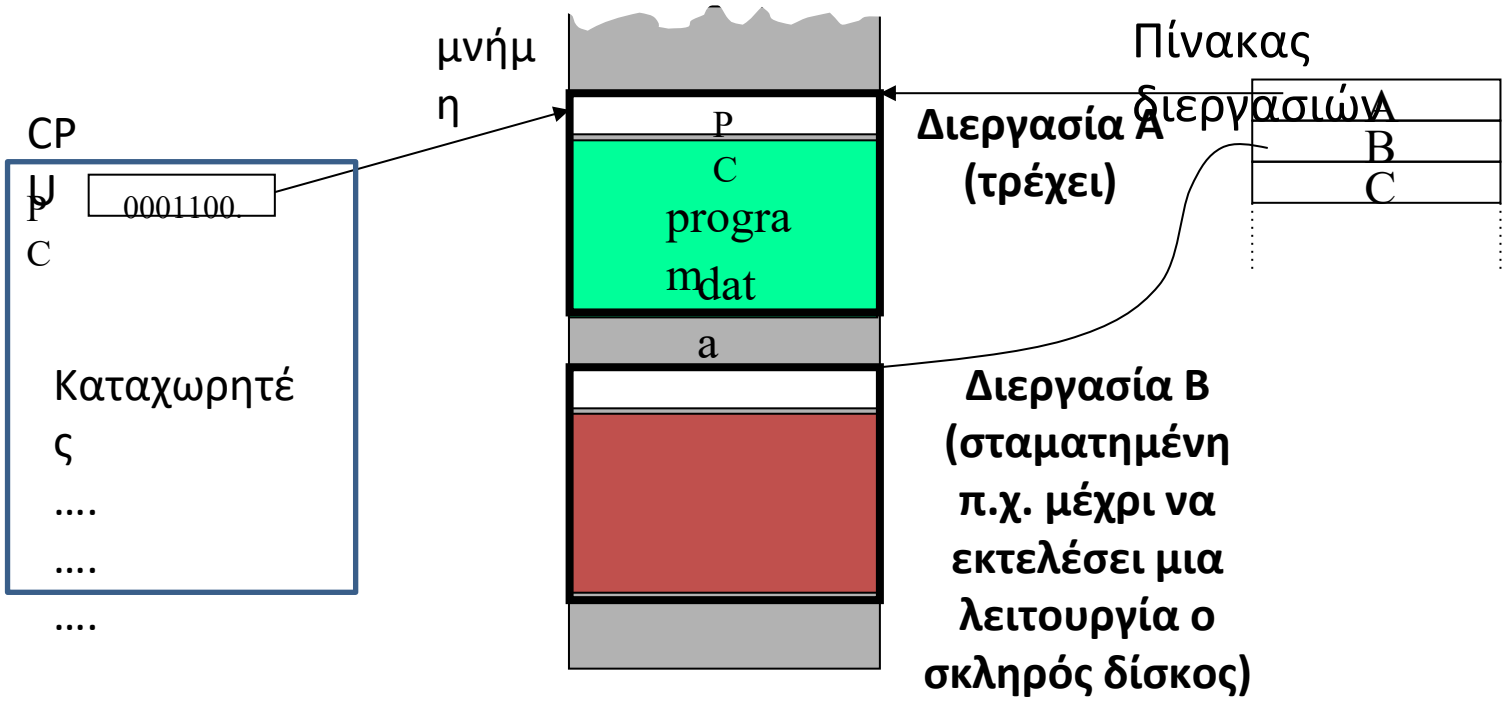
- **Φλοιός** ή **κέλυφος (shell)**: software μέσω του οποίου γίνεται η επικοινωνία του χρήστη με τη μηχανή
  - Γραφικό περιβάλλον διεπαφής με χρήστη (Graphical User Interface, GUI)
  - Διαχειριστής παραθύρων (windows manager)
- **Πυρήνας (kernel)**: περιέχει software που εκτελεί τις βασικές λειτουργίες 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

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)

