Διαχείριση Εισόδου/Εξόδου

Καθοδήγηση και Συντονισμός της λειτουργίας των περιφερειακών συσκευών ώστε:

→ Ανεξαρτησία από τους κώδικες παράστασης χαρακτήρων των συσκευών (*character code independence*)

→ Ανεξαρτησία από τα χαρακτηριστικά και τον τρόπο λειτουργίας των συσκευών (*device independence*)

- ♦ Πρόσβαση ως: Λογικά (εικονικά) ρεύματα (streams)
- ♦ Πρόσβαση ως: Κοινά Αρχεία (files)

→ Ομοιόμορφη μεταχείριση των συσκευών (*uniform treatment of devices*)

→ Βελτιστοποίηση του βαθμού χρήσης/απασχόλησης (*degree of utilization*)

ΕΙΔΗ ΠΕΡΙΦΕΡΕΙΑΚΩΝ ΣΥΣΚΕΥΩΝ ΚΑΙ ΔΙΑΧΕΙΡΙΣΤΩΝ ΕΙΣΟΔΟΥ/ΕΞΟΔΟΥ

♦ Αφιερωμένες συσκευές (dedicated devices)

- π.χ. μονάδες μαγνητικών ταινιών, εκτυπωτές γραμμών/χαρακτήρων, οπτικοί αναγνώστες

♦ Καταμεριζόμενες συσκευές (shared/sharable devices)

- π.χ. μονάδες μαγνητικών δίσκων

♦ Λογικές/εικονικές συσκευες (logical/virtual devices)

- προσομοίωση ‘dedicated’ με ‘shared/sharable’

Περιγραφητής Συσκευής (device descriptor)

- ή Ομάδα Ελέγχου Συσκευής/Μονάδας (DCB/UCB)
  - → φυσική Διεύθυνση (αριθμός) συσκευής
  - → όνομα συσκευής
  - → κατάσταση συσκευής (busy, ready, off)
  - → αριθμός (διεύθυνση) ελεγκτή
  - → Επιπρόσθετες πληροφορίες (εντολές λειτουργίας, δείκτες σε πίνακες μετάφρασης χαρακτήρων, διεύθυνση προγράμματος διαχειριστή, διεύθυνση buffer και μέγεθος, τρέχουσα διεργασία κ.α.)
- Πίνακας Περιγραφητών των Συσκευών (DDT)
- Πίνακας Εκχώρησης των Συσκευών (DAT)

Διαχειριστές Περιφερειακών Συσκευών

→ Τερματικών, Δίσκου, Spoolers κ.λ.π.

→ Διεργασίες ατέρμονης ανακύκλωσης (infinite loop)

- Δέχεται αίτηση εξυπηρέτησης από διεργασία (PID)
- Μετάφραση σε εντολές ελεγκτή, διαύλου κ.λ.π.
- Δρομολόγηση εξυπηρέτησης – δέσμευση συσκευής
- Έκδοση προνομιούχων εντολών εισόδου/εξόδου
- Αναμονή – μεταφορά δεδομένων
- Αντιμετώπιση σφαλμάτων (λειτουργίας/δεδομένων)
- Γνωστοποίηση αποτελεσμάτων στη διεργασία
- Τερματισμός σύνδεσης – αποδέσμευση διεργασίας

ΕΙΚΟΝΙΚΕΣ ΕΝΤΟΛΕΣ ΕΙΣΟΔΟΥ/ΕΞΟΔΟΥ

Για τα προγράμματα των χρηστών παρέχονται συνήθως (απλότητα, διαφάνεια κ.λ.π.) 'εικονικές εντολές' μέσω των οποίων ο χρήστης ορίζει

- Το ρεύμα ή το αρχείο
- Τη διεύθυνση μνήμης για τη μεταφορά δεδομένων

Π.χ.: read (fd, buf, nbyte)

SPOOLERS (print spoolers…)

Προσομοίωση 'dedicated' devices με 'shared/sharable' (χρήση ενταμιευτών – μεταφορά σε physical blocks)

Οταν μία διεργασία θέλει να εκτυπώσει ένα αρχείο, τότε το αρχείο και ο περιγραφητής του εισάγεται σε ένα ειδικό 'ευρετήριο' του Spooler εξόδου.

Ο spooler είναι η μόνη διεργασία που επιτρέπεται να χρησιμοποιήσει (δέσμευση) τον εκτυπωτή

Τί επιτυγχάνεται.....

- Ομοιόμορφη κατανομή του φόρτου στις αφιερωμένες (dedicated) συσκευές – επιτάχυνση
- Ελάττωση της πιθανότητας να συμβεί αδιέξοδο
- Ευκολία με την οποία μπορούμε να πάρουμε πολλαπλά αντίγραφα του ιδίου αρχείου

...όπως με την ομαδοποίηση/απομαδοποίηση δεδομένων κατά την προσπέλαση των Δίσκων

Spoolers δικτύου (network spoolers) – π.χ. μετάδοση μηνυμάτων ηλεκτρονικού ταχυδρομείου

ΔΙΑΧΕΙΡΙΣΤΕΣ ΤΕΡΜΑΤΙΚΩΝ

Είσοδος (από πληκτρολόγιο) και Εξοδος (στην οθόνη)