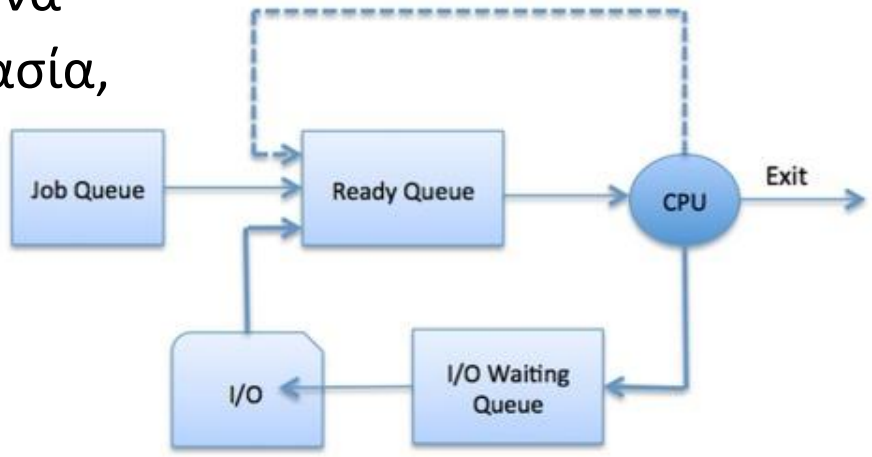
Χρονοπρογραμματιστής (Scheduler)

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.