- → Συλλογή χαρακτήρων από πληκτρολόγιο και μεταφορά τους στη διεργασία
- → 'raw' data (όλοι οι χαρακτήρες ανεπεξέργαστοι - π.χ. εξειδικευμένοι screen editors) και '
- → 'cooked' data (επεξεργασία χαρακτήρων –γραμμές)
- → Ανάγκη μετάφρασης χαρακτήρων (κώδικες)
- → Ανάγκη ενταμίευσης (buffering) χαρακτήρων – (προσωρινοί buffers και dedicated buffers)
- → Echo (ηχώς) χαρακτήρων στην οθόνη
- → Υπολογισμός της θέσης του κέρσορα (με βάση την πληκτρολόγηση και τις εξόδους των διεργασιών)

Κάθε πάτημα πλήκτρου προκαλεί μία 'διακοπή' που ξυπνά τον διαχειριστή τερματικού

Εξοδος στην οθόνη: αντίστοιχα αλλά πιο απλή

Διαφορετικοί Διαχειριστές για Πληκτρολόγιο και Οθόνη

Περιγραφητές Τερματικών (terminal descriptors)

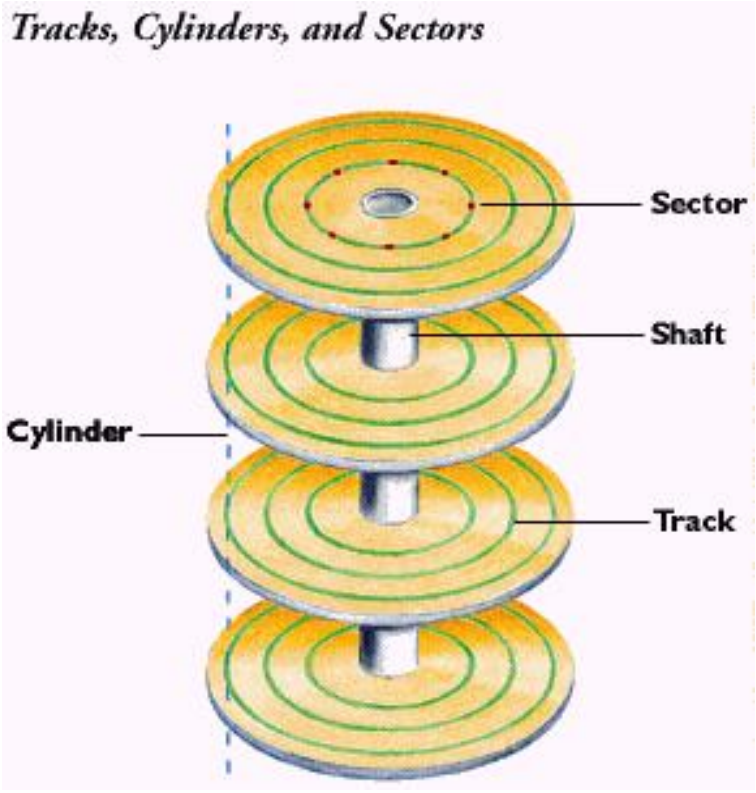
- → Αριθμός (διεύθυνση τερματικού)
- → Κατάσταση (ready, busy, off)
- → Δείκτες σε buffers
- → Ταυτότητα διεργασίας
- → Τρόπος μετάφρασης χαρακτήρων κ.λ.π.

ΔΙΑΧΕΙΡΙΣΤΕΣ ΔΙΣΚΩΝ

Shaft = Αξονας
Cylinder = Κίλυνδρος
Track = Ιχνος (αυλάκι)
Sector = Τομέας

Περιστροφική καθυστέρηση (latency)

Seek Time



Ελεγκτής (controller)

Κρυφή Μνήμη Δίσκου (disk cache)

Είτε στον Ελεγκτή (ακριβότερο, μικρότερη χωρητικότητα) είτε στην Κύρια Μνήμη

'Write-back' και 'Write-through' cache

Δίσκοι Κινητής Κεφαλής

Μία κεφαλή για κάθε επιφάνεια η οποία κινείται κάθετα στην επιφάνεια προς το ζητούμενο 'track'. Ολες οι κεφαλές για όλες τις επιφάνειες κινούνται μαζί (στο ίδιο track).

Μοναδικός Προσδιορισμός (ίχνος, κεφαλή, τομέας)

Δίσκοι Σταθερής Κεφαλής

Μία κεφαλή για κάθε 'track' (πολύ ακριβό)

Πολιτικές Διαχείρισης Δίσκων (disk scheduling)

ΔΙΣΚΟΙ ΚΙΝΗΤΗΣ ΚΕΦΑΛΗΣ

→ Πολλαπλές αιτήσεις για συγκεκριμένο κύλινδρο (track), κεφαλή και τομέα (ομαδοποιούνται ανά κύλινδρο με χρήση διαφορετικών ουρών/λιστών)

→ Η κεφαλή κινείται κάθε φορά προς ένα track/κύλινδρο και διαβάζει/γράφει με τη σειρά που εμφανίζονται οι αιτούμενοι τομείς κατά την περιστροφή

→ *Στόχοι: (α)Ελαχιστοποίηση Μετακίνησης της Κεφαλής, (β) Ελαχιστοποίηση Μέσου Χρόνου Εξυπηρέτησης (πέραν των απλών FCFS και Priority-based)*

- FCFS (First Come First Serve)

- Δίκαιος αλλά στη γενική περίπτωση ...αργός

- SSTF (Shortest Seek Time First)

- ο πιο αποδοτικός (πρόβλήματα: 'starvation')

- SCAN algorithms

- αποφεύγεται το starvation (στη γενική περίπτωση χειρότερος από τον SSTF)
- παραλλαγές: SCAN, C-SCAN, LOOK, C-LOOK

- OPT (optimal – βέλτιστος)

ΔΙΣΚΟΙ ΣΤΑΘΕΡΗΣ ΚΕΦΑΛΗΣ

- Ο κύλινδρος/track είναι πλέον δεδομένος
- Απαιτείται μόνο να εξυπηρετηθούν οι αιτούμενοι τομείς με τη σειρά που εμφανίζονται κατά την περιστροφή

ΑΣΚΗΣΗ 1

Κατά το χρονικό διάστημα που κινείται ο βραχίονας αναζήτησης ενός δίσκου κινητής κεφαλής για να μεταφέρει πληροφορίες από ή προς τον κιλυνδρο 11, φθάνουν στον διαχειριστή του δίσκου απαιτήσεις, με τη σειρά που δίνονται παρακάτω, για τη μεταφορά δεδομένων από ή προς τους κιλύνδρους:

1, 36, 16, 34, 9, 12

Πόσους κιλύνδρους θα διανύσει συνολικά ο βραχίονας του δίσκου για να εξυπηρετήσει τις απαιτήσεις αυτές, όταν χρησιμοποιηθούν οι παρακάτω αλγόριθμοι;

A) FCFS B) SSTF

Γ) LOOK, όταν η κεφαλή κινείται προς αύξοντες αριθμούς (διευθύνσεις) κυλίνδρων

Απάντηση:

A) FCFS: 111 κίλυνδροι

B) SSTF: 61 κίλυνδροι

Γ) LOOK: 60 κίλυνδροι … **(best)**

Μέσοι Χρόνοι Εξυπηρέτησης:

A) FCFS: = (10+45+65+83+108+111)/6 = 422/6 = ...

B) SSTF: = (1+4+11+26+59+61)/6 = 162/6 = ... **(best)**

Γ) LOOK: = (1+5+23+25+52+60)/6 = 166/6 = ...

ΑΣΚΗΣΗ 2

Υποθέστε ότι η κεφαλή ανάγνωσης/εγγραφής ενός δίσκου κινητής κεφαλής, που έχει 200 ίχνη (0 ως 199), αρχίζει από το ίχνος 53 και ότι η ουρά των απαιτήσεων περιέχει απαιτήσεις για τα ίχνη:

98, 183, 37, 122, 14, 124, 65, 67

Πόσα ίχνη θα πρέπει η κεφαλή να διανύσει συνολικά για να εξυπηρετήσει τις απαιτήσεις αυτές, όταν χρησιμοποιηθούν οι παρακάτω αλγόριθμοι;

Α) FCFS

Β) SSTF

Γ) OPT, βέλτιστος χρονοπρογραμματισμός

Δ) LOOK, όταν η κεφαλή κινείται προς το ίχνος 0

Ε) C-LOOK, όταν η κεφαλή κινείται προς το ίχνος 199

Απάντηση:

Α) FCFS: 640 ίχνη

Β) SSTF: 236 ίχνη

Γ) OPT: 208 ίχνη

Δ) LOOK: 208 ίχνη

Ε) C-LOOK: 322 ίχνη

Τί θα έβγαζε η χρήση του LOOK με την η κεφαλή να κινείται προς το ίχνος 199 ?

ΑΣΚΗΣΗ 3

Για ένα σύστημα δίσκου με μία επιφάνεια, 50 ίχνη (αυλάκια) και 8 τομείς ανά ίχνος, καταφτάνουν οι ακόλουθες αιτήσεις μία δεδομένη χρονική στιγμή (στη μορφή [ίχνος, τομέας]). Εστω οι αιτήσεις για εξυπηρέτηση

(10,3), (22,4) (20,4), (10,6), (2,7), (2,5), (20,2), (40,0), (6,1), (10,7), (22,5) (38,2), (40,2)

Υποθέτοντας ότι (α) για τη μετακίνηση από ίχνος σε γειτονικό ίχνος απαιτούνται 6ms, (β) η κεφαλή αρχικά βρίσκεται στο ίχνος 20

Α. Υπολογίστε το συνολικό απαιτούμενο χρόνο αναζήτησης για τους αλγορίθμους

(α) FCFS (β) SSTF (γ) SCAN (κίνηση άνω) (δ) LOOK (κίνηση άνω)

Β. Υπολογίστε το μέγιστο συνολικό χρόνο εξυπηρέτησης, υποθέτοντας ότι

(α) για μία περιστροφή του δίσκου απαιτούνται 24ms

(β) η εξυπηρέτηση δεδομένων (διάβασμα/γράψιμο) σε έναν τομέα γίνεται σε μηδενικό χρόνο, καθώς περνάει ο συγκεκριμένος τομέας κάτω από την κεφαλή.