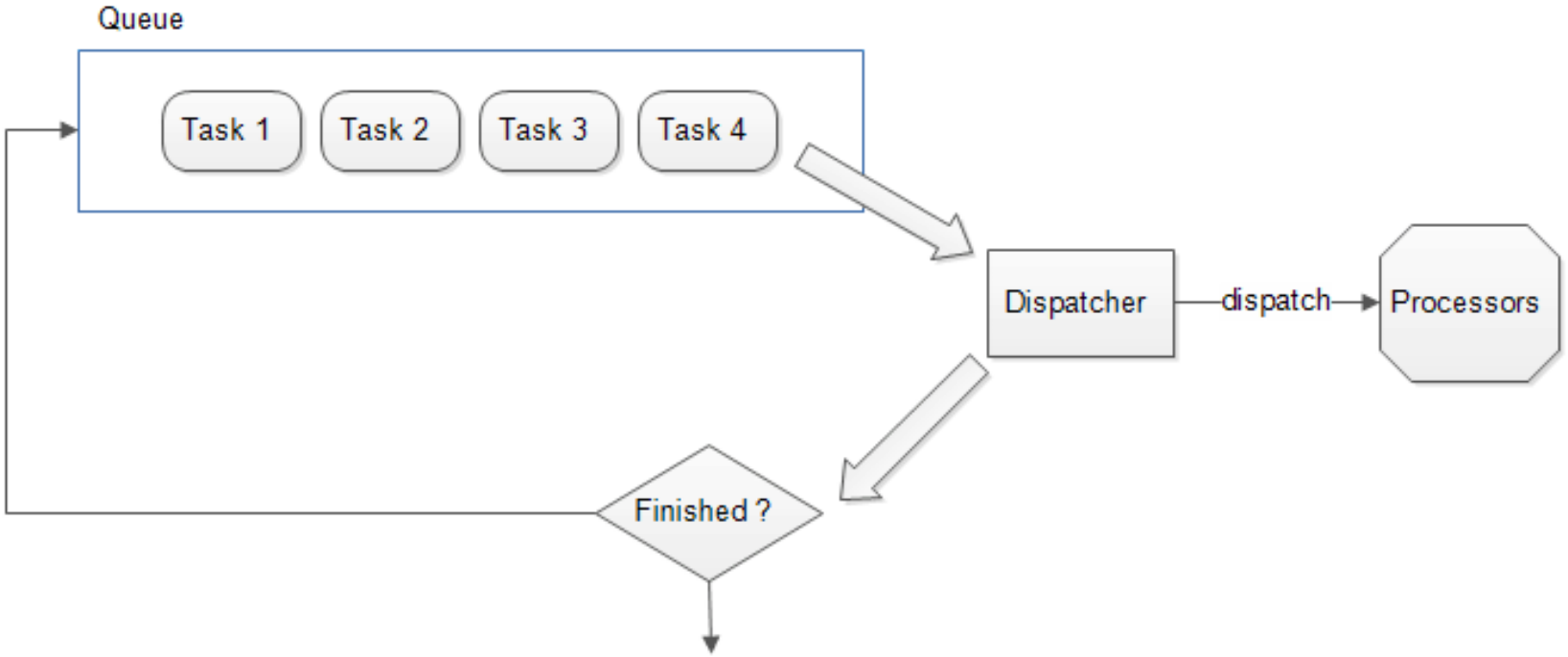


β. Αναθέτει **προτεραιότητες** για την εκτέλεση

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

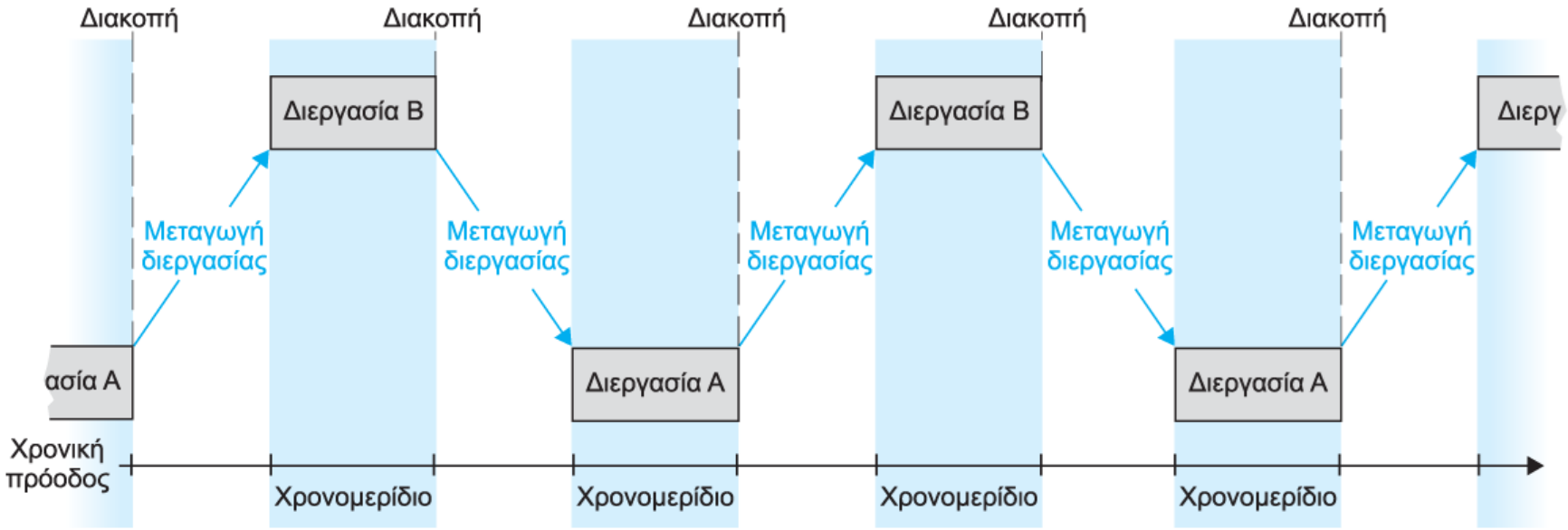
Διεκπεραιωτής (Dispatcher)

- 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)

1. Η. Ο **Διεκπεραιωτής επιλέγει μια διεργασία** που είναι έτοιμη για εκτέλεση από τον πίνακα. π.χ. τη διεργασία Α
   - Το ποια διεργασία θα επιλέξει, εξαρτάται από την προτεραιότητά της (που την παρέχει ο **χρονοπρογραμματιστής** - scheduler)
2. Ξεκινά ένας **μετρητής** που μετράει αντίστροφα
   - Μετρητής: υπόλοιπος χρόνος για να τελειώσει το χρονομερίδιο της Α
3. Ο Διεκπεραιωτής ανακτά την κατάστασή της Α
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 επόμενη διεργασία
8. Ξεκινά ένας μετρητή (διάρκεια = time slice) και ξεκινά η νέα διεργασία. Πήγαινε στο Βήμα 3.