Hints:

Για το Α:

Πρέπει να ομαδοποιήσετε τις αιτήσεις ανά ίχνος (κύλινδρο), ανεξάρτητα με τη σειρά κατά την οποία ήρθαν τυχόν 'επόμενοι' τομείς για κάποιο ήδη αναφερθέν/αιτηθέν ίχνος, δηλαδή

10 (3,6,7) 22 (4,5) 20 (4,2) 2 (7,5) 40 (0,2) 6 (1) 38 (2)

και μετά να υπολογίσετε τους χρόνους αναζήτησης (σε αριθμό ιχνών συνολικής μετακίνησης) για κάθε έναν από τους ζητούμενους αλγορίθμους, σαν να ήταν μονές/απλές αιτήσεις για κάθε ίχνος/κύλινδρο (και να πολαπλασιάσετε μετά με τα 6ms)

Για τους αλγόριθμους SCAN, LOOK θεωρείστε ότι αν υπάρχει αίτηση αρχικά για τον κύλινδρο/ίχνος όπου ήδη βρίσκεται η κεφαλή πρώτα εξυπηρετείται αυτή και μετά συνεχίζεται η μετακίνηση (εκτός αν σας λέει κάτι διαφορετικό η

εκφώνηση – π.χ. αν έλεγε ότι ‘η κεφαλή μόλις έχει τελειώσει με την εξυπηρέτηση μίας αίτησης στο ίχνος/κύλινδρο 20’ θα έπρεπε να υποθέσετε ότι ήδη έχει αρχίσει να κινείται προς κάποια κατεύθυνση και έτσι ‘χάνει’ την επόμενη αίτηση για τον κύλινδρο/ίχνος 20)

<u>Για το Β:</u>

Για κάθε επίσκεψη σε κάποιον κύλινδρο/ίχνος, θεωρείστε ότι θα πρέπει να προσθέσετε εκτός από το χρόνο που απαιτείται για την μετακίνηση σε αριθμό ιχνών (προκειμένου να φτάσει η κεφαλή στο συγκεκριμένο ίχνος/κύλινδρο), και τον χρόνο που απαιτείται για να εξυπηρετήσει όλες τις αιτήσεις τομέων που αφορούν τον συγκεκριμένο κύλινδρο/ίχνος ως εξής:

Θεωρούμε ότι:

- για το διάβασμα ενός τομέα (πέρασμα όλου του τομέα κάτω από την κεφαλή) απαιτείται χρόνος 24/8 = 3ms.

- εξυπηρετείται πρώτα ο τομέας ο οποίος θα βρεθεί πρώτος κάτω από την κεφαλή ανεξάρτητα από τη σειρά της αίτησής τους σε σύγκριση με τους υπόλοιπους τομείς για τον ίδιο κύλινδρο/ίχνος.

- Υποθέτετε στη συνέχεια ότι (θα σας το ξεκαθαρίζει η εκφώνηση αυτό),

    - Είτε ότι κάθε φορά που επισκέφτεστε έναν κύλινδρο/ίχνος η χειρότερη περίπτωση είναι μόλις να εχει αρχίσει να περνάει κάτω από την κεφαλή ένας από τους αιτούμενους/αναφερόμενους τομείς για τον συγκεκριμένο κύλινδρο/ίχνος . Π.χ. για το 10(3,6,7) υποθέτουμε ότι μόλις έχει π.χ. αρχίσει να περνάει ο 3 οπότε μέγιστος απαιτούμενος χρόνος είναι να γίνει μία ολόκληρη περιστροφή (όπου ενδιάμεσα θα διαβαστούν ο 6 και ο 7) και μετά να ξαναέρθει ο 3 και να περάσει/διαβαστεί ολόκληρος από την κεφαλή, δηλαδή σύνολο χρόνος περιστροφής/διαβάσματος 9 τομέων (9x3=27ms) (σσ. το ίδιο απτέλεσμα βγαίνει και αν υποθέσουμε αρχικά ότι ότι μόλις έχει π.χ. αρχίσει να περνάει ο 6 ή ο 7).

    - Είτε ότι το παραπάνω συμβαίνει μόνο την πρώτη φορά και στη συνέχεια πρέπει να υπολογίσετε για κάθε επίσκεψη σε νέο κύλινδρο/ίχνος πόση περιστροφή έχει κάνει ήδη επιπλεον ο δίσκος και ποιός τομέας βρίκσεται κάθε φορά κάτω από την κεφαλή)

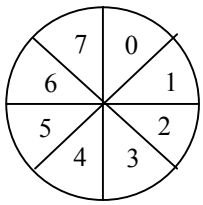
ΑΣΚΗΣΗ 4

Ο ελεγκτής παρότι ναι μεν μπορεί να διαβάσει ή γράψει αυτόματα από/κάθε τομέα καθώς αυτός περνά κάτω από την κεφαλή, χρειάζεται στη συνέχεια συνήθως (για κάθε διάβασμα/γράψιμο ενός τομέα) και κάποιο χρόνο για να μεταφέρει τα αντίστοιχα δεδομένα από/προς τη μνήμη.

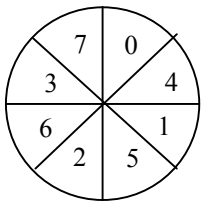
Για το λόγο αυτό (και επειδή κατά το χρόνο μεταφοράς) ο επόμενος/προηγούμενος τομέας περνά κανονικά κάτω από την κεφαλή, ο ελεγκτής συνήθως δεν προλαβαίνει να διαβασει/γράψει ακολουθιακά blocks/τομείς.

Για να λυθεί το πρόβλημα αυτό (και έχοντας στο μυαλό μας ότι πολύ συχνά απαιτείται η προσπέλαση ακολουθιακών μπλοκς/τομέων καθώς έτσι αποθηκεύονται συνήθως μεγάλα αρχεία στο δίσκο), συνηθίζεται να αποθηκεύουμε τα ακολουθιακά (στη λογική) μπλοκς/τομείς, ανά δύο ή τρεις πραγματικούς τομείς/μπλοκς σε κάθε ίχνος του δίσκου (ανάλογα με το πόσο χρόνο χρειάζεται ο ελεγκτής) για την μεταφορά των δεδομένων.

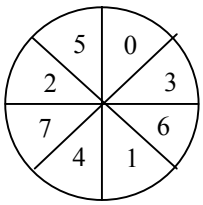
Οταν τα αποθηκεύουμε ανά 2 τομείς έχουμε τη λεγόμενη ‘μονή υπερπήδηση’ (interleaving)
Οταν τα αποθηκεύουμε ανά 3 τομείς έχουμε τη λεγόμενη ‘διπλή υπερπήδηση’ (interleaving)



Χωρίς υπερπήδηση



Μονή υπερπήδηση



Διπλή υπερπήδηση

Πρόβλημα:

Αν έχουμε ένα δίσκο με διπλή υπερπήδηση, με 8 τομείς των 512 bytes σε κάθε ίχνος και ταχύτητα περιστροφής 300 rpm (περιστροφές το λεπτό),

(i) Πόσο χρόνο χρειάζεται να διαβάσει όλους τους τομείς που βρίσκονται σε ένα ίχνος, αν υποθέσουμε ότι ο βραχίονας βρίσκεται στη σωστή θέση και ότι χρειάζεται ½ περιστροφής για να φτάσει ο τομέας ‘0’ κάτω από την κεφαλή?
(ii) Ποιός είναι ο ρυθμός δεδομένων ?
(iii) Επαναλάβατε το πρόβλημα σε ένα δίσκο με τα ίδια χαρακτηριστικά, χωρίς υπερπήδηση.
(iv) Πόσο μειώνεται ο ρυθμός δεδομένων ?

Λύση:

(i) Απαιτείται ½ περιστροφή για να φτάσει ο τομέας '0' κάτω από την κεφαλή και ακολούθως 2,75 περιστροφές μέχρι να διαβαστούν και οι 8 τομείς (τελευταίος θα περάσει και θα διαβαστεί ο τομέας 7 κατά την τρίτη περιστροφή). Αρα σύνολο 3,25 περιστροφές.

Σε αυτές κανονικά πρέπει να προστεθεί και το maximum (όπως υπαγορεύεται από τη διπλή υπερπήδηση) του χρονου μεταφοράς των δεδομένων του τελευταίου διαβασμένου τομέα (7) στη μνήμη. Δηλαδή σύνολο τελικά (με αυτήν την παραδοχή): 3,5 περιστροφές.

Εφόσον ο δίσκος περιστρέφεται με ρυθμό 300rpm, ο χρόνος που απαιτείται για 3,5 περιστροφές είναι ίσος με (απλή μέθοδος των τριών):
T = 3,5/300 min = 0,7sec (αντίστοιχα T = 0,65 sec αν υπολογίσουμε με 3,25 περιστροφές).

(ii) Διαβάζονται συνολικά 8x512=4096 bytes σε 0,7 sec.
Αρα ρυθμός δεδομένων (απλή μέθοδος των τριών)
R = 4096 x 1/0,7 = 5851 bytes/sec
Ανίστοιχα αν υπολογίσουμε με 3,25 περιστροφές
R = 4096 x 1/0,65 = 6301 bytes/sec

(iii) Χωρίς υπερπήδηση (αυτόματο διάβασμα), θα απαιτηθούν συνολικά μόνο 1,5 περιστροφές άρα T = 1,5/300 min = 0,3sec

(iv) R = 4096 x 1/0,3 = 13653 bytes/sec
Ο ρυθμός μειώθηκε κατά (13653-5851)/13653= 57,1%
Ανίστοιχα αν υπολογίσουμε με 3,25 περιστροφές
Ο ρυθμός μειώθηκε κατά (13653-6301)/13653 = 53,8%

(iii) Αν υποθέσουμε 'χωρίς υπερπήδηση' μεν, χωρίς όμως αυτόματο διάβασμα (χάσιμο δηλαδή κάθε επόμενου τομέα), θα απαιτηθούν συνολικά 8,5 περιστροφές άρα T = 8,5/300 min = 1,7sec

(iv) R = 4096 x 1/1,7 = 2409 bytes/sec
Ο ρυθμός μειώθηκε κατά (5851-2409)/5851= 58,8%
Ανίστοιχα αν υπολογίσουμε με 3,25 περιστροφές
Ο ρυθμός μειώθηκε κατά (6301-2409)/6301= 61,7%