Πολυπρογραμματισμός

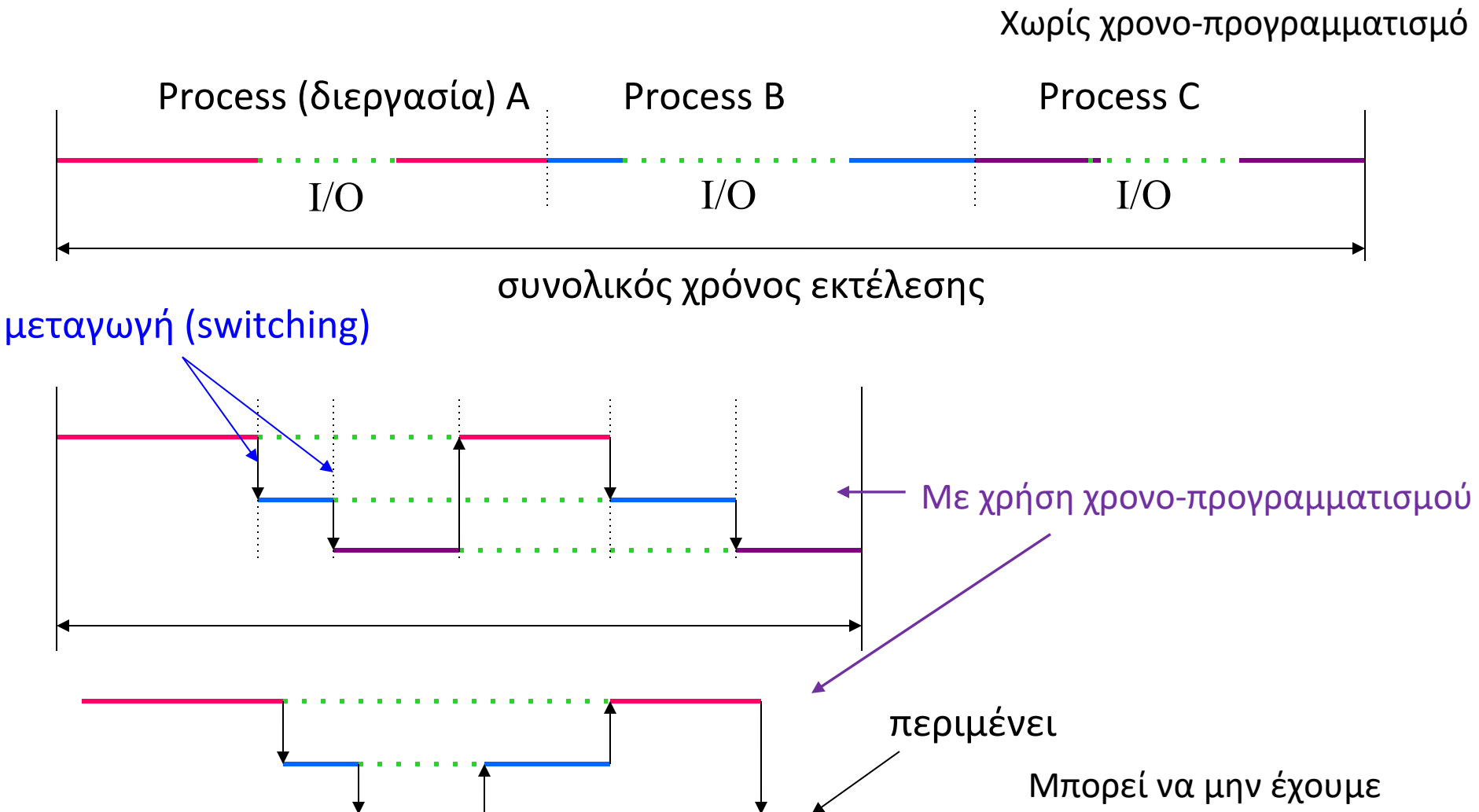
Είναι τελικά καλός ο πολυπρογραμματισμός;

- **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

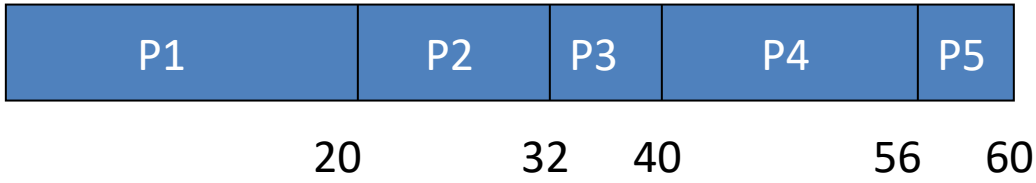
- FCFS = First-come First-serve ή FIFO = First-in First-out
- Εκτέλεση διεργασιών με την σειρά άφιξής τους
- Πλεονέκτημα: απλή μέθοδος
- Μειονέκτημα: Η απόδοση εξαρτάται από την σειρά άφιξης
  - Όχι καλή (όχι δίκαιη) για εργασίες που φτάνουν αργότερα ή αν μια μεγάλη εργασία φτάνει νωρίς
- Διεργασία Α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

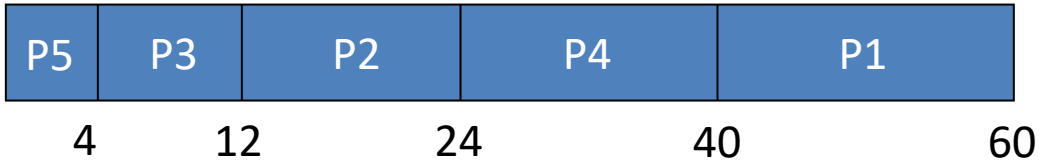
- Να υπολογιστεί ο μέσος χρόνος αναμονής για διεργασίες που έρχονται με σειρά άφιξης P1, P2, P3, P4, P5 και έχουν την παρακάτω διάρκεια
- P1: 20, P2: 12, P3: 8, P4: 16, P5: 4