Ασκηση 5

Θεωρήστε έναν δίσκο με 60 συνολικά κυλίνδρους (1...60) και την κεφαλή του να βρίσκεται στον κίλυνδρο 40.
Εστω η ουρά αιτήσεων για τους κυλίνδρους 10, 15, 20, 45, 50, 55
Εστω επίσης ότι:
- Χρόνος μετακίνησης 1 κιλύνδρου = 6ms
- Χρόνος εξυπηρέτησης 1 αίτησης = 3 ms
Εστω επίσης ότι:
- στα 30ms ήρθε αίτηση για τον κίλυνδρο 42
- στα 50 ms έρχεται αίτηση για τον κύλινδρο 36
- στα 70 ms έρχεται αίτηση για τον κύλινδρο 25

(α) Υπολογίστε πόσος χρόνος απαιτείται για την εξυπηρέτηση όλων των αιτήσεων με χρήση του SSTF. Ποιά η διαφορά με τον να έρχονταν οι παραπάνω επιπλέον αιτήσεις στα 60, 80 και 100 ms αντίστοιχα ?
(β) Επαναλάβατε το ίδιο για τον FCFS
(γ) Επαναλάβατε το ίδιο για τον LOOK με κατεύθυνση άνω

Λύση

(α) για τον SSTF:

40→45 = 5x6+3=33ms
έχει έρθει η αίτηση για τον κύλινδρο 42
45→42 = 3x6+3=21ms (σύνολο 54ms)
έχει έρθει η αίτηση για τον κύλινδρο 36
42→36 = 6x6+3=39ms (σύνολο 93ms)
έχει έρθει η αίτηση για τον κύλινδρο 25
36→25 =11x6+3 = 69ms (σύνολο 162ms)
συνεχίζουμε κατά τον ίδιο τρόπο εξυπηρετώντας κατά σειρά τις αιτήσεις για τους κυλίνδρους 20, 15, 10, 50, 55
(η συνολική σειρά εξυπηρέτησης δηλαδή είναι 45, 42, 36, 25, 20, 15, 10, 50, 55)
(συνεχίστε μόνοι σας υπολογίζοντας τον τελικό συνολικό χρόνο εξυπηρέτησης)

αν οι παραπάνω επιπλέον αιτήσεις για τους κυλίνδρους 42, 36, 25 έρχονταν στα
60, 80 και 100 ms αντίστοιχα, δεν θα επηρέαζαν τόσο πολύ τη σειρά εξυπηρέτησης (θα έμπαιναν δηλαδή απλά ανάμεσα στις δεξιές και τις αριστερές αιτήσεις) και η συνολική σειρά εξυπηρέτησης θα ήταν
45-50-55-42-36-25-20-15-10
(συνεχίστε μόνοι σας υπολογίζοντας τον τελικό συνολικό χρόνο εξυπηρέτησης)

(β) για τον FCFS η συνολική σειρά εξυπηρέτησης θα ήταν σε κάθε περίπτωση
10, 15, 20, 45, 50, 55, 42, 36, 25

(γ) για τον LOOK με κατεύθυνση άνω, πάλι σε κάθε περίπτωση η συνολική σειρά εξυπηρέτησης θα ήταν 45, 50, 55, 42, 36, 25, 20, 15, 10.
Αντίθετα (για τον LOOK) αν οι επιπλέον αιτήσεις έρχονταν π.χ. σε χρόνο 350, 360, 370, η συνολική σειρά εξυπηρέτησης θα ήταν 45, 50, 55, 20, 15, 10, 42, 36, 25

Ασκηση 6

A disk arm is currently positioned over cylinder 500 of a disk and is moving upwards. There are outstanding requests for cylinders 100 210 300 520 600 and 740.

(α) Identify the order in which these requests will be satisfied if the disk scheduling uses the **Elevator Algorithm** (LOOK) and calculate the number of cylinders which the arm traverses.

(β) Explain the advantage of this algorithm compared to servicing requests on a "First Come First Served" basis or on a "Shortest Seek First" basis.

Απαντήσεις:

(α) The Elevator Algorithm causes the arm to sweep across the disk in the same direction until all requests in that direction have been satisfied and then reverse.
The requests will be satisfied in the order

	500	520	600	740	300	210	100	
movements	20	80	140	440	90	110	880	cyls

Επαναλάβατε μόνοι σας για τον LOOK με κατεύθυνση κάτω, και για τον SSTF

(β) Since the disk may be servicing requests for many different files, FCFS may result in excessive arm movements which will seriously degrade retrieval time. Elevator algorithm (LOOK) reduces the overall movements and smoothes them by making them in one direction. It also suits the typical organisation of modern file systems that put frequently accessed data in the middle of the disk.
Shortest Seek First minimises arm movement but may result in the driver continuously servicing requests in one block of cylinders and starving requests outside this.