- 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

Παράδειγμα FIFO

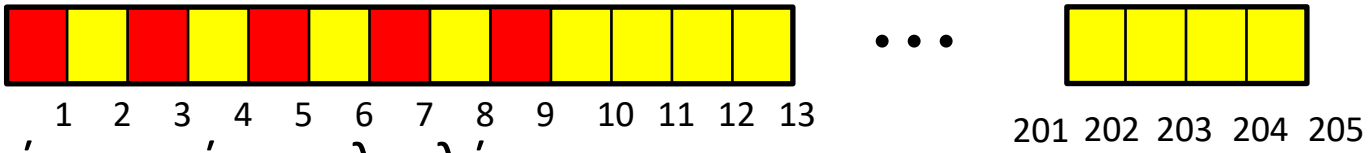
- Να υπολογιστεί ο μέσος χρόνος αναμονής για διεργασίες που έρχονται με σειρά άφιξης P5, P3, P2, P4, P1 και έχουν την παρακάτω διάρκεια
- P1: 20, P2: 12, P3: 8, P4: 16, P5: 4



- Χρόνος αναμονής μιας διεργασίας: ο χρόνος μέχρι αυτή να εκκινήσει
- Χρόνοι αναμονής:
  - P1: 40
  - P2: 12
  - P3: 4
  - P4: 24
  - P5: 0
- Μέσος χρόνος αναμονής: 16 (πολύ καλύτερα από πριν)
- Η σειρά άφιξης παίζει ρόλο!

Χρονοδρομολόγηση με Εναλλαγές (Round Robin Scheduling , RR)

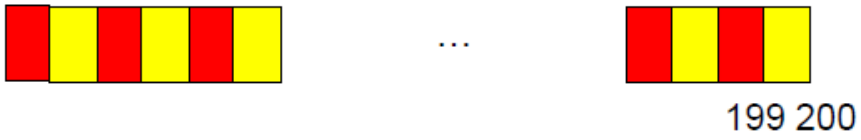
- Το προηγούμενο πρόβλημα δεν υφίσταται αν οι υπολογιστικά μεγάλες διεργασίες δεν μονοπωλούν τον επεξεργαστή
- Εκτέλεση μέρους κάθε διεργασίας σε γύρους
- Σε κάθε γύρο, η CPU εκτελεί διαδοχικά και κυκλικά όλες τις εργασίες, την καθεμία για ένα συγκεκριμένο χρονικό διάστημα (περίοδο) T
  - Δίκαιος τρόπος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



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

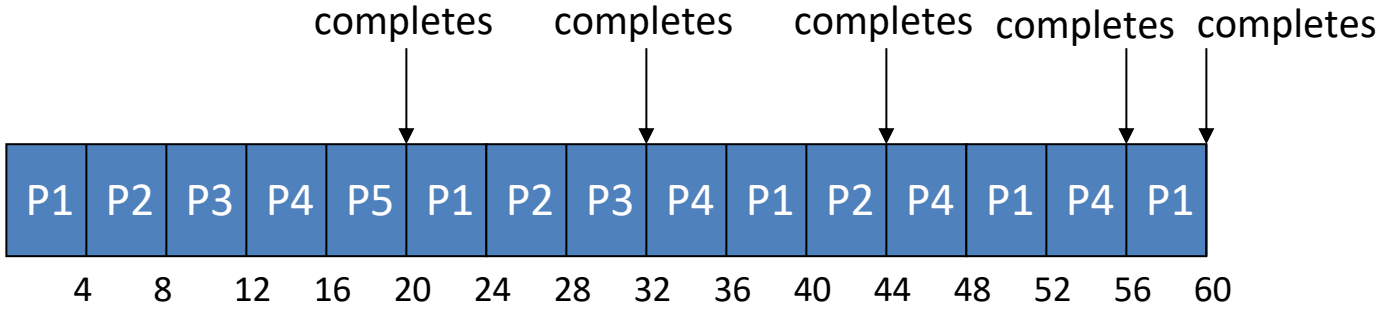
RR: Σχεδιαστικά ζητήματα

- Περίοδος Τ εκτέλεσης κάθε διεργασίας:
  - Πολύ μεγάλη περίοδος: Η πολιτική τείνει προς την FIFO
  - Πολύ 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)

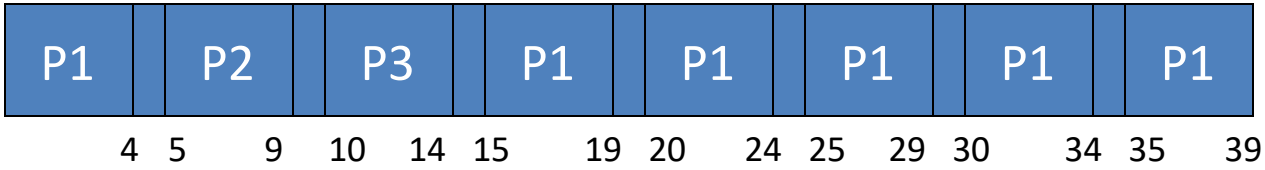
- Διάρκεια της διεργασίας στην CPU και σειρά:
  - P1: 20, P2: 12, P3: 8, P4: 16, P5: 4 που εκτελούνται με αυτή τη σειρά (έστω ότι εμφανίζονται όλες τη στιγμή 0)
  - Περίοδος RR: T = 4



- 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 που εκτελούνται με αυτή τη σειρλα (έστω ότι εμφανίζονται όλες τη στιγμή 0)
  - Περίοδος T = 4, χρόνος για context-switching δ = 1
  - Συνολικός χρόνος αναμονής για κάθε διεργασία:
  - P1: 0 + 11 + 1 + 1 + 1 + 1 = 15 (ή 39 - 24 = 15)
  - P2: 5
  - P3:10
- Μέσος συνολικός χρόνος αναμονής: 10

Ερώτηση για το σπίτι
(εξεταστική Σεπτεμβρίου 2024)

Δίνονται οι διεργασίες P1, P2, P3, με χρονική στιγμή άφιξης και διάρκεια όπως φαίνεται στον παρακάτω πίνακα:

(α) Να υπολογιστεί ο συνολικός χρόνος **αναμονής στην ουρά** της διεργασίας P2, όταν ακολουθείται η πολιτική

1. FIFO (FCFS)
2. Round Robin με περίοδο T=2 sec, όπου σε κάθε διεργασία που μπαίνει στην ουρά δίνεται η μικρότερη προτεραιότητα. (Υποθέστε μηδενικό χρόνο μεταγωγής, δηλαδή δ=0)

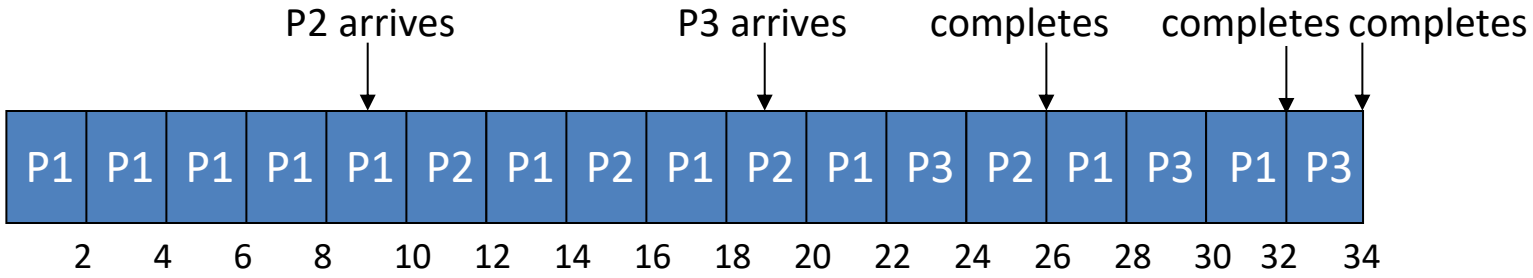
(β) Να υπολογιστεί ο **μέσος συνολικός χρόνος αναμονής στην ουρά** όταν χρησιμοποιείται η πολιτική FIFO (FCFS)

Διεργασία	Χρονική στιγμή άφιξης	Διάρκεια διεργασίας
P1	0 sec	20 sec
P2	9 sec	8 sec
P3	19 sec	6 sec

Ερώτηση για το σπίτι
(εξεταστική Σεπτεμβρίου 2024)

(α1) Χρόνος **αναμονής στην ουρά** της P2 με FIFO: 20-9=11 sec

(α2) Χρόνος **αναμονής στην ουρά** της P2 με Round Robin και περίοδο T=2 sec:



Μετράμε το χρόνο που είναι στην ουρά από τη στιγμή άφιξης ή 26-8-9=9 sec

(β) Χρόνος **αναμονής στην ουρά** με FIFO: της P1 -> 0 sec, της P2 -> 20-9=11 sec, της P3 -> (20+8)-19=9 sec. **Μέσος χρόνος αναμονής**: (0+11+9)/3 = 20/3 sec