ΒΑΣΙΚΟΙ ΑΛΓΟΡΙΘΜΟΙ ΧΡΟΝΟΔΡΟΜΟΛΟΓΗΣΗΣ ΔΙΣΚΩΝ ΚΙΝΗΤΗΣ ΚΕΦΑΛΗΣ (συνέχεια)

- FCFS (First Come First Serve)

Ικανοποιεί κάθε φορά την επόμενη κατά σειρά άφιξης αίτηση

- SSTF (Shortest Seek Time First)

Ικανοποιεί κάθε φορά την αίτηση που αφορά/βρίσκεται στο κοντινότερο ίχνος (για την οποία δηλαδή χρειάζεται να διανύσει το μικρότερο αριθμό ιχνών για να την ικανοποιήσει)

- SCAN (Elevator) algorithms[1,2]

Διατρέχει (σαρώνει) το δίσκο προς μία κατεύθυνση (π.χ. από το ίχνος 0 προς το ίχνος N-1) και ικανοποιεί κάθε αίτηση που συναντά μπροστά του (με τη σειρά που τις συναντά). Όταν φτάσει σε κάποιο άκρο (0 ή N-1) γυρίζει και σαρώνει προς την αντίθετη κατεύθυνση (ικανοποιώντας πάλι κάθε αίτηση που συναντά μπροστά του - με τη σειρά που τις συναντά) κοκ. Παραλλαγές (έστω ένας δίσκος με N ίχνη: 0…N-1):

SCAN: σαρώνει προς κάθε κατεύθυνση μέχρι ακριβώς το εκάστοτε επόμενο άκρο (0 ή N-1)

LOOK: όπως ο SCAN αλλά σαρώνει προς κάθε κατεύθυνση μέχρι το ίχνος μόνο που υπάρχει κάποια αίτηση έως εκείνη τη χρονική στιγμή. Εάν τη στιγμή που μόλις εξυπηρέτησε την τρέχουσα αίτηση, δεν υπάρχει κάποια άλλη αίτηση προς το άκρο της κατεύθυνσης που σαρώνει, γυρνά αμέσως και εξυπηρετεί προς την αντίθετη κατεύθυνση.

---

[1] **C-SCAN / C-LOOK (Circular Scan / Look):** διατρέχει πάντα προς μία κατεύθυνση (από μικρότερα προς μεγαλύτερα ίχνη: δηλαδή από 0 έως N-1 αν είναι ο C-SCAN ή μέχρι εκεί που υπάρχει αίτηση αν είναι ο C-LOOK) -> και όταν φτάσει στο αντίστοιχο άκρο (ίχνος N-1 ή ίχνος τελευταίας υπάρχουσας αίτησης αντίστοιχα), επιστρέφει αυτόματα πάλι στην αρχή (ίχνος 0 αν είναι ο C-SCAN ή ίχνος πρώτης υπάρχουσας από την αρχή αν είναι ο C-LOOK) χωρίς να ικανοποιεί καμία αίτηση ενδιάμεσα, και αρχίζει πάλι να ικανοποιεί προς την ίδια κατεύθυνση (από μικρότερα δηλαδή προς μεγαλύτερα ίχνη).

[2] **OPT (optimal – βέλτιστος /** ως προς τον συνολικό αριθμό διανυόμενων ιχνών): Ισοδύναμος με τον LOOK - με την κεφαλή να κινείται προς το κοντινότερο άκρο.

ΑΣΚΗΣΗ #0

Κατά το χρονικό διάστημα που κινείται ο βραχίονας αναζήτησης ενός δίσκου κινητής κεφαλής **(με 50 συνολικά ίχνη 0…49)** για να μεταφέρει πληροφορίες **από ή προς τον κύλινδρο 11**, φθάνουν στον διαχειριστή του δίσκου αιτήσεις, με τη σειρά που δίνονται παρακάτω, για τη μεταφορά δεδομένων από ή προς τους κυλίνδρους:
1, 36, 16, 34, 9, 12
Πόσους κυλίνδρους θα διανύσει συνολικά ο βραχίονας του δίσκου για να εξυπηρετήσει τις αιτήσεις αυτές;

A) FCFS
11 -> 1 -> 36 -> 16 -> 34 -> 9 -> 12
B) SSTF
11 -> 12 -> 9 -> 16 -> 1 -> 34 -> 36
Γ) SCAN, με την κεφαλή να κινείται προς τα πάνω
11 -> 12 -> 16 -> 34 -> 36 -> 49 -> 9 -> 1
Δ) SCAN, με την κεφαλή να κινείται προς τα κάτω
11 -> 9 -> 1 -> 0 -> 12 -> 16 -> 34 -> 36
E) LOOK, με την κεφαλή να κινείται προς τα πάνω
11 -> 12 -> 16 -> 34 -> 36 -> 9 -> 1
Z) LOOK, με την κεφαλή να κινείται προς τα κάτω
11 -> 9 -> 1 -> 12 -> 16 -> 34 -> 36
H) C-SCAN
11 -> 12 -> 16 -> 34 -> 36 -> 49 -> 0 -> 1 -> 9
Θ) C-LOOK
11 -> 12 -> 16 -> 34 -> 36 -> 1 -> 9

A) FCFS: 111 κύλινδροι
B) SSTF: 61 κύλινδροι
Γ) SCAN πάνω: 86 κύλινδροι …
Δ) SCAN κάτω: 47 κύλινδροι …
E) LOOK πάνω: 60 κύλινδροι …
Z) LOOK κάτω: 45 κύλινδροι … **(best)**
H) C-SCAN: 96 κύλινδροι …
Θ) C-LOOK: 68 κύλινδροι …

Selecting a Disk-Scheduling Algorithm

- SSTF is common and has a natural appeal
- SCAN and C-SCAN perform better for systems that place a heavy load on the disk.
- Performance depends on the number and types of requests.
- Requests for disk service can be influenced by the file-allocation method.
- The disk-scheduling algorithm should be written as a separate module of the operating system, allowing it to be replaced with a different algorithm if necessary.
- Either SSTF or LOOK is a reasonable choice for the default algorithm.

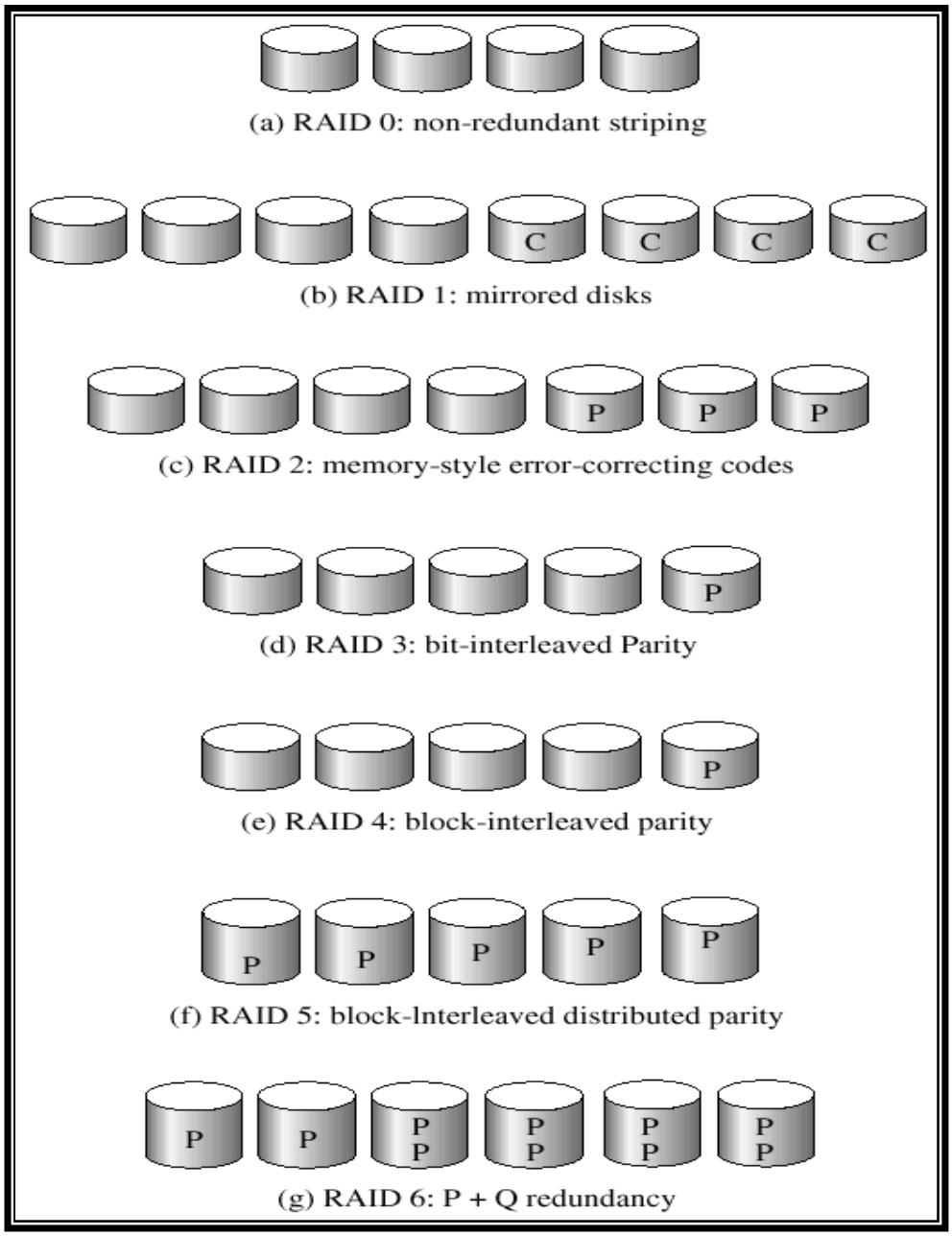
RAID

- **RAID** – multiple disk drives provides **reliability** via **redundancy**.
- Redundant Array of Independent Disks (RAID)
- RAID is arranged into six different levels.
- Several improvements in disk-use techniques involve the use of multiple disks working cooperatively.

- Disk striping uses a group of disks as one storage unit.

- RAID schemes improve performance and improve the reliability of the storage system by storing redundant data.
  - *Mirroring* or *shadowing* keeps duplicate of each disk.
  - *Block interleaved parity* uses much less redundancy.

RAID levels


(a) RAID 0: non-redundant striping
C
C
C
C
(b) RAID 1: mirrored disks
P
P
P
(c) RAID 2: memory-style error-correcting codes
P
(d) RAID 3: bit-interleaved Parity
P
(e) RAID 4: block-interleaved parity
P
P
P
P
P
(f) RAID 5: block-Interleaved distributed parity
P
P
P P
P P
P P
P P
(g) RAID 6: P + Q redundancy