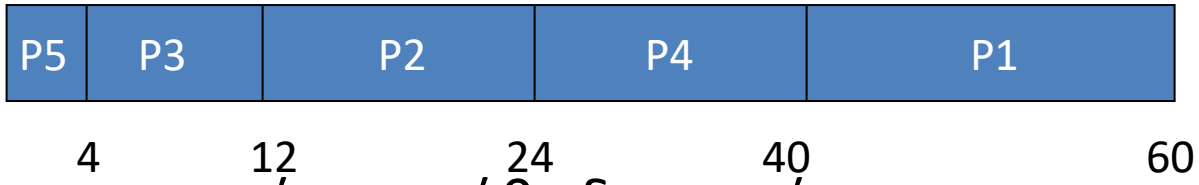
Διεργασία	Χρονική στιγμή άφιξης	Διάρκεια διεργασίας
P1	0 sec	20 sec
P2	9 sec	8 sec
P3	19 sec	6 sec

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)

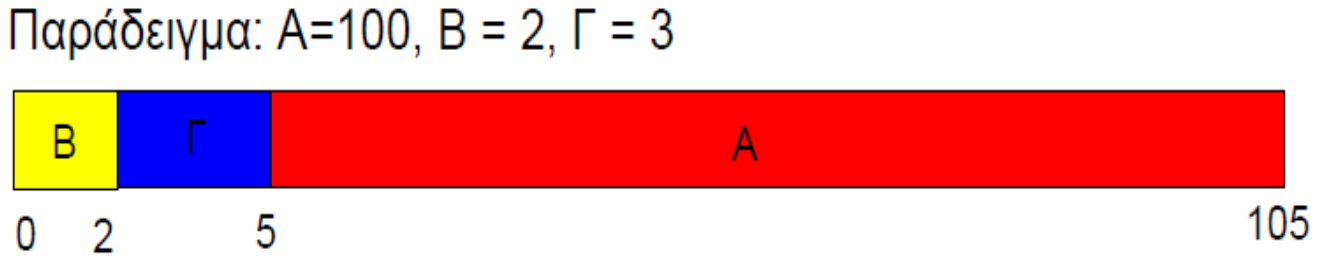
- Διεργασίες και διάρκειες: P1: 20, P2: 12, P3: 8, P4: 16, P5: 4



- 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: Προεκτοπισιμότητα και Μη

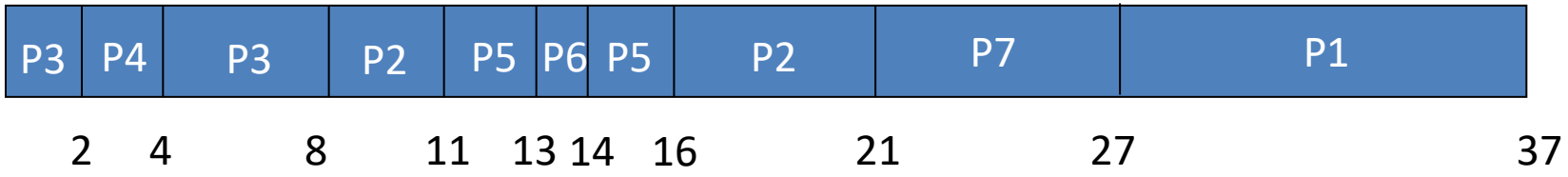
- Έστω Διεργασίες Α,Β,Γ,... με διαφορετική προτεραιότητα
- Προεκτοπίσιμη (pre-emptive) πολιτική χρονοδρομολόγησης με προτεραιότητες: Αν εκτελείται μια διεργασία x και καταφτάσει μια διεργασία y, η x διακόπτεται μόνο αν η y είναι μεγαλύτερης προτεραιότητας από την x
- Εφόσον δεν φτάσει άλλη διεργασία 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 και ποιες οχι;
- 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:
  - Προεκτοπίσιμη ή μη, ανάλογα με το τι γίνεται όταν καταφτάνει μια μεγαλύτερης προτεραιότητας διεργασία ενώ εξυπηρετείται μια άλλη

Παράδειγμα Προεκτοπίσιμης SJF

Διεργασία	Χρονική στιγμή άφιξης	Διάρκεια διεργασίας
P1	0 sec	10 sec
P2	0 sec	8 sec
P3	0 sec	6 sec
P4	2 sec	2 sec
P5	11 sec	4 sec
P6	13 sec	1 sec
P7	15 sec	6 sec



Μέσος χρόνος **ολοκλήρωσης** με προεκτοπίσιμη SJF :
(37 + 21 + 8 + (4-2) + (16-11) + (14-13) + (27-15)) / 7 = 86/7
Μέσος χρόνος **ολοκλήρωσης** με FIFO:
(10 + 18 + 24 + (26-2) + (30-11) + (31-13) + (37-15)) / 7 = 135/7

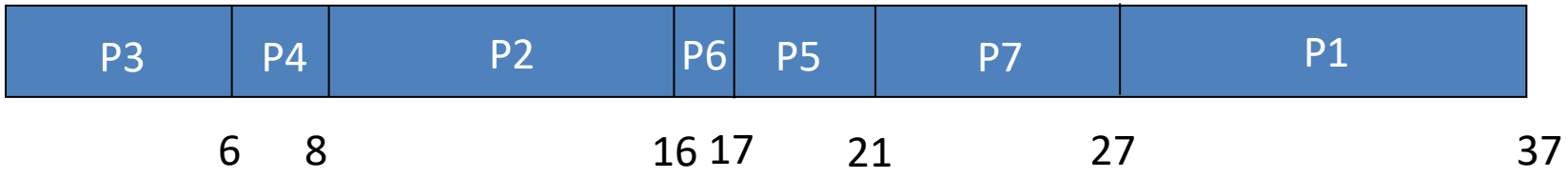
Παράδειγμα Μη Προεκτοπίσιμης SJF

Διεργασία	Χρονική στιγμή άφιξης	Διάρκεια διεργασίας
P1	0 sec	10 sec
P2	0 sec	8 sec
P3	0 sec	6 sec
P4	2 sec	2 sec
P5	11 sec	4 sec
P6	13 sec	1 sec
P7	15 sec	6 sec

Άσκηση για το σπίτι…

Παράδειγμα Μη Προεκτοπίσιμης SJF

Διεργασία	Χρονική στιγμή άφιξης	Διάρκεια διεργασίας
P1	0 sec	10 sec
P2	0 sec	8 sec
P3	0 sec	6 sec
P4	2 sec	2 sec
P5	11 sec	4 sec
P6	13 sec	1 sec
P7	15 sec	6 sec



Μέσος χρόνος **ολοκλήρωσης** με **μη** προεκτοπίσιμη SJF :

(37 + 16 + 6 + (8-2) + (21-11) + (17-13) + (27-15)) / 7 = 91/7

Ενώ ο μέσος χρόνος **ολοκλήρωσης** με προεκτοπίσιμη SJF ήταν: 86/7

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) από τη cpu 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

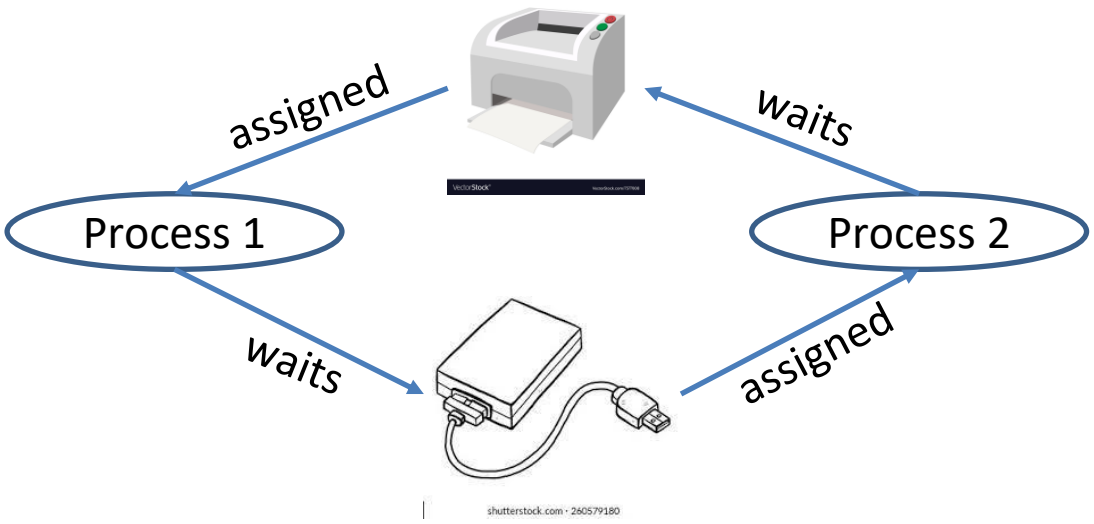
Ανταγωνισμός μεταξύ διεργασιών (2)

- **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

A
B
C
full

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 λειτουργίες 